

# Quick Reference to Using vi

Stephen W. Turner

## A Very Brief History

The **vi** editor (correctly pronounced “vee eye”) exists in many forms, as different variations of it (with different names) are available for several operating systems and architectures. It originated on an early version of Unix as a “visual” improvement over **ed** and **ex**, two “relatively primitive” (pronounced “horrible”) line-oriented editors that are still available today. The early versions of vi were well-liked enough that it was eventually included in distributions of System V Unix. Whatever the version, vi has been improved to the point where it is a very powerful text editor. The information contained in this document covers only a fraction of the functionality of vi.

The “vi” editor that you commonly use on machines running Linux is called vim, which is a “vi emulator”, meaning that it’s not legally allowed to be called vi, but it acts completely like vi (with some improvements). Other versions of vi that you may encounter include Elvis, Vile, Nvi, and WinVi, to name a few.

## Using vi

To begin, you can think of vi as existing in one of two modes: text-entry mode (I call it insert mode) and command mode. The command mode is exactly what the label implies: you can move about within the file without actually modifying any of the data, issuing various sorts of commands. If you are in insert mode, it means that you are modifying data and usually much more limited with respect to where you can move (for example, in most versions of vi, if you are in insert mode, then you cannot use the arrow keys to move up or down).

Editing things in vi is accomplished mainly through numerous single-character commands, many of which can be preceded by a number to imply “do the following n times”. Thus, going from command mode to insert mode usually involves typing a single-character command. Going from insert mode back to command mode often (but not always) involves hitting the escape key. There are also some meta commands that are issued by typing a colon (“:”) followed by some character sequence. The meta commands are used for functions like naming the output file, saving the file, quitting, etc. What follows is a series of tables containing commonly used commands and notions in vi.

### Meta Information:

<b>word</b>	Any alphanumeric text delimited by whitespace and/or various punctuation characters. Examples: fred, a123 x_y_z
<b>Word</b>	Any alphanumeric text delimited only by whitespace (may include punctuation characters). Example: swturner@umflint.edu
<esc>	Hit the escape key. This <i>always</i> takes you out of insert mode.
^X	control-x - that is, hit control and x at the same time. (Uppercase is not necessary)

### Meta operations (you must be in command mode to execute these):

:w (filename)	Write to whatever file name you specify
:wq	Write and quit. This only works if vi has a filename in its buffer
:q	Quit. This only works if you have already saved changes or made no changes
:q!	Quit without saving changes
ZZ	Shortcut for quit or write and quit This only works if vi has a filename in its buffer
.	Perform exactly the same as the last destructive operation you did.
u	Undo whatever destructive operation you just did. Some versions maintain a stack of these operations.
/(pattern)	search forward for whatever pattern. Regular expressions work.
/	Search forward for the last pattern you looked for.
?(pattern)	search backward for a pattern.
?	Search backward for the last pattern you looked for.
n	find the next occurrence of the last pattern you looked for (works forward or backward, depending on your last search)

**Operations that place you into Insert Mode** (may overwrite or delete or insert):

o	Insert a new line below the current cursor position
O	Insert a new line above the current cursor position
a	Insert after the cursor position
A	Append to the end of the line
i	Insert before the current cursor position
I	Insert at the beginning of the line
ce	replace starting from current cursor to the end of the next <b>word</b>
cE	replace starting from current cursor to the end of the next <b>Word</b>
s	Replace the current character and place into insert mode.
5s	Replace this and the next four characters, place into insert mode.
S	Replace the entire current line, place into insert mode.
4S	Replace this and the next 3 current lines, place into insert mode.

**Operations that insert or destroy things but don't place you into insert mode:**

x	Delete the character under the cursor.
X	Delete the character just before the cursor.
5x	Delete 5 characters, including the one under the cursor.
5X	Delete the 5 characters prior to the cursor.
dw	Delete a <b>word</b>
dW	Delete a <b>Word</b>
dd	delete the current line
5dd	Delete this and the next four lines (5 total)
p	Copy the buffer to the nearest position after the cursor (if it's a line, then it goes on the line below. If it's less than a line, then it goes on this line, after the cursor)
P	Copy the buffer to the nearest position before the cursor (same reasoning as with 'p', but "before")
yy	Yank (copy) the current line into the buffer.
5yy	Yank (copy) the current line and the next four into the buffer

**Movement operations:**

j	Move down one line. Same as the down arrow.
k	Move up one line. Same as the up arrow.
h	Move left one character. Same as the left arrow.
l	Move right one character. Same as the right arrow.
w	Move right one <b>word</b>
W	Move right one <b>Word</b>
b	Move back one <b>word</b>
B	Move back one <b>Word</b>
0 (zero)	Move the cursor to the beginning of the line
\$ (dollar-sign)	Move the cursor to the end of the line
G	go to the end of the file
1G	go to line 1
35G	go to line 35 (if you have at least 35 lines)
^D (ctrl-D)	Go down about a half screen.
^U (ctrl-U)	Go up about a half screen.
^F (ctrl-F)	Go down about a full screen.
^B (ctrl-B)	Go up about a full screen.