

UNIX: vi Editor

General Introduction

The vi editor (short for visual editor) is a screen editor which is available on almost all Unix systems. Once you have learned vi, you will find that it is a fast and powerful editor. vi has no menus but instead uses combinations of keystrokes in order to accomplish commands. If you are just beginning to learn Unix, you might find the pico editor easier to use (most command options are displayed at the bottom of the screen). Please refer to the pico handout for more information.

Starting vi

To start using vi, at the Unix prompt type **vi** followed by a file name. If you wish to edit an existing file, type in its name; if you are creating a new file, type in the name you wish to give to the new file.

```
%vi filename
```

Then hit **Return**. You will see a screen similar to the one below which shows blank lines with tildes and the name and status of the file.

vi's Modes and Moods

vi has two modes: the command mode and the insert mode. It is essential that you know which mode you are in at any given point in time. When you are in command mode, letters of the keyboard will be interpreted as commands. When you are in insert mode the same letters of the keyboard will type or edit text. vi always starts out in command mode. When you wish to move between the two modes, keep these things in mind. You can type **i** to enter the insert mode. If you wish to leave insert mode and return to the command mode, hit the **ESC** key. If you're not sure where you are, hit **ESC** a couple of times and that should put you back in command mode.

General Command Information

As mentioned previously, vi uses letters as commands. It is important to note that in general vi commands:

- are case sensitive: lowercase and uppercase command letters do different things
- are not displayed on the screen when you type them
- generally do not require a **Return** after you type the command.

You will see some commands which start with a colon (:). These commands are *ex* commands which are used by the *ex* editor. *ex* is the true editor which lies underneath vi -- in other words, vi is the interface for the ex editor.

Entering Text

To begin entering text in an empty file, you must first change from the command mode to the insert mode. To do this, type the letter **i**. When you start typing, anything you type will be entered into the file. Type a few short lines and hit **Return** at the end of each of line. Unlike word processors, vi does not use word wrap. It will break a line at the edge of the screen. If you make a mistake, you can use the Backspace key to remove your errors. If the Backspace key doesn't work properly on your system, try using the Ctrl h key combination.

Cursor Movement

You must be in command mode if you wish to move the cursor to another position in your file. If you've just finished typing text, you're still in insert mode and will need to press **ESC** to return to the command mode.

Moving One Character at a Time

Try using your direction keys to move up, down, left and right in your file. Sometimes, you may find that the direction keys don't work. If that is the case, to move the cursor one character at the time, you may use the **h**, **j**, **k**, and **l** keys. These keys move you in the following directions:

h	left one space	l	right one space
j	down one space	k	up one space

If you move the cursor as far as you can in any direction, you may see a screen flash or hear a beep.

Moving among Words and Lines

While these four keys (or your direction keys) can move you just about anywhere you want to go in your file, there are some shortcut keys that you can use to move a little more quickly through a document. To move more quickly among words, you might use the following:

w	moves the cursor forward one word
b	moves the cursor backward one word (if in the middle of a word, b will move you to the beginning of the current word).
e	moves to the end of a word.

To build on this further, you can precede these commands with a number for greater movement. For example, 5w would move you forward five words; 12b would move you backwards twelve words. [You can also use numbers with the commands mentioned earlier. For example, 5j would move you down 5 characters.]

Command Keys and Case

You will find when using vi that lower case and upper case command keys are interpreted differently. For example, when using the lower case **w**, **b**, and **e** commands, words will be defined by a space or a punctuation mark. On the other hand, **W**, **B**, and **E** commands may be used to move between words also, but these commands ignore punctuation.

Shortcuts

Two short cuts for moving quickly on a line include the **\$** and the **0** (zero) keys. The **\$** key will move you to the end of a line, while the **0** will move you quickly to the beginning of a line.

Screen Movement

To move the cursor to a line within your current screen use the following keys:

H moves the cursor to the top line of the screen.
M moves the cursor to the middle line of the screen.
L moves the cursor to the last line of the screen.

To scroll through the file and see other screens use:

ctrl-f scrolls down one screen
ctrl-b scrolls up one screen
ctrl-u scrolls up a half a screen
ctrl-d scrolls down a half a screen

Two other useful commands for moving quickly from one end to the other of a document are **G** to move to the end of the file and **1G** to move to the beginning of the file. If you precede **G** with a number, you can move to a specific line in the document (e.g. 15G would move you to line 15).

Moving by Searching

One method for moving quickly to a particular spot in your file is to search for specific text. When you are in command mode, type a **/** followed the text you wish to search for. When you press **Return**, the cursor will move to the first incidence of that string of text. You can repeat the search by typing **n** or search in a backwards direction by using **N**.

Basic Editing

To issue editing commands, you must be in command mode. As mentioned before, commands will be interpreted differently depending upon whether they are issued in lower or upper case. Also, many of the editing commands can be preceded by a number to indicate a repetition of the command.

Deleting (or Cutting) Characters, Words, and Lines

To delete a character, first place your cursor on that character. Then, you may use any of the following commands:

x deletes the character under the cursor.
X deletes the character to the left of your cursor.
dw deletes from the character selected to the end of the word.
dd deletes all the current line.
D deletes from the current character to the end of the line.

Preceding the command with a number will delete multiple characters. For example, **10x** will delete the character selected and the next 9 characters; **10X** will delete the 10 characters to the left of the currently selected character. The command **5dw** will delete 5 words, while **4dd** deletes four lines.

Pasting Text using Put

Often, when you delete or cut text, you may wish to reinsert it in another location of the document. The Put command will paste in the last portion of text that was deleted since deleted text is stored in a buffer. To use this command, place the cursor where you wish the deleted text to appear. Then use **p** to reinsert the text. If you are inserting a line or paragraph use the lower case **p** to insert on the line below the cursor or upper case **P** to place in on the line above the cursor.

Copying Text with Yank

If you wish to make a duplicate copy of existing text, you may use the yank and put commands to accomplish this function. Yank copies the selected text into a buffer and holds it until another yank or deletion occurs. Yank is usually used in combination with a word or line object such as the ones shown below:

```
yw  copies a word into a buffer (7yw copies 7 words)
yy  copies a line into a buffer (3yy will copy 3 lines)
```

Once the desired text is yanked, place the cursor in the spot in which you wish to insert the text and then use the put command (**p** for line below or **P** for line above) to insert the contents of the buffer.

Replacing or Changing Characters, Words, and Lines

When you are using the following commands to replace text, you will be put temporarily into insert mode so that you can change a character, word, line, or paragraph of text.

```
r    replaces the current character with the next character you enter/type.
      Once you enter the character you are returned to command mode.
R    puts you in overtype mode until you hit ESC which will then return
      you to command mode.
cw  changes and replaces the current word with text that you type.
      A dollar sign marks the end of the text you're changing.
      Pressing ESC when you finish will return you to command mode.
```

Inserting Text

If you wish to insert new text in a line, first position the cursor to the right of where you wish the inserted text to appear. Type **i** to get into insert mode and then type in the desired text (note that the text is inserted before the cursor). Press **ESC** to return to command mode.

Inserting a Blank Line

To insert a blank line below the line your cursor is currently located on, use the **o** key and then hit **ESC** to return to the command mode . Use **O** to insert a line above the line the cursor is located on.

Appending Text

You can use the append command to add text at any place in your file. Append (**a**) works very much like Insert (**i**) except that it insert text *after* the cursor rather than before it. Append is probably used most often for adding text to the end of a line. Simply place your cursor where you wish to append text and press **a**. Once you've finished appending, press **ESC** to go back to command mode.

Joining Lines

Since vi does not use automatic word wrap, it is not unusual in editing lines to end up with lines that are too short and that might be improved if joined together. To do this, place your cursor on the first line to be joined and type **J**. As with other commands, you can precede **J** with a number to join multiple lines (**4J** joins 4 lines).

Undoing

Be sure to remember this command. When you make a mistake you can undo it. **DO NOT** move the cursor from the line where you made the change. Then try using one of the following two commands:

- u** undoes the last change you made anywhere in the file. Using **u** again will "undo the undo".
- U** undoes all recent changes to the current line. You cannot have moved from the line to recover the original line.

Closing and Saving Files

When you edit a file in vi, you are actually editing a copy of the file rather than the original. The following sections describe methods you might use when closing a file, quitting vi, or both.

Quitting and Saving a File

The command **ZZ** (notice that it is in uppercase) will allow you to quit vi and save the edits made to a file. You will then return to a Unix prompt. Note that you can also use the following commands:

- :w** to save your file but not quit vi (this is good to do periodically in case of machine crash!).
- :q** to quit if you haven't made any edits.
- :wq** to quit and save edits (basically the same as ZZ).

Quitting without Saving Edits

Sometimes, when you create a mess (when you first start using vi this is easy to do!) you may wish to erase all edits made to the file and either start over or quit. To do this, you can choose from the following two commands:

- :e!** reads the original file back in so that you can start over.
- :q!** wipes out all edits and allows you to exit from vi.

More about Combining Commands, Objects, and Numbers

Now that you've learned some basic vi commands you might wish to expand your skills by trying some fancy combination steps. Some commands are generally used in combination with a text object. We've already seen some examples of this. For example, when you use the command **dw** to delete a word, that combines the delete (**d**) command with the word (**w**) text object. When you wish to delete multiple words, you might add a number to this combination. If you wished to delete 2 words you might use **2dw** or **d2w**. Either of these combinations would work. So, as you can see, the general format for a command can be

(number) (command) (text object) or (command) (number) (text object)

You might wish to try out some of the following combinations of commands and objects:

Command	Text Object
d (delete)	w (word to the left)
y (yank/copy)	b (word to the right or backward)
c (change)	e (end of word)
	H (top of the screen)
	L (bottom of the screen)
	M (middle of the screen)
	0 (zero - first character on a line)
	\$ (end of a line)
	((previous sentence)
) (next sentence)
	[(previous section)
] (next section)

Repeating a Command

If you are doing repetitive editing, you may wish to use the same command over and over. vi will allow you to use the dot (.) to repeat the last basic command you issued. If for example, you wished to delete several lines, you could use **dd** and then . (dot) in quick succession to delete a few lines.

A Quick Word about Customizing Your vi Environment

There are several options that you can set from within vi that can affect how you use vi. For example, one option allows you to set a right margin that will then force vi to automatically wrap your lines as you type. To do this, you would use a variation of the **:set** command. The **:set** command can be used to change various options in vi. In the example just described, you could, while still in vi, type **:set wrapmargin=10** to specify that you wish to have a right margin of 10. Another useful option is **:set number**. This command causes vi to display line numbers in the file you are working on.

Other Options

To view a listing of other options, you could type **:set all**. To view only those options which are currently in effect, you can type **set:** by itself. Options that you set while in a vi session will apply during that session only. To make permanent changes to your vi environment, you could edit your **.exrc** file. However, you should not edit this file unless you know what you are doing!

Useful vi Commands

Cut/Paste Commands:

x	delete one character (destructive backspace)
dw	delete the current word (Note: ndw deletes n numbered words)
dd	delete the current line (Note: ndd deletes n numbered lines)
D	delete all content to the right of the cursor
d\$	same as above
:u	undo last command
p,P	paste line starting one line below/above current cursor location
J	combine the contents of two lines
"[a-z]nyy	yank next n lines into named buffer [a-z]
"[a-z]p/P	place the contents of selected buffer below/above the current line

Extensions to the Above Commands:

:3,18d	delete lines 3 through 18
16,25m30	move lines 16 through 25 to after line 30
23,29co62	copy specified lines and place after line 62

Cursor Relocation commands:

:[n]	goto line [n]
shift g	place cursor on last line of text
h/l/j/k	move cursor left, right, down and up
^f/^b	move forward, backward in text, one page
^u/^d	move up, down one half page
\$	move to end of line
0	move to beginning of line

Extensions to the Above:

b	move backwards one word (Note: nb moves back n number of word(s))
e	move to end of current word
(move to beginning of current block
)	move to the end of current block

Searching and Substitution commands:

/ [string]	search forward for string
? [string]	search backwards for string
n	repeat last search
N	repeat search in opposite direction
cw	change the contents of the current word, (use ESC to stop replacement mode)
c\$	Replace all content to the right of cursor (exit replacement mode with ESC)
c0	Replace all content to the left of cursor (exit with ESC)
:1,\$s/s1/s2/g	(Yow!) global replacement of string1 with string2
r	replace current character with next character type

Entering the Insert Mode:

i	Begin inserting text at current cursor location
I	Begin inserting text at the beginning of the current line
a	Begin appending text, one character to the right of current cursor location
A	Begin appending text at the end of the current line
o/O	Begin entering text one line below/above current line
ESC	Exit insertion mode and return to command mode

Exiting and Entering vi

ZZ	save file and exit vi
:wq	same as above
:e!	return to last saved version of current file
:q	quit without save, (Note :q! is required if changes have been made)
:w	write without exit (:w! to force write)

Fancy Stuff:

:1,10w file	write lines 1 through 10 to file newfile
:340,\$w >> file	write lines 340 through the end of the file and append to file newfile
:sh	escape temporarily to a shell
^d	return from shell to vi
:[command]	execute UNIX command without leaving vi
:r![command]	read output of command into vi
:r[filename]	read filename into vi
:\$r newfile	read in newfile and attach at the end of current document
:r !sort file	read in contents of file after it has been passed through the UNIX sort
:n	open next file (works with wildcard filenames, ex: vi file*)
:^g	list current line number
:set number	show line numbers
:set showinsert	show flag ("I") at bottom of screen when in insert mode
:set all	display current values of vi variables
:set ai	set autoindent; after this enter the insert mode and tab, from this point on vi will indent each line to this location. Use ESC to stop the indentations.
^T	set the autoindent tab one tab stop to the right
^D	set the autoindent tab one stop to the left
:set tabstop=n	sets default tab space to number n
>>	shift contents of line one tab stop to the right
<<	shift contents of line one tab stop to the left

Access this document at <http://www.ccsf.org/Pub/Fac/vi.html>

It was found at a website adopted by Sai Anand Balu of University of Carolina at Chapel Hill, Academic Technology and Networks.

More detailed help pages for vi editor can be found at <http://www.ccsf.org/Pub/UNIXhelp/vi/ref.html>