Chapter 20. Introduction to vi

The **vi** editor is installed on almost every Unix. Linux will very often install **vim** (**vi improved**) which is similar. Every system administrator should know **vi(m)**, because it is an easy tool to solve problems.

The **vi** editor is not intuitive, but once you get to know it, **vi** becomes a very powerful application. Most Linux distributions will include the **vimtutor** which is a 45 minute lesson in **vi(m)**.

20.1. command mode and insert mode

The vi editor starts in **command mode**. In command mode, you can type commands. Some commands will bring you into **insert mode**. In insert mode, you can type text. The **escape key** will return you to command mode.

Table 20.1. getting to command mode

key	action
Esc	set vi(m) in command mode.

20.2. start typing (a A i I o O)

The difference between a A i I o and O is the location where you can start typing. a will append after the current character and A will append at the end of the line. i will insert before the current character and I will insert at the beginning of the line. o will put you in a new line after the current line and O will put you in a new line before the current line.

Table 20.2. switch to insert mode

command	action
a	start typing after the current character
A	start typing at the end of the current line
i	start typing before the current character

I	start typing at the start of the current line
o	start typing on a new line after the current line
O	start typing on a new line before the current line

20.3. replace and delete a character (r x X)

When in command mode (it doesn't hurt to hit the escape key more than once) you can use the x key to delete the current character. The big X key (or shift x) will delete the character left of the cursor. Also when in command mode, you can use the r key to replace one single character. The r key will bring you in insert mode for just one key press, and will return you immediately to command mode.

Table 20.3. replace and delete

command	action
x	delete the character below the cursor
X	delete the character before the cursor
r	replace the character below the cursor
p	paste after the cursor (here the last deleted character)
xp	switch two characters

20.4. undo and repeat (u .)

When in command mode, you can undo your mistakes with u. You can do your mistakes twice with . (in other words, the . will repeat your last command).

Table 20.4. undo and repeat

command	action
u	undo the last action
.	repeat the last action

20.5. cut, copy and paste a line (dd yy p P)

When in command mode, dd will cut the current line. yy will copy the current line. You can paste the last copied or cut line after (p) or before (P) the current line.

Table 20.5. cut, copy and paste a line

command	action
dd	cut the current line
yy	(yank yank) copy the current line
p	paste after the current line
P	paste before the current line

20.6. cut, copy and paste lines (3dd 2yy)

When in command mode, before typing dd or yy, you can type a number to repeat the command a number of times. Thus, 5dd will cut 5 lines and 4yy will copy (yank) 4 lines. That last one will be noted by vi in the bottom left corner as "4 line yanked".

Table 20.6. cut, copy and paste lines

command	action
3dd	cut three lines
4yy	copy four lines

20.7. start and end of a line (0 or ^ and $)

When in command mode, the 0 and the caret ^ will bring you to the start of the current line, whereas the $ will put the cursor at the end of the current line. You can add 0 and $ to the d command, d0 will delete every character between the current character and the start of the line.

Likewise d$ will delete everything from the current character till the end of the line. Similarly y0 and y$ will yank till start and end of the current line.

Table 20.7. start and end of line

command	action
0	jump to start of current line
^	jump to start of current line
$	jump to end of current line
d0	delete until start of line
d$	delete until end of line

20.8. join two lines (J) and more

When in command mode, pressing **J** will append the next line to the current line. With **yyp** you duplicate a line and with **ddp** you switch two lines.

Table 20.8. join two lines

command	action
J	join two lines
yyp	duplicate a line
ddp	switch two lines

20.9. words (w b)

When in command mode, **w** will jump to the next word and **b** will move to the previous word. w and b can also be combined with d and y to copy and cut words (dw db yw yb).

Table 20.9. words

command	action
w	forward one word
b	back one word
3w	forward three words
dw	delete one word
yw	yank (copy) one word
5yb	yank five words back
7dw	delete seven words

20.10. save (or not) and exit (:w :q :q!)

Pressing the colon : will allow you to give instructions to vi (technically speaking, typing the colon will open the **ex** editor). **:w** will write (save) the file, **:q** will quit an unchanged file without saving, and **:q!** will quit vi discarding any changes. **:wq** will save and quit and is the same as typing **ZZ** in command mode.

Table 20.10. save and exit vi

command	action
:w	save (write)
:w fname	save as fname
:q	quit
:wq	save and quit
ZZ	save and quit
:q!	quit (discarding your changes)
:w!	save (and write to non-writable file!)

The last one is a bit special. With **:w! vi** will try to **chmod** the file to get write permission (this works when you are the owner) and will **chmod** it back when the write succeeds. This should always work when you are root (and the file system is writable).

20.11. Searching (/ ?)

When in command mode typing / will allow you to search in vi for strings (can be a regular expression). Typing /foo will do a forward search for the string foo and typing ?bar will do a backward search for bar.

Table 20.11. searching

command	action
/string	forward search for string
?string	backward search for string
n	go to next occurrence of search string
/^string	forward search string at beginning of line
/string$	forward search string at end of line
/br[aeio]l	search for bral brel bril and brol
/\<he\>	search for the word **he** (and not for **he**re or t**he**)

20.12. replace all (:1,$ s/foo/bar/g)

To replace all occurrences of the string foo with bar, first switch to ex mode with : . Then tell vi which lines to use, for example 1,$ will do the replace all from the first to the last line. You can write 1,5 to only process the first five lines. The s/foo/bar/g will replace all occurrences of foo with bar.

Table 20.12. replace

command	action
:4,8 s/foo/bar/g	replace foo with bar on lines 4 to 8

:1,$ s/foo/bar/g	replace foo with bar on all lines

20.13. reading files (:r :r !cmd)

When in command mode, :r foo will read the file named foo, :r !foo will execute the command foo. The result will be put at the current location. Thus :r !ls will put a listing of the current directory in your text file.

Table 20.13. read files and input

command	action
:r fname	(read) file fname and paste contents
:r !cmd	execute cmd and paste its output

20.14. text buffers

There are 36 buffers in vi to store text. You can use them with the **"** character.

Table 20.14. text buffers

command	action
"add	delete current line and put text in buffer a
"g7yy	copy seven lines into buffer g
"ap	paste from buffer a

20.15. multiple files

You can edit multiple files with vi. Here are some tips.

Table 20.15. multiple files

command	action
vi file1 file2 file3	start editing three files
:args	lists files and marks active file
:n	start editing the next file
:e	toggle with last edited file
:rew	rewind file pointer to first file

20.16. abbreviations

With **:ab** you can put abbreviations in vi. Use **:una** to undo the abbreviation.

Table 20.16. abbreviations

command	action
:ab str long string	abbreviate **str** to be 'long string'
:una str	un-abbreviate str

20.17. key mappings

Similarly to their abbreviations, you can use mappings with **:map** for command mode and **:map!** for insert mode.

This example shows how to set the F6 function key to toggle between **set number** and **set nonumber**. The <bar> separates the two commands, **set number!** toggles the state and **set number?** reports the current state.

```
:map <F6> :set number!<bar>set number?<CR>
```

20.18. setting options

Some options that you can set in vim.

```
:set number  ( also try :se nu )
:set nonumber
:syntax on
:syntax off
:set all  (list all options)
:set tabstop=8
:set tx   (CR/LF style endings)
:set notx
```

You can set these options (and much more) in **~/.vimrc** for vim or in **~/.exrc** for standard vi.

```
paul@barry:~$ cat ~/.vimrc
set number
set tabstop=8
set textwidth=78
map <F6> :set number!<bar>set number?<CR>
paul@barry:~$
```