**Chapter 8**

**vi Editor**

**1. General Introduction**

The vi editor (short for visual editor) is a screen editor which is available on almost all Unix systems. Once we have learned vi, we 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 we are just beginning to learn Unix, we might find the Pico editor easier to use (most command options are displayed at the bottom of the screen). If we use the Pine email application and have composed or replied to a message we have probably already used Pico as it is used for text entry. For more information please refer to the Pine/Pico page.

**1.1 Starting vi**

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

**$ vi filename**

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

**~**

**"myfile" [New file]**

**1.2 vi's Modes**

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

**2. General Command Informati on**

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 we type them
* Generally do not require a Return after we type the command.

We 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.

**2.1 Entering Text**

To begin entering text in an empty file, we must first change from the command mode to the insert mode. To do this, type the letter i. When we start typing, anything we 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 we make a mistake, we 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.

**2.2 Cursor Movement**

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

**2.2.1 Moving One Character at a Time**

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

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

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

**2.2.2 Moving among Words and Lines**

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

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

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

**2.3 Command Keys and Case**

We 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.

**2.4 Shortcuts**

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

**2.5 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 we precede G with a number, we can move to a specific line in the document (e.g. 15G would move us to line 15).

**2.6 Moving by Searching**

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

**Basic Editing**

To issue editing commands, we 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.

**3. Deleting (or Cutting) Characters, Words, and Lines**

To delete a character, first place your cursor on that character. Then, we 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.

**4. Pasting Text using Put**

When we delete or cut text, we may need 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 we need the deleted text to appear. Then use p to reinsert. If we 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.

**5. Copying Text with Yank**

If we wish to make a duplicate copy of existing text, we 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 we 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.

**6. Replacing or Chang ing Characters, Words, and Li nes**

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

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

**6.1 Inserting Text**

If we wish to insert new text in a line, first position the cursor to the right of where we 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.

**6.2 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.

**6.3 Appending Text**

We 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 we wish to append text and press a. Once we've finished appending, press ESC to go back to command mode.

**6.4 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, we can precede J with a number to join multiple lines (4J joins 4 lines).

**6.5 Undoing**

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

|  |  |
| --- | --- |
| u | undoes the last change we made anywhere in the file. Using **u** again will "undo the undo". |
| U | undoes all recent changes to current line. We can not have moved from the line to recover the original line. |

**7. Closing a nd Saving Files**

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

**7.1 Quitting and Saving a File**

The command ZZ (notice that it is in uppercase) will allow us to quit vi and save the edits made to a file. We will then return to a Unix prompt. Note that we 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 we haven't made any edits. |
| :wq | to quit and save edits (basically the same as ZZ). |

**7.2 Quitting without Saving Edits**

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

|  |  |
| --- | --- |
| :e! | Reads the original file back in so that we can start over. |
| :q! | Wipes out all edits and allows us to exit from vi. |

**More about Combining Commands, Objects, and Numbers**

Now that we've learned some basic vi commands we 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 we use the command dw to delete a word, that combines the delete (d) command with the word (w) text object. When we wish to delete multiple words, we might add a number to this combination. If we wished to delete 2 words we might use 2dw or d2w. Either of these combinations would work. So, as we can see, the general format for a command can be

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

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

|  |  |
| --- | --- |
| **Command** | **Description** |
| **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) |

**8. Repeating a Command**

If we are doing repetitive editing, we may wish to use the same command over and over. vi will allow we to use the dot (.) to repeat the last basic command we issued. If for example, we wished to deleted several lines, we 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 we can set from within vi that can affect how we use vi. For example, one option allows us to set a right margin that will then force vi to automatically wrap your lines as we type. To do this, we 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, we could, while still in vi, type :set wrapmargin=10 to specify that we 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 we are working on.

**8.1 Other Options**

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

**9. Useful vi Commands**

**9.1 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 |

**9.2 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 |

**9.3 Cursor Relocation commands:**

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

**9.4 Extensions to the Above:**

|  |  |
| --- | --- |
| b | move backwards one word (Note: nb moves back n number of words) |
| e | move to end of current word |
| ( | move to beginning of curent 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 typed |

**9.5 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 |

**9.6 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) |

**9.7 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 |

**10. COMMAND SUMMARY TEST**

Answer the following questions before continuing with the remainder of the exercises.

10. What is the command to move up one line?

11. What is the command to move down one line?

12. What is the command to delete a line?

13. What is the command to abandon the editor, and return to the shell?

14. What is the command to undo the last command?

15. What is the command to position the cursor at the end of the current line?

16. What is the command to write the buffer to the file and remain in vi?

17. What is the command to erase the character at the cursor position?

18. What is the command to position the cursor at the beginning of the previous word?