VI REFERENCE

CONTENTS

ENTERING VI

%**vi** [−t tag] [−r] [−x] [+position] [−**w**n] [files]

Options:

vi	screen editor or **view** (read only) or **ex** (line editor) or **edit** (friendly **ex**)
−t	equivalent to initial **tag** cmd; edits file containing tag and positions editor at function (see UNIX **ctags**)
−r	retrieves last saved version of named file after system or editor crash (default is list of all saved files.)
−x	edit encrypted file
+position	if position is omitted, positions cursor at end of file useful cmds include /pat, ?pat, line numbers
−wn	sets default window size to n (useful for dial-ups)
files	indicates files to be edited

VI MODES

command	anything typed is taken as an editing command, <ESC> cancels partial command
insert	entered by any of **aiAIoOcCsSR**, terminate with <ESC>

DEFINITIONS

x or X or Y	represent a single character
^x	control char . hold <CTRL> key and press x. can be upper case or lower case
^	caret. usually a shifted 6
status line	bottom screen line. shows error messages. echos input after :, /, ?, ! feedback about i/o and large changes
@lines	on screen only. not in file (deleted lines)
~lines	lines past end of file
text	one or more characters
string	zero or more text or pattern matching characters
operator	a command to **vi** that takes an object (**c d y p P**)
object	objects are strings such as words. sentences. characters. cursor to end of line. cursor to end of file... Objects specify what the operator is operating on (eg.. in **dw**. **d** is operator. **w** is object) (see OBJECTS. page 3. for list of objects)
n	number of objects (eg . in **d8w** – delete 8 words) except before **[[ ]] : ? ^d ^u x G | H L M 0 'x 'x**. n always defaults to 1 (n**d**n causes n's to multiply: 3**d**4**w** deletes 12 words
sentence	ends at . ! ? followed by <CR> or two spaces
paragraph	begins after blank line. or paragraph macros
section	defined as ending with a section macro
pattern	text and pattern matching characters (abbrev pat)
<CR>	return key on your terminal
<ESC>	escape key on your terminal
[item]	square brackets indicate optional items. don't type brackets

INSERTING TEXT

(n preceding insert adds n copies of inserted text)

a	add after cursor
A	add to end of line
i	insert before cursor
I	insert before first nonblank on the line
o	open a line below line cursor is on. ready for insert
O	open a line above line cursor is on. ready for insert

INSERT MODE COMMANDS

<ESC>	terminate insert mode
<backspace>	erase character
^h or <erase>	erase char. (your erase. set with UNIX **stty** cmd)
^w	erase last word. defined by **b**
<kill>	erase current line of insert (set with **stty**)
<interrupt>	terminate insert abnormally (set with **stty**)
^d	backtab over autoindent
^^d	kill autoindent. restore it on next line
0^d	kills all autoindent
^t	move shift width to right (if **autoindent** set)
^vc	insert non-printing character
^@	if 1st char. of insertion. repeat last insertion
^i	insert tab
<CR>	start a new line
	escape a following **^h**. or your erase char

OPERATORS

format: operator n object

c	change specified object
d	delete specified object
y	yank specified object
p	put buffer contents after cursor
P	put buffer contents before cursor

OBJECTS
CURSOR MOVEMENT

format: operator n object

- n specifies # of objects
- if no operator, cursor moves to specified object

w	one word forward (defined by punctuation or blank)
b	one word backward
e	end of current word
W B E	same as **w b e**, ignoring punctuation
) (	next sentence, previous sentence
} {	next paragraph, previous paragraph
]] [[	next section, previous section
fx	forward to n'th x in line
Fx	backward to n'th previous x in line
tx	forward to character before n'th x
Tx	backward to character after n'th x
j	down line, same column
k	up line, same column
h	character to left of cursor
l	character to right of cursor
#	current line

same format: n acts differently

H	top line on screen (n lines below top)
M	middle line on screen (n does nothing)
L	last line on screen (n lines up from last)
G	end of file (line n in file)
0	beginning of current line (n does nothing)
$	end of current line (end of n-1'th line down)
'a	line containing mark (n does nothing)
`a	marked position (n does nothing)
/pat/	forward to pattern (n doesn't work)
?pat?	backward to pattern (n doesn't work)
+ or <CR>	current and next line (current to n'th line)
–	current and line above (current to n'th line up)
^	cursor to beginning of line (n does nothing)
|	cursor to 1st column (cursor to column n)

LIMITS

510 or 1024	characters/line
256	characters/global command list
128	characters/file name
128	characters in previous insert or delete in **vi**
100	characters in shell escape cmds
30	characters in a tag name
250,000	characters/file
32	macros defined by **map**
512	characters in macros

DELETING TEXT

d	delete operator, format: **d** n object
Other deletes that don't fit the format **d** n object	
ndd	delete n lines
nx (or **dn**<space>)	delete n chars, cursor and chars to right
nX	delete n characters to left of cursor
D	delete rest of line
"bdd	delete a line into buffer b (any delete can be deleted into a buffer)
"Bdd	append line to buffer b

UNDO, REDO, RETRIEVE

u	undo last change
U	restore current line
.	repeat last change
"np	retrieve n'th last delete (9 numbered buffers), usable only while in current file
"bp	retrieve contents of buffer b (26 lettered buffers), can use if going to another file with **:e** or **:n**
"1pu.u.u.	scan through last few deletes to find a particular one

CHANGING OR REPLACING TEXT

.	repeat last change
n.	repeat last change n times
rc	replace character under cursor with c
nrc	replace n characters with c
Rtext	write over old text, begin at cursor, <ESC> to end
nRtext	like **R**, insert n copies of text, <ESC> to end
stext	substitute text for char. under cursor, <ESC> ends
nstext	substitute text for next n characters, <ESC> ends
Stext	substitute entire line, <ESC> ends
nStext	substitute text for next n lines, <ESC> ends
:s/X/Y/opt	substitute string Y for X in line
opt formed from:	
g	change every occurence in line
c	confirm each change
p	print changed lines
&	repeat the last **:s** request
:g/X/cmd	run cmd on all the lines that match string X
:g/X/**s**//Y/opt	global change string X to Y (same options as in **:s** above)
:g!/X/cmd	run cmd on all lines not matching string X
:g/pattern	move to last line in file containing pattern
:g/X/**s**/Y/Z/opt	on lines matching X, change Y to Z
:v/X/cmd	run command on all lines that do not match string X
n,m	format :n,m preceding **s**, **g** or **v** command; limits range of pattern matching from lines n through m
c	change operator, format: **c** n object
Special cases that don't fit the format: **c** n object	
ncc	change n entire lines, <ESC> to end
nC	change from cursor to end of n'th line, <ESC> ends
xp, **dwwP**, **ddp**	reverse order of two letters, words, lines

INDENTING

(indent size defined by **shiftwidth** option)

n<<.n>>	shift n lines left, right

MOVING WITHIN A LINE

^ or __ move to first non white space
n| or n→ or n<space> move n characters to right
n^h or n← or n<backspace> move n characters to left
• see OBJECTS, page 3 for more

MOVING TO OTHER LINES

:n go to line n
G go to end of file (go to line n in file)
(the following commands may be preceded by n)
<CR> or + to start of next line
– to start of previous line
j (or ↓ or ^n or LF) next line, same column
k (or ↑ or ^p) previous line, same column
H, M, L go to first, middle, last line on screen
• see OBJECTS, page 3 for more

COPYING AND MOVING TEXT

Operators: format: operator n object
y yanks object into buffer
p puts buffer contents after the cursor
P puts buffer contents before the cursor
Special cases that don't fit the format: operator n object
nY or nyy yank copy of n lines into an unnamed buffer
"bY yank copy into buffer b (b is any letter from a–z)
"bp puts the contents of buffer b after cursor
"BY append copy to buffer b
:m,n**co#** or :m,n**t#** copy lines m – n after line #
($ indicates end of file)
:m,n**m#** move lines m – n after line #
:n,m**w** file write lines n–m to file (may overwrite)
:n,m**w**>> file append lines n–m to end of file

ADJUSTING THE SCREEN

^l or ^r clear and redraw the screen
Redraw screen: format [/pat/] [n] z [m] [<CR>.–]
(items in square brackets are optional)
Options
/pat/ or n indicate cursor line after screen redrawn
<CR>.–.. indicate cursor line is at top(<CR>), bottom(–) or middle(.) of screen when screen is redrawn
m use m line window
^g or :f display status line on bottom line of screen
n^f move n full screens forward (up)
n^d move n lines forward (up) (½ window default)
n^b move n full screens backward (down)
n^u move n lines backward (down) (½ window default)
n^e scroll window down n lines
n^y scroll window up n lines

SAVING YOUR PLACE

mx mark current cursor position with character x
'x move to blank character in line of mark x
'' move to first non-blank character in line of previous change
`x move to position marked with x
`` return to previous place after motion of cursor by command (eg., /,**G**,?)
operator`x operation takes place from marked place to cursor
operator'x operation takes place over complete lines

SETTING OPTIONS

- Options set with **set** last only while you are in the editor
- To make them permanent, put them in your **.exrc** file

.exrc file in home dir listing default for options on entry to **vi**

set EXINIT = 'set options' sets default on entry to **vi** (C shell)

EXINIT = 'set options' sets default on entry to **vi** (Bourne shell)

:**set all** prints all option settings

:**set** prints settings set with **EXINIT**, & current changes

:**set** option–name or :**set** option–name=value
sets option (eg., **set sw = 8**)

:**set no**option–name unsets option (eg., **noai**)

:**set** option–name?
prints option–name setting on status line

OPTIONS

	Default	What Option Does if On
autoindent(**ai**)	**noai**	supply indentation automatically
autoprint(**ap**)	**ap**	print line after **d c J m : s t u**
autowrite(**aw**)	**noaw**	automatic write before **:n**, **!**, **e#**, ^^, **:rew**, ^], **:tag**
beautify(**bf**)	**nobf**	discard control chars from input except tab, newline, formfeed
directory(**dir**)	**dir = /tmp**	specify directory of buffer file
edcompatible	**noed**	**g**, **c** are toggles in substitutes
errorbells(**eb**)	**noeb**	precede error messages with bell
hardtabs(**ht**)	**ht = 8**	set terminal hardware tabs
ignorecase(**ic**)	**noic**	ignore case in searches
lisp	**nolisp**	modify **)** **{** **{** **[[** **]]** to be compatible with lisp
list	**nolist**	show tabs (^I), end of line (**$**)
magic	**magic**	if **nomagic**, only ^ and **$** are metacharacters
mesg	**mesg**	allow **write** from other users
number	**nonu**	prefix lines with line number
open	**open**	if **noopen**, **open** and **vi** are not permitted
optimize(**opt**)	**opt**	speed output by eliminating automatic <CR>
paragraphs	**para = LI IPLPPPQPbpP**	macro names that start paragraphs for **{** and **}** operators
prompt	**prompt**	prompt for command mode input with ':'
readonly(**ro**)	**noro**	change file status to readonly
redraw(**re**)	**nore**	simulate smart terminal on dumb
remap	**remap**	accept macros within macros
report	**report = 10**	specify threshold size of changes reported on status line
scroll	**scroll = 11**	# lines scrolled with ^**d**, **z**
sections(**sect**)	**sect = NHSHHHU**	macro names which start new section for **[[** and **]]** operators
shell	**sh = /bin/sh**	pathname of new shell for **!** and **:sh** (default from **$SHELL** if present)
shiftwidth(**sw**)	**sw = 8**	# spaces for **<**,**>** input ^**d**, ^**t**
showmatch(**sm**)	**nosm**	show matching **(** or **{** as **)** or **}** is typed
slowopen	**slow**	postpone display updates during inserts
tabstop(**ts**)	**ts = 8**	^**i** tab stops set for text input
taglength(**tl**)	**tl = 0**	tags not significant beyond this many chars. (**0** means all chars)

OPTIONS, continued

tags	**tags =** **/usr/lib/tags**	path for files checked for tags, (current dir. included in default)
term	**$TERM**	name of terminal being used, set by UNIX **$TERM**
terse	**noterse**	produce shorter error diagnostics
timeout(**to**)	**noto**	no 1 sec time limit for **map**s
warn	**warn**	warn if "No write since last change" before **!** command
window	**window** = n	number of lines in a text window, n is speed dependent
wrapmargin (**wm**)	**wm** = **0**	cause all lines to be broken at a space at least n spaces from right edge of screen
wrapscan(**ws**)	**ws**	searches wrap around end of file
writeany(**wa**)	**nowa**	inhibit normal checks before write commands

MOVEMENT: SEARCHING TEXT

/pattern<CR>	search forward for pattern
?pattern<CR>	search backward for pattern
//	last regular expression
/pat/;//;//	find 3rd occurrence of pattern
n	repeat previous search in direction of initial search
N	repeat previous search in reverse direction
/<CR>	repeat previous search in forward direction
?<CR>	repeat previous search in backward direction
/pat/ ± n	go to n'th line before/after pattern (forward search)
?pat? ± n	go to n'th line before/after pattern (backward search)
fx	forward to x (one letter – not pattern) within line only
Fx	back to x within line
tx	move forward to character before x in line
Tx	move backward to character after x in line
;	repeat last **f F T t**
,	reverse direction of last **f F T t**
%	find matching ( ) or { } or **[ ]**

PATTERNS WITHIN SEARCHES

(strings in searches may be regular expressions)

^	beginning of line (if first char. in search)
$	end of line (if last character in search)
\	escape next character (\ / **$. ^ [' & • | ~** must be escaped in searches)
\\	escape \

the following patterns work only if the **magic** option set

.	any character (eg., x.y matches x = y, x – y, xry)
*	finds **0** or more occurrences of previous character
[A – Z]	matches any character, uppercase A – Z
[abc]	matches a, b or c
[^abc]	any char but a, b or c (^ must be char. after [)
\(...\)	mark part of pattern to use in replacement
\<	beginning of a word
\>	end of a word

Patterns in Replacement String:

&	entire string found
~	text of previous replacement pattern
\n	n'th marked pattern
\u	make next character upper case
\l	make next character lower case
\U	characters are upper case till **\e** or **\E**
\L	characters are lower case till **\e** or **\E**
\E or **\e**	turn off **\U** or **\L**

EX COMMANDS

• all : commands are **ex** commands. use without : in **ex**

:**cd** [dir]	change working directory of editor. **$HOME** default
:**l**	display tab(^I). backspace(^H). backslash(\). bell(^G). formfeed(^L). newline(**$**) in current line on status line
:**nu**	display current line preceded by line number
:**pre**	preserve buffer (useful if write fails)
:**rec**	retrieve buffer after system crash. :**pre**. disconnect
:**sh**	start subshell. shell set from **$SHELL** unless changed by :**set**. **$EXINIT** or .**exrc** (^d returns)
:**so** file	read and execute cmds from file
:**stop** (or ^z)	stop editor. return to top level shell (UNIX **fg** resumes edit. Berkeley only)
:**ta** [tag]	edit file containing function tag at tag (see UNIX **ctags**)
^]	like :**ta**. cursor is on 1st char. of tag name
:**ve**	display version # and date of last update
:**vi**	go back to **vi** when in **ex**
:**=**	displays number of lines in file

DEFINING NEW REQUESTS

macros put macro body in buffer (using yank or delete operators)

@x	invoke macro
@@	repeat last macro

:**map** lhs rhs defines lhs (any character or escape sequence) to be equivalent to a combination of requests (rhs). ^v is used to escape <CR>,<ESC>,newline, space and tab in definition. Use **KVgqv**••= or function keys (unused commands in **vi**)

:**map!** lhs rhs	map applies to insert and command mode
:**map**	display created maps on status line
:**map!**	display insert mode maps on status line
:**unmap** lhs	end specific map
:**unmap!** ^v lhs	end insert mode map

:**ab** ex example abbreviate: every time ex inserted, insert shows example

:**ab**	display abbreviations
:**una** ex	unabbreviate ex

FILE MANIPULATION

- all : commands are **ex** commands, use without : in **ex**
- if no file specified, use current file (don't type brackets)

Writing Information to a File: w

- all **w** cmds can be preceded by line numbers **:1,4w** file

:w [file]	save file, remain in current file
:w >> [file]	append to file, remain in current file
:w! [file]	write to file, override normal checking
:w#	write to file specified in last **ex** command

Editing Other Files: e, n

:e [file]	edit file
:e +pos [file]	edit file, starting at pos
:e! [file]	edit file, ignore changes to current file, if no file, re-edit current file
:e# or **^^**	edit last file edited, cursor on last edited line
:n	edit next file in list
:n[pos] files	specify new list of files, start next file at position pos
:n!	ignore changes to current file, edit next one
:args	show list of files; **[ ]** delimits current file
:rew	rewind list of files, edit first file in list
:rew!	like **:rew**, discard changes to current file

Reading Info Into a File: r

:r [file]	read in and insert file, current file default
:r !cmd	read in and insert output of UNIX cmd
:nr [file]	read in and insert file after line n

Executing UNIX Commands

:!cmd	execute single UNIX command, then return to **vi**
%	name of current file
#	name of last file edited
!	use text of last command
:!!	repeat last UNIX command
:w !cmd	run file through UNIX cmd, output not inserted
:posw !cmd	run specified lines through UNIX command
pos	n,m, +, −, n+, n−, /pat/±n ?pat?±n
n!obj cmd	use n objects as input to UNIX cmd, execute command, output replaces object (eg., **2!)sort** sorts lines from current line to end of second sentence) (see OBJECTS, page 3)
!!cmd	replace current line with output of cmd

Naming Files

:f or **^g**	show current line and file number
:f file	rename current file

Leaving VI

:wq [file]	save file in file, exit from editor
:wq! [file]	save file, no checking done, exit from editor
:q	exit from editor
:q!	exit from editor, discarding changes
:x [file]	save changes if made, exit from editor
ZZ	save changes if made, exit from editor
Q	escape to line editor **ex** (to return, type **vi**)