Vi tutorial

Author : Vasudev Ram

Introduction :

This is a brief tutorial on the vi editor available on all UNIX/Linux platforms. The purpose of this tutorial is to get reader acquainted with the basic features of this editor, and quickly start using it for everyday tasks.

For more details, refer to the man page for vi.

Vi is a text editor with 'modes' of operation.
The modes are :
1. Command mode (the default when you start vi).
2. Input mode (various versions of this exist).
3. Last line mode (also called ex mode).

Most commands are mnemonic, i.e. the letter for the command is also the 1st letter of a word which tells you what the command does.

e.g. w to move forward by one _w_ord.
b to move _b_ack by one word.
y to _y_ank
p to _p_ut
etc.

This makes it easy to remember the letters for commands.

Starting vi :

Type :

vi filename

or

vedit filename

You can also use the vedit command instead of vi. Vedit has the same functionality as vi but is slightly more user-friendly (e.g. it shows which mode you are currently in, tells you how many lines were affected by the last command, etc). Vedit can be useful when you are learning to use vi for the 1st time. After you are familiar with the vi commands, if you prefer, you can go back to using vi which is less distracting since it doesn't give so many information messages (it assumes that you know what you are doing).

1. Command mode

In command mode, keys entered are treated as commands rather than as input. Most commands are single letters. Case is significant. Some commands can be combined, or prefixed with qualifiers. Commands are of various types, e.g. to move around in the file, to change (add/delete/modify) text, etc.

Common commands include :
- cursor movement commands
- text modification commands
- search commands
- file-related commands
- miscellaneous commands

Cursor movement commands :

The simplest movement commands are :

h j k l

These commands move the cursor up, down, left and right by one character respectively. If you keep the key pressed, the command repeats. e.g. k moves down by one line. kkkkk moves down by 5 lines. This can also be given as 5k. Most commands accept a number prefix indicating how many times to repeat the command.

The ENTER (carriage return) and + keys work same as j.
The - key is the same as k.

It is preferable not to use arrow (cursor) keys for movement. These keys generate escape sequences which are treated as other commands by vi. Although some versions of vi are implemented in such a way that the arrow keys work the way they do in DOS, this is not guaranteed on all UNIX platforms. So if you don't get into the habit of using arrow keys for cursor movement, you can avoid some unpleasant surprises when you move to another UNIX version. However, this is a personal choice.

Movement in larger increments :

n| (that's the pipe or vertical bar character) (where n is any number) moves the cursor to the n'th character on the line. If the character position is invalid (e.g. 85| on a line of 60 characters), vi beeps. Just | by itself moves to the beginning of the line.

0 and $ move to the start and end of the line respectively.
^ moves to the 1st non-blank character in the line.

fx (where x is any character) searches forward (finds) for the next occurrence of that character from the cursor position, on the current line ONLY, i.e. it does not work across lines. If the character exists on the line, the cursor moves to it. If not, vi beeps. After a successful find, you can repeat (again on the same line only) by typing a semicolon (;). You can also type a comma (,) to repeat the find in the opposite direction.

Fx works exactly like fx, except that it scans from right to left instead of left to right. Comma and semi-colon (see previous paragraph) also work the same after an Fx command.

w and b move forward and backward by one word respectively.

W and B do the same, but the definition of a word is different (see man vi for details of this, as well as for more detailed descriptions of all the commands).

H moves to the top (highest) line on the screen.
M moves to the middle line on the screen.
L moves to the lowest line on the screen.
(These commands refer to the screen, not the file as a whole).

Ctl-d (control-d) scrolls down by half a page.
Ctl-u scrolls up by the same amount.
Ctl-f and ctl-b scroll forward and backward a page.
10ctl-f scrolls forward 10 pages.

nG where n is a number, moves to that line number in the file. E.g. 1G goes to the top of the file, 157G to the 157th line, and just G goes to the end of the file.

mx where x is any single letter, marks the current line with a marker called x. Later you can move back to that line with the command 'x, where x is the same letter.

e.g. ma (marks current line as a).
Move around in the file a bit.
'a (takes you back to the line you were on when you gave the command ma).

You can use the letters a to z as markers.
The markers are forgotten when you reload the file.

`` (which acts as a toggle) takes you to the position you were at (before the latest movement command).

Text modification commands :

x deletes the character under the cursor.

rn (where n is any character) replaces the character under the cursor with the character n.

cw (change word) puts you into input mode (see below). The last letter in the word you are on, gets changed to a $ sign. Whatever you type, until you type Escape, replaces the letters between the cursor and the $ sign. The most common use of this command is to change a single word to another word. But it is not limited to this alone. You can type two words, an entire line or even many lines. Everything you type until Escape, will be used to replace the characters from the cursor position till the $ sign.

dw (delete word) deletes the current word you are on.

Miscellaneous commands :

Ctl-l to redraw the screen.

2. Input mode :

Input mode means that whatever you type (except Escape key) is understood by vi to be text that you want to enter into the file, and is not a command. There are many different ways of entering input mode, but only one way of exiting input mode - by typing Escape.

Keys to enter input mode :

i (meaning insert before the cursor) puts you in input mode. All keys you type after this are treated as text and entered in the file before the current cursor position. The characters ahead are shifted as needed.

I also puts you in input mode, but it starts at the beginning of the line (actually at the 1st non-blank character on the line).

a (meaning append after the cursor) is like insert (i) but appends characters after the cursor, instead of before.

A is like a but starts appending at the end of the line.

R (replace characters) puts you in another kind of input mode (called replace mode) in which whatever you type overwrites whatever is under the cursor. This goes on till you type Escape. As with cw command, you can replace 1 character (but the r command is easier for this as you don't have to type Escape), many characters or even many lines. But if you press ENTER and continue typing, from the 2nd line onwards, vi behaves as if you are in insert mode. That is, only what was on the 1st line you were on, will replace the characters on that line which you typed over; the remaining lines you type will be inserted and will not overwrite existing text.

o (open mode) opens (inserts) a new line below the current line and puts you into input mode on that line. This goes on until you type Escape.

O is like o but opens a new line above the current line, rather than below.

All of the above input modes (i, I, a, A, R, o, O) are terminated by pressing Escape, which takes you back to command mode.

If you press Escape while already in command mode, vi beeps.

Don't use the arrow keys in input mode either. As mentioned above, they generate Escape sequences - and escape sequences begin with the Escape key. This can have the unintended effect of taking you back to command mode and even deleting your text. E.g. if the left arrow key generates the sequence Escape-D, pressing it will put you back into command mode and execute the D command, which deletes to the end of the line!

3. Last line mode :

In this mode, you are actually entering commands to the ex editor, on which vi is built (vi is basically a visual extension of the ex or ed editor, which is a command-line-only editor like edlin in older DOS versions). You get into the last line mode from command mode, by entering a : (colon) character. The colon and whatever you type is displayed on the last line of the screen (which is how this mode gets its name). All characters you type after the colon until a carriage return, are treated as though they are ex commands. There are a large number of ex commands. Refer to the ex man page for complete details. Here we will discuss some of the commonly used commands.
Note: Last line differs from command in that the command is not completed till you press the ENTER key. In command mode, the ENTER key is not needed to end the command - in fact it is itself a command (same functionality as j key).
In the examples below, it is assumed that you press the ENTER key after the command shown.

Saving your work :

w command (write)

:w
This saves the file to disk under the same name you opened it. I.e. if you opened the file with the command "vi filemame", the contents will be saved to filename.

:w filename
This command (write) saves the current contents of the file to the given file name.

:n,mwfilename
This saves the contents of line numbers n to m to the given filename. Use $ in place of m to refer to the last line number without knowing what its value is.

If the filename to which you are trying to save already exists, vi will warn you. In this case use w! instead of w. But this will still only work if you have permission to write to the file. For root (the superuser), this restriction is not enforced (may vary on different UNIX versions).

e command (edit)

:e filename

This is to start editing another file instead of the current one. If you have not saved your latest changes, vi will warn you. To override anyway, use e! instead of e. Once you have used the e command once, you can toggle back and forth between the two files by the command e#.

End of Vi tutorial