index.txt For Vim version 9.1. Last change: 2025 Nov 09

VIM REFERENCE MANUAL by Bram Moolenaar

index

This file contains a list of all commands for each mode, with a tag and a short description. The lists are sorted on ASCII value.

Tip: When looking for certain functionality, use a search command. E.g., to look for deleting something, use: "/delete".

For an overview of options see option-list.
For an overview of built-in functions see functions.
For a list of Vim variables see vim-variable.
For a complete listing of all help items see help-tags.

==========

1. Insert mode *insert-index*

tag	char	action in Insert mode
i_CTRL-@	CTRL-@	insert previously inserted text and stop insert
i_CTRL-A	CTRL-A	insert previously inserted text
	CTRL-B	not used i_CTRL-B-gone
i_CTRL-C	CTRL-C	quit insert mode, without checking for abbreviation, unless 'insertmode' set.
i_CTRL-D	CTRL-D	delete one shiftwidth of indent in the current line
i_CTRL-E	CTRL-E	insert the character which is below the cursor
	CTRL-F	not used (but by default it's in 'cinkeys' to re-indent the current line)
i_CTRL-G_j	CTRL-G CTRL-J	line down, to column where inserting started
i_CTRL-G_j	CTRL-G j	line down, to column where inserting started
i_CTRL-G_j	CTRL-G <Down>	line down, to column where inserting started

i_CTRL-G_k CTRL-G CTRL-K line up, to column where inserting started
i_CTRL-G_k CTRL-G k line up, to column where inserting started
i_CTRL-G_k CTRL-G <Up> line up, to column where inserting started
i_CTRL-G_u CTRL-G u start new undoable edit
i_CTRL-G_U CTRL-G U don't break undo with next cursor movement
i_<BS> <BS> delete character before the cursor
i_digraph {char1}<BS>{char2}
enter digraph (only when 'digraph' option set)
i_CTRL-H CTRL-H same as <BS>
i_<Tab> <Tab> insert a <Tab> character
i_CTRL-I CTRL-I same as <Tab>
i_<NL> <NL> same as <CR>
i_CTRL-J CTRL-J same as <CR>
i_CTRL-K CTRL-K {char1} {char2}
enter digraph
i_CTRL-L CTRL-L when 'insertmode' set: Leave Insert mode
i_<CR> <CR> begin new line
i_CTRL-M CTRL-M same as <CR>
i_CTRL-N CTRL-N find next match for keyword in front of the cursor
i_CTRL-O CTRL-O execute a single command and return to insert mode
i_CTRL-P CTRL-P find previous match for keyword in front of the cursor
i_CTRL-Q CTRL-Q same as CTRL-V, unless used for terminal control flow
i_CTRL-SHIFT-Q CTRL-SHIFT-Q {char}
like CTRL-Q unless modifyOtherKeys is active
i_CTRL-R CTRL-R {register}
insert the contents of a register
i_CTRL-R_CTRL-R CTRL-R CTRL-R {register}
insert the contents of a register literally
i_CTRL-R_CTRL-O CTRL-R CTRL-O {register}
insert the contents of a register literally and don't auto-indent
i_CTRL-R_CTRL-P CTRL-R CTRL-P {register}
insert the contents of a register literally and fix indent.
CTRL-S not used or used for terminal control flow
i_CTRL-T CTRL-T insert one shiftwidth of indent in current line
i_CTRL-U CTRL-U delete all entered characters in the current line
i_CTRL-V CTRL-V {char} insert next non-digit literally
i_CTRL-SHIFT-V CTRL-SHIFT-V {char}
like CTRL-V unless modifyOtherKeys is active
i_CTRL-V_digit CTRL-V {number} insert three digit decimal number as a single byte.
i_CTRL-W CTRL-W delete word before the cursor
i_CTRL-X CTRL-X {mode} enter CTRL-X sub mode, see i_CTRL-X_index
i_CTRL-Y CTRL-Y insert the character which is above the cursor
i_CTRL-Z CTRL-Z when 'insertmode' set: suspend Vim
i_<Esc> <Esc> end insert mode (unless 'insertmode' set)
i_CTRL-[CTRL-[same as <Esc>
i_CTRL-_CTRL-N CTRL-\ CTRL-N go to Normal mode
i_CTRL-_CTRL-G CTRL-\ CTRL-G go to mode specified with 'insertmode'

	CTRL-\ a - z	reserved for extensions
	CTRL-\ others	not used
i_CTRL-]	CTRL-]	trigger abbreviation
i_CTRL-^	CTRL-^	toggle use of :lmap mappings
i_CTRL-_	CTRL-_	When 'allowrevins' set: change language (Hebrew, Farsi) {only when compiled with the +rightleft feature}
	<Space> to '~'	not used, except '0' and '^' followed by CTRL-D
i_0_CTRL-D	0 CTRL-D	delete all indent in the current line
i_^_CTRL-D	^ CTRL-D	delete all indent in the current line, restore it in the next line
i_<Del>	<Del>	delete character under the cursor
	Meta characters (0x80 to 0xff, 128 to 255)	not used
i_<Left>	<Left>	cursor one character left
i_<S-Left>	<S-Left>	cursor one word left
i_<C-Left>	<C-Left>	cursor one word left
i_<Right>	<Right>	cursor one character right
i_<S-Right>	<S-Right>	cursor one word right
i_<C-Right>	<C-Right>	cursor one word right
i_<Up>	<Up>	cursor one line up
i_<S-Up>	<S-Up>	same as <PageUp>
i_<Down>	<Down>	cursor one line down
i_<S-Down>	<S-Down>	same as <PageDown>
i_<Home>	<Home>	cursor to start of line
i_<C-Home>	<C-Home>	cursor to start of file
i_<End>	<End>	cursor past end of line
i_<C-End>	<C-End>	cursor past end of file
i_<PageUp>	<PageUp>	one screenful backward
i_<PageDown>	<PageDown>	one screenful forward
i_<F1>	<F1>	same as <Help>
i_<Help>	<Help>	stop insert mode and display help window
i_<Insert>	<Insert>	toggle Insert/Replace mode
i_<LeftMouse>	<LeftMouse>	cursor at mouse click
i_<ScrollWheelDown>	<ScrollWheelDown>	move window three lines down
i_<S-ScrollWheelDown>	<S-ScrollWheelDown>	move window one page down
i_<ScrollWheelUp>	<ScrollWheelUp>	move window three lines up
i_<S-ScrollWheelUp>	<S-ScrollWheelUp>	move window one page up
i_<ScrollWheelLeft>	<ScrollWheelLeft>	move window six columns left
i_<S-ScrollWheelLeft>	<S-ScrollWheelLeft>	move window one page left
i_<ScrollWheelRight>	<ScrollWheelRight>	move window six columns right
i_<S-ScrollWheelRight>	<S-ScrollWheelRight>	move window one page right

commands in CTRL-X submode *i_CTRL-X_index*

i_CTRL-X_CTRL-D	CTRL-X CTRL-D	complete defined identifiers
i_CTRL-X_CTRL-E	CTRL-X CTRL-E	scroll up
i_CTRL-X_CTRL-F	CTRL-X CTRL-F	complete file names
i_CTRL-X_CTRL-I	CTRL-X CTRL-I	complete identifiers
i_CTRL-X_CTRL-K	CTRL-X CTRL-K	complete identifiers from dictionary

i_CTRL-X_CTRL-L	CTRL-X CTRL-L	complete whole lines
i_CTRL-X_CTRL-N	CTRL-X CTRL-N	next completion
i_CTRL-X_CTRL-O	CTRL-X CTRL-O	omni completion
i_CTRL-X_CTRL-P	CTRL-X CTRL-P	previous completion
i_CTRL-X_CTRL-R	CTRL-X CTRL-R	complete contents from registers
i_CTRL-X_CTRL-S	CTRL-X CTRL-S	spelling suggestions
i_CTRL-X_CTRL-T	CTRL-X CTRL-T	complete identifiers from thesaurus
i_CTRL-X_CTRL-Y	CTRL-X CTRL-Y	scroll down
i_CTRL-X_CTRL-U	CTRL-X CTRL-U	complete with 'completefunc'
i_CTRL-X_CTRL-V	CTRL-X CTRL-V	complete like in : command line
i_CTRL-X_CTRL-Z	CTRL-X CTRL-Z	stop completion, text is unchanged
i_CTRL-X_CTRL-]	CTRL-X CTRL-]	complete tags
i_CTRL-X_s	CTRL-X s	spelling suggestions

commands in completion mode (see popupmenu-keys)

complete_CTRL-E	CTRL-E	stop completion and go back to original text
complete_CTRL-Y	CTRL-Y	accept selected match and stop completion
	CTRL-L	insert one character from the current match
	<CR>	insert currently selected match
	<BS>	delete one character and redo search
	CTRL-H	same as <BS>
	<Up>	select the previous match
	<Down>	select the next match
	<PageUp>	select a match several entries back
	<PageDown>	select a match several entries forward
	other	stop completion and insert the typed character

==========

2. Normal mode *normal-index*

CHAR	any non-blank character
WORD	a sequence of non-blank characters
N	a number entered before the command
{motion}	a cursor movement command
Nmove	the text that is moved over with a {motion}
SECTION	a section that possibly starts with '}' instead of '{'

note: 1 = cursor movement command; 2 = can be undone/redone

tag	char	note	action in Normal mode
	CTRL-@		not used
CTRL-A	CTRL-A	2	add N to number at/after cursor
CTRL-B	CTRL-B	1	scroll N screens Backwards
CTRL-C	CTRL-C		interrupt current (search) command
CTRL-D	CTRL-D		scroll Down N lines (default: half a screen)
CTRL-E	CTRL-E		scroll N lines upwards (N lines Extra)
CTRL-F	CTRL-F	1	scroll N screens Forward
CTRL-G	CTRL-G		display current file name and position
<BS>	<BS>	1	same as "h"
CTRL-H	CTRL-H	1	same as "h"
<Tab>	<Tab>	1	go to N newer entry in jump list
CTRL-I	CTRL-I	1	same as <Tab>
<NL>	<NL>	1	same as "j"
<S-NL>	<S-NL>	1	same as CTRL-F

CTRL-J	CTRL-J	1	same as "j"
	CTRL-K		not used
CTRL-L	CTRL-L		redraw screen
<CR>	<CR>	1	cursor to the first CHAR N lines lower
<S-CR>	<S-CR>	1	same as CTRL-F
CTRL-M	CTRL-M	1	same as <CR>
CTRL-N	CTRL-N	1	same as "j"
CTRL-O	CTRL-O	1	go to N older entry in jump list
CTRL-P	CTRL-P	1	same as "k"
	CTRL-Q		not used, or used for terminal control flow
CTRL-R	CTRL-R	2	redo changes which were undone with 'u'
	CTRL-S		not used, or used for terminal control flow
CTRL-T	CTRL-T		jump to N older Tag in tag list
CTRL-U	CTRL-U		scroll N lines Upwards (default: half a screen)
CTRL-V	CTRL-V		start blockwise Visual mode
CTRL-W	CTRL-W {char}		window commands, see CTRL-W
CTRL-X	CTRL-X	2	subtract N from number at/after cursor
CTRL-Y	CTRL-Y		scroll N lines downwards
CTRL-Z	CTRL-Z		suspend program (or start new shell)
	CTRL-[<Esc>		not used
CTRL-_CTRL-N	CTRL-\ CTRL-N		go to Normal mode (no-op)
CTRL-_CTRL-G	CTRL-\ CTRL-G		go to mode specified with 'insertmode'
	CTRL-\ a - z		reserved for extensions
	CTRL-\ others		not used
CTRL-]	CTRL-]		:ta to ident under cursor
CTRL-^	CTRL-^		edit Nth alternate file (equivalent to ":e #N")
	CTRL-_		not used
<Space>	<Space>	1	same as "l"
!	!{motion}{filter}	2	filter Nmove text through the {filter} command
!!	!!{filter}	2	filter N lines through the {filter} command
quote	"{register}		use {register} for next delete, yank or put ({.%#:} only work with put)
#	#	1	search backward for the Nth occurrence of the ident under the cursor
$	$	1	cursor to the end of Nth next line
%	%	1	find the next (curly/square) bracket on this line and go to its match, or go to matching comment bracket, or go to matching preprocessor directive.
N%	{count}%	1	go to N percentage in the file
&	&	2	repeat last :s
'	'{a-zA-Z0-9}	1	cursor to the first CHAR on the line with mark {a-zA-Z0-9}
''	''	1	cursor to the first CHAR of the line where the cursor was before the latest jump.
'(	'(	1	cursor to the first CHAR on the line of the start of the current sentence
')	')	1	cursor to the first CHAR on the line of the end of the current sentence
'<	'<	1	cursor to the first CHAR of the line where highlighted area starts/started in the

			current buffer.
'>	'>	1	cursor to the first CHAR of the line where highlighted area ends/ended in the current buffer.
'[	'[	1	cursor to the first CHAR on the line of the start of last operated text or start of put text
']	']	1	cursor to the first CHAR on the line of the end of last operated text or end of put text
'{	'{	1	cursor to the first CHAR on the line of the start of the current paragraph
'}	'}	1	cursor to the first CHAR on the line of the end of the current paragraph
(	(	1	cursor N sentences backward
)	)	1	cursor N sentences forward
star	*	1	search forward for the Nth occurrence of the ident under the cursor
+	+	1	same as <CR>
<S-Plus>	<S-+>	1	same as CTRL-F
,	,	1	repeat latest f, t, F or T in opposite direction N times
-	-	1	cursor to the first CHAR N lines higher
<S-Minus>	<S-->	1	same as CTRL-B
.	.	2	repeat last change with count replaced with N
/	/{pattern}<CR>	1	search forward for the Nth occurrence of {pattern}
/<CR>	/<CR>	1	search forward for {pattern} of last search
0	0	1	cursor to the first char of the line
count	1		prepend to command to give a count
count	2		"
count	3		"
count	4		"
count	5		"
count	6		"
count	7		"
count	8		"
count	9		"
:	:	1	start entering an Ex command
N:	{count}:		start entering an Ex command with range from current line to N-1 lines down
;	;	1	repeat latest f, t, F or T N times
<	<{motion}	2	shift Nmove lines one 'shiftwidth' leftwards
<<	<<	2	shift N lines one 'shiftwidth' leftwards
=	={motion}	2	filter Nmove lines through "indent"
==	==	2	filter N lines through "indent"
>	>{motion}	2	shift Nmove lines one 'shiftwidth' rightwards
>>	>>	2	shift N lines one 'shiftwidth' rightwards
?	?{pattern}<CR>	1	search backward for the Nth previous occurrence of {pattern}
?<CR>	?<CR>	1	search backward for {pattern} of last search
@	@{a-z}	2	execute the contents of register {a-z} N times

@:	@:		repeat the previous ":" command N times
@@	@@	2	repeat the previous @{a-z} N times
A	A	2	append text after the end of the line N times
B	B	1	cursor N WORDS backward
C	["x]C	2	change from the cursor position to the end of the line, and N-1 more lines [into register x]; synonym for "c$"
D	["x]D	2	delete the characters under the cursor until the end of the line and N-1 more lines [into register x]; synonym for "d$"
E	E	1	cursor forward to the end of WORD N
F	F{char}	1	cursor to the Nth occurrence of {char} to the left
G	G	1	cursor to line N, default last line
H	H	1	cursor to line N from top of screen
I	I	2	insert text before the first CHAR on the line N times
J	J	2	Join N lines; default is 2
K	K		lookup Keyword under the cursor with 'keywordprg'
L	L	1	cursor to line N from bottom of screen
M	M	1	cursor to middle line of screen
N	N	1	repeat the latest '/' or '?' N times in opposite direction
O	O	2	begin a new line above the cursor and insert text, repeat N times
P	["x]P	2	put the text [from register x] before the cursor N times
Q	Q		switch to "Ex" mode
R	R	2	enter replace mode: overtype existing characters, repeat the entered text N-1 times
S	["x]S	2	delete N lines [into register x] and start insert; synonym for "cc".
T	T{char}	1	cursor till after Nth occurrence of {char} to the left
U	U	2	undo all latest changes on one line
V	V		start linewise Visual mode
W	W	1	cursor N WORDS forward
X	["x]X	2	delete N characters before the cursor [into register x]
Y	["x]Y		yank N lines [into register x]; synonym for "yy"
ZZ	ZZ		write if buffer changed and close window
ZQ	ZQ		close window without writing
[	[{char}		square bracket command (see [below)
	\		not used
]	]{char}		square bracket command (see] below)
^	^	1	cursor to the first CHAR of the line
_	_	1	cursor to the first CHAR N - 1 lines lower
`	`{a-zA-Z0-9}	1	cursor to the mark {a-zA-Z0-9}
`(	`(	1	cursor to the start of the current sentence
`)	`)	1	cursor to the end of the current sentence
`<	`<	1	cursor to the start of the highlighted area
`>	`>	1	cursor to the end of the highlighted area
`[	`[	1	cursor to the start of last operated text

			or start of putted text
`]	`]	1	cursor to the end of last operated text or end of putted text
``	``	1	cursor to the position before latest jump
`{	`{	1	cursor to the start of the current paragraph
`}	`}	1	cursor to the end of the current paragraph
a	a	2	append text after the cursor N times
b	b	1	cursor N words backward
c	["x]c{motion}	2	delete Nmove text [into register x] and start insert
cc	["x]cc	2	delete N lines [into register x] and start insert
d	["x]d{motion}	2	delete Nmove text [into register x]
dd	["x]dd	2	delete N lines [into register x]
do	do	2	same as ":diffget"
dp	dp	2	same as ":diffput"
e	e	1	cursor forward to the end of word N
f	f{char}	1	cursor to Nth occurrence of {char} to the right
g	g{char}		extended commands, see g below
h	h	1	cursor N chars to the left
i	i	2	insert text before the cursor N times
j	j	1	cursor N lines downward
k	k	1	cursor N lines upward
l	l	1	cursor N chars to the right
m	m{A-Za-z}		set mark {A-Za-z} at cursor position
n	n	1	repeat the latest '/' or '?' N times
o	o	2	begin a new line below the cursor and insert text, repeat N times
p	["x]p	2	put the text [from register x] after the cursor N times
q	q{0-9a-zA-Z"}		record typed characters into named register {0-9a-zA-Z"} (uppercase to append)
q	q		(while recording) stops recording
q:	q:		edit : command-line in command-line window
q/	q/		edit / command-line in command-line window
q?	q?		edit ? command-line in command-line window
r	r{char}	2	replace N chars with {char}
s	["x]s	2	(substitute) delete N characters [into register x] and start insert
t	t{char}	1	cursor till before Nth occurrence of {char} to the right
u	u	2	undo changes
v	v		start characterwise Visual mode
w	w	1	cursor N words forward
x	["x]x	2	delete N characters under and after the cursor [into register x]
y	["x]y{motion}		yank Nmove text [into register x]
yy	["x]yy		yank N lines [into register x]
z	z{char}		commands starting with 'z', see z below
{	{	1	cursor N paragraphs backward
bar	|	1	cursor to column N
}	}	1	cursor N paragraphs forward
~	~	2	'tildeop' off: switch case of N characters under cursor and move the cursor N characters to the right

~	~{motion}		'tildeop' on: switch case of Nmove text
<C-End>	<C-End>	1	same as "G"
<C-Home>	<C-Home>	1	same as "gg"
<C-Left>	<C-Left>	1	same as "b"
<C-LeftMouse>	<C-LeftMouse>		":ta" to the keyword at the mouse click
<C-Right>	<C-Right>	1	same as "w"
<C-RightMouse>	<C-RightMouse>		same as "CTRL-T"
<C-Tab>	<C-Tab>		same as "g<Tab>"
<Del>	["x]<Del>	2	same as "x"
N<Del>	{count}<Del>		remove the last digit from {count}
<Down>	<Down>	1	same as "j"
<End>	<End>	1	same as "$"
<F1>	<F1>		same as <Help>
<Help>	<Help>		open a help window
<Home>	<Home>	1	same as "0"
<Insert>	<Insert>	2	same as "i"
<Left>	<Left>	1	same as "h"
<LeftMouse>	<LeftMouse>	1	move cursor to the mouse click position
<MiddleMouse>	<MiddleMouse>	2	same as "gP" at the mouse click position
<PageDown>	<PageDown>		same as CTRL-F
<PageUp>	<PageUp>		same as CTRL-B
<Right>	<Right>	1	same as "l"
<RightMouse>	<RightMouse>		start Visual mode, move cursor to the mouse click position
<S-Down>	<S-Down>	1	same as CTRL-F
<S-Left>	<S-Left>	1	same as "b"
<S-LeftMouse>	<S-LeftMouse>		same as "*" at the mouse click position
<S-Right>	<S-Right>	1	same as "w"
<S-RightMouse>	<S-RightMouse>		same as "#" at the mouse click position
<S-Up>	<S-Up>	1	same as CTRL-B
<Undo>	<Undo>	2	same as "u"
<Up>	<Up>	1	same as "k"

<ScrollWheelDown>	<ScrollWheelDown>	move window three lines down
<S-ScrollWheelDown>	<S-ScrollWheelDown>	move window one page down
<ScrollWheelUp>	<ScrollWheelUp>	move window three lines up
<S-ScrollWheelUp>	<S-ScrollWheelUp>	move window one page up
<ScrollWheelLeft>	<ScrollWheelLeft>	move window six columns left
<S-ScrollWheelLeft>	<S-ScrollWheelLeft>	move window one page left
<ScrollWheelRight>	<ScrollWheelRight>	move window six columns right
<S-ScrollWheelRight>	<S-ScrollWheelRight>	move window one page right

==========

2.1 Text objects *objects*

These can be used after an operator or in Visual mode to select an object.

tag	command	action in op-pending and Visual mode
v_aquote	a"	double quoted string
v_a'	a'	single quoted string
v_a(	a(	same as ab
v_a)	a)	same as ab
v_a<	a<	"a <>" from '<' to the matching '>'
v_a>	a>	same as a<
v_aB	aB	"a Block" from "[{" to "]}" (with brackets)
v_aW	aW	"a WORD" (with white space)

v_a[	a[	"a []" from '[' to the matching ']'
v_a]	a]	same as a[
v_a`	a`	string in backticks
v_ab	ab	"a block" from "[(" to "])" (with braces)
v_ap	ap	"a paragraph" (with white space)
v_as	as	"a sentence" (with white space)
v_at	at	"a tag block" (with white space)
v_aw	aw	"a word" (with white space)
v_a{	a{	same as aB
v_a}	a}	same as aB
v_iquote	i"	double quoted string without the quotes
v_i'	i'	single quoted string without the quotes
v_i(	i(	same as ib
v_i)	i)	same as ib
v_i<	i<	"inner <>" from '<' to the matching '>'
v_i>	i>	same as i<
v_iB	iB	"inner Block" from "[{" and "]}"
v_iW	iW	"inner WORD"
v_i[	i[	"inner []" from '[' to the matching ']'
v_i]	i]	same as i[
v_i`	i`	string in backticks without the backticks
v_ib	ib	"inner block" from "[(" to "])"
v_ip	ip	"inner paragraph"
v_is	is	"inner sentence"
v_it	it	"inner tag block"
v_iw	iw	"inner word"
v_i{	i{	same as iB
v_i}	i}	same as iB

==========

2.2 Window commands *CTRL-W*

tag	command	action in Normal mode
CTRL-W_CTRL-B	CTRL-W CTRL-B	same as "CTRL-W b"
CTRL-W_CTRL-C	CTRL-W CTRL-C	no-op
CTRL-W_CTRL-D	CTRL-W CTRL-D	same as "CTRL-W d"
CTRL-W_CTRL-F	CTRL-W CTRL-F	same as "CTRL-W f"
	CTRL-W CTRL-G	same as "CTRL-W g ."
CTRL-W_CTRL-H	CTRL-W CTRL-H	same as "CTRL-W h"
CTRL-W_CTRL-I	CTRL-W CTRL-I	same as "CTRL-W i"
CTRL-W_CTRL-J	CTRL-W CTRL-J	same as "CTRL-W j"
CTRL-W_CTRL-K	CTRL-W CTRL-K	same as "CTRL-W k"
CTRL-W_CTRL-L	CTRL-W CTRL-L	same as "CTRL-W l"
CTRL-W_CTRL-N	CTRL-W CTRL-N	same as "CTRL-W n"
CTRL-W_CTRL-O	CTRL-W CTRL-O	same as "CTRL-W o"
CTRL-W_CTRL-P	CTRL-W CTRL-P	same as "CTRL-W p"
CTRL-W_CTRL-Q	CTRL-W CTRL-Q	same as "CTRL-W q"
CTRL-W_CTRL-R	CTRL-W CTRL-R	same as "CTRL-W r"
CTRL-W_CTRL-S	CTRL-W CTRL-S	same as "CTRL-W s"
CTRL-W_CTRL-T	CTRL-W CTRL-T	same as "CTRL-W t"
CTRL-W_CTRL-V	CTRL-W CTRL-V	same as "CTRL-W v"
CTRL-W_CTRL-W	CTRL-W CTRL-W	same as "CTRL-W w"
CTRL-W_CTRL-X	CTRL-W CTRL-X	same as "CTRL-W x"
CTRL-W_CTRL-Z	CTRL-W CTRL-Z	same as "CTRL-W z"
CTRL-W_CTRL-]	CTRL-W CTRL-]	same as "CTRL-W]"

CTRL-W_CTRL-^	CTRL-W CTRL-^	same as "CTRL-W ^"
CTRL-W_CTRL-_	CTRL-W CTRL-_	same as "CTRL-W _"
CTRL-W_+	CTRL-W +	increase current window height N lines
CTRL-W_-	CTRL-W -	decrease current window height N lines
CTRL-W_:	CTRL-W :	same as :, edit a command line
CTRL-W_<	CTRL-W <	decrease current window width N columns
CTRL-W_=	CTRL-W =	make all windows the same height & width
CTRL-W_>	CTRL-W >	increase current window width N columns
CTRL-W_H	CTRL-W H	move current window to the far left
CTRL-W_J	CTRL-W J	move current window to the very bottom
CTRL-W_K	CTRL-W K	move current window to the very top
CTRL-W_L	CTRL-W L	move current window to the far right
CTRL-W_P	CTRL-W P	go to preview window
CTRL-W_R	CTRL-W R	rotate windows upwards N times
CTRL-W_S	CTRL-W S	same as "CTRL-W s"
CTRL-W_T	CTRL-W T	move current window to a new tab page
CTRL-W_W	CTRL-W W	go to N previous window (wrap around)
CTRL-W_]	CTRL-W]	split window and jump to tag under cursor
CTRL-W_^	CTRL-W ^	split current window and edit alternate file N
CTRL-W__	CTRL-W _	set current window height to N (default: very high)
CTRL-W_b	CTRL-W b	go to bottom window
CTRL-W_c	CTRL-W c	close current window (like :close)
CTRL-W_d	CTRL-W d	split window and jump to definition under the cursor
CTRL-W_f	CTRL-W f	split window and edit file name under the cursor
CTRL-W_F	CTRL-W F	split window and edit file name under the cursor and jump to the line number following the file name.
CTRL-W_g_CTRL-]	CTRL-W g CTRL-]	split window and do :tjump to tag under cursor
CTRL-W_g]	CTRL-W g]	split window and do :tselect for tag under cursor
CTRL-W_g}	CTRL-W g }	do a :ptjump to the tag under the cursor
CTRL-W_gf	CTRL-W g f	edit file name under the cursor in a new tab page
CTRL-W_gF	CTRL-W g F	edit file name under the cursor in a new tab page and jump to the line number following the file name.
CTRL-W_gt	CTRL-W g t	same as *gt*: go to next tab page
CTRL-W_gT	CTRL-W g T	same as *gT*: go to previous tab page
CTRL-W_g<Tab>	CTRL-W g <Tab>	same as g<Tab>: go to last accessed tab page.
CTRL-W_h	CTRL-W h	go to Nth left window (stop at first window)
CTRL-W_i	CTRL-W i	split window and jump to declaration of identifier under the cursor
CTRL-W_j	CTRL-W j	go N windows down (stop at last window)
CTRL-W_k	CTRL-W k	go N windows up (stop at first window)
CTRL-W_l	CTRL-W l	go to Nth right window (stop at last window)
CTRL-W_n	CTRL-W n	open new window, N lines high
CTRL-W_o	CTRL-W o	close all but current window (like :only)
CTRL-W_p	CTRL-W p	go to previous (last accessed) window
CTRL-W_q	CTRL-W q	quit current window (like :quit)
CTRL-W_r	CTRL-W r	rotate windows downwards N times

CTRL-W_s	CTRL-W s	split current window in two parts, new window N lines high
CTRL-W_t	CTRL-W t	go to top window
CTRL-W_v	CTRL-W v	split current window vertically, new window N columns wide
CTRL-W_w	CTRL-W w	go to N next window (wrap around)
CTRL-W_x	CTRL-W x	exchange current window with window N (default: next window)
CTRL-W_z	CTRL-W z	close preview window
CTRL-W_bar	CTRL-W |	set window width to N columns
CTRL-W_}	CTRL-W }	show tag under cursor in preview window
CTRL-W_<Down>	CTRL-W <Down>	same as "CTRL-W j"
CTRL-W_<Up>	CTRL-W <Up>	same as "CTRL-W k"
CTRL-W_<Left>	CTRL-W <Left>	same as "CTRL-W h"
CTRL-W_<Right>	CTRL-W <Right>	same as "CTRL-W l"

==========

2.3 Square bracket commands [ ]

tag	char	note	action in Normal mode
[_CTRL-D	[CTRL-D		jump to first #define found in current and included files matching the word under the cursor, start searching at beginning of current file
[_CTRL-I	[CTRL-I		jump to first line in current and included files that contains the word under the cursor, start searching at beginning of current file
[#	[#	1	cursor to N previous unmatched #if, #else or #ifdef
['	['	1	cursor to previous lowercase mark, on first non-blank
[(	[(	1	cursor N times back to unmatched '('
[star	[*	1	same as "[/"
[`	[`	1	cursor to previous lowercase mark
[/	[/	1	cursor to N previous start of a C comment
[D	[D		list all defines found in current and included files matching the word under the cursor, start searching at beginning of current file
[I	[I		list all lines found in current and included files that contain the word under the cursor, start searching at beginning of current file
[P	[P	2	same as "[p"
[[	[[	1	cursor N sections backward
[]	[]	1	cursor N SECTIONS backward
[c	[c	1	cursor N times backwards to start of change
[d	[d		show first #define found in current and included files matching the word under the cursor, start searching at beginning of current file
[f	[f		same as "gf"
[i	[i		show first line found in current and included files that contains the word under

			the cursor, start searching at beginning of current file
[m	[m	1	cursor N times back to start of member function
[p	[p	2	like "P", but adjust indent to current line
[s	[s	1	move to the previous misspelled word
[z	[z	1	move to start of open fold
[{	[{	1	cursor N times back to unmatched '{'
[<MiddleMouse>	[<MiddleMouse>	2	same as "[p"
]_CTRL-D	] CTRL-D		jump to first #define found in current and included files matching the word under the cursor, start searching at cursor position
]_CTRL-I	] CTRL-I		jump to first line in current and included files that contains the word under the cursor, start searching at cursor position
]#	]#	1	cursor to N next unmatched #endif or #else
]'	]'	1	cursor to next lowercase mark, on first non-blank
])	])	1	cursor N times forward to unmatched ')'
]star	]*	1	same as "]/"
]`	]`	1	cursor to next lowercase mark
]/	]/	1	cursor to N next end of a C comment
]D	]D		list all #defines found in current and included files matching the word under the cursor, start searching at cursor position
]I	]I		list all lines found in current and included files that contain the word under the cursor, start searching at cursor position
]P	]P	2	same as "[p"
][	][	1	cursor N SECTIONS forward
]]	]]	1	cursor N sections forward
]c	]c	1	cursor N times forward to start of change
]d	]d		show first #define found in current and included files matching the word under the cursor, start searching at cursor position
]f	]f		same as "gf"
]i	]i		show first line found in current and included files that contains the word under the cursor, start searching at cursor position
]m	]m	1	cursor N times forward to end of member function
]p	]p	2	like "p", but adjust indent to current line
]s	]s	1	move to next misspelled word
]z	]z	1	move to end of open fold
]}	]}	1	cursor N times forward to unmatched '}'
]<MiddleMouse>	]<MiddleMouse>	2	same as "]p"

==========

2.4 Commands starting with 'g' *g*

tag	char	note	action in Normal mode
g_CTRL-A	g CTRL-A		only when compiled with MEM_PROFILE

			defined: dump a memory profile
g_CTRL-G	g CTRL-G		show information about current cursor position
g_CTRL-H	g CTRL-H		start Select block mode
g_CTRL-]	g CTRL-]		:tjump to the tag under the cursor
g#	g#	1	like "#", but without using "\<" and "\>"
g$	g$	1	when 'wrap' off go to rightmost character of the current line that is on the screen; when 'wrap' on go to the rightmost character of the current screen line
g&	g&	2	repeat last ":s" on all lines
g'	g'{mark}	1	like ' but without changing the jumplist
g`	g`{mark}	1	like ` but without changing the jumplist
gstar	g*	1	like "*", but without using "\<" and "\>"
g+	g+		go to newer text state N times
g,	g,	1	go to N newer position in change list
g-	g-		go to older text state N times
g0	g0	1	when 'wrap' off go to leftmost character of the current line that is on the screen; when 'wrap' on go to the leftmost character of the current screen line
g8	g8		print hex value of bytes used in UTF-8 character under the cursor
g;	g;	1	go to N older position in change list
g≤	g≤		display previous command output
g?	g?	2	Rot13 encoding operator
g?g?	g??	2	Rot13 encode current line
g?g?	g?g?	2	Rot13 encode current line
gD	gD	1	go to definition of word under the cursor in current file
gE	gE	1	go backwards to the end of the previous WORD
gH	gH		start Select line mode
gI	gI	2	like "I", but always start in column 1
gJ	gJ	2	join lines without inserting space
gN	gN	1,2	find the previous match with the last used search pattern and Visually select it
gP	["x]gP	2	put the text [from register x] before the cursor N times, leave the cursor after it
gQ	gQ		switch to "Ex" mode with Vim editing
gR	gR	2	enter Virtual Replace mode
gT	gT		go to the previous tab page
gU	gU{motion}	2	make Nmove text uppercase
gV	gV		don't reselect the previous Visual area when executing a mapping or menu in Select mode
g]	g]		:tselect on the tag under the cursor
g^	g^	1	when 'wrap' off go to leftmost non-white character of the current line that is on the screen; when 'wrap' on go to the leftmost non-white character of the current screen line
g_	g_	1	cursor to the last CHAR N - 1 lines lower
ga	ga		print ascii value of character under the cursor
gd	gd	1	go to definition of word under the cursor

			in current function
ge	ge	1	go backwards to the end of the previous word
gf	gf		start editing the file whose name is under the cursor
gF	gF		start editing the file whose name is under the cursor and jump to the line number following the filename.
gg	gg	1	cursor to line N, default first line
gh	gh		start Select mode
gi	gi	2	like "i", but first move to the '^ mark
gj	gj	1	like "j", but when 'wrap' on go N screen lines down
gk	gk	1	like "k", but when 'wrap' on go N screen lines up
gm	gm	1	go to character at middle of the screenline
gM	gM	1	go to character at middle of the text line
gn	gn	1,2	find the next match with the last used search pattern and Visually select it
go	go	1	cursor to byte N in the buffer
gp	["x]gp	2	put the text [from register x] after the cursor N times, leave the cursor after it
gq	gq{motion}	2	format Nmove text
gr	gr{char}	2	virtual replace N chars with {char}
gs	gs		go to sleep for N seconds (default 1)
gt	gt		go to the next tab page
gu	gu{motion}	2	make Nmove text lowercase
gv	gv		reselect the previous Visual area
gw	gw{motion}	2	format Nmove text and keep cursor
g@	g@{motion}		call 'operatorfunc'
g~	g~{motion}	2	swap case for Nmove text
g<Down>	g<Down>	1	same as "gj"
g<End>	g<End>	1	same as "g$" but go to the rightmost non-blank character instead
g<Home>	g<Home>	1	same as "g0"
g<LeftMouse>	g<LeftMouse>		same as <C-LeftMouse>
	g<MiddleMouse>		same as <C-MiddleMouse>
g<RightMouse>	g<RightMouse>		same as <C-RightMouse>
g<Tab>	g<Tab>		go to the last accessed tab page.
g<Up>	g<Up>	1	same as "gk"

==========

2.5 Commands starting with 'z' *z*

tag	char	note	action in Normal mode
z<CR>	z<CR>		redraw, cursor line to top of window, cursor on first non-blank
zN<CR>	z{height}<CR>		redraw, make window {height} lines high
z+	z+		cursor on line N (default line below window), otherwise like "z<CR>"
z-	z-		redraw, cursor line at bottom of window, cursor on first non-blank
z.	z.		redraw, cursor line to center of window, cursor on first non-blank
z=	z=		give spelling suggestions

zA	zA		open a closed fold or close an open fold recursively
zC	zC		close folds recursively
zD	zD		delete folds recursively
zE	zE		eliminate all folds
zF	zF		create a fold for N lines
zG	zG		temporarily mark word as correctly spelled
zH	zH		when 'wrap' off scroll half a screenwidth to the right
zL	zL		when 'wrap' off scroll half a screenwidth to the left
zM	zM		set 'foldlevel' to zero
zN	zN		set 'foldenable'
zO	zO		open folds recursively
zR	zR		set 'foldlevel' to the deepest fold
zW	zW		temporarily mark word as incorrectly spelled
zX	zX		re-apply 'foldlevel'
z^	z^		cursor on line N (default line above window), otherwise like "z-"
za	za		open a closed fold, close an open fold
zb	zb		redraw, cursor line at bottom of window
zc	zc		close a fold
zd	zd		delete a fold
ze	ze		when 'wrap' off scroll horizontally to position the cursor at the end (right side) of the screen
zf	zf{motion}		create a fold for Nmove text
zg	zg		permanently mark word as correctly spelled
zh	zh		when 'wrap' off scroll screen N characters to the right
zi	zi		toggle 'foldenable'
zj	zj	1	move to the start of the next fold
zk	zk	1	move to the end of the previous fold
zl	zl		when 'wrap' off scroll screen N characters to the left
zm	zm		subtract one from 'foldlevel'
zn	zn		reset 'foldenable'
zo	zo		open fold
zp	zp		paste in block-mode without trailing spaces
zP	zP		paste in block-mode without trailing spaces
zr	zr		add one to 'foldlevel'
zs	zs		when 'wrap' off scroll horizontally to position the cursor at the start (left side) of the screen
zt	zt		redraw, cursor line at top of window
zuw	zuw		undo zw
zug	zug		undo zg
zuW	zuW		undo zW
zuG	zuG		undo zG
zv	zv		open enough folds to view the cursor line
zw	zw		permanently mark word as incorrectly spelled
zx	zx		re-apply 'foldlevel' and do "zv"
zy	zy		yank without trailing spaces
zz	zz		redraw, cursor line at center of window
z<Left>	z<Left>		same as "zh"
z<Right>	z<Right>		same as "zl"

2.6 Operator-pending mode *operator-pending-index*

These can be used after an operator, but before a {motion} has been entered.

tag	char	action in Operator-pending mode
o_v	v	force operator to work characterwise
o_V	V	force operator to work linewise
o_CTRL-V	CTRL-V	force operator to work blockwise

3. Visual mode *visual-index*

Most commands in Visual mode are the same as in Normal mode. The ones listed here are those that are different.

tag	command	note	action in Visual mode
v_CTRL-_CTRL-N	CTRL-\ CTRL-N		stop Visual mode
v_CTRL-_CTRL-G	CTRL-\ CTRL-G		go to mode specified with 'insertmode'
v_CTRL-A	CTRL-A	2	add N to number in highlighted text
v_CTRL-C	CTRL-C		stop Visual mode
v_CTRL-G	CTRL-G		toggle between Visual mode and Select mode
v_<BS>	<BS>	2	Select mode: delete highlighted area
v_CTRL-H	CTRL-H	2	same as <BS>
v_CTRL-O	CTRL-O		switch from Select to Visual mode for one command
v_CTRL-V	CTRL-V		make Visual mode blockwise or stop Visual mode
v_CTRL-X	CTRL-X	2	subtract N from number in highlighted text
v_<Esc>	<Esc>		stop Visual mode
v_CTRL-]	CTRL-]		jump to highlighted tag
v_!	!{filter}	2	filter the highlighted lines through the external command {filter}
v_:	:		start a command-line with the highlighted lines as a range
v_<	<	2	shift the highlighted lines one 'shiftwidth' left
v_=	=	2	filter the highlighted lines through the external program given with the 'equalprg' option
v_>	>	2	shift the highlighted lines one 'shiftwidth' right
v_b_A	A	2	block mode: append same text in all lines, after the highlighted area
v_C	C	2	delete the highlighted lines and start insert
v_D	D	2	delete the highlighted lines
v_b_I	I	2	block mode: insert same text in all lines, before the highlighted area
v_J	J	2	join the highlighted lines
v_K	K		run 'keywordprg' on the highlighted area
v_O	O		move horizontally to other corner of area
v_P	P		replace highlighted area with register

			contents; registers are unchanged
	Q		does not start Ex mode
v_R	R	2	delete the highlighted lines and start insert
v_S	S	2	delete the highlighted lines and start insert
v_U	U	2	make highlighted area uppercase
v_V	V		make Visual mode linewise or stop Visual mode
v_X	X	2	delete the highlighted lines
v_Y	Y		yank the highlighted lines
v_aquote	a"		extend highlighted area with a double quoted string
v_a'	a'		extend highlighted area with a single quoted string
v_a(	a(		same as ab
v_a)	a)		same as ab
v_a<	a<		extend highlighted area with a <> block
v_a>	a>		same as a<
v_aB	aB		extend highlighted area with a {} block
v_aW	aW		extend highlighted area with "a WORD"
v_a[	a[		extend highlighted area with a [] block
v_a]	a]		same as a[
v_a`	a`		extend highlighted area with a backtick quoted string
v_ab	ab		extend highlighted area with a () block
v_ap	ap		extend highlighted area with a paragraph
v_as	as		extend highlighted area with a sentence
v_at	at		extend highlighted area with a tag block
v_aw	aw		extend highlighted area with "a word"
v_a{	a{		same as aB
v_a}	a}		same as aB
v_c	c	2	delete highlighted area and start insert
v_d	d	2	delete highlighted area
v_g_CTRL-A	g CTRL-A	2	add N to number in highlighted text
v_g_CTRL-X	g CTRL-X	2	subtract N from number in highlighted text
v_gJ	gJ	2	join the highlighted lines without inserting spaces
v_gq	gq	2	format the highlighted lines
v_gv	gv		exchange current and previous highlighted area
v_iquote	i"		extend highlighted area with a double quoted string (without quotes)
v_i'	i'		extend highlighted area with a single quoted string (without quotes)
v_i(	i(		same as ib
v_i)	i)		same as ib
v_i<	i<		extend highlighted area with inner <> block
v_i>	i>		same as i<
v_iB	iB		extend highlighted area with inner {} block
v_iW	iW		extend highlighted area with "inner WORD"
v_i[	i[		extend highlighted area with inner [] block
v_i]	i]		same as i[
v_i`	i`		extend highlighted area with a backtick quoted string (without the backticks)
v_ib	ib		extend highlighted area with inner () block

v_ip	ip		extend highlighted area with inner paragraph
v_is	is		extend highlighted area with inner sentence
v_it	it		extend highlighted area with inner tag block
v_iw	iw		extend highlighted area with "inner word"
v_i{	i{		same as iB
v_i}	i}		same as iB
v_o	o		move cursor to other corner of area
v_p	p		replace highlighted area with register contents; deleted text in unnamed register
v_r	r	2	replace highlighted area with a character
v_s	s	2	delete highlighted area and start insert
v_u	u	2	make highlighted area lowercase
v_v	v		make Visual mode characterwise or stop Visual mode
v_x	x	2	delete the highlighted area
v_y	y		yank the highlighted area
v_~	~	2	swap case for the highlighted area

==========

4. Command-line editing *ex-edit-index*

Get to the command-line with the ':', '!', '/' or '?' commands.
Normal characters are inserted at the current cursor position.
"Completion" below refers to context-sensitive completion. It will complete file names, tags, commands etc. as appropriate.

tag	command	action in Command-line editing mode
	CTRL-@	not used
c_CTRL-A	CTRL-A	do completion on the pattern in front of the cursor and insert all matches
c_CTRL-B	CTRL-B	cursor to begin of command-line
c_CTRL-C	CTRL-C	same as <Esc>
c_CTRL-D	CTRL-D	list completions that match the pattern in front of the cursor
c_CTRL-E	CTRL-E	cursor to end of command-line
'cedit'	CTRL-F	default value for 'cedit': opens the command-line window; otherwise not used
c_CTRL-G	CTRL-G	next match when 'incsearch' is active
c_<BS>	<BS>	delete the character in front of the cursor
c_digraph	{char1} <BS> {char2}	enter digraph when 'digraph' is on
c_CTRL-H	CTRL-H	same as <BS>
c_<Tab>	<Tab>	if 'wildchar' is <Tab>: Do completion on the pattern in front of the cursor
c_<S-Tab>	<S-Tab>	same as CTRL-P
c_wildchar	'wildchar'	Do completion on the pattern in front of the cursor (default: <Tab>)
c_CTRL-I	CTRL-I	same as <Tab>
c_<NL>	<NL>	same as <CR>
c_CTRL-J	CTRL-J	same as <CR>
c_CTRL-K	CTRL-K {char1} {char2}	enter digraph
c_CTRL-L	CTRL-L	do completion on the pattern in front of the cursor and insert the longest common part
c_<CR>	<CR>	execute entered command

c_CTRL-M	CTRL-M	same as <CR>
c_CTRL-N	CTRL-N	after using 'wildchar' with multiple matches: go to next match, otherwise: recall older command-line from history.
	CTRL-O	not used
c_CTRL-P	CTRL-P	after using 'wildchar' with multiple matches: go to previous match, otherwise: recall older command-line from history.
c_CTRL-Q	CTRL-Q	same as CTRL-V, unless it's used for terminal control flow
c_CTRL-R	CTRL-R {regname}	insert the contents of a register or object under the cursor as if typed
c_CTRL-R_CTRL-R	CTRL-R CTRL-R {regname}	
c_CTRL-R_CTRL-O	CTRL-R CTRL-O {regname}	insert the contents of a register or object under the cursor literally
	CTRL-S	not used, or used for terminal control flow
c_CTRL-T	CTRL-T	previous match when 'incsearch' is active
c_CTRL-U	CTRL-U	remove all characters
c_CTRL-V	CTRL-V	insert next non-digit literally, insert three digit decimal number as a single byte.
c_CTRL-W	CTRL-W	delete the word in front of the cursor
	CTRL-X	not used (reserved for completion)
	CTRL-Y	copy (yank) modeless selection
	CTRL-Z	not used (reserved for suspend)
c_<Esc>	<Esc>	abandon command-line without executing it
c_CTRL-[	CTRL-[	same as <Esc>
c_CTRL-_CTRL-N	CTRL-\ CTRL-N	go to Normal mode, abandon command-line
c_CTRL-_CTRL-G	CTRL-\ CTRL-G	go to mode specified with 'insertmode', abandon command-line
	CTRL-\ a - d	reserved for extensions
c_CTRL-_e	CTRL-\ e {expr}	replace the command line with the result of {expr}
	CTRL-\ f - z	reserved for extensions
	CTRL-\ others	not used
c_CTRL-]	CTRL-]	trigger abbreviation
c_CTRL-^	CTRL-^	toggle use of :lmap mappings
c_CTRL-_	CTRL-_	when 'allowrevins' set: change language (Hebrew, Farsi)
c_<Del>	<Del>	delete the character under the cursor
c_<Left>	<Left>	cursor left
c_<S-Left>	<S-Left>	cursor one word left
c_<C-Left>	<C-Left>	cursor one word left
c_<Right>	<Right>	cursor right
c_<S-Right>	<S-Right>	cursor one word right
c_<C-Right>	<C-Right>	cursor one word right
c_<Up>	<Up>	recall previous command-line from history that matches pattern in front of the cursor
c_<S-Up>	<S-Up>	recall previous command-line from history
c_<Down>	<Down>	recall next command-line from history that matches pattern in front of the cursor
c_<S-Down>	<S-Down>	recall next command-line from history
c_<Home>	<Home>	cursor to start of command-line
c_<End>	<End>	cursor to end of command-line

c_<PageDown>	<PageDown>	same as <S-Down>
c_<PageUp>	<PageUp>	same as <S-Up>
c_<Insert>	<Insert>	toggle insert/overstrike mode
c_<LeftMouse>	<LeftMouse>	cursor at mouse click

commands in wildmenu mode (see 'wildmenu')

<Up>	move up to parent / select the previous match
<Down>	move down to submenu / select the next match
<Left>	select the previous match / move up to parent
<Right>	select the next match / move down to submenu
<CR>	move into submenu when doing menu completion
CTRL-E	stop completion and go back to original text
CTRL-Y	accept selected match and stop completion
other	stop completion and insert the typed character

commands in wildmenu mode with 'wildoptions' set to "pum"

<PageUp>	select a match several entries back
<PageDown>	select a match several entries forward

==========

5. Terminal-Job mode *terminal-job-index*

Most Normal mode commands except for window commands (CTRL-W) do not work in a terminal window. Switch to Terminal-Normal mode to use them.
This assumes 'termwinkey' is not set.

tag	char	action in Terminal-Job mode
t_CTRL-_CTRL-N	CTRL-\ CTRL-N	switch to Terminal-Normal mode
t_CTRL-W_N	CTRL-W N	switch to Terminal-Normal mode
t_CTRL-W_:	CTRL-W :	enter an Ex command
t_CTRL-W_.	CTRL-W .	type CTRL-W in the terminal
	CTRL-W CTRL-\	send a CTRL-\ to the job in the terminal
t_CTRL-W_quote	CTRL-W " {register}	paste register in the terminal
t_CTRL-W_CTRL-C	CTRL-W CTRL-C	forcefully ends the job
t_CTRL-W_CTRL-W	CTRL-W CTRL-W	move focus to the next window
t_CTRL-W_gt	CTRL-W gt	go to next tabpage, same as *gt*
t_CTRL-W_gT	CTRL-W gT	go to previous tabpage, same as *gT*

You found it, Arthur! *holy-grail :smile*

==========

6. EX commands *ex-cmd-index :index*

This is a brief but complete listing of all the ":" commands, without mentioning any arguments. The optional part of the command name is inside []. The commands are sorted on the non-optional part of their name.

tag	command	action
:	:	nothing
:range	:{range}	go to last line in {range}
:!	:!	filter lines or execute an external command

:!!	:!!	repeat last ":!" command
:#	:#	same as ":number"
:&	:&	repeat last ":substitute"
:star	:*	use the last Visual area, like :'<,'>
:<	:<	shift lines one 'shiftwidth' left
:=	:=	print the last line number
:>	:>	shift lines one 'shiftwidth' right
:@	:@	execute contents of a register
:@@	:@@	repeat the previous ":@"
:2match	:2mat[ch]	define a second match to highlight
:3match	:3mat[ch]	define a third match to highlight
:Next	:N[ext]	go to previous file in the argument list
:Print	:P[rint]	print lines
:X	:X	ask for encryption key
:append	:a[ppend]	append text
:abbreviate	:ab[breviate]	enter abbreviation
:abclear	:abc[lear]	remove all abbreviations
:aboveleft	:abo[veleft]	make split window appear left or above
:abstract	:abstract	declare a Vim9 abstract class
:all	:al[l]	open a window for each file in the argument list
:amenu	:am[enu]	enter new menu item for all modes
:anoremenu	:an[oremenu]	enter a new menu for all modes that will not be remapped
:args	:ar[gs]	print the argument list
:argadd	:arga[dd]	add items to the argument list
:argdedupe	:argded[upe]	remove duplicates from the argument list
:argdelete	:argd[elete]	delete items from the argument list
:argedit	:arge[dit]	add item to the argument list and edit it
:argdo	:argdo	do a command on all items in the argument list
:argglobal	:argg[lobal]	define the global argument list
:arglocal	:argl[ocal]	define a local argument list
:argument	:argu[ment]	go to specific file in the argument list
:ascii	:as[cii]	print ascii value of character under the cursor
:autocmd	:au[tocmd]	enter or show autocommands
:augroup	:aug[roup]	select the autocommand group to use
:aunmenu	:aun[menu]	remove menu for all modes
:buffer	:b[uffer]	go to specific buffer in the buffer list
:bNext	:bN[ext]	go to previous buffer in the buffer list
:ball	:ba[ll]	open a window for each buffer in the buffer list
:badd	:bad[d]	add buffer to the buffer list
:balt	:balt	like ":badd" but also set the alternate file
:bdelete	:bd[elete]	remove a buffer from the buffer list
:behave	:be[have]	set mouse and selection behavior
:belowright	:bel[owright]	make split window appear right or below
:bfirst	:bf[irst]	go to first buffer in the buffer list
:blast	:bl[ast]	go to last buffer in the buffer list
:bmodified	:bm[odified]	go to next buffer in the buffer list that has been modified
:bnext	:bn[ext]	go to next buffer in the buffer list
:botright	:bo[tright]	make split window appear at bottom or far right
:bprevious	:bp[revious]	go to previous buffer in the buffer list
:brewind	:br[ewind]	go to first buffer in the buffer list
:break	:brea[k]	break out of while loop
:breakadd	:breaka[dd]	add a debugger breakpoint
:breakdel	:breakd[el]	delete a debugger breakpoint

:breaklist	:breakl[ist]	list debugger breakpoints
:browse	:bro[wse]	use file selection dialog
:bufdo	:bufd[o]	execute command in each listed buffer
:buffers	:buffers	list all files in the buffer list
:bunload	:bun[load]	unload a specific buffer
:bwipeout	:bw[ipeout]	really delete a buffer
:change	:c[hange]	replace a line or series of lines
:cNext	:cN[ext]	go to previous error
:cNfile	:cNf[ile]	go to last error in previous file
:cabbrev	:ca[bbrev]	like ":abbreviate" but for Command-line mode
:cabclear	:cabc[lear]	clear all abbreviations for Command-line mode
:cabove	:cabo[ve]	go to error above current line
:caddbuffer	:cad[dbuffer]	add errors from buffer
:caddexpr	:cadde[xpr]	add errors from expr
:caddfile	:caddf[ile]	add error message to current quickfix list
:cafter	:caf[ter]	go to error after current cursor
:call	:cal[l]	call a function
:catch	:cat[ch]	part of a :try command
:cbefore	:cbe[fore]	go to error before current cursor
:cbelow	:cbel[ow]	go to error below current line
:cbottom	:cbo[ttom]	scroll to the bottom of the quickfix window
:cbuffer	:cb[uffer]	parse error messages and jump to first error
:cc	:cc	go to specific error
:cclose	:ccl[ose]	close quickfix window
:cd	:cd	change directory
:cdo	:cdo	execute command in each valid error list entry
:cfdo	:cfd[o]	execute command in each file in error list
:center	:ce[nter]	format lines at the center
:cexpr	:cex[pr]	read errors from expr and jump to first
:cfile	:cf[ile]	read file with error messages and jump to first
:cfirst	:cfir[st]	go to the specified error, default first one
:cgetbuffer	:cgetb[uffer]	get errors from buffer
:cgetexpr	:cgete[xpr]	get errors from expr
:cgetfile	:cg[etfile]	read file with error messages
:changes	:changes	print the change list
:chdir	:chd[ir]	change directory
:checkpath	:che[ckpath]	list included files
:checktime	:checkt[ime]	check timestamp of loaded buffers
:chistory	:chi[story]	list the error lists
:class	:class	start of a class declaration
:clast	:cla[st]	go to the specified error, default last one
:clearjumps	:cle[arjumps]	clear the jump list
:clipreset	:clip[reset]	reset 'clipmethod'
:clist	:cl[ist]	list all errors
:close	:clo[se]	close current window
:cmap	:cm[ap]	like ":map" but for Command-line mode
:cmapclear	:cmapc[lear]	clear all mappings for Command-line mode
:cmenu	:cme[nu]	add menu for Command-line mode
:cnext	:cn[ext]	go to next error
:cnewer	:cnew[er]	go to newer error list
:cnfile	:cnf[ile]	go to first error in next file
:cnoremap	:cno[remap]	like ":noremap" but for Command-line mode
:cnoreabbrev	:cnorea[bbrev]	like ":noreabbrev" but for Command-line mode
:cnoremenu	:cnoreme[nu]	like ":noremenu" but for Command-line mode
:copy	:co[py]	copy lines
:colder	:col[der]	go to older error list

:colorscheme	:colo[rscheme]	load a specific color scheme
:command	:com[mand]	create user-defined command
:comclear	:comc[lear]	clear all user-defined commands
:compiler	:comp[iler]	do settings for a specific compiler
:continue	:con[tinue]	go back to :while
:confirm	:conf[irm]	prompt user when confirmation required
:const	:cons[t]	create a variable as a constant
:copen	:cope[n]	open quickfix window
:cprevious	:cp[revious]	go to previous error
:cpfile	:cpf[ile]	go to last error in previous file
:cquit	:cq[uit]	quit Vim with an error code
:crewind	:cr[ewind]	go to the specified error, default first one
:cscope	:cs[cope]	execute cscope command
:cstag	:cst[ag]	use cscope to jump to a tag
:cunmap	:cu[nmap]	like ":unmap" but for Command-line mode
:cunabbrev	:cuna[bbrev]	like ":unabbrev" but for Command-line mode
:cunmenu	:cunme[nu]	remove menu for Command-line mode
:cwindow	:cw[indow]	open or close quickfix window
:delete	:d[elete]	delete lines
:debug	:deb[ug]	run a command in debugging mode
:debuggreedy	:debugg[reedy]	read debug mode commands from normal input
:def	:def	define a Vim9 user function
:defcompile	:defc[ompile]	compile Vim9 user functions in current script
:defer	:defe[r]	call function when current function is done
:delcommand	:delc[ommand]	delete user-defined command
:delfunction	:delf[unction]	delete a user function
:delmarks	:delm[arks]	delete marks
:diffupdate	:dif[fupdate]	update 'diff' buffers
:diffget	:diffg[et]	remove differences in current buffer
:diffoff	:diffo[ff]	switch off diff mode
:diffpatch	:diffp[atch]	apply a patch and show differences
:diffput	:diffpu[t]	remove differences in other buffer
:diffsplit	:diffs[plit]	show differences with another file
:diffthis	:difft[his]	make current window a diff window
:digraphs	:dig[raphs]	show or enter digraphs
:display	:di[splay]	display registers
:disassemble	:disa[ssemble]	disassemble Vim9 user function
:djump	:dj[ump]	jump to #define
:dl	:dl	short for :delete with the 'l' flag
:dlist	:dli[st]	list #defines
:doautocmd	:do[autocmd]	apply autocommands to current buffer
:doautoall	:doautoa[ll]	apply autocommands for all loaded buffers
:dp	:d[elete]p	short for :delete with the 'p' flag
:drop	:dr[op]	jump to window editing file or edit file in current window
:dsearch	:ds[earch]	list one #define
:dsplit	:dsp[lit]	split window and jump to #define
:edit	:e[dit]	edit a file
:earlier	:ea[rlier]	go to older change, undo
:echo	:ec[ho]	echoes the result of expressions
:echoconsole	:echoc[onsole]	like :echomsg but write to stdout
:echoerr	:echoe[rr]	like :echo, show like an error and use history
:echohl	:echoh[l]	set highlighting for echo commands
:echomsg	:echom[sg]	same as :echo, put message in history
:echon	:echon	same as :echo, but without <EOL>
:echowindow	:echow[indow]	same as :echomsg, but use a popup window

:else	:el[se]	part of an :if command
:elseif	:elsei[f]	part of an :if command
:emenu	:em[enu]	execute a menu by name
:endclass	:endclass	end of a class declaration
:enddef	:enddef	end of a user function started with :def
:endenum	:endenum	end of an enum declaration
:endif	:en[dif]	end previous :if
:endinterface	:endinterface	end of an interface declaration
:endfor	:endfo[r]	end previous :for
:endfunction	:endf[unction]	end of a user function started with :function
:endtry	:endt[ry]	end previous :try
:endwhile	:endw[hile]	end previous :while
:enew	:ene[w]	edit a new, unnamed buffer
:enum	:enum	start of an enum declaration
:eval	:ev[al]	evaluate an expression and discard the result
:ex	:ex	same as ":edit"
:execute	:exe[cute]	execute result of expressions
:exit	:exi[t]	same as ":xit"
:export	:exp[ort]	Vim9: export an item from a script
:exusage	:exu[sage]	overview of Ex commands
:file	:f[ile]	show or set the current file name
:files	:files	list all files in the buffer list
:filetype	:filet[ype]	switch file type detection on/off
:filter	:filt[er]	filter output of following command
:find	:fin[d]	find file in 'path' and edit it
:final	:final	declare an immutable variable in Vim9
:finally	:fina[lly]	part of a :try command
:finish	:fini[sh]	quit sourcing a Vim script
:first	:fir[st]	go to the first file in the argument list
:fixdel	:fix[del]	set key code of <Del>
:fold	:fo[ld]	create a fold
:foldclose	:foldc[lose]	close folds
:folddoopen	:foldd[oopen]	execute command on lines not in a closed fold
:folddoclosed	:folddoc[losed]	execute command on lines in a closed fold
:foldopen	:foldo[pen]	open folds
:for	:for	for loop
:function	:fu[nction]	define a user function
:global	:g[lobal]	execute commands for matching lines
:goto	:go[to]	go to byte in the buffer
:grep	:gr[ep]	run 'grepprg' and jump to first match
:grepadd	:grepa[dd]	like :grep, but append to current list
:gui	:gu[i]	start the GUI
:gvim	:gv[im]	start the GUI
:hardcopy	:ha[rdcopy]	send text to the printer
:help	:h[elp]	open a help window
:helpclose	:helpc[lose]	close one help window
:helpfind	:helpf[ind]	dialog to open a help window
:helpgrep	:helpg[rep]	like ":grep" but searches help files
:helptags	:helpt[ags]	generate help tags for a directory
:highlight	:hi[ghlight]	specify highlighting methods
:hide	:hid[e]	hide current buffer for a command
:history	:his[tory]	print a history list
:horizontal	:hor[izontal]	following window command work horizontally
:insert	:i[nsert]	insert text
:iabbrev	:ia[bbrev]	like ":abbrev" but for Insert mode
:iabclear	:iabc[lear]	like ":abclear" but for Insert mode

:if	:if	execute commands when condition met
:ijump	:ij[ump]	jump to definition of identifier
:ilist	:il[ist]	list lines where identifier matches
:imap	:im[ap]	like ":map" but for Insert mode
:imapclear	:imapc[lear]	like ":mapclear" but for Insert mode
:imenu	:ime[nu]	add menu for Insert mode
:import	:imp[ort]	Vim9: import an item from another script
:inoremap	:ino[remap]	like ":noremap" but for Insert mode
:inoreabbrev	:inorea[bbrev]	like ":noreabbrev" but for Insert mode
:inoremenu	:inoreme[nu]	like ":noremenu" but for Insert mode
:intro	:int[ro]	print the introductory message
:interface	:interface	start of an interface declaration
:iput	:ip[ut]	like :put, but adjust the indent to the current line
:isearch	:is[earch]	list one line where identifier matches
:isplit	:isp[lit]	split window and jump to definition of identifier
:iunmap	:iu[nmap]	like ":unmap" but for Insert mode
:iunabbrev	:iuna[bbrev]	like ":unabbrev" but for Insert mode
:iunmenu	:iunme[nu]	remove menu for Insert mode
:join	:j[oin]	join lines
:jumps	:ju[mps]	print the jump list
:k	:k	set a mark
:keepalt	:keepa[lt]	following command keeps the alternate file
:keepmarks	:kee[pmarks]	following command keeps marks where they are
:keepjumps	:keepj[umps]	following command keeps jumplist and marks
:keeppatterns	:keepp[atterns]	following command keeps search pattern history
:lNext	:lN[ext]	go to previous entry in location list
:lNfile	:lNf[ile]	go to last entry in previous file
:list	:l[ist]	print lines
:labove	:lab[ove]	go to location above current line
:laddexpr	:lad[dexpr]	add locations from expr
:laddbuffer	:laddb[uffer]	add locations from buffer
:laddfile	:laddf[ile]	add locations to current location list
:lafter	:laf[ter]	go to location after current cursor
:last	:la[st]	go to the last file in the argument list
:language	:lan[guage]	set the language (locale)
:later	:lat[er]	go to newer change, redo
:lbefore	:lbe[fore]	go to location before current cursor
:lbelow	:lbel[ow]	go to location below current line
:lbottom	:lbo[ttom]	scroll to the bottom of the location window
:lbuffer	:lb[uffer]	parse locations and jump to first location
:lcd	:lc[d]	change directory locally
:lchdir	:lch[dir]	change directory locally
:lclose	:lcl[ose]	close location window
:lcscope	:lcs[cope]	like ":cscope" but uses location list
:ldo	:ld[o]	execute command in valid location list entries
:lfdo	:lfd[o]	execute command in each file in location list
:left	:le[ft]	left align lines
:leftabove	:lefta[bove]	make split window appear left or above
:legacy	:leg[acy]	make following command use legacy script syntax
:let	:let	assign a value to a variable or option
:lexpr	:lex[pr]	read locations from expr and jump to first
:lfile	:lf[ile]	read file with locations and jump to first
:lfirst	:lfir[st]	go to the specified location, default first one
:lgetbuffer	:lgetb[uffer]	get locations from buffer

:lgetexpr	:lgete[xpr]	get locations from expr
:lgetfile	:lg[etfile]	read file with locations
:lgrep	:lgr[ep]	run 'grepprg' and jump to first match
:lgrepadd	:lgrepa[dd]	like :grep, but append to current list
:lhelpgrep	:lh[elpgrep]	like ":helpgrep" but uses location list
:lhistory	:lhi[story]	list the location lists
:ll	:ll	go to specific location
:llast	:lla[st]	go to the specified location, default last one
:llist	:lli[st]	list all locations
:lmake	:lmak[e]	execute external command 'makeprg' and parse error messages
:lmap	:lm[ap]	like ":map!" but includes Lang-Arg mode
:lmapclear	:lmapc[lear]	like ":mapclear!" but includes Lang-Arg mode
:lnext	:lne[xt]	go to next location
:lnewer	:lnew[er]	go to newer location list
:lnfile	:lnf[ile]	go to first location in next file
:lnoremap	:ln[oremap]	like ":noremap!" but includes Lang-Arg mode
:loadkeymap	:loadk[eymap]	load the following keymaps until EOF
:loadview	:lo[adview]	load view for current window from a file
:lockmarks	:loc[kmarks]	following command keeps marks where they are
:lockvar	:lockv[ar]	lock variables
:lolder	:lol[der]	go to older location list
:lopen	:lop[en]	open location window
:lprevious	:lp[revious]	go to previous location
:lpfile	:lpf[ile]	go to last location in previous file
:lrewind	:lr[ewind]	go to the specified location, default first one
:ls	:ls	list all buffers
:ltag	:lt[ag]	jump to tag and add matching tags to the location list
:lunmap	:lu[nmap]	like ":unmap!" but includes Lang-Arg mode
:lua	:lua	execute Lua command
:luado	:luad[o]	execute Lua command for each line
:luafile	:luaf[ile]	execute Lua script file
:lvimgrep	:lv[imgrep]	search for pattern in files
:lvimgrepadd	:lvimgrepa[dd]	like :vimgrep, but append to current list
:lwindow	:lw[indow]	open or close location window
:move	:m[ove]	move lines
:mark	:ma[rk]	set a mark
:make	:mak[e]	execute external command 'makeprg' and parse error messages
:map	:map	show or enter a mapping
:mapclear	:mapc[lear]	clear all mappings for Normal and Visual mode
:marks	:marks	list all marks
:match	:mat[ch]	define a match to highlight
:menu	:me[nu]	enter a new menu item
:menutranslate	:menut[ranslate]	add a menu translation item
:messages	:mes[sages]	view previously displayed messages
:mkexrc	:mk[exrc]	write current mappings and settings to a file
:mksession	:mks[ession]	write session info to a file
:mkspell	:mksp[ell]	produce .spl spell file
:mkvimrc	:mkv[imrc]	write current mappings and settings to a file
:mkview	:mkvie[w]	write view of current window to a file
:mode	:mod[e]	show or change the screen mode
:mzscheme	:mz[scheme]	execute MzScheme command
:mzfile	:mzf[ile]	execute MzScheme script file
:nbclose	:nbc[lose]	close the current Netbeans session

:nbkey	:nb[key]	pass a key to Netbeans
:nbstart	:nbs[tart]	start a new Netbeans session
:next	:n[ext]	go to next file in the argument list
:new	:new	create a new empty window
:nmap	:nm[ap]	like ":map" but for Normal mode
:nmapclear	:nmapc[lear]	clear all mappings for Normal mode
:nmenu	:nme[nu]	add menu for Normal mode
:nnoremap	:nn[oremap]	like ":noremap" but for Normal mode
:nnoremenu	:nnoreme[nu]	like ":noremenu" but for Normal mode
:noautocmd	:noa[utocmd]	following commands don't trigger autocommands
:noremap	:no[remap]	enter a mapping that will not be remapped
:nohlsearch	:noh[lsearch]	suspend 'hlsearch' highlighting
:noreabbrev	:norea[bbrev]	enter an abbreviation that will not be remapped
:noremenu	:noreme[nu]	enter a menu that will not be remapped
:normal	:norm[al]	execute Normal mode commands
:noswapfile	:nos[wapfile]	following commands don't create a swap file
:number	:nu[mber]	print lines with line number
:nunmap	:nun[map]	like ":unmap" but for Normal mode
:nunmenu	:nunme[nu]	remove menu for Normal mode
:oldfiles	:ol[dfiles]	list files that have marks in the viminfo file
:open	:o[pen]	start open mode (not implemented)
:omap	:om[ap]	like ":map" but for Operator-pending mode
:omapclear	:omapc[lear]	remove all mappings for Operator-pending mode
:omenu	:ome[nu]	add menu for Operator-pending mode
:only	:on[ly]	close all windows except the current one
:onoremap	:ono[remap]	like ":noremap" but for Operator-pending mode
:onoremenu	:onoreme[nu]	like ":noremenu" but for Operator-pending mode
:options	:opt[ions]	open the options-window
:ounmap	:ou[nmap]	like ":unmap" but for Operator-pending mode
:ounmenu	:ounme[nu]	remove menu for Operator-pending mode
:ownsyntax	:ow[nsyntax]	set new local syntax highlight for this window
:packadd	:pa[ckadd]	add a plugin from 'packpath'
:packloadall	:packl[oadall]	load all packages under 'packpath'
:pbuffer	:pb[uffer]	edit buffer in the preview window
:pclose	:pc[lose]	close preview window
:pedit	:ped[it]	edit file in the preview window
:perl	:pe[rl]	execute Perl command
:print	:p[rint]	print lines
:profdel	:profd[el]	stop profiling a function or script
:profile	:prof[ile]	profiling functions and scripts
:promptfind	:pro[mptfind]	open GUI dialog for searching
:promptrepl	:promptr[epl]	open GUI dialog for search/replace
:perldo	:perld[o]	execute Perl command for each line
:pop	:po[p]	jump to older entry in tag stack
:popup	:popu[p]	popup a menu by name
:ppop	:pp[op]	":pop" in preview window
:preserve	:pre[serve]	write all text to swap file
:previous	:prev[ious]	go to previous file in argument list
:psearch	:ps[earch]	like ":ijump" but shows match in preview window
:ptag	:pt[ag]	show tag in preview window
:ptNext	:ptN[ext]	:tNext in preview window
:ptfirst	:ptf[irst]	:trewind in preview window
:ptjump	:ptj[ump]	:tjump and show tag in preview window
:ptlast	:ptl[ast]	:tlast in preview window
:ptnext	:ptn[ext]	:tnext in preview window

:ptprevious	:ptp[revious]	:tprevious in preview window
:ptrewind	:ptr[ewind]	:trewind in preview window
:ptselect	:pts[elect]	:tselect and show tag in preview window
:public	:public	prefix for a class or object member
:put	:pu[t]	insert contents of register in the text
:pwd	:pw[d]	print current directory
:py3	:py3	execute Python 3 command
:python3	:python3	same as :py3
:py3do	:py3d[o]	execute Python 3 command for each line
:py3file	:py3f[ile]	execute Python 3 script file
:python	:py[thon]	execute Python command
:pydo	:pyd[o]	execute Python command for each line
:pyfile	:pyf[ile]	execute Python script file
:pyx	:pyx	execute python_x command
:pythonx	:pythonx	same as :pyx
:pyxdo	:pyxd[o]	execute python_x command for each line
:pyxfile	:pyxf[ile]	execute python_x script file
:quit	:q[uit]	quit current window (when one window quit Vim)
:quitall	:quita[ll]	quit Vim
:qall	:qa[ll]	quit Vim
:read	:r[ead]	read file into the text
:recover	:rec[over]	recover a file from a swap file
:redo	:red[o]	redo one undone change
:redir	:redi[r]	redirect messages to a file or register
:redraw	:redr[aw]	force a redraw of the display
:redrawstatus	:redraws[tatus]	force a redraw of the status line(s)
:redrawtabline	:redrawt[abline]	force a redraw of the tabline
:redrawtabpanel	:redrawtabp[anel]	force a redraw of the tabpanel
:registers	:reg[isters]	display the contents of registers
:resize	:res[ize]	change current window height
:retab	:ret[ab]	change tab size
:return	:retu[rn]	return from a user function
:rewind	:rew[ind]	go to the first file in the argument list
:right	:ri[ght]	right align text
:rightbelow	:rightb[elow]	make split window appear right or below
:ruby	:rub[y]	execute Ruby command
:rubydo	:rubyd[o]	execute Ruby command for each line
:rubyfile	:rubyf[ile]	execute Ruby script file
:rundo	:rund[o]	read undo information from a file
:runtime	:ru[ntime]	source vim scripts in 'runtimepath'
:rviminfo	:rv[iminfo]	read from viminfo file
:substitute	:s[ubstitute]	find and replace text
:sNext	:sN[ext]	split window and go to previous file in argument list
:sandbox	:san[dbox]	execute a command in the sandbox
:sargument	:sa[rgument]	split window and go to specific file in argument list
:sall	:sal[l]	open a window for each file in argument list
:saveas	:sav[eas]	save file under another name.
:sbuffer	:sb[uffer]	split window and go to specific file in the buffer list
:sbNext	:sbN[ext]	split window and go to previous file in the buffer list
:sball	:sba[ll]	open a window for each file in the buffer list
:sbfirst	:sbf[irst]	split window and go to first file in the buffer list

:sblast	:sbl[ast]	split window and go to last file in buffer list
:sbmodified	:sbm[odified]	split window and go to modified file in the buffer list
:sbnext	:sbn[ext]	split window and go to next file in the buffer list
:sbprevious	:sbp[revious]	split window and go to previous file in the buffer list
:sbrewind	:sbr[ewind]	split window and go to first file in the buffer list
:scriptnames	:scr[iptnames]	list names of all sourced Vim scripts
:scriptencoding	:scripte[ncoding]	encoding used in sourced Vim script
:scriptversion	:scriptv[ersion]	version of Vim script used
:scscope	:scs[cope]	split window and execute cscope command
:set	:se[t]	show or set options
:setfiletype	:setf[iletype]	set 'filetype', unless it was set already
:setglobal	:setg[lobal]	show global values of options
:setlocal	:setl[ocal]	show or set options locally
:sfind	:sf[ind]	split current window and edit file in 'path'
:sfirst	:sfir[st]	split window and go to first file in the argument list
:shell	:sh[ell]	escape to a shell
:simalt	:sim[alt]	Win32 GUI: simulate Windows ALT key
:sign	:sig[n]	manipulate signs
:silent	:sil[ent]	run a command silently
:sleep	:sl[eep]	do nothing for a few seconds
:sleep!	:sl[eep]!	do nothing for a few seconds, without the cursor visible
:slast	:sla[st]	split window and go to last file in the argument list
:smagic	:sm[agic]	:substitute with 'magic'
:smap	:smap	like ":map" but for Select mode
:smapclear	:smapc[lear]	remove all mappings for Select mode
:smenu	:sme[nu]	add menu for Select mode
:smile	:smi[le]	make the user happy
:snext	:sn[ext]	split window and go to next file in the argument list
:snomagic	:sno[magic]	:substitute with 'nomagic'
:snoremap	:snor[emap]	like ":noremap" but for Select mode
:snoremenu	:snoreme[nu]	like ":noremenu" but for Select mode
:sort	:sor[t]	sort lines
:source	:so[urce]	read Vim or Ex commands from a file
:spelldump	:spelld[ump]	split window and fill with all correct words
:spellgood	:spe[llgood]	add good word for spelling
:spellinfo	:spelli[nfo]	show info about loaded spell files
:spellrare	:spellra[re]	add rare word for spelling
:spellrepall	:spellr[epall]	replace all bad words like last z=
:spellundo	:spellu[ndo]	remove good or bad word
:spellwrong	:spellw[rong]	add spelling mistake
:split	:sp[lit]	split current window
:sprevious	:spr[evious]	split window and go to previous file in the argument list
:srewind	:sre[wind]	split window and go to first file in the argument list
:stop	:st[op]	suspend the editor or escape to a shell
:stag	:sta[g]	split window and jump to a tag

:startinsert	:star[tinsert]	start Insert mode
:startgreplace	:startg[replace]	start Virtual Replace mode
:startreplace	:startr[eplace]	start Replace mode
:static	:static	prefix for a class member or function
:stopinsert	:stopi[nsert]	stop Insert mode
:stjump	:stj[ump]	do ":tjump" and split window
:stselect	:sts[elect]	do ":tselect" and split window
:sunhide	:sun[hide]	same as ":unhide"
:sunmap	:sunm[ap]	like ":unmap" but for Select mode
:sunmenu	:sunme[nu]	remove menu for Select mode
:suspend	:sus[pend]	same as ":stop"
:sview	:sv[iew]	split window and edit file read-only
:swapname	:sw[apname]	show the name of the current swap file
:syntax	:sy[ntax]	syntax highlighting
:syntime	:synti[me]	measure syntax highlighting speed
:syncbind	:sync[bind]	sync scroll binding
:t	:t	same as ":copy"
:tNext	:tN[ext]	jump to previous matching tag
:tabNext	:tabN[ext]	go to previous tab page
:tabclose	:tabc[lose]	close current tab page
:tabdo	:tabd[o]	execute command in each tab page
:tabedit	:tabe[dit]	edit a file in a new tab page
:tabfind	:tabf[ind]	find file in 'path', edit it in a new tab page
:tabfirst	:tabfir[st]	go to first tab page
:tablast	:tabl[ast]	go to last tab page
:tabmove	:tabm[ove]	move tab page to other position
:tabnew	:tabnew	edit a file in a new tab page
:tabnext	:tabn[ext]	go to next tab page
:tabonly	:tabo[nly]	close all tab pages except the current one
:tabprevious	:tabp[revious]	go to previous tab page
:tabrewind	:tabr[ewind]	go to first tab page
:tabs	:tabs	list the tab pages and what they contain
:tab	:tab	create new tab when opening new window
:tag	:ta[g]	jump to tag
:tags	:tags	show the contents of the tag stack
:tcd	:tc[d]	change directory for tab page
:tchdir	:tch[dir]	change directory for tab page
:tcl	:tcl	execute Tcl command
:tcldo	:tcld[o]	execute Tcl command for each line
:tclfile	:tclf[ile]	execute Tcl script file
:tearoff	:te[aroff]	tear-off a menu
:terminal	:ter[minal]	open a terminal window
:tfirst	:tf[irst]	jump to first matching tag
:throw	:th[row]	throw an exception
:this	:this	prefix for an object member during initialization (e.g. on new())
:tjump	:tj[ump]	like ":tselect", but jump directly when there is only one match
:tlast	:tl[ast]	jump to last matching tag
:tlmenu	:tlm[enu]	add menu for Terminal-Job mode
:tlnoremenu	:tln[oremenu]	like ":noremenu" but for Terminal-Job mode
:tlunmenu	:tlu[nmenu]	remove menu for Terminal-Job mode
:tmapclear	:tmapc[lear]	remove all mappings for Terminal-Job mode
:tmap	:tma[p]	like ":map" but for Terminal-Job mode
:tmenu	:tm[enu]	define menu tooltip
:tnext	:tn[ext]	jump to next matching tag

:tnoremap	:tno[remap]	like ":noremap" but for Terminal-Job mode
:topleft	:to[pleft]	make split window appear at top or far left
:tprevious	:tp[revious]	jump to previous matching tag
:trewind	:tr[ewind]	jump to first matching tag
:try	:try	execute commands, abort on error or exception
:tselect	:ts[elect]	list matching tags and select one
:tunmap	:tunma[p]	like ":unmap" but for Terminal-Job mode
:tunmenu	:tu[nmenu]	remove menu tooltip
:type	:type	create a type alias
:undo	:u[ndo]	undo last change(s)
:undojoin	:undoj[oin]	join next change with previous undo block
:undolist	:undol[ist]	list leafs of the undo tree
:unabbreviate	:una[bbreviate]	remove abbreviation
:unhide	:unh[ide]	open a window for each loaded file in the buffer list
:uniq	:uni[q]	uniq lines
:unlet	:unl[et]	delete variable
:unlockvar	:unlo[ckvar]	unlock variables
:unmap	:unm[ap]	remove mapping
:unmenu	:unme[nu]	remove menu
:unsilent	:uns[ilent]	run a command not silently
:update	:up[date]	write buffer if modified
:vglobal	:v[global]	execute commands for not matching lines
:var	:var	variable declaration in Vim9
:version	:ve[rsion]	print version number and other info
:verbose	:verb[ose]	execute command with 'verbose' set
:vertical	:vert[ical]	make following command split vertically
:vim9cmd	:vim9[cmd]	make following command use Vim9 script syntax
:vim9script	:vim9s[cript]	indicates Vim9 script file
:vimgrep	:vim[grep]	search for pattern in files
:vimgrepadd	:vimgrepa[dd]	like :vimgrep, but append to current list
:visual	:vi[sual]	same as ":edit", but turns off "Ex" mode
:viusage	:viu[sage]	overview of Normal mode commands
:view	:vie[w]	edit a file read-only
:vmap	:vm[ap]	like ":map" but for Visual+Select mode
:vmapclear	:vmapc[lear]	remove all mappings for Visual+Select mode
:vmenu	:vme[nu]	add menu for Visual+Select mode
:vnew	:vne[w]	create a new empty window, vertically split
:vnoremap	:vn[oremap]	like ":noremap" but for Visual+Select mode
:vnoremenu	:vnoreme[nu]	like ":noremenu" but for Visual+Select mode
:vsplit	:vs[plit]	split current window vertically
:vunmap	:vu[nmap]	like ":unmap" but for Visual+Select mode
:vunmenu	:vunme[nu]	remove menu for Visual+Select mode
:windo	:wind[o]	execute command in each window
:write	:w[rite]	write to a file
:wNext	:wN[ext]	write to a file and go to previous file in argument list
:wall	:wa[ll]	write all (changed) buffers
:while	:wh[ile]	execute loop for as long as condition met
:winsize	:wi[nsize]	get or set window size (obsolete)
:wincmd	:winc[md]	execute a Window (CTRL-W) command
:winpos	:winp[os]	get or set window position
:wlrestore	:wl[restore]	restore the Wayland compositor connection
:wnext	:wn[ext]	write to a file and go to next file in argument list
:wprevious	:wp[revious]	write to a file and go to previous file in

		argument list
:wq	:wq	write to a file and quit window or Vim
:wqall	:wqa[ll]	write all changed buffers and quit Vim
:wundo	:wu[ndo]	write undo information to a file
:wviminfo	:wv[iminfo]	write to viminfo file
:xit	:x[it]	write if buffer changed and close window
:xall	:xa[ll]	same as ":wqall"
:xmapclear	:xmapc[lear]	remove all mappings for Visual mode
:xmap	:xm[ap]	like ":map" but for Visual mode
:xmenu	:xme[nu]	add menu for Visual mode
:xrestore	:xr[estore]	restores the X server connection
:xnoremap	:xn[oremap]	like ":noremap" but for Visual mode
:xnoremenu	:xnoreme[nu]	like ":noremenu" but for Visual mode
:xunmap	:xu[nmap]	like ":unmap" but for Visual mode
:xunmenu	:xunme[nu]	remove menu for Visual mode
:yank	:y[ank]	yank lines into a register
:z	:z	print some lines
:~	:~	repeat last ":substitute"

vim:tw=78:ts=8:noet:ft=help:norl:

This site is maintained by Carlo Teubner (*(my first name) at cteubner dot net*).