

VI QUICK REFERENCE

Character Positioning

h	move cursor left one char
j	move cursor up one char
k	move cursor down one char
l	move cursor right one char
^	first non white-space character
O	beginning of line
\$	end of line
fx	find next "x" on current line
Fx	find previous "x" on current line
tx	move to character prior to next "x" on current line
Tx	move to character following previous "x" on current line
;	repeat last f F t or T
,	repeat inverse of last f F t or T
n	move to column n
%	find matching ({) or }

Words, Sentences, Paragraphs

w	forward a word
5w	forward 5 words
W	forward a blank-delimited word
b	back a word
B	back a blank-delimited word
e	end of word
E	end of a blank-delimited word
)	to next sentence
}	to next paragraph
(back a sentence
{	back a paragraph

Line Positioning

H	top line on screen
L	last line on screen
M	middle line on screen
+ or <CR>	next line, at first non-white
-	previous line, at first non-white

Yank and Put

Put inserts the text most recently deleted or yanked; however, if a buffer is named (using the ASCII lower-case letters a - z), the text in that buffer is put instead.

yy	yank one line to buffer
3yy	yank 3 lines
3yl	yank 3 characters
p	put back text after cursor
P	put back text before cursor
"xp	put from buffer x
"x7yy	yank 7 lines to buffer x
"xdd	delete line into buffer x

Insert and Replace

a	append after cursor
A	append at end of line
i	insert before cursor
I	insert before first non-blank
o	open line below
O	open above
r	replace single char
R	replace characters

Delete

x	delete a character
dw	delete a word
dd	delete a line
ndd	delete n number of lines
D	delete rest of line
dO	delete from cursor to begin of line
dtx	delete to "x" in a line
d/pat	delete to "pattern"
"g5dd	delete 5 lines and save them in buffer g

Undo, Redo, Retrieve

u	undo last change
U	restore current line
.	repeat last change
"np	retrieve n'th last delete

Positioning Within File

/pat	next occurrence matching "pat"
?pat	previous occurrence matching "pat"
n	repeat last / or ? command
N	reverse last / or ? command
/pat/+n	nth line after "pat"
?pat?-n	nth line before "pat"
G	go to beginning of the last line in the file
nG	go to the beginning of the specified line, where n is a line number
^F	forward screen
^B	backward screen
^D	scroll down half screen
^U	scroll up half screen
]]	next section/function
[[previous section/function
(beginning of sentence
)	end of sentence
{	beginning of paragraph
}	end of paragraph

File Manipulations

:wq	write and quit
:w	write back changes
:w!	forced write, if permission originally not valid
:w file	write filename
:w! file	overwrite file name
:5,30w file	writes lines 5-30 to filename
:5,30w>>file	appends lines 5-30 to filename
:q	quit
:q!	quit, discard changes
:e file	edit filename
:e!	reedit, discard changes
:e + file	edit, starting at end
:e +n	edit starting at line n
:e #	edit alternate file
:e! #	edit alternate file, discard changes
:5r file	read filename into current vi session starting at line 5
:r !cmd	reads output of command into current vi session
:sh	run shell, then return
:!cmd	run cmd, then return
:n	edit next file in arglist
:n args	specify new arglist
^G	show current file and line
:ta tag	position cursor to tag

VI QUICK REFERENCE

Adjusting the Screen

`^L` clear and redraw window
`z<CR>` redraw screen with current line at top of window
`z-` redraw screen with current line at bottom of window
`z.` redraw screen with current line at center of window
`/pat/z-` move line with "pat" to bottom of window
`zn.` use n-line window
`^E` scroll window down 1 line
`^Y` scroll window up 1 line

Marking and Returning

```` move cursor to previous context  
`"` move cursor to first non-white space in line  
`mx` mark current position with the ASCII lower-case letter x  
``x` move cursor to mark x  
`'x` move cursor to first non-white space in line marked by x

## Corrections During Insert

`^H` erase last character (backspace)  
`^W` erase last word  
`\` quotes your erase and kill characters  
`<ESC>` ends insertion, back to command mode  
`DEL` interrupt, terminates insert mode  
`^D` backtab one character; reset left margin autoindent  
`^D` caret (^) followed by control-d (^D); backtab to beginning of line; do not reset left margin of autoindent  
`0^D` backtab to beginning of line; reset left margin of autoindent  
`^V` quote non-printable character

## Miscellaneous Operations

`c` changes rest of ??? depending other arguments; uses same syntax as d (delete)  
`C` change rest of line (c\$)  
`s` substitute chars (cl)  
`S` substitute lines (cc)  
`J` join lines  
`<ESC>` end insert or incomplete cmd  
`!` filter through command

## EX COMMANDS: Global Replace

`:g/old/s//new/opt` replace "old" with "new" - see options

### options:

`g` globally change every occurrence in file  
`s` change every occurrence in line  
`c` confirm each change  
`p` print each change  
`:1,$!sort` sort lines 1-end alphabetically  
`:1,2co4` copy lines 1-2 to line 4  
`:1,8m17` move lines 1-8 to line 17

## ex Metacharacters for Pattern Match

`/` turn off special meaning of next char  
`^` matches beginning of line  
`$` matches end of line  
`.` matches any single character  
`*` matches any character any number of times  
`[string]` matches any one of the enclosed characters. A dash (-) specifies a range ([a-d] = [abcd])  
`[^string]` matches any string not enclosed  
`&` stands for text in a search (/giant/s//\*/s/ replaces giant with giants)  
`\(pat\)` escaped parentheses used to define a regular expression or pattern for substitution  
`/n` The number n is used to refer to previous patterns defined by parentheses  
`<ctl>v` used to escape an <ESC> or <CR> in the replacement part of the command

## Examples

Deletes all blank lines

`:g/^$/d`

find all lines (g) that have a beginning (^) and an end (\$) with no characters between and delete (d) them

Double space all lines

`:g/$s//<CTL>v<CR>/g`

find end of each line (\$) and substitute (s//) a <CR>

Remove lead blanks on each line

`:g/^\ *(.*)/s//\1/g`

find the first blank of each line (^) and any other blanks (\*), followed by any single character string \(.\*) and retain the character string (\1). This demonstrates escaped parentheses to define a regular expression. An easier way to accomplish this is: `:g/^\ */s//g`

Swap the strings on either side of a hyphen

`:g/^(.*) - \(.*)/s//\2 - \1/gc`

find the pattern (abc - xyz) and reverse the pattern (xyz - abc)