

Vi Quick Reference

Entering/leaving vi

% vi <i>name</i>	edit <i>name</i> at top
% vi + <i>n</i> <i>name</i>	... at line <i>n</i>
% vi + <i>name</i>	... at end
% vi - r	list saved files
% vi - r <i>name</i>	recover file <i>name</i>
% vi <i>name1 name2</i> ...	edit first; rest via :n
% view <i>name</i>	read only mode

Vi states

Command	Normal and initial state. Others return here. ESC (escape) cancels partial command.
Insert	Entered by a i A I o O c C s S R . Enter your arbitrary text then terminates with ESC character.

The display

Last line	Error messages, echoing input to : / ? and ! , feedback about i/o and large changes.
@ lines	On screen only, not in file.
~ lines	Lines past end of file.
^ <i>x</i>	Control characters, ^? is delete.

File manipulation

:w	write out changes
:wq	write and quit <i>vi</i>
ZZ	exit from vi, saving changes
:w <i>name</i>	write file <i>name</i>
:w! <i>name</i>	overwrite file <i>name</i>
:q	quit
:q!	quit, discard ALL changes
:e <i>name</i>	edit file <i>name</i>
:e!	reedit, discard changes
:e #	edit alternate file
:n	edit next file in arglist
:sh	run shell, type <i>exit</i> to return
!:cmd	run <i>cmd</i> , then return
^G	shows current filename and line

Insert and replace

a	append after cursor
A	append at end of line
i	insert before
I	insert before first non-space
o	open line below
O	open above
rx	replace single char with <i>x</i>
R	replace multiple characters

Corrections during inserting

^H	erase last character
^W	erases last word
^D	backtab over <i>autoindent</i>
^V^X	puts control character ^ <i>X</i> in text
ESC	ends insertion, back to command

Undo, redo, retrieve

u	undo last change
U	restore current line
.	repeat last change
"d <i>p</i>	retrieve <i>d</i> 'th last delete

Counts before vi commands

Typing a number before a vi command will tell vi to "repeat" the command that many times. More specifically:

line/column number	z G
scroll amount	^D ^U
replicate insert	a i A I
repeat effect	most rest

Operators (double to affect lines)

These operators act like a prefix, you type the operator ("d" for example) then type a cursor movement command to specify what the operator will affect (example: "dw" deletes a word)

d	delete
c	change
y	yank lines to buffer
<	left shift
>	right shift
!	filter through UNIX command

Simple commands

dw	delete a word
de	... leaving punctuation
dd	delete a line
3dd	... 3 lines
i abc ESC	insert text <i>abc</i>
cw new ESC	change current word to <i>new</i>
xp	transpose characters

Character positioning

^	first non blank
0	beginning of line
\$	end of line
h or →	forward
l or ←	backwards
^H	same as ←
space	same as →
fx	find character <i>x</i> forward
Fx	... backwards
tx	up to character <i>x</i> forward
Tx	... backwards
;	repeat last f F t or T
,	opposite direction of ;
 	to specified column
10 	to 10th column
%	find matching ({) or }

Vi Quick Reference

Words, sentences, paragraphs

w	word forward
W	... ignore punctuation
b	word backwards
B	... ignore punctuation
e	end of word
E	... ignore punctuation
)	sentence forward
(sentence backwards
}	paragraph forward
{	paragraph backward

Line positioning

H	home window line
L	last window line
M	middle window line
+	next line, at first non-white
-	previous line, at first non-white
↓ or j	next line, same column
↑ or k	previous line, same column

Marking and returning

``	previous place in file
''	... at first non-blank in line
mx	mark position with letter <i>x</i>
`x	to mark <i>x</i>
ˆx	... at first non-blank in line

Scanning pattern formation

/pattern	next line matching "pattern"
?pattern	... find backwards
^	beginning of line
\$	end of line
.	any character
*	any number of preceding
.*	matches anything or nothing
<	beginning of word
>	end of word
[abc]	a single char (<i>a</i> or <i>b</i> or <i>c</i>)
[^abc]	... any char except <i>a</i> or <i>b</i> or <i>c</i>
[x-y]	... between <i>x</i> and <i>y</i>

Positioning within file

^F	forward one screenfull
^B	backward one screenfull
^D	scroll down half screen
^U	scroll up half screen
^E	scroll window down 1 line
^Y	scroll window up 1 line
G	goto line (end default)
10G	goto 10th line in file
/pattern	next line matching <i>pattern</i>
?pattern	prev line matching <i>pattern</i>
n	repeat last / or ?
N	reverse last / or ?
%	find matching () { or }
]]	next section/function
[[previous section/function

Adjusting the screen

^L	clear and redraw messy screen
^R	retype, eliminate @ lines
zCR	redraw, current at window top
z-	... at bottom
z.	... at center
zn.	use <i>n</i> line window

Miscellaneous operations

C	change rest of line
D	delete rest of line
Y	yank lines
s	substitute chars
J	join lines
x	delete characters
X	... before cursor

Yank and put

p	put back lines
P	put before
"xy	yank (copy) to buffer <i>x</i>
"xd	delete into buffer <i>x</i>
"xp	put from buffer <i>x</i>

NOTE: the yank and delete commands above are followed by a cursor movement command to specify what will be yanked or deleted. (see Operators section)

Initializing VI options

set option	enable option
set nooption	disable option
set option=value	give value <i>val</i>
set all	show all options
set	show changed options

Useful options

autoindent	ai	supply indent
autowrite	aw	write before changing files
ignorecase	ic	in scanning
lisp		() { } are s-exp's
list		print ^I for tab, \$ at end
magic		. [* special in patterns
number	nu	number lines
paragraphs	para	macro names which start ...
redraw		simulate smart terminal
scroll		command mode lines
sections	sect	macro names ...
shiftwidth	sw	for < >, and input ^D
showmatch	sm	to) and } as typed
window		visual mode lines
wrapscan	ws	around end of buffer?
wrapmargin	wm	automatic line splitting