**https://docs.oracle.com/cd/E19455-01/806-2902/6jc3b36e2/index.html**

**Chapter 6 Using the vi Editor**

vi (pronounced "vee-eye," short for visual display editor) is the standard SunOS text editor. Since vi is not window-based, this multipurpose editor can be used on any kind of terminal to edit a wide range of file types.

You can enter and edit text with vi, but it is not a word-processor. It was not created to process formatted text in the familiar manner of a commercial word-processor. To produce formatted printouts, vi relies on a typesetting emulation program, such as nroff, troff, or ditroff. These programs allow you to format vi text by inserting codes that are then interpreted by the emulator.

vi contains a huge array of commands, many of which have overlapping functions. At first, it's quite normal for new users to feel overloaded by this. The purpose of this chapter, however, is to provide you with an overview of the most essential vi commands. As you begin to use vi, you'll find that it is an extremely powerful text editor, and that it may take you a while to become proficient.

Note that there is a read-only version of vi called view. When you open a file with view, you can use vi commands, but you cannot write (or save) your changes. This allows you or others to read a vi file without accidentally changing it.

**6.1 Starting vi**

In the sub-sections that follow, you'll learn how to start vi, enter text in a file, save (write) the file, and quit vi. You'll also create a practice file that you'll use for the rest of this chapter.

**6.1.1 Creating a File**

Start vi and edit the file paint as shown in this example:

|  |
| --- |
| $ **vi paint** |

If paint already exists, vi will open the existing file; if this is a new file, vi will create it. For the purposes of this example, paint should be a new file.

The vi editing screen appears in a moment:

**Figure 6-1 The vi Editing Screen**



The cursor appears in the upper left corner of the screen. Blank lines are indicated by a vertical series of tildes (~).

Note that you can also start vi without specifying a file name by just typing vi. You can then name the file later when you exit vi.

**6.1.2 The Status Line**

The last line of the screen, called the **status line**, shows the name of the file and the number of lines and characters in the file. When you create a new file, as is the case with our example, the status line indicates that it's a new file.

**6.2 The Two Modes of vi**

There are two modes of operation in vi: entry mode and command mode. You use **entry mode** to enter text into a file, while **command mode** is used to enter commands that perform specific vi functions. Command mode is the default mode for vi.

Since vi doesn't indicate which mode you're currently in, distinguishing between command mode and entry mode is probably the single greatest cause of confusion among new vi users. However, if you remember just a few basic concepts from the beginning, you should be able to avoid most of the usual "vi stress."

When you first open a vi file, it's always in command mode. Before you can enter text in the file, you must type one of the vi entry commands, such as i ("insert"), to insert text **at** the current cursor location, or a ("append"), to insert text **after** the current cursor location. (These and other vi entry commands are covered in greater detail later in this chapter.)

Whenever you want to return vi to command mode, press Esc. If you're not sure which mode vi is presently in, simply press Esc to make sure it's in command mode and continue from there. If you press Esc while vi is already in command mode, the system beeps and the screen flashes, but no harm is done.

**6.2.1 Entry Mode**

To enter text in the sample file paint, type the vi "insert" command i. This takes vi out of command mode and puts it into entry mode.

Now type a few short lines of text, ending every line with a Return. Characters you type appear to the left of the cursor and push any existing characters to the right. For the moment, you can correct your mistakes by backspacing and retyping a line before you press Return. Later you will learn how to edit the text you entered.

When you finish entering text in paint, press Esc to return to command mode. The cursor moves back onto the last character entered. Now you can enter more vi commands.

If vi seems to act unpredictably, make sure that you are not in "Caps Lock" mode, which would cause your entries to be all capital letters. On some systems, the F1 key (which is usually located next to the Esc key) acts as the Caps Lock. Pressing this key instead of Esc is a common error.

**Note -**

Occasionally you may need to instruct vi to clear or redraw the screen to eliminate, for example, extraneous system messages. To redraw the screen, enter command mode and press Ctrl-L. This is similar to the OpenWindows Refresh command.

**6.2.2 Command Mode**

When you open a file with vi, you are in command mode. In this mode, you can enter commands to implement a wide range of functions. Most vi commands consist of one or two letters and an optional number. Usually, there are upper and lowercase versions of commands that perform related but different functions. As an example, typing a appends the file to the right of the cursor, while typing A appends the file at the **end** of the line.

Most vi commands don't require that you press Return to execute them. Commands beginning with a colon (:), however, do require that you press Return after the command. Some discussions of the vi editor refer to commands preceded with a colon as a third, and uniquely separate mode of vi, **last-line mode**. This is because when you type the colon while in command mode, the colon and the remainder of what is typed appear on the bottom line of the screen. For the purpose of this discussion, however, all vi commands are initiated from command mode.

Commands preceded with a colon are actually **ex** commands. vi and ex are two separate interfaces to the same text editing program. While vi is a screen-oriented interface, ex is a line-oriented interface. The full set of ex commands is available from within vi. When you press the colon, you are actually switching to the line-oriented, ex interface. This allows you to perform many file manipulation commands without ever leaving vi. See ["6.6 Using ex Commands"](https://docs.oracle.com/cd/E19455-01/806-2902/6jc3b36eh/index.html), in this chapter, for further information.

**6.3 Ending a Session**

When you edit a file in vi, your changes are not made directly to the file. Instead, they are applied to a copy of the file that vi creates in a temporary memory space called the **buffer**. The permanent disk copy of the file is modified only when you **write** (save) the contents of the buffer.

This arrangement has its good and bad points. On the one hand, it means that you can quit a file and discard all the changes that you have made during an editing session, leaving the disk copy intact. On the other hand, you could lose the (unsaved) contents of the work buffer if the system crashes. (People on remote terminals connected by phone lines are especially vulnerable to unplanned interruptions.)

The best policy is to save your work frequently, especially when making substantive changes.

Caution -

Although it's possible to run multiple, simultaneous vi sessions on one file, it is not a good idea. Great confusion could result when you try to determine which changes have been written to the file and which changes have been overwritten from a simultaneous session.

**6.3.1 Saving Changes and Quitting vi**

vi is rich in more or less synonymous commands that control saving the buffer contents to a file and quitting vi. These commands give you the option of saving, saving-and-quitting, or quitting-without-saving.

**6.3.1.1 Saving**

Save the contents of the buffer (write the buffer to the file on disk) by typing:

|  |
| --- |
| **:w** |

followed by Return.

**6.3.1.2 Saving and Quitting**

Save and quit by typing:

|  |
| --- |
| **:wq** |

followed by Return. Alternatively, type ZZ.

Note that the command ZZ is neither preceded by a colon nor followed by Return.

**6.3.1.3 Quitting Without Saving**

When you've made no changes to a file and simply want to quit, type:

|  |
| --- |
| **:q** |

followed by Return. If you have made changes, vi will not let you quit with :q. Instead, it will display the message, No write since last change (:quit! overrides).

If you do not want to save your changes, type:

|  |
| --- |
| **:q!** |

followed by Return.

**6.4 Printing a File**

Once you have quit a vi file, you can print the file with the following command:

|  |
| --- |
| $ **lp** *filename* |

where **filename** is the name of the vi file to be printed. This command prints the file to your default printer. The file is printed without any formatting, line for line, just as it appears on the screen. See [Chapter 8, Using Printers](https://docs.oracle.com/cd/E19455-01/806-2902/6jc3b36ft/index.html), for more information on printer commands.

**6.5 Basic vi Commands**

The following sections explain several categories of vi commands. These include:

* Moving around in a file
* Inserting text
* Changing and substituting text
* Undoing changes to text
* Deleting text
* Copying and moving text
* Repeating commands

**6.5.1 Moving Around in a File**

In the previous sections you learned how to create, save, print, and exit a vi file. Now that you have created a file, you'll need to understand the concepts required to navigate within it. Open your practice file now, and try out each of the commands discussed in this section.

**6.5.1.1 Moving the Cursor**

When you start vi, the cursor is in the upper left corner of the vi screen. In command mode, you can move the cursor with a number of keyboard commands. Certain letter keys, the arrow keys, and the Return key, Back Space (or Delete) key, and the Space Bar can all be used to move the cursor when you're in command mode.

**Note -**

Most vi commands are case-sensitive; the "same" command typed in lowercase and uppercase characters could have radically different effects.

**Moving with Arrow Keys**

If your machine is equipped with arrow keys, try these now. You should be able to move the cursor freely about the screen using combinations of the up, down, right, and left arrow keys. Notice that you can only move the cursor across already existing text or input spaces.

If you're using vi from a remote terminal, the arrow keys may not work correctly. This will depend on your terminal emulator. If the arrow keys don't work in your case, you can use the following substitutes:

* To move left, press h.
* To move right, press l.
* To move down, press j.
* To move up, press k.

**Moving One Word**

Press w ("word") to move the cursor to the right one word at a time.

Press b ("back") to move the cursor to the left one word at a time.

Press W or B to move the cursor past the adjacent punctuation to the next or previous blank space.

Press e ("end") to move the cursor to the last character of the current word.

**Moving to Start or End of Line**

Press ^ to move the cursor to the start of the current line.

Press $ to move the cursor to the end of the current line.

**Moving Down One Line**

Press the Return key to move the cursor to the beginning of the next line down.

**Moving Left**

Press the Back Space key to move the cursor one character to the left.

**Moving Right**

Press the Space Bar to move the cursor one character to the right.

**Moving to the Top**

Press H ("high") to move the cursor to the top of the screen.

**Moving to the Middle**

Press M ("middle") to move the cursor to the middle of the screen.

**Moving to the Bottom**

Press L ("low") to move the cursor to the bottom of the screen.

**6.5.1.2 Paging and Scrolling**

If you move down when the cursor is at the bottom of the screen, or move up when the cursor is at the top of the screen, you will see the text scroll up or down. This can be an effective way to display more text in a very short file, but it can be tedious to move this way through a long file.

You may have noticed that moving the cursor either past the bottom or past the top of the screen has the effect of scrolling text up or down. This works for a very short file, but it is a tedious way to move through a long file.

You can page or scroll backward or forward through a file, a screen or a half-screen at a time. (To try out these commands on paint, you might want to add text so you have a longer file to work with.)

Note that there is a fundamental difference between paging and scrolling. Scrolling actually scrolls the cursor up or down through the text **a line at a time,**as though it were on a paper scroll. Paging moves the cursor up or down through the text **a screenful at a time**. On a fast system, you might not notice the difference. However, if you're working from a remote terminal or in some other situation where your system is running slower than usual, this difference can become painfully apparent.

**Page Forward One Screen**

To scroll forward (move down) one screenful, press Ctrl-F. (Hold down the Control key and press the F key.) The cursor moves to the upper left corner of the new screen.

**Scroll Forward One-Half Screen**

To scroll forward one half of a screen, press Ctrl-D.

**Page Backward One Screen**

To scroll backward (i.e., move up) one screenful, press Ctrl-B.

**Scroll Backward One-Half Screen**

To scroll backward one half of a screen, press Ctrl-U.

**6.5.2 Inserting Text**

vi provides many commands for inserting text. This section introduces you to the most useful of these commands. Note that each of these commands places vi in entry mode. To use any of these commands, you must first be in command mode. Remember to press Esc to make sure you are in command mode.

**6.5.2.1 Append**

Type a (append) to insert text to the **right** of the cursor. Experiment by moving the cursor anywhere on a line and typing a, followed by the text you want to add. Press Esc when you're finished.

Type A to add text to the **end** of a line. To see how this works, position the cursor anywhere on a text line and type A. The cursor will move to the end of the line, where you can type your additions. Press Esc when you're done.

**6.5.2.2 Insert**

Insert text to the left of the cursor by typing i from command mode.

Type I to insert text at the beginning of a line. (The command will move the cursor from any position on that line.) Again, as with all the commands in this section, press Esc to return to command mode after entering the desired text.

**6.5.2.3 Open Line**

Use these commands to open new lines, either above or below the current cursor position.

Type o to open a line **below** the current cursor position. To experiment, type o followed by a bit of text. You can enter several lines of text if you like. Press Esc when you are finished.

Type O to open a line **above** the current cursor position.

**6.5.3 Changing Text**

Changing text involves substituting one section of text for another. vi has several ways to do this, depending on circumstances.

**6.5.3.1 Changing a Word**

To replace a word, position the cursor at the beginning of the word to be replaced. Type cw, followed by the new word. To finish, press Esc.

To change **part** of a word, place the cursor on the word, to the **right** of the portion to be saved. Type cw, enter the correction, and press Esc.

**6.5.3.2 Changing a Line**

To replace a line, position the cursor anywhere on the line and type cc. The line disappears, leaving a blank line for your new text (which can be of any length). Press Esc to finish.

**6.5.3.3 Changing Part of a Line**

To replace part of a line, place the cursor to the **right** of the portion to be saved. Type C, enter the correction, and press Esc. This changes the portion of the line from the current cursor position to the end of the line.

**6.5.3.4 Substituting Character(s)**

To substitute one or more characters for the character under the cursor, type s, followed by the new text. Press Esc to return to command mode.

**6.5.3.5 Replacing One Character**

Use this command to replace the character highlighted by the cursor with another character. Position the cursor over the character and type r, followed by just one replacement character. After the substitution, vi automatically returns to command mode (there's no need to press Esc).

**6.5.3.6 Transposing Characters**

Correcting transposed characters takes just two keystrokes in vi. Suppose you find that you've typed "teh" when you meant to enter "the". Make the correction by putting the cursor over the first letter to be moved (in this case, e), and then type xp. The e and h will trade places - and vi will automatically return to command mode.

**6.5.3.7 Breaking or Joining Lines**

To break a line without affecting text, move the cursor to a space where you want the line to break and type r (for "replace") followed by Return. Note that if you type r with the cursor on a character and then press Return, that character will be replaced by the Return.

To join two lines, place the cursor on the upper line and type an uppercase J. (There's no need to press Esc after typing J.)

**6.5.4 Undoing Changes**

When editing text and making changes to a vi file, there will no doubt be times when you'll wish that you had **not** changed something. vi's undo commands allow you to back up one operation and continue on from there.

**6.5.4.1 Undoing the Previous Command**

If you make a mistake in vi or if you just change your mind once an operation is completed, you can undo your last command by pressing u immediately after the command. (There's no need to press Esc after typing u.) Pressing u a **second** time undoes the undo.

**6.5.4.2 Undoing Changes to a Line**

Type U to undo all changes you've made to a line. This command works only if you haven't moved the cursor off the line. (There's no need to press Esc after typing U.)

**6.5.5 Deleting Text**

These vi commands delete the character, word, or line you indicate. vi stays in command mode, so any subsequent text insertions must be preceded by additional commands to enter entry mode.

**6.5.5.1 Deleting One Character**

To delete one character, position the cursor over the character to be deleted and type x.

The x command also deletes the space the character occupied--when a letter is removed from the middle of a word, the remaining letters will close up, leaving no gap. You can also delete blank spaces in a line with the x command.

To delete one character before (to the left of) the cursor, type X (uppercase).

**6.5.5.2 Deleting a Word or Part of a Word**

To delete a word, position the cursor at the beginning of the word and type dw. The word and the space it occupied are removed.

To delete part of a word, position the cursor on the word to the **right** of the part to be saved. Type dw to delete the rest of the word.

**6.5.5.3 Deleting a Line**

To delete a line, position the cursor anywhere on the line and type dd. The line and the space it occupied are removed.

**6.5.5.4 Deleting Part of a Line**

You can also delete part of a line.

To delete everything to the **right** of the cursor, position the cursor to the right of the part of the line you want to save, and type D.

To delete everything to the **left** of the cursor, position the cursor to the right of the part of the line you want to delete and type d0 (d-zero).

**6.5.5.5 Deleting to the End of the File**

To delete everything from the current line to the end of the file, type dG. This also deletes the line on which the cursor is located.

**6.5.5.6 Deleting from Beginning of File**

To delete everything from the beginning of the file to the current line, type d1G. This also deletes the line on which the cursor is located.

**6.5.6 Copying and Moving Text -- Yank, Delete, and Put**

Many word-processors allow you to "copy and paste" and "cut and paste" lines of text. The vi editor also includes these features. The vi command-mode equivalent of "copy and paste" is **yank and put**; the equivalent of "cut and paste" is **delete and put**.

The methods for copying or moving small blocks of text in vi involves using a combination of the yank, delete, and put commands.

**6.5.6.1 Copying Lines**

To copy a line requires two commands: yy or Y ("yank") and either p ("put below") or P ("put above"). Note that Y does the same thing as yy.

To yank one line, position the cursor anywhere on the line and type yy. Now move the cursor to the line above where you want the yanked line to be put (copied), and type p. A copy of the yanked line will appear in a new line **below** the cursor.

To place the yanked line in a new line **above** the cursor, type P.

The yy command works well with a count: to yank 11 lines, for example, just type 11yy. Eleven lines, counting down from the cursor, will be yanked, and vi indicates this with a message at the bottom of the screen: 11 lines yanked.

You can also use the P or p commands immediately after any of the deletion commands discussed earlier. This puts the text you deleted above or below the cursor, respectively.

Caution -

Use only cursor-moving commands between yanking or deleting and putting. If you delete or yank any other text before putting the new text in place, the lines you yanked or deleted will be lost.

**6.5.6.2 Moving Lines**

Moving lines also requires two commands: dd ("delete") and either p or P.

To move one line, position the cursor anywhere on the line and type dd. For example, to delete 5 lines, type 5dd.

Next, move the cursor to the line above where you want the deleted line reinserted and type p. This inserts the text on a new line below the cursor.

Alternatively, you can put the deleted line above the cursor by typing P.

**6.5.6.3 Using Named Buffers**

To repeatedly insert a group of lines in various places within a document, you can yank (or delete) the lines into a named buffer. You specify named buffers by preceding a command with double quotes (") and a name for the buffer. For example, to yank four lines into the named buffer **a**, type "a4yy". You can use several different buffers. For example, you might also delete text from one location and add it to several others. To delete 12 lines into the named buffer **b**, type "b12dd".

To insert the text, precede the p or P command with *n*, where **n** is the named buffer. For example, to insert the lines saved in buffer **b**, type "bP".

You can overwrite named buffers with new lines. The buffers are saved until you exit vi.

When you use named buffers, you can safely delete and yank other text without affecting the lines you have already saved in the named buffers -- unless, of course, you purposely overwrite the named buffer.

**6.5.7 Using a Count to Repeat Commands**

Many vi commands can be preceded by a repeat factor (called a **count**) -- a number that precedes the command and tells it how many times to repeat the operation.

Most of the commands in the previous sections take counts. For instance, 3dd repeats the command to delete a line three times, therefore deleting three lines. 2dw deletes two words, and 4x deletes four characters or spaces. You can also use counts with commands to move the cursor, such as 3w and 2Ctrl-F. This will all become evident as you learn the vi commands. In the section ["6.12 Summary of Basic vi Commands"](https://docs.oracle.com/cd/E19455-01/806-2902/6jc3b36f0/index.html)each command that takes a count is indicated by "[count]" before the command name.

Typing a period (.) repeats the previous text-changing command. For example, if you have just deleted a line with dd, you can move the cursor to another line and delete it by simply typing a period.

**6.6 Using ex Commands**

ex commands are more accurate and convenient than yank, delete, and put when you're dealing with large blocks of text. Rather than counting lines on the screen and then searching for an insertion point, you give vi a range of lines to be moved or copied and then specify the line before the insertion point. (Of course, with a delete command there is no insertion point.)

**6.6.1 Turning Line Numbers On and Off**

To turn line numbers **on**, type :set nu and press Return.

Line numbers appear in the left margin. Note that these numbers do not appear when you print out the file. They are visible only on the screen.

|  |
| --- |
| 1 Oh, when I die, take my saddle from the wall,  2 Put it on my pony, lead him out of the stall.  3 Tie my bones to his back, point our faces to the west,  4 And we'll ride the prairies that we love the best.  5  6 Ride around, little doggies,  7 Ride around real slow.  8 Firey and Snuffy are rarin' to go.  ~  ~  ~  ~  ~  ~  :set nu |

To turn line numbers **off**, type :set nonu and press Return.

**6.6.2 Copying Lines**

The basic form of the ex copy command is:

|  |
| --- |
| **:line#,line# co line#** |

The first two numbers (separated by a comma) specify the range of lines to be copied. The third number is the line **before** the insertion point.

For example, to copy lines 1 through 5 of paint and place the copy after line 12, you would type the following:

|  |
| --- |
| **:1,5 co 12** |

and press Return.

When specifying line ranges, use the abbreviations

* Period (.) to denote "from the current line."
* Dollar sign ($) to denote "to end of file."

Thus, to copy the range "from the current line through line 5" and insert this block after line 12, you would type:

|  |
| --- |
| **:.,5 co 12** |

To copy the range "from line 6 through the end of the file" and insert this block after line 2, you would type:

|  |
| --- |
| **:6,$ co 2** |

**6.6.3 Moving Lines**

The basic form of the ex move command is similar to the copy command discussed above:

|  |
| --- |
| :*line#*,*line#* m *line#* |

Line ranges and insertion points are specified in the same ways, including use of the abbreviations . and $. The difference in function is simply that "move" removes a block from one location and reinserts it elsewhere.

For example, to move lines 1 through 5 to the line following 12, you would type:

|  |
| --- |
| **:1,5 m 12** |

and press Return.

**6.6.4 Deleting Lines**

To delete a range of lines, use the command form:

|  |
| --- |
| :*line#*,*line#* **d** |

For example, to delete lines 1 through 5, you would type:

|  |
| --- |
| **:1,5 d** |

**6.7 Searching and Replacing with vi**

vi provides several ways to find your place in a file by locating a specified string of characters. It also has a powerful global replacement function.

**6.7.1 Finding a Character String**

A **character string** is simply one or more characters in a row. It may include letters, numbers, punctuation, special characters, blank spaces, tabs, or carriage returns. A string may be a grammatical word or it may be part of a word.

To find a character string, type / followed by the string you want to search for, and then press Return. vi positions the cursor at the next occurrence of the string. For example, to find the string "meta", type /meta followed by Return.

Type n to go to the **next** occurrence of the string; type N to go to the **previous** occurrence.

To search backward in a file, you can use ? instead of /. In this case, the directions of n and N are reversed.

Searches normally are case-sensitive: a search for "china" will not find "China." If you want vi to ignore case during a search, type :set ic. To change it back to the default, case-sensitive mode, type :set noic.

If vi finds the requested string, the cursor will stop at its first occurrence. If the string is not found, vi will display Pattern not found on the last line of the screen.

Certain special characters ( / & ! . ^ \* $ \ ?) have special significance to the search process and must be "escaped" when used in a search. To escape a special character, precede it with a backslash (\). For example, to search for the string "anything?" type /anything\? and press Return.

These special characters can be used as commands to the search function, so if you want to search for a string including one or more of these characters, you must indicate this by preceding the character with a backslash. To escape a backslash itself, type \\.

**6.7.2 Refining the Search**

You can make searches more precise by tagging the string with indicators for the following characteristics:

* Beginning of line
* End of line
* Beginning of word
* End of word
* Wild card characters

To match the beginning of a line, start the search string with a caret (^). For example, to find the next line beginning with "Search", type:

|  |
| --- |
| **/^Search** |

To match the end of a line, end the search string with a dollar sign ($). For example, to find the next line ending with "search.", type:

|  |
| --- |
| **/search\.$** |

(Note that the period is escaped with a backslash.)

To match the beginning of a word, type \< at the beginning of the string; to match the end of a word, type \> at the end of the string. Thus, to match a word, rather than a string, combine the end-of-word and beginning-of-word tags in the search pattern. For example, to find the next occurrence of the word--as opposed to the string--"search", type:

|  |
| --- |
| **/\<search\>** |

To match any character, type a period (.) in the string at the location to be matched. For example, to find the next occurrence of "disinformation" or "misinformation", type:

|  |
| --- |
| **/.isinformation** |

Since this is a search for a string and not a word, this search pattern may also find such constructions as "misinformationalist" and "disinformationism".

To search for alternative characters in a string, enclose the alternatives in brackets. The search pattern /[md]*string* will find strings beginning with either m or d. On the other hand, /[d-m]*string* will find strings beginning with any letter from d through m.

To match zero or more occurrences of the last character, type an asterisk (\*) in the string. You can effectively combine brackets and the asterisk to look for well-defined alternatives. For example, to find all strings beginning with a through z and ending with "isinformation" **and** to find all occurrences of the string "isinformation", type:

|  |
| --- |
| **/[a-z]\*isinformation** |

**6.7.3 Replacing a Character String**

The procedure for replacing a text string is based on the search procedures discussed above. All the special matching characters for searches can be used in search-and-replace.

The basic command form is:

|  |
| --- |
| **:g/search-string/s//replace-string/g** |

followed by the Return key.

Therefore, to replace every occurrence of the string "disinformation" with "newspeak", you would type:

|  |
| --- |
| **:g/disinformation/s//newspeak/g** |

and press Return.

You can modify this command to halt the search and make vi query whether you want to make the replacement in each instance. The following command uses gc (adding c for "consult") to make vi stop at every occurrence of "disinformation" and ask whether you want to make the substitution. Respond with y for yes or n for no.

|  |
| --- |
| **:g/disinformation/s//newspeak/gc** |

**Note -**

You can cancel a "consulted" search-and-replace by pressing Ctrl-C.

**6.7.4 Going to a Specific Line**

To go to the last line of an open file, by type G. To return to the first line of the file, by type 1G.

You can go to any other line by typing its number followed by G.

For example, suppose that you quit the file paint while editing line 51. You can access that line by opening the file and typing 51G.

**6.8 Inserting One File into Another**

vi makes it convenient to "read" (insert) a file into the file you are editing. The general form of the command is:

|  |
| --- |
| **:line#** r **filename** |

If you do not specify a line number, vi inserts the file at the current cursor position.

For example, if you wanted to insert the file orwell at line 84 of the file paint, you would type:

|  |
| --- |
| **:84 r orwell** |

Or you could position the cursor on line 84 and type:

|  |
| --- |
| **:r orwell** |

**6.9 Editing Multiple Files**

vi allows you to edit multiple files. For example, to edit the file orwell while you are editing paint:

1. First, save your current work in paint. Type :w and press Return.
2. To edit orwell, type :n orwell and press Return.
3. Make editing changes to orwell and save your work.
4. When you are finished working with orwell and have saved your work, you have three choices:
   * Exit vi. Type :q and press Return.
   * Return to paint. Type :n # and press Return.
   * Swap back and forth between two files with the command :n #.

**6.9.1 Editing a Series of Files**

To edit a series of files, list the file names after the vi command when you start vi from the command prompt:

|  |
| --- |
| **$** **vi paint orwell** |

The files appear in the order in which they are listed. First paint appears. When you finish editing paint, type :n to go on to the next file, orwell. To go on to the next file without saving changes in the current file, type :n! instead of :n.

If you have a series of files with related names (for example, test1, test2, test3), you can use wildcard characters to specify a group of files:

|  |
| --- |
| **$ vi test\*** |

The files will appear for editing in alphabetical order by name.

**6.9.2 Copying Lines Between Files**

To copy lines from one file to another, do the following:

1. Edit the first file.
2. Save the desired lines in named buffers, using the yank command. For example, to save 10 lines in buffer **a**, type a10Y.
3. Without exiting vi, edit the next file (orwell in this example):

|  |
| --- |
| **:n orwell** |

1. Add the lines from the first file with the put command. For example, to put the contents of buffer **a** below the current cursor position, type ap.

Remember that the contents of all named buffers are lost whenever you exit vi. Don't use the quit (:q) command until you have finished any operations involving named buffers.

**6.10 Setting vi Parameters**

vi has many variables that affect its behavior and appearance. You can view a list of these variables (with their current settings) while running vi by typing:

|  |
| --- |
| **:set all** |

followed by Return.

**6.11 Recovering from a Crash**

If the system crashes, the contents of your buffer are at risk. Often, though, you can recover most of your work by restarting vi with the command form:

|  |
| --- |
| **vi -r** **filename** |

where **filename** is the file you were editing at the time of the crash. The system will usually send you mail after the system is restarted, telling you there is a recover file.

**6.12 Summary of Basic vi Commands**

The following table provides a convenient reference for basic vi commands.

Table 6-1 Basic vi Commands

| **Command** | **Meaning** |
| --- | --- |
| **Starting** vi |  |
| vi **filename** | Open or create file |
| vi | Open new file to be named later |
| vi **-r** **filename** | Recover crashed file |
| view **filename** | Open file read-only |
| **Cursor** **Commands** |  |
| h | Move left one character |
| j | Move down one line |
| k | Move up one line |
| l | Move right one character |
| w | Move right one word |
| W | Move right one word (past punctuation) |
| b | Move left one word |
| B | Move left one word (past punctuation) |
| e | Move to end of current word |
| Return | Move down one line |
| Backspace | Move left one character |
| Spacebar | Move right one character |
| H | Move to top of screen |
| M | Move to middle of screen |
| L | Move to bottom of screen |
| Ctrl-F | Page forward one screen |
| Ctrl-D | Scroll forward one-half screen |
| Ctrl-B | Page backward one screen |
| Ctrl-U | Scroll backward one-half screen |
| **Inserting Characters and Lines** |  |
| a | Insert characters to right of cursor |
| A | Insert characters at end of line |
| i | Insert characters to left of cursor |
| I | Insert characters at beginning of line |
| o | Insert line below cursor |
| O | Insert line above cursor |
| **Changing Text** |  |
| cw | Change word (or part of word) to right of cursor |
| cc | Change line |
| C | Change from cursor to end of line |
| s | Substitute string for character(s) from cursor forward |
| r | Replace character at cursor with one other character |
| r Return | Break line |
| J | Join current line and line below |
| xp | Transpose character at cursor and character to right |
| ~ | Change case of letter (upper or lower) |
| u | Undo previous command |
| U | Undo all changes to current line |
| :u | Undo previous last-line command |
| **Deleting Text** |  |
| x | Delete character at the cursor |
| X | Delete character to the left of the cursor |
| dw | Delete word (or part of word to right of cursor) |
| dd | Delete line containing the cursor |
| D | Delete part of line to right of cursor |
| dG | Delete to end of file |
| d1G | Delete from beginning of file to cursor |
| :5,10 d | Delete lines 5-10 |
| **Copying and Moving Text** |  |
| yy | Yank or copy line |
| Y | Yank or copy line |
| p | Put yanked or deleted line below current line |
| P | Put yanked or deleted line above current line |
| :1,2 co 3 | Copy lines 1-2 and put after line 3 |
| :4,5 m 6 | Move lines 4-5 and put after line 6 |
| **Setting Line Numbers** |  |
| :set nu | Show line numbers |
| :set nonu | Hide line numbers |
|  | **Setting Case-sensitivity** |
| :set ic | Searches should ignore case |
| :set noic | Searches should be case-sensitive |
| **Finding a Line** |  |
| G | Go to last line of file |
| 1G | Go to first line of file |
| 21G | Go to line 21 |
| **Searching and Replacing** |  |
| /**string** | Search for **string** |
| ?**string** | Search backward for **string** |
| n | Find next occurrence of **string** in search direction |
| N | Find previous occurrence of **string** in search direction |
| :g/*search*/s//*replace*/g | Search and replace |
| **Clearing the Screen** |  |
| Ctrl-L | Clear (refresh) scrambled screen |
| **Inserting a File into a File** |  |
| :r **filename** | Insert (read) file after cursor |
| :34 r **filename** | Insert file after line 34 |
| **Saving and Quitting** |  |
| :w | Save changes (write buffer) |
| :w **filename** | Write buffer to named file |
| :wq | Save changes and quit vi |
| ZZ | Save changes and quit vi |
| :q! | Quit without saving changes |

#To add some in 1 group:

useradd -G qa-others rmanu

The command “**wc**” basically means “word count” and with different optional parameters one can use it to count the number of lines, words, and characters in a text file.

To count the number of lines, use “**wc**” with “**l**” as

|  |  |
| --- | --- |
| 1 | wc -l yourTextFile |

To count the number of words, use “**wc**” with “**w**” option as

|  |  |
| --- | --- |
| 1 | wc -w yourTextFile  Space in a dir  df -k  Filesystem 1K-blocks Used Available Use% Mounted on  udev 16452660 0 16452660 0% /dev  tmpfs 3294532 330144 2964388 11% /run  /dev/mapper/os-root 4735064 896300 3575188 21% /  /dev/mapper/os-usr 103087648 13453156 85306388 14% /usr  tmpfs 16472656 3120 16469536 1% /dev/shm  tmpfs 5120 0 5120 0% /run/lock  tmpfs 16472656 0 16472656 0% /sys/fs/cgroup  /dev/sda1 463844 105930 329447 25% /boot  /dev/mapper/os-tmp 7742856 18404 7308096 1% /tmp  /dev/mapper/os-var 51481468 1385276 47886484 3% /var  /dev/mapper/os-var+log 206319288 393640 197424640 1% /var/log  /dev/mapper/os-var+log+audit 9710112 51220 9142592 1% /var/log/audit  /dev/mapper/os-opt 20516476 1827448 17803836 10% /opt  10.50.38.17:/vol/home4\_003cf1/homes4/smazumder 2684354560 1836938880 847415680 69% /homes/smazumder |

split [options] filename prefix

-l linenumber

-b bytes