Quick Reference for vi

This quick reference lists commands you can use in the `vi` editor on Hewlett-Packard's UNIX™ System, HP-UX.

Notations

- Commands beginning with `:` (colon) must end with [Return] These commands represent escape commands to the `ex` editor. You can reference the `ex` tutorial for more details.
- *`file`* is the name of file
- *`cursor_cmd`* is a cursor movement command (e.g., `G` `j` `w` `b`)
- *`char`* is a single character
- `str` is a character string (can contain pattern matching characters)
- CTRL-x means you press CTRL, hold it down, and press the x key.
- *`n,m`* can be two line numbers (e.g., `4,50`), a line marker (e.g. `.,$`), or a search expression (e.g., `/string1/,/string2/`).
- *`(a-z)`* means you choose a letter from `a` through `z`

Modes

Command Mode	When you are not inserting or changing text, you can move the cursor and run commands (e.g., searching, deleting, saving). Pay attention to the case of the commands; check the [Caps] lock key if `vi` behaves strangely.
Insert Mode	When you insert or change more than one character of text, you cannot use command mode commands. To leave the insert mode, press Esc.

Start a vi Session

`vi file`	Edit *file*
`vi -r file`	Edit last saved version of *file* after system or editor crash
`vi + n file`	Edit *file* and place cursor at line *n*
`vi + file`	Edit *file* and place cursor on last line
`vi file1 … filen`	Edit *file1* through *filen*; After saving changes in *file1*, enter `:n` for next file, `:p` for previous file
`vi +/str file`	Edit *file* and place cursor at line containing *str*

Save Text and Exit `vi`

`ZZ` or `:wq` or `:x`	Save file and exit `vi`
`:w file`	Save *file* but do not exit; omitting file saves current file
`:w! file`	Save *file* overriding normal checking
`:n,mw file`	Write lines *n* through *m* to *file*
`:n,mw>>file`	Append lines *n* through *m* to end of *file*
`:q`	Leave `vi`, saving changes before last write (you may be prompted to save first)
`:q!`	Leave `vi` without saving any changes since last write
`Q`	Escape `vi` into ex editor with same file; : vi returns
`:e!`	Re-edit current file, disregarding changes since last write
`:e file`	Edit new file

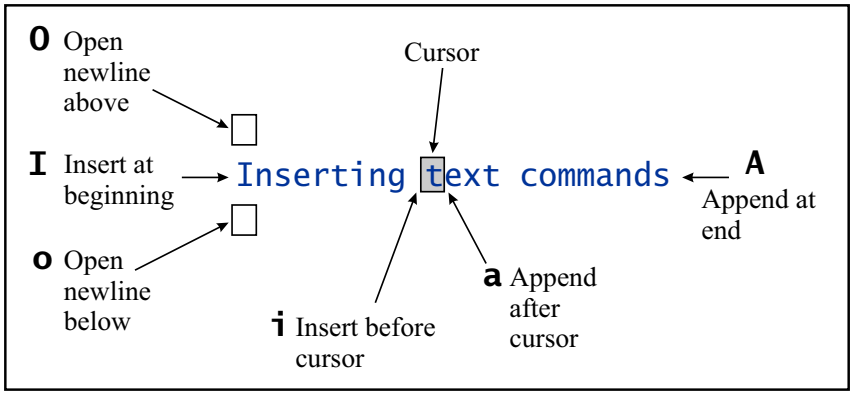
Status Commands

`:.=`	Print current line number
`:=`	Print number of lines in file
CTRL-g	Show file name, current line number, total lines in file, and percent of file location
`:l` (letter "l")	Display tab (`^I`) backslash (`\`) backspace (`^H`) newline (`$`) bell (`^G`) formfeed (`^L`) of current line in status line

Inserting Text

To leave the insert mode, press Esc.

`a`	Append after cursor
`A`	Append after end of current line
`I`	Insert before cursor
`I`	Insert before beginning of current line
`o`	Open new line below current line and insert
`O`	Open new line above current line and insert
CTRL-v *char*	While inserting, ignore special meaning of *char* (e.g., for inserting characters like Esc and control characters)
`:r` *file*	Read *file*, and insert after current line
`:r !`*program*	Insert output of *program* after current line
`:`*n*`r` *file*	Read *file*, and insert after line *n*

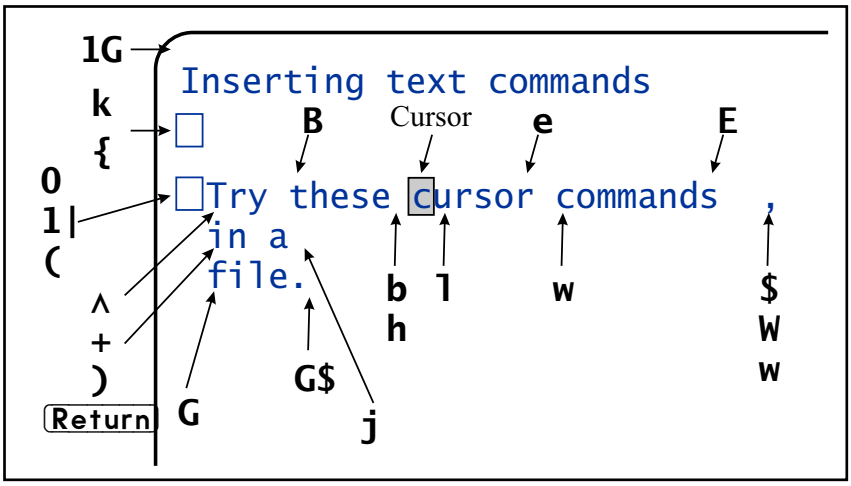


Undoing and Repeating Commands

`u`	Undo last command
`U`	Restore current line to original state
`"`*n*`p`	Retrieve last *n*th delete (last 9 deletes are in buffer)
`"1pu.u.`	Scroll through the delete buffer until you retrieve desired delete (repeat `u.`)
`n`	Repeat last `/` or `?` search command
`N`	Repeat, in reverse direction, last `/` or `?` search command
`;` (semi-colon)	Repeat last `f F t` or `T` search command
`,` (comma)	Repeat, in reverse direction, last `f F t` or `T` search command
`.` (period)	Repeat last text change command

Moving the Cursor

k or CTRL-p	Up
j or CTRL-j	Down
h or CTRL-h	Left
l or Space	Right
w or W	Start of next word; W ignores punctuation
b or B	Start of previous word; B ignores punctuation
e or E	End of next word; E ignores punctuation
0 (zero) or |	First column in current line
n|	Column *n* in current line
^ (caret)	First non-blank character in current line
$	Last character in current line
+ or Return	First character in next line
-	First non-blank character in previous line
1G	First line in file
G	Last line in file
G$	Last character in file
*n*G	Line *n* in file
(	Back to beginning of sentence
)	Forward to beginning of next sentence
{	Back to beginning of paragraph
}	Forward to beginning of next paragraph



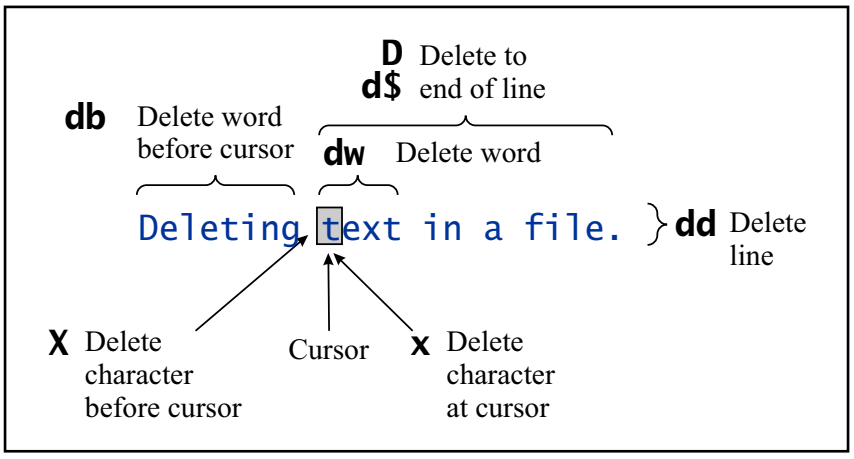
Section Positioning

Mark sections by placing { in first column.

[[	Back to beginning of section
]]	Forward to beginning of next section

Deleting Text

CTRL-h or Backspace	While inserting, delete previous character
CTRL-w	While inserting, delete previous word
CTRL-x	While inserting, delete to start of inserted text
*n*x	Delete *n* characters beginning with current; omitting *n* deletes current character
*n*X	Delete previous *n* characters; omitting *n* deletes previous character
xp	Switch character at cursor with following character
*n*dw	Delete next *n* words beginning with current; omitting *n* deletes current word
*n*db	Delete previous *n* words; omitting *n* deletes previous word
*n*dd	Delete *n* lines beginning with current; omitting *n* deletes current line
:*n*,*m*d	Delete lines *n* through *m*
D or d$	Delete from cursor to end of current line
d*cursor_cmd*	Delete text to *cursor_cmd* (e.g., dG deletes from current line to end of file)



Placing Marks in the Text

m(*a-z*)	Mark current position with a letter *a* through *z* (e.g ., m*a*)
'(*a-z*)	Move cursor to position (*a-z*) (e.g., 'a)
‘‘ or ’’ (single quotes or grave accents)	Move cursor to location before last / ? or G

Pattern Matching

Pattern matching characters help find strings with similar characteristics.

`:set magic`	Allow pattern matching with special characters (default)
`:set nomagic`	Allow only `^` and `$` as special characters
`^` (caret)	Match beginning of line
`$`	Match end of line
`.` (period)	Match any single character
`\<`	Match beginning of word
`\>`	Match end of word
`[`*`str`*`]`	Match any single character in *`str`*
`[~`*`str`*`]`	Match any character not in *`str`*
`[`*`a`*`-`*`n`*`]`	Match any character between *`a`* and *`n`*
`*`	Match zero or more occurrences of previous character in expression
`\`	Escape meaning of next character (e.g., `\$` lets you search for `$`)
`\\`	Escape the `\` character

Indenting Text

CTRL-i or Tab	While inserting, insert one shift width
`:set ai`	Turn on auto-indentation
`:set sw=`*`n`*	Set shift width to *`n`* characters
`n``<<` or *`n`*`>>`	Shift *`n`* lines left or right (respectively) by one shift width; omitting *`n`* shifts one line
`<` or `>`	Use with cursor command to shift multiple lines left or right

HEWLETT PACKARD
HP Part Number
98597-90000
Microfiche No. 98597-99000
Printed in U.S.A. 9/87
98597-90630
For Internal Use Only

Searching

`%`	Search to beginning of balancing `()` `[]` or `{}`
`f`*`char`*	Search forward in current line to *`char`*
`F`*`char`*	Search backward in current line to *`char`*
`t`*`char`*	Search forward in current line to character before *`char`*
`T`*`char`*	Search backward in current line to character after *`char`*
`/str` Return	Find `str`
`?str` Return	Search in reverse for *`str`*
`:set ic`	Ignore case when searching
`:set noic`	Pay attention to case when searching (default)

Global Search and Replace

`:n.ma/str1/str2/opt`	Search from *`n`* to *`m`* for *`str1`*. Replace *`str1`* with *`str2`*, using *`opt`*. *`opt`* can be `g` for global change, `c` to confirm change (press (`y` to acknowledge, Return to suppress), and `p` to print changed lines.
`&`	Repeat last `:a` command
`:g/str/cmd`	Run *`cmd`* on all lines that contain *`str`*
`:g/str1/s/str2/str3/`	Find line containing *`str1`*, replace *`str2`* with *`str3`*
`:v/str/cmd`	Execute *`cmd`* on all lines that do not match *`str`*

Copying and Placing Text

`nyy` or `nY`	Yank *`n`* lines (place in buffer); omitting *`n`* yanks current line
`y`*`cursor_cmd`*	Yank from cursor to *`cursor_cmd`* (e.g., `yG` yanks current line to last line in file)
`"(a-z)nyy` or `"(a-z)`*`n`*`dd`	Copy or delete *`n`* lines into named buffer *`a`* through *`z`*; omit *`n`* for current line
`p` (lower-case)	Put yanked text after cursor (print buffer); also prints last deleted text
`P` (upper case)	Put yanked text before cursor; also prints last deleted text
`"(a-z)p` or `"(a-z)P`	Put lines from named buffer *`a`* through *`z`* after or before current line

Changing Text

Preceding these commands with *n* (a number) repeats the command *n* times.

r*char*	Replace current character with *char*
R*text* Esc	Replace current character(s) with *text*
s*text* Esc	Substitute *text* for current character
S or cc *text* Esc	Substitute *text* for entire line
cw*text* Esc	Change current word to *text*
C*text* Esc	Change rest of current line to *text*
c*cursor_cmd text* Esc	Change to *text* from current position to *cursor_cmd*

Joining Lines

J	Join next line to end of current line
*n*J	Join next *n* lines

Cursor Placement and Adjusting the Screen

H	Move cursor to top line of screen
nH	Move cursor to line *n* from top of screen
M	Move cursor to middle of screen
L	Move cursor to bottom line of screen
nL	Move cursor to line *n* from bottom of screen
CTRL-e	Move screen up one line
CTRL-y	Move screen down one line
CTRL-u	Move screen up ½ page
CTRL-d	Move screen down ½ page
CTRL-b	Move screen up one page
CTRL-f	Move screen down one page
CTRL-i or letter "l"	Redraw screen
z Return	Make current line top line on screen
*n*z Return	Make line *n* top line on screen
z.	Make current line middle line
nz.	Make line *n* middle line on screen
z-	Make current line bottom line
nz-	Make line *n* bottom line on screen

Shell Escape Commands

`:! cmd`	Execute shell command *cmd*; you can add these special characters to indicate: `%` name of the current file `#` name of last tile edited
`:!!`	Execute last shell command
`:r! cmd`	Read and insert output from *cmd*
`:f file`	Rename current file to *file*
`:w !cmd`	Send currently edited file to *cmd* as standard input and execute *cmd*
`:cd dir`	Change the current working directory to *dir* (`$HOME` is default)
`:sh`	Start a sub-shell (CTRL-d returns to editor)
`:so file`	Read and execute commands in *file* (*file* is a shell script)

Shell Filters

`!cursor_cmd cmd`	Send text from current position to *cursor_cmd* to shell command *cmd*. Replace original text in file with output from *cmd*
`!}sort` Return	Example: Sort from current position to end of paragraph and replace text with sorted text

Macros and Abbreviations

`:map key cmd_seq`	Define *key* to run *cmd_seq* when pressed
`:map`	Display all created macros on status line
`:unmap key`	Remove macro definition for *key*
`:ab str string`	When *str* is inserted, replace with *string*
`:ab`	Display all abbreviations
`:una str`	Unabbreviate *str*

Map allows you to define strings of `vi` commands. Place in `.exrc` to run each time you enter `vi`. For long macros, set the `notimeout` option. If you embed control characters (e.g., keys like Esc in the macro, you need to precede them with CTRL-v.
If you need to include quotes (`"`), precede them with `\` (backslash).
Unused keys in `vi` are: `K V g q v * =` and the function keys.
Example:
`:map v /I` CTRL-v Esc `dwiYou` CTRL-v Esc Esc
When `v` is pressed, search for "`I`" (`/I` Esc), delete word (`dw`) and insert "`You`" (`iYou` Esc). CTRL-v allows Esc to be inserted.

Setting Options

Options shown here are default. To change them, either set them (`:set option`) or unset them (`:set nooption`). To run options each time you enter `vi`, place in `.exrc` file in home directory and omit preceding colons (`:`).

`:set all`	Print all options
`:set nooption`	Turn off option
`:set noai`	Set automatic indentation
`:set ap`	Print line after `d c J m : s t u` command
`:set bf`	Discard control characters from input
`:set eb`	Precede error messages with bell
`:set noic`	Ignore case when searching
`:set dir=tmp`	Set directory of buffer file
`:set lisp`	Modify brackets for Lisp compatibility
`:set magic`	Pattern match with special characters
`:set mesg`	Allow other users to send messages
`:set nolist`	Show tabs (^I) and end of line($)
`:set nonu`	Prefix lines with line number
`:set opt`	Speed output: eliminate automatic Return
`:set prompt`	Prompt for command mode input with :
`:set nore`	Simulate smart terminal on dumb terminal
`:set remap`	Allow macros within macros
`:set report`	Indicate largest size of changes reported on status line
`:set ro`	Change file to read only
`:set scroll=n`	Set *n* lines for CTRL-d and z
`:set sh=`*`shell_path`*	Set shell escape (default is `/bin/sh`)
`:set showmode`	Indicate input or replace mode
`:set sw=`*`8`*	Set the shift width to *8* characters
`:set term`	Print terminal type
`:set terse`	Shorten error messages with terse
`:set notimeout`	Eliminate one second time limit for macros
`:set tl=`*`0`*	Set significance of tags beyond this many characters (*0* means all)
`:set te=`*`8`*	Set tab stops for text input to *8* characters
`:set nowa`	Inhibit normal checks before write commands

`:set warn`	Warn “No write since last change”
`:set window=`*`n`*	Set number of lines in a text window to *`n`*
`:set wm=`*`n`*	Set automatic word wrap around *`n`* spaces from right margin (e.g., `:set wm=8`)

Ranges

`:n,m`	lines `n` to `m`
`:.`	current line
`:$`	last line
`:'c`	marker `c`
`:%`	all lines
`:g/pattern/`	all matching lines

Examples:

`:7,12d` deletes lines lines 7 to 12

`:.,$s/pattern/string/g` replace `pattern` matches with `string` from current line (`.`) to end of file (`$`).