

Vi-editor

- Vi- editor is one of the most versatile editors of UNIX.
- The vi-editor was created by Bill Joy for BSD versions for unix.
- It has now become standard on almost all versions of UNIX.
- Bram Moolenaar improved the editor and called it "vim" (vi-improved) editor.
- Vi uses number of internal commands to navigate to any point in a text file and edit the text there.
- It allows to copy and move text within a file and also from one file to another.
- Vi offers cryptic and sometimes mnemonic, internal commands for editing work.

Vi editor operates in three mode.

✓ Input/Insert mode

If a user wish to insert content in the file the user has to be in insert mode. To switch from command mode to insert mode user has to use insert mode commands. To turn back to command mode one has to press <ESC> key.

✓ Ex-mode (Last line mode)

This mode is used for applying line commands. To enter into this mode one has to be in command mode and then type (:) colon. The cursor then moves to last line in the editor.

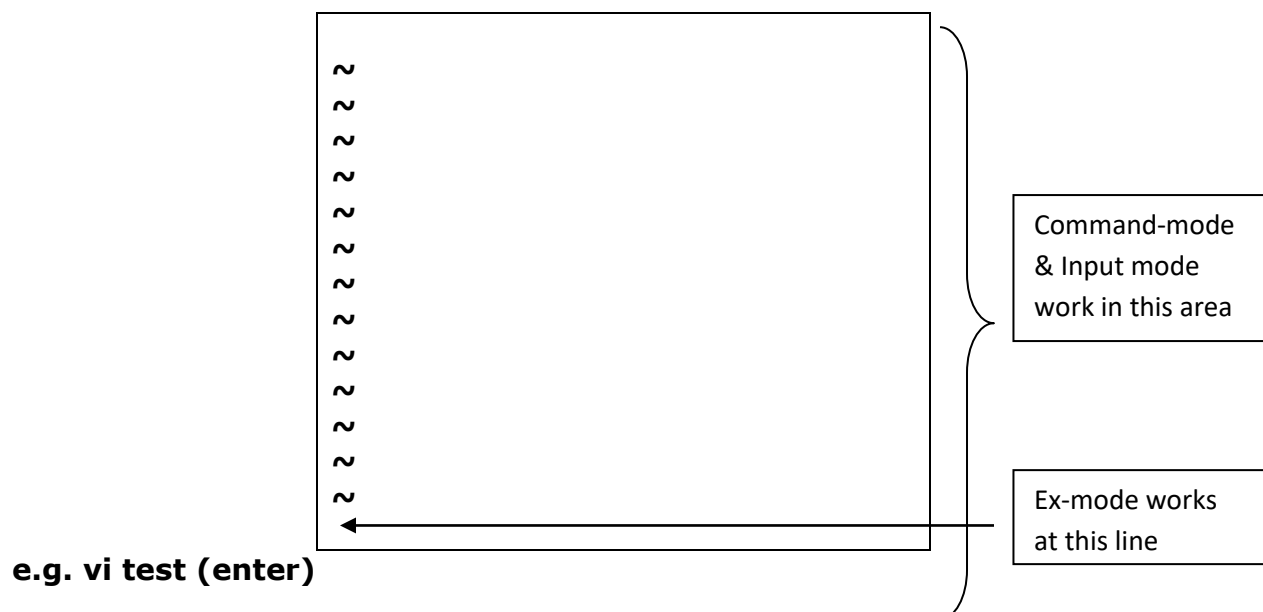
✓ Command mode

When user opens the editor he/she is in command mode. User can perform various editing operations in this mode.

Syntax (for invoking vi editor)

\$ vi (enter) [will open vi editor with a temporary file name]

\$ vi filename (enter) [will open vi editor with the given file name in the current directory]



After applying above command the vi editor gets open as shown in the figure above.

Input mode commands

- ✓ The vi editor when opens by default is in command mode.
- ✓ To switch over to the input mode the following commands are used.

- **I - Switches to input mode and inserts text at the starting position of the current line.**

e.g. This is just an example. ← The cursor denotes the current line
Esc + I

The cursor moves at starting place and text will be inserted there and existing text will be shifted to right.

- **i - Switches to input mode and inserts the text at the current cursor position**

e.g. This is just an example. ← The cursor denotes the current line
Esc + i

The text will be inserted here and existing text will be shifted to right. (i.e. the text will be inserted after space and "example" will be shifted to right.)

- **a - Appends text to right of cursor (Existing text shifted right)**

e.g. This is just an example. ← The cursor denotes the current line
Esc + a

The text will be inserted here and existing text will be shifted to right. (i.e. the text will be inserted after "e" and "xample" will be shifted to right.)

- **A - Appends text at the end of line**

e.g. This is just an example. ← The cursor denotes the current line
Esc + A

The text will be inserted here at the end of line.

- **O- Opens line above the current line**

←	The new line is added over here.
←	The cursor denotes the current line
←	

- **o - Opens line below the current line**

←	The cursor denotes the current line
←	The new line is added over here.
←	

- **r – Replaces single character with a single character**
 e.g. This is just an example. ←----- if "r" is pressed and then a is typed the "e" will be replaced by "a"
 output: **This is just an axample.**
When one uses the r command then after replacing the single character the mode is changed back to command mode again.
- **R – Replaces single character with multiple characters (the characters are overwritten).**
 e.g. This is just an example. ←----- if "r" is pressed and then a is typed the "e" will be replaced by "a"
 output: **This is just an axample.**
When one uses the r command then after replacing the single character the mode is changed back to command mode again.
- **s – Replaces single character with multiple characters (the characters are not overwritten).**
- **S – Replaces entire line**

Command mode commands

Navigation Commands

Movement in four directions

k- Moves cursor up (above line)

j - Moves cursor down (below line)

h- Moves cursor left (one character)

l- Moves cursor right (one character)

Repeat factor can be used with the navigation commands

e.g. 3l will take 3 character right

3k will take 3 lines up.

Word navigation commands

b-Moves back to beginning of the word

e-Moves forward to end of the word

w-Moves forward to beginning of word

Repeat factor can be used with the above commands to speed up the navigation

Note:

B,E, W also performs the similar functions except that punctuations are skipped.

Moving to line Extremes (0, | and \$)

0-will take you to the starting position of the current line

| - will take you to the starting position of the current line

If repeat factor is used with "|" then it will move that many number of columns.

e.g. 10| will move the cursor to the 10th column.

\$-will take you to the end of line

Scrolling

Ctrl f : Scrolls the screen(page) forward

Ctrl b : Scrolls the screen(page) backward

Ctrl d : Scrolls the half screen (page) in forward direction

Ctrl u : Scrolls the half screen (page) in forward direction

Note: user can use repeat factor with any of the above scrolling command.

e.g. 10 ctrl f will scroll 10 pages forward.

Absolute Movement (G)

The "G" command is used to move the cursor to the specific line.

e.g. 10G : will place the cursor at the 10th line

1G : will place the cursor at the 1st line

G : will place the cursor at the last line

Ctrl g – to know the line number of the current line

When ctrl g is pressed it displays the following message:

“

Editing Text

yy- yank (copy)

dd- delete

x & X

p- paste the text below(dd or yy) & right (x & X) the current line

P –paste the text above(dd or yy) & left (x & X)the current line

U- works on line,it helps to discard all the changes made in th

u

dw

d\$ or D

Changing the text (c)

c0 – will change from cursor to beginning of the line

c\$ or C – will change the text from cursor to the end of the line

3cw- will change three words

cc – will change the current line

Filtering Text (!)

!

!! -will work on the current line

Paste

Searching for a Pattern (/ or ?)

- ✓ Vi editor is very strong in searching and replacement.
- ✓ Searching can be done in forward as well as in backward direction and also can be repeated.
- ✓ It is applied at Command mode by pressing / which takes to the last line as given below:
 - **/pattern (enter)**
 - In a default case it searches in forward direction.
- ✓ The search begins forward to position the cursor on the first instance of the word.
- ✓ The entire file is searched; if end of file is reached the search is wrapped around the beginning of the file.
- ✓ If the search fails the corresponding message is displayed.
 - **(Using "?" command)**

- ✓ The above character is used to search in backward direction for the most previous instance of the pattern. The wraparound feature is applied here but in backward direction

- **? pattern (enter)**

❖ **Repeating the last pattern Search (n & N)**

- ✓ The n and N commands are used to repeat the last searched pattern.
- ✓ **n – is used to Repeat the pattern in the same direction of the original command**
 - i.e if the search is made with / then n will repeat the search in forward direction
 - if the search is made with ? then n will repeat the search in backward direction
- ✓ **N – is used to Repeat the pattern in the opposite direction of the original command'**
 - i.e if the search is made with / then n will repeat the search in backward direction
 - if the search is made with ? then n will repeat the search in forward direction

Note: We can combine / (search), n (repeat search) and . (repeat last command) to perform various operations.

If one wants to search any pattern when opening vi editor then one has to type the following command

\$ vi +/pattern Name of the file

❖ **Substitution – Search and Replace (:s)**

- ✓ The substitute is an ex mode command
- ✓ It lets one replace a pattern in the file with another text.
- ✓ Syntax:
 - :address/source_pattern/target_pattern/flags
 - E.g. :1,50s/test//g
The above command will replace all the occurrences of test from 1st to 50th line with a space.
- ✓ Interactive Substitution:
 - If the user wants to replace the string selectively then **c flag** with **g flag** is used.
 - All the lines are selected one by one in which pattern is found and the user is asked to substitute the pattern or not for each line.

❖ **The ex Mode Handling Multiple files**

- Switching files
 - :e filename (enter) – this command will open the new file(name given with the command) without closing the existing file.
 - e! – this command will reload the last saved version of the file which means if the file is modified since last open all the changes made to the file will be discarded.

- To move between open file **[Ctrl^]** and **:e#** commands are used.
- Opening multiple files with vi command
 - If one opens multiple file at a time with vi editor.
e.g. \$ vi file1 file2 file3 In that case one can move from one file to another file given at command line **:n** and **:rew** commands are used.
 - **:n – will move to next file in command line**
 - **:rew – will move to the first file in the command line.**

Note: while using any of this command if any of the file is not saved then it gives message before moving to the next file.

To move to the next file without saving the existing file, one can use **:e!#** or **:n!** or **:rew!** Commands.

- Inserting file and command output
- In vi editor one can insert some data of one file into another file without leaving the editor using **:r** command
 - :r note1 (enter) – will insert file note1**
 - :r !cat note1**
 - :r !date (enter) – will insert the output of the command date.**

❖ **Named buffers: storing multiple text sections**

Vi stores the deleted text in an unnamed buffer.

This technique has two limitations.

1. At a time only one buffer can be used.
2. When the contents of a file are moved from one file to another the contents of the buffer are lost.

Vi has 26 special buffers named after the letters of alphabet [a-z].

To use these buffers “ and the name of the buffer has to be preceded by the command.

e.g. “a4yy – Copies 4 lines into buffer a

“ap – Pastes the content of buffer a, below the current line.

One can also move text from one file to another using named buffer provided one has not to leave vi editor.

e.g. “a10yy Copies 10 lines into buffer a

:e file1 Switch to file file1

30G Move to line 30

“ap Put the 10 lines after line 10

❖ **Numbered Buffer**

vi stores up to last nine complete line deletions using numbered buffers(1 to 9).

These buffers work only on entire line when deleted and not on part of line.

e.g. “1p will restore most recent deletion

One can go on like this up to “9p.

We can use (.) command with the numbered buffer to repeat and use the next numbered buffer.

e.g. "1p will print the content in 1 buffer.

Then u.u.u. will paste the content of 2nd buffer then 3rd buffer and so on.

❖ **Entering control characters ([Ctrl-v])**

If in vi editor one wants to send escape sequences to printer or terminal in that case need to enter control characters.

Vi uses [ctrl-v] to precede any control character.

e.g. [ctrl-v] then press [ctrl-h] will print ^H

[ctrl-v] then press [Esc] will print ^[

❖ **Searching for a character**

- ✓ To search a specific character in vi editor **f** and **t** commands are used.
- ✓ The commands are used only with the current line. i.e. it finds the occurrence of the character in the current line only.
- ✓ The command can be used with repeat factor. Eg. 3fi – will place the cursor on the 3rd occurrence of the character i.

f – Moves the cursor to the first occurrence of the specific character.

e.g. fi (enter) will position the cursor at first occurrence of i

t – Moves the cursor a single character before the first occurrence of the specific character.

e.g. ti (enter) will position the cursor one step ahead at first occurrence of i

; - Repeats search in same direction along which previous search was made with f and t

, (comma) - Repeats search in a direction opposite to that along which previous search was made with f or t.

Note: F & T performs the same action but they work in reverse direction.

❖ **Marking Text**

One can mark up the position of any text in vi editor and can move to it later.

m : command is used to mark the text.

Label for the marking can be any of the character from **a-z** i.e. there are total 26 characters that can be used as label for marking text.

To move to the marked location **`** is used with the name of the label.

e.g. **ma** at certain location will mark the text at that location with the label a

To move to that location **`a** is used.

Note: To jump from one marked location to another marked location **`` (twice the single quote)** is used.

One can also use the **``** with G command. i.e. when one moves from current cursor location to some specific line using 20G and then G i.e. last line. Moving from line 20 to last and last to 20th one can use **``**

❖ **Customizing vi**

vi-editor provides facility to customize the functioning of keys, abbreviate frequently used words or phrases into short strings.

There are three ex Mode commands through which we can customize vi-editor.


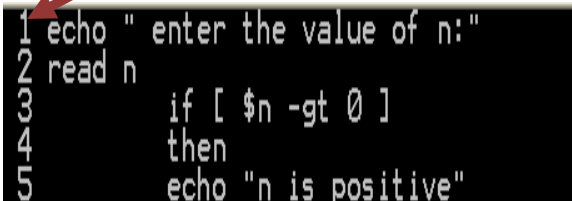
All the modifications are saved in `~/.exrc`. This file is read upon startup and its behavior is determined by statements placed in that file.

- **The set command**

The general vi environment is determined by its variable settings.

These variables are controlled by **:set** command.

Various variables used with set command are:

Command	Abbr.	Description
:set showmode		Displays message regarding the mode in which the user is working. Insert mode: 
:set autowrite	Aw	Writes(Saves) the current file automatically whenever user switches files with :n or :e#.
:set autoindent	Ai	Next line starts at previous indented level.
:set number	Nu	Displays line numbers on screen. Line numbers are not preserved with the file. i.e. line numbers are not saved with the file. Line numbers: 
:set nonumber		To remove the line numbers this option is used.
:set magic		Treats regular expression character special when searching for patterns. This is the default setting. While searching characters * , . are treated with their special meaning.
:set nomagic		This option despecialize the regular expression character when searching for a pattern.
:set showmatch	Sm	Shows momentarily match to a) and } When text in input mode is entered

		whenever the user types the } or) the cursor moves to its corresponding starting { or (for a moment and then comes back to the original cursor position.
:set tabstop	ts	The default tabstop is 8 spaces in vi-editor. To change it user needs to use the tabstop command E.g. :set tabstop=5
:set wrapscan	Ws	Continues the pattern search by moving to other end of file so that entire file is scanned. To avoid the wrapscan facility "nowrapscan" can be used.
:set ignorecase	Ic	Ignores case when searching the pattern.

- map: Mapping keys of keyboard
 - this command allows user to assign the undefined keys or reassign the defined ones so that when such key is pressed, it is expanded to a command sequence.
 - It is used like a macro.
 - Syntax:

:map key key sequence to be mapped

e.g. :map g :w^M

This command will assign the save procedure into "g". i.e. When the g key is pressed then file is saved.

Note: g,q,v,K,V and Z are undefined letters in vi editor.

- User can map keys in input mode also in which case press the mapped key while entering or changing text itself.
e.g. :map! #2 ^[:w^M will map F2 for saving the content of the file while in input mode.
 - the command mode map is canceled by :unmap and Input mode map is canceled by :unmap!.
 - The symbol :! Invokes a temporary shell escape which is used to run cc (compiler)
e.g. :map! #3 !cc %^M (% here means the current file)
:map! #3 !%^M
 - These are two important mappings used by the author.
- abbr: Abbreviating Text Input
The abbreviate command itself is abbreviated to "ab".
This command is used to expand short string to long words.

e.g. :ab re regular expression

:ab me metacharacter

When re or me is typed and the next key is neither alphanumeric or underscore then re is expanded to regular expression and me is expanded to metacharacter.