vi Reference Card

This reference card lists commands which can be used in the `vi` editor on Hewlett-Packard's UNIX™ system, HP-UX.

Definitions

- All commands beginning with : (colon) must end with the Return key.
- *file* is the name of a file.
- *cursor_cmd* is a cursor movement command.
- *str* is a character string.
- CTRL-x means you press the control key and the following key, x, simultaneously.

Entering vi

`vi` *file*	Edit *file*
`vi -r` *file*	Edit last saved version of *file* after system or editor crash

Saving Text and Leaving vi

`ZZ` or `:wq` or `:x`	Save file and leave vi
`:w` *file*	Save file to *file* but do not leave vi (omitting *file* saves current file)
`:`*n*`,`*m*`w` *file*	Write lines *n* through *m* to *file*
`:`*n*`,`*m*`w>>`*file*	Append lines *n* through *m* to *file*
`:q`	Leave vi
`:q!`	Leave vi without saving any changes
`Q`	Escape `vi` into `ex`

Inserting Text

`a`	After cursor
`A`	After end of current line
`i`	Before cursor
`I`	Before beginning of current line
`o`	New line below current line
`O`	New line above the current line
CTRL-v *char*	While inserting text, ignore special meaning of *char* (allows inserting characters like ESC)
`:r` *file*	Read *file*, and insert after current line

Moving The Cursor

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

Section Positioning

Mark section by placing `{` in the first column.

`[[`	Beginning of previous section
`]]`	End of next section

Cursor Placement and Adjusting the Screen

`H`	Top line of screen
n`H`	Line *n* from top of screen
`M`	Middle of screen
`L`	Bottom line of screen
n`L`	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-l	Redraw screen
`z` Return	Make current line top line on screen
n`z` Return	Make line *n* top line in screen
`z.`	Make current line middle line
n`z.`	Make line *n* middle line in screen
`z-`	Make current line bottom line
n`z-`	Make line *n* bottom line in screen

Placing Marks in the Text

`m`*(a-z)*	Mark current position with *(a-z)*
`'`*(a-z)*	Move cursor to position named *(a-z)*
`''`	Move cursor to where you were before the last `/` `?` or `G` command

Searching

`%`	Beginning of balancing (), [], or {}
`f`*character*	Forward in current line to *character*
`F`*character*	Backward in current line to *character*
`t`*character*	Forward in current line to character before *character*
`T`*character*	Backward in current line to character after *character*
`/`*str* Return	Find *str*
`?`*str* Return	Find in reverse, *str*
`:se ic`	Ignore case when searching
`:se noic`	Pay attention to case when searching (default)

HEWLETT PACKARD

HP Part Numbers
98597-90000
Microfiche No. 98597-99000
Printed in U.S.A. 12/86

98597-90615
For Internal Use Only

Global Search and Replace

`:a,n`s`/fstr/rstr/opt`	Search from line *a* to line *n* for *fstr* and replace with *rstr*, where *opt* is: `g` for global change (change all occurrences), `c` for confirm changes (press Return to acknowledge), and `p` is for print the changed lines. *a* and *n* can be line numbers or: "." for current line, "$" for last line.
`:g/str/cmd`	Run *cmd* on all lines that match *str*
`:g/fstr/s/fstr2/rstr/`	On lines matching *fstr*, change *fstr2* to *rstr*

Using Pattern Matching in Searching

Pattern matching characters can be used to find strings with similar characteristics.

`:se magic`	Allow pattern matching (default)
`:se nomagic`	Allow only ^ and $ characters
`^`	Match beginning of line
`$`	Match end of line
`.`	Match any character
`\<`	Match beginning of word
`\>`	Match end of word
`[`*str*`]`	Match any single character in *str*
`[^`*str*`]`	Matches any character not in *str*
`[`*x-y*`]`	Match any character between *x* and *y*
`*`	Match any number of characters

Deleting Text

CTRL-h or Back space	While inserting, delete previous character
CTRL-w	While inserting, delete previous word
CTRL-x	While inserting, delete to start of inserted text
`x`	Delete current character
n`x`	Delete *n* characters
`X`	Delete previous character
n`X`	Delete previous *n* characters
`dw`	Delete next word
`db`	Delete previous word
`dd`	Delete current line
n`dd`	Delete next *n* lines, including current
`:`*n*`,`*m*`d`	Delete lines *n* through *m*
`D`	Delete rest of current line
`d`*cursor_cmd*	Delete text to *cursor_cmd* (cursor movement command)

Copying Text

`yy` or `Y`	*Yank* current line; see *Placing Text*
n`Y` or *n*`yy`	*Yank* next *n* lines
`y`*cursor_cmd*	*Yank* from current to *cursor_cmd*

Placing Text

`p` (*lower-case*)	Place *yanked* text after cursor
`P`	Place *yanked* text before cursor

Changing Text

`r`*char*	Replace next character with *char*
`R`*text* ESC	Replace next character(s) with *text*
`s`*text* ESC	Substitute *text* for next character
`S` or `cc` *text* ESC	Substitute *text* for entire line
`cw`*text* ESC	Change next word to *text*
`C`*text* ESC	Change rest of current line to *text*
`c`*cursor_cmd text* ESC	Change text from current position to *cursor_cmd* position, to *text*
n change_cmd	Repeat *change_cmd* *n* times

Joining Text

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

Indenting Text

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

Restoring and Repeating

`u`	Undo last command
`U`	Restore current line to previous state
`n`	Repeat last multi-character search command
`N`	Repeat, in reverse direction, last multi-character search command
`;`	Repeat last single-character search command
`,` (*comma*)	Repeat, in reverse direction, last single-character search command
`.` (*period*)	Repeat last command which changed the file

Shell Commands

`:!` *cmd*	Execute shell command
`:!!`	Execute last shell command
`:r!` *cmd*	Read and insert output from *cmd*
`:f` *file*	Rename current file

Status Commands

`:=`	Print current line number
CTRL-g	Show file name and current line location

Filters

`!`*cursor_cmd cmd*	Text can be used as input into a shell command. The output replaces the text from the current position to *cursor_cmd*. The text is input to the shell command *cmd*.

For example, sort from the current position to the end of the paragraph:

`!}sort` Return

Setting Options

Place permanent options in `.exrc` file.

`:set all`	Print all options
`:set no`*option*	Turn off *option*
`:set bf`	Discard control characters from input
`:set eb`	Precede error messages with bell
`:set lisp`	Modify brackets to be compatible with Lisp
`:set mesg`	Allow other users to send messages
`:set ro`	Change file to read only
`:set remap`	Allow macros within macros
`:set scroll=`*num*	Set *num* of lines for CTRL-d and `z`
`:set sh=`*shell_path*	Set shell escape (default is `/bin/sh`)
`:set warn`	Warn "No write since last change"
`:set wm=`*n*	Allow automatic wraparound *n* spaces from right margin (inserts Return without splitting words; example: `:set wm=8`)
`:set notimeout`	No 1 second time limit for macros

Defining Macros

Macros function during current `vi` session unless placed in the `.exrc` file (a file created to define options for all `vi` sessions; create this file if it does not exist, and place `:set` *option* and `:map` commands in it). For long macros, set the `notimeout` option.

`:map` *key cmd_seq*	Define *key* to perform command sequence *cmd_seq* when pressed.

Example:

`:map v /I` CTRL-v ESC `dwiYou` CTRL-v ESC ESC

When `v` is pressed, `vi` searches for "`I`", then deletes the word (`dw`) and replaces it with "`You`". The CTRL-v allows ESC to be typed into the string to end the insert mode, and the last ESC terminates the macro.