

Linux Commands

4. File Editing Commands

a. You can use **head** to display the first ten lines of a file.

\$ head filename

The **head** command can also display the first **n** lines of a file.

\$ head -5 filename → will display first 5 lines of the specified file.

The **head** command can also display the first **n bytes** of a file.

\$ head -c20 filename → will display first 20 bytes of the specified file.

b. The **tail** command displays the last ten lines of a file.

\$ tail filename

Same as head command tail can display specific number of lines of a file.

\$ tail -17 filename → will display the last 17 lines of the specified file.

c. To display the contents of a file **cat** (concatenate) command is used.

\$ cat filename

\$ cat file1 file2 → will combine the contents of both the files and display on the screen.

You can use **cat** to create text files. Type the **cat > test1.txt** command. Then type one or more lines, finishing each line with the enter key. Once finished typing after the last line, hold the **Control (Ctrl)** key and press **d**.

\$ cat > test2.txt

\$ cat file1 file2 > file3 → will combine the contents of file1 and file2 and put in a new file named file3

\$ cat file4 > file5 → will copy the contents of file4 into file5

d. The **tac** command displays the last lines first (cat backwards). The following screen shot displays the difference.

```
admin@server1:~  
File Edit View Search Terminal Help  
[root@server1 ~]# cat file1  
This is first line.  
This is second line.  
This is third line.  
[root@server1 ~]# tac file1  
This is third line.  
This is second line.  
This is first line.  
[root@server1 ~]# █
```

e. The **more** command is useful for displaying files that take up more than one screen. **More** will allow you to see the contents of the file page by page. Use the space bar to see the next page, or q to quit. The **more** command allows scrolling in one direction only. Scroll back is not allowed.

```
$ cat file1 | more
```

f. The **less** command works same as more, but allows scrolling back and forth. Using up and down arrow keys you can scroll the lines back and forth. Thus **less** is preferred by most users.

```
$ cat file1 | less
```

Note:- The sign | (shift key and \ key) is known as pipe.

5. Some Useful commands

a. To find out whether a command given to the shell will be executed as an external command or as a built-in command, use the **type** command.

```
$ type ls
```

b. The **which** command will search for binaries in the \$PATH environment variable.

```
$ which ls → gives you the location from where this ls command is executed.
```

c. The shell allows you to create **aliases**. **Aliases** are often used to create an easier to remember name for an existing command or to easily supply parameters.

```
$ alias c=clear
```

```
$ c → now this will clear the terminal screen. You do not require to type entire clear command now.
```

d. You can provide one or more aliases as arguments to the alias command to get their definitions.

```
$ alias c → will display to which command c is mapped.
```

```
$ alias → Providing no arguments gives a complete list of current aliases.
```

e. You can undo an alias with the **unalias** command.

```
$ unalias c → will remove the mapping of c to the clear command.
```

f. To repeat the last command in bash, type **!!**. This is pronounced as bang bang.

```
$ !! → Will run the last command.
```

g. You can repeat other commands using one bang followed by one or more characters. The shell will repeat the last command that started with those characters.

```
$ !ca → will run the last cat command.
```

```
$ !to → will run the last touch command.
```

h. To see older commands, use **history** to display the shell command history.

\$ history

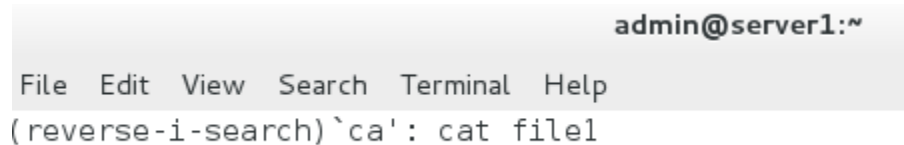
Use **history n** to see the last **n** commands

\$ history 15 → will display last 15 commands

By typing **!** followed by the number preceding the command you want to repeat, then the shell will echo the command and execute it.

\$!15 → will execute the command on number 15 in the output of the history command.

i. Use **ctrl-r** to search in the history. Type **ctrl-r** followed by the characters in the command and it finds the last command containing these consecutive characters. The following screen shot displays the output of **ctrl-r**.



The screenshot shows a terminal window with the prompt **admin@server1:~**. Below the prompt is a menu bar with the options **File Edit View Search Terminal Help**. The terminal content shows the command **(reverse-i-search)`ca': cat file1**, which is the result of searching for 'ca' in the command history.

j. The **find** command can be very useful to search for files.

\$ find / -name file1 → will search for the file named file1 in the entire file system.

\$ find /data -name file1 → will search for the file named file1 in the /data directory only.

\$ find . -name test1.txt → will search file named test1.txt in the current directory.

\$ find / -type d -name data → will search only for directory named data below /.

\$ find /data --newer file1 → will display all the files in the /data directory that are newer than file1.

k. The **locate** tool is very different from **find** in that it uses an index to locate files. This is a lot faster than traversing all the directories, but it also means that it is always outdated. If the index does not exist yet, then you have to create it (as root on Red Hat Enterprise Linux) with the **updatedb** command.

updatedb → # denotes root login

locate filename

l. The **date** command can display the date, time, time zone and more.

\$ date

m. The **cal** command displays the current month, with the current day highlighted.

\$ cal

\$ cal 1 2016 → will display the calendar for the month of January of year 2016.

n. The **grep** filter is very useful. The most common use of **grep** is to filter lines of text containing (or not containing) a certain string.

\$ cat filename | grep string → will display the lines which contain the word identified by the string.

\$ grep string filename → displays the same output as above.

One of the most useful options of **grep** is **grep -i** which filters in a case insensitive way.

\$ grep -i test filename → will display all the lines which contain test word written in any format like Test, TEST, TeSt etc.

Another very useful option is **grep -v** which outputs lines not matching the string.

\$ grep -v test filename → will display all the line which do not contain the word test.

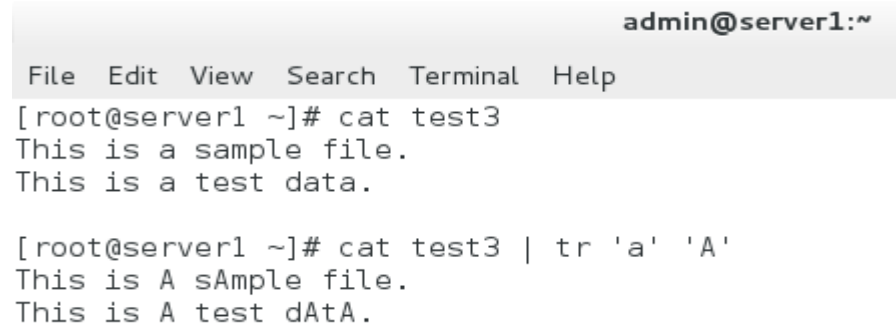
With **grep -A1** one line after the result is also displayed.

With **grep -B1** one line before the result is also displayed.

With **grep -C1** one line before and one after are also displayed.

All three options (A,B, and C) can display any number of lines (using e.g. A2, B4 or C20).

o. You can translate characters with **tr**. The following screen shot displays the effect of **tr**.



```
admin@server1:~
File Edit View Search Terminal Help
[root@server1 ~]# cat test3
This is a sample file.
This is a test data.

[root@server1 ~]# cat test3 | tr 'a' 'A'
This is A sAmple file.
This is A test dAtA.
```

\$ cat test1.txt | tr 'a-z' 'A-Z' → will convert all a-z small case characters to uppercase.

\$ cat test1.txt | tr -d e → will remove(delete) the character **e** from the output.

p. **wc** makes counting words, lines and characters easy.

\$ wc -l filename → will display number of lines in the specified file.

\$ wc -w filename → will display the number of words within the file.

\$ wc -c filename → will display total number of characters within the file.

q. The **sort** filter will default to an alphabetical sort.

\$ sort filename → will sort the file based on first characters of the file.

\$ sort -k1 filename → will sort the output according to column 1

\$ sort -k2 filename → will sort the output according to column 2

\$ sort -n -k5 filename → will perform a numerical sort instead of alphabetical.

r. With **uniq** you can remove duplicates from a sorted list. The following screen shot displays the result of **uniq**.

```
File Edit View Search Terminal Help
[root@server1 ~]# cat test5
Sachin
Rohan
Ajay
Sachin
Dinesh
Rohan
[root@server1 ~]# sort test5
Ajay
Dinesh
Rohan
Rohan
Sachin
Sachin
[root@server1 ~]# sort test5 | uniq
Ajay
Dinesh
Rohan
Sachin
```

uniq can also count occurrences with the **-c** option

```
admin@server1:~
File Edit View Search Terminal Help
[root@server1 ~]# sort test5 | uniq -c
  1 Ajay
  1 Dinesh
  2 Rohan
  2 Sachin
[root@server1 ~]# █
```

s. Comparing streams or files can be done with the **comm**. By default **comm** will output three columns. The first column displays the unique contents of the first file. The second column displays the unique contents of the second file. The third column displays the common contents of both files.

```
admin@server1:~
File Edit View Search Terminal Help
[root@server1 ~]# comm test5 test6
          Ajay
Dinesh
Girish
Jack
Mahesh
      Rakesh
      Rohan
      Sachin
      Virat
```