The Vi Editor:

A far-from complete guide

Jonathan V. Phillips

THE FREQUENTLY-ASKED QUESTIONS (FAQ) COLUMN

------------ What is vi? ------------

Vi is a fully-featured text editor that features multiple-file editing during one session, interaction with a UNIX shell during operation, macros, and numerous other advanced functions. Vi was developed by Bill Joy as part of the BSD UNIX project, and has become a standard on any UNIX-like system.

----- Why should I use vi? -----

Despite the bad reputation vi has acquired as arcane or cryptic, it deserves a close examination in choosing your text editor. Don't simply try using this editor once to make a decision—play around with it. You'll find the commands are much more intuitive than the complicated Control-key sequences of emacs, and its features leave Pico in the dust. Give vi a try!

------- How do I start vi? -------

Vi can be started with a variety of command-line options. A few of the most common ones are listed below. From your home directory (or any directory where you have "write" rights), simply type:

`vi` *`file`*	To start editing file *file* with vi.
`vi +`*`n file`*	To start editing *file* at line *n*.
`vi +/`*`xxx file`*	Edits *file* at first occurrence of *xxx*.

---- How do I pronounce vi? ----

By the way, this question is not actually frequently asked, but it needs to be addressed. The correct pronunciation is "vee-eye."

NOTE: In this guide, letters or words typed in italics are meant to be replaced by appropriate letters or words depending on what you wish to do.

-- ENTERING TEXT (INPUT MODE)--

The vi editor starts up in "command mode." This means that you cannot at first type text directly into your file. To do so, you must invoke "input mode" by typing the appropriate letter. Some commands associated with this process are listed below:

`a`	append typed text after the cursor
`A`	append text at end of line
`i`	insert text before cursor
`I`	insert text at beginning of line
`R`	"overwrite" text at cursor
`o`	insert a line below current line and begin text there
`O`	insert a line above current line and begin text there

Once you have entered input mode by one of the above commands, you may type your text. You may use `BACKSPACE` if you make an error, but you cannot move around on the page otherwise. To do so, you must reenter command mode by pressing the `ESCAPE` key.

-- COMMAND MODE OPTIONS--

NOTE: Typing a number before a command repeats that command that number of times.

• Moving Around

Vi provides a multitude of movement commands. Here are some of the more useful ones (remember you must be in command mode to use these).

`h,j,k,l`	move left,down,up,and right (on some systems the cursor keys will function as well)
`SPACEBAR`	move right one character
`w` or `W`	move forward by one word
`b` or `B`	move backward by one word
`e` or `E`	move to the end of current word

`0`	first character of current line
`$`	last character of current line
`^`	first nonblank character of current line
`-`	first nonblank character of previous line
`+`	first nonblank character of next line

`H`	top line of screen
`M`	middle line of screen
`L`	last line of screen
n`H`	*n* lines after top line
n`L`	*n* lines before last line

`CTRL-F`	scroll forward one screen
`CTRL-B`	scroll backward one screen
`CTRL-D`	scroll down one-half screen
`CTRL-U`	scroll up one-half screen
`CTRL-E`	scroll to show one more line at bottom of screen
`CTRL-Y`	scroll to show one more line at top of screen
`z RETURN`	put current line at top of screen
`z.`	put current line at middle of screen
`z-`	put current line at bottom of screen
`CTRL-L`	redraw screen

• Editing Commands

In general (although not entirely), editing commands in vi have the following format:

`[n] operator object`

where the *operator* is one of the following:

`c`	begin a change
`d`	begin a deletion
`y`	begin a yank (or copy)

The *object* can represent a character, word, sentence, paragraph, or section. In general, the object is the key you would press in command mode to refer to the desired text. For example, w is a word, ^ is the first character of the current line, and) represents a sentence. If the operation is to be performed on the entire current line, simply repeat the letter twice (`cc`, `dd`, or `yy`).

The *n* represents the number of times the following operation is to be performed. As usual, if no *n* is specified, 1 is assumed. Here are some examples of the normal use of these commands (and some exceptions):

n`cw`	change *n* words
`cc`	change current line
`C`	change text from current position up to end of line
n`dd`	delete *n* lines
`d^`	delete back to beginning of line
`D`	delete remainder of line
`d/`*`pat`*	delete up to first occurrence of the pattern *pat*
`dn`	repeat pattern delete
`dG`	delete to end of file
`yw`	copy (yank) a word
n`yy`	copy *n* lines
`y)`	copy to beginning of next sentence
`p`	place text previously copied
n`x`	delete *n* characters starting at cursor
n`X`	delete previous *n* characters
`.`	repeat last change
`~`	reverse case

• Large Movement Commands

If you're working with big files, you may need to search for specific text, go directly to line numbers far away, or mark positions.

Here are some ways to search:

`/`*`text`*	search forward for text
`/`	repeat forward search
`?`*`text`*	search backward for text
`?`	repeat backward search
`n`	repeat previous search
`N`	repeat previous search in the opposite direction

A few commands relating to line numbers:

`CTRL-G`	display current line number
n`G`	go to line *n*
`:`*`n`*	go to line *n*
`G`	go to last line in file

And some marking commands:

`m`*`x`*	mark current position with character *x*
`` ` ``*`x`*	move cursor to mark *x*
`'`*`x`*	move to start of line containing *x*
` `` `	move to previous mark (or location prior to last search)
`''`	like ` `` `, but go to start of line

• Saving and Exiting

Alas, all good things must come to and end, and sometime you will wish to leave the vi editor. Here are some ways to save your work and to exit vi:

`ZZ`	Quit vi, and write (save) file only if changes were made
`:q!`	Quit vi and DO NOT WRITE
`:wq`	Write and quit current file
`:w`	Write file
`:w `*`file`*	Save copy to *file*
`:q`	Quit file
`:e!`	return to version of current file at time of last write

• Multiple File Access

As your skill increases, you might want to handle multiple files in one session of vi. Here are some commands which will get you started with multiple files:

`:e `*`file`*	edit another *file*, current file becomes alternate
:e!	return to last saved version of file
:e + *file*	begin editing at end of *file*
:e +*n file*	open *file* at line *n*
:e #	open to previous position in

	alternate file
:n	edit next file
:args	display multiple files to be edited
:rew	rewind list of multiple files to top

• UNIX Interaction

Inserting files and text output from UNIX commands is a vital feature of any editor (even Pico can sort of do this). Here are some related commands:

`:r` *`file`*	read in contents of *file* after cursor
`:r !`*`command`*	read in output from *command* after current line
`:`*`nr`* `!`*`command`*	like above, but put output after line *n* (0 for top of file)
`:!`*`command`*	run *command*, then return to vi
`CTRL-Z`	suspend vi, return with `fg`

• Macros

You might find it helpful to abbreviate long sequences of commands you perform frequently. Here are some ways to do so:

`:ab` *`in out`*	use *in* as an abbreviation for *out*
`:unab` *`in`*	remove abbreviation for *in*
`:ab`	list abbreviations
`:map` *`c sequence`*	map character c as a *sequence* of commands
`:unmap` *`c`*	disable map for character *c*
`:map`	list characters that are mapped

Note: The following characters are not used by command mode and may be mapped by the user:

Letters:	`g K q V v`
Control Keys:	`^A ^K ^O ^T ^W ^X`
Symbols:	`_ * \`

• Miscellaneous

Here are some commands which really didn't fit anywhere else.

`J`	Join two lines
`:j!`	Join two lines, preserving blank spaces
<<	shift current line left by one shift width (default=8 spaces)
>>	shift current line right by one shift width (default=8 spaces)
`:%s/`*`text`*`/new-`*`text`*`/`	Replaces text once
`:%s/`*`text`*`/new-`*`text`*`/g`	Replaces text globally

• Credits

Primary sources (and great UNIX references in general):

UNIX in a Nutshell: System V Edition. By Daniel Gilly and the staff of O'Reilly & Associates, Inc. Sebastopol, CA: O'Reilly & Associates, Inc., 1992.

UNIX for the Impatient. By Paul W. Abrahams and Bruce R. Larson. New York: Addison-Wesley Publishing Company.