Basic Linux Commands

Srihari Kalgi
M.Tech, CSE (KReSIT),
IIT Bombay

May 5, 2009

Calender

- **cal**: Command to see calender for any specific month or a complete year
  - cal [[month] year]

```
$ cal april 2009
      April 2009
Su Mo Tu We Th Fr Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30
```

date

- **date**: displays the current date

```
$ date
Tue Apr 21 21:33:49 IST 2009
kuteer$ date +"%D %H:%M:%S"
04/21/09 21:35:02
```

- Options:
  - d - The da of the month (1-31)
  - y - The last two digits of the year
  - H,M,S - Hour Minute and second respectively
  - D - the date in mm/dd/yy
- For more information see **man date**

echo and printf

- **echo**: Print message on the terminal
- usage: echo “<message>”

```
$ echo "Welcome to the workshop"
Welcome to the workshop
```

- **printf**: Print the formatted message on the terminal
- Syntax of printf is same as C language printf statement
- usage: printf “<formatted message”

```
$ printf "the amount is %d\n" 100
the amount is 100
```

Calculator

- **bc**: A text based calculator

  ```
  $ bc
  2*10+20-9+4/2 [Input]
  33 [Output]
  [ctrl+d] [Quit]
  ```

- **xcalc** is graphical based calculator

script: Record your session

- **script** command records your session and stores it in a file

```
$ script
Script started, file is typescript
$ echo "this is a sample script"
this is a sample script
$ [ctrl+d]
Script done, file is typescript
```

- By default if you dont specify any file name the contents will be stored in file name **typescipt**

```
 $ cat typescript
Script started on Tuesday 21 April 2009 10:07:0
$ echo "this is a sample script"
this is a sample script
$
Script done on Tuesday 21 April 2009 10:07:34 P
```

passwd: Changing your password

- **passwd** command allows you to change your password

```
 kuteer:~/workshop$ passwd
Changing password for srihari.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

WHO: Who are the users?

- **who** command tells you the users currently logged on to the system

```
 kuteer:~$ who
srihari pts/0 2009-04-15 11:58 (:10.129.41.3)
nithin pts/1  2009-04-15 16:09 (:10.129.20.5)
avadhut pts/2 2009-04-13 14:39 (:10.129.45.20)
anil pts/3    2009-04-13 16:32 (:10.129.23.45)
```

man - The reference Manual

- **man** displays the documentation for a command
- usage: man <command name>

```
        ls - list directory contents
SYNOPSIS
        ls [OPTION]... [FILE]...
DESCRIPTION
        List  information  about  the  FILEs (the
        none of -cftuvSUX nor --sort.
```

Linux file system

- Standard directory structure
  - / - the topmost
  - /dev - all the devices are accessible as files
  - /var - “variable” data such as mails, log files, databases
  - /usr - almost all the packages installed
  - /etc - configuration files
  - /home - home directories for all the users
  - /root - home directory of the privileged user root
  - /mnt - used to mount other directories/partitions.

File Attributes

- To see the file attributes type **ls -l** on your terminal

```
kuteer:˜$ ls -l
$<$permissions$>$ $<$owner$>$ $<$group$>$
drwxr-xr-x  2 srihari srihari      144 2009-04-0
-rw-r--r--  1 srihari srihari     1548 2009-03-2
drwxr-xr-x  2 srihari srihari       48 2009-03-1
-rw-r--r--  1 srihari srihari     3570 2009-03-2
```

- The file Testing.java has the following permissions **-rw-r–r–**
- It has 10 characters, first character is **d** if its directory and **-** if its file.
- Next 9 characters are divided into three groups with a set of 3 characters each

File Attributes Contd. . .

- First 3 characters - Owner of the file or directory
- Next 3 characters - Group
- Last 3 characters - Others
- **r** - Read i.e. File or directory is readable
- **w** - Write i.e. File or directory is writable
- **x** - Execute i.e. File or directory is executable
- **-rw-r–r–** means it has read, write but not execute permissions for the owner of the file, only read permissions for the group and only read permissions for others

File Attributes Contd. . .

- The third column of the command **ls -l** tells about the owner of the file, next column tells to which group it belongs

```
-rw-r--r--  1 srihari srihari     3570 2009-03-
```

- The file Testing.java has the owner as srihari and also belongs to a group called srihari

Changing the File attributes

- **chmod** Changing the permissions of the file

```
kuteer:~$ chmod o+x Testing.java
kuteer:~$ ls -l Testing.java
-rw-r--r-x 1 srihari srihari 3570 2009-03-23 10:
kuteer:~$ chmod 655 Testing.java
kuteer:~$ ls -l Testing.java
-rw-r-xr-x 1 srihari srihari 3570 2009-03-23 10:
```

Changing ownership

- **chown** command is used for changing the ownership and also group of the file

```
 kuteer:~$ chown guest Testing.java
kuteer:~$ ls -l Testing.java
-rw-r-xr-x 1 geust srihari 3570 2009-03-23 10:52
 kuteer:~$ chown guest:guest Testing.java
kuteer:~$ ls -l Testing.java
-rw-r-xr-x 1 geust guest 3570 2009-03-23 10:52 T
```

File system commands

- Deleting Files - rm
- Copying and moving files - cp, mv
- Creating directories - mkdir
- Deleting Empty Directory - rmdir

```
$ rm Testing.java
//deletes the file Testing.java
$ cp Testing.java Copy.java
//creates the copy of Testing.java
$ mv Testing.java Test.java
//renames the file Testing.java to Test.java
$ mkdir newDir
//Creates directory newDir
$ rmdir newDir
//deletes directory newDir newDir should be empt
```

cat : Concatenate Files

- **cat** command is used to display the contents of a small file on terminal
- usage: **cat** <file_name>

```
$ cat sample3.txt
Unix (officially trademarked as UNIX, sometimes
......
```

- **cat** when supplied with more than one file will concatenate the files without any header information

```
 $ cat sample3.txt sample4.txt
/*contents of sameple3.txt*/
/*Followed by contents of sample4.txt without an
```

tac : concatenate files in reverse

- **tac** command is used to display the contents of a small file in reverse order on terminal
- usage: tac <file_name>

```
 $ tac sample3.txt
/*displays sample3.txt in reverse order*/
```

- **tac** when supplied with more than one file will concatenate the reverse contents of files without any header information

```
 $ tac sample3.txt sample4.txt
/*print sample3.txt in reverse order*/
/*print sample4.txt in reverse order without any
```

more, less : paging output

- **more** and **less** commands are used to view large files one page at a time
- usage: more <file_name>
- usage: less <file_name>

```
 $ more sample1.txt
/*sample1.txt will be displayed one page
at a time */
 $ less sample1.txt
/*sample1.txt will be displayed one page
at a time */
```

- **less** is the standard pager for linux and in general **less** is more powerful than **more**

wc : statistic of file

- **wc** command is used to count lines, words and characters, depending on the option used.
- usage: **wc** [options] [file_name]

```
$ wc sample1.txt
  65  2776 17333 sample1.txt
```

- Which means sample1.txt file has 65 lines, 2776 words, and 17333 characters
- you can just print number of lines, number of words or number of charcters by using following options:
  - -l : Number of lines
  - -w : Number of words
  - -c : Number of characters

cmp: comparing two files

- **cmp** command is used to compare two files whether they are identical or not
- usage: cmp <file1> <file2>
- The two files are compared byte by byte and the location of the first mismatch is printed on the screen
- If two files are identical, then it doesnot print anything on the screen

```
 $ cmp sample1.txt sample2.txt
sample1.txt sample2.txt differ: byte 1, line 1
 $ cmp sample1.txt sample1_copy.txt
 $ /*No output prompt returns back*/
```

comm : what is common?

- **comm** command displays what is common between both the files
- usage: **comm** <file1> <file2>
- The input files to **comm** command should be sorted alphabetically

```
 $ comm sample5.txt sample6.txt
                anil
        barun
                dasgupta
        lalit
                shukla
singhvi
sumit
```

comm: contd. . .

- Column 1 gives the names which are present in sample5.txt but not in sample6.txt
- Column 2 gives the names which are not present in sample5.txt but present in sample6.txt
- Column 3 gives the names which are present in both the files

gzip and gunzip

- **gzip** command is used to compress the file, and **gunzip** is used to de-compress it.
- usage: gzip <file_name>
- It provides the extension .gz and removes the original file

```
$ wc sample_copy.txt
   65  2776 17333 sample_copy.txt
$ gzip sample_copy.txt
$ wc sample_copy.txt.gz
  26  155 7095 sample_copy.txt.gz
```

- The compression ratio depends on the type, size and nature of the file
- usage: gunzip <file_name_with.gz>

```
$ gunzip sample_copy.txt.gz
$ /*do ls and you can see the original file*/
```

- If you want to compress the directory contents recursively, use **-r** option with **gzip** command and unzip it use the same option with **gunzip** command

tar : The archival program

- **tar** command is used to create archive that contains a group or file or entire directory structure.
- It is generally used for back ups.
- usage: **tar** [options] <output_file.tar> <file1 or dir> ...
- The following are the options:
  - -c Create an archive
  - -x Extract files from archive
  - -t Display files in archive
  - -f `arch` Name the archive `arch`

```
 $ tar -cvf compression.tar compression
compression/                    //v for verbose
compression/temp/
compression/temp/sample2.txt
compression/sample1.txt
```

tar contd. . .

- We can use **tar** and **gzip** command in succession to compress the tar file.

```
 $ tar -cvf compression.tar compression
 $ gzip compression.tar
 $ //will create compression.tar.gz file
```

- For un-compression the file first use **gunzip** command, which will create a tar file and then use **tar** command to untar the contents

```
 $ gunzip compression.tar.gz
 $ tar -xvf compression.tar
```

- To just view the contents of the tar file use **-t** option

```
 $ tar -tvf compression.tar
$ tar -tvf compression.tar
drwxr-xr-x srihari/srihari   0 2009-04-22 11:29
drwxr-xr-x srihari/srihari   0 2009-04-22 11:29
-rw-r--r-- srihari/srihari 17663 2009-04-22 11:2
-rw-r--r-- srihari/srihari 17333 2009-04-22 11:2
```

tar contd. . .

- Instead of doing **tar** first and then **gzip** next, we can combine both of them using the option **-z**

```
$ tar -cvzf compression.tar.gz compression
compression/
compression/temp/
compression/temp/sample2.txt
compression/sample1.txt
```

- We can de-compress .tar.gz agin in a single command using the option **-z** with **-x**

```
 $ tar -xvzf compression.tar.gz
```

zip and unzip: compressing and archiving

- **zip** command can be used for archiving as well as compressing the contents of the directory or the file
- usage: zip [options] output.zip <files_to_be_zipped or directory>

```
$ zip sample1.zip sample1.txt
//will create sample1.zip file
```

- Use **-r** option to recursively zip the contents of the directory

```
 $ zip -r compression.zip compression
// will create compression.zip file
```

- To un-compress the file use **unzip** command

```
 $ unzip compression.zip
// will uncompress the compression.zip file
```

Filters

- Filters are commands which accept data from standard input, manupulate it and write the results to standard output
- **head** command displays the top of the file, when used without any option it will display first 10 lines of the file

```
 $ head sample1.txt
/*display first 10 lines*/
```

- Similarly **tail** command displays the end of the file. By default it will display last 10 lines of the file

```
 $ tail sample1.txt
/*display last 10 lines*/
```

- **tail** or **head** with **-n** followed by a number will display that many number of lines from last and from first respectively

```
 $ head -n 20 sample1.txt
/* will display first 20 lines*/
 $ tail -n 15 sample1.txt
/* will display last 15 lines */
```

cut : cutting columns

- **cut** command can be used to cut the columns from a file with **-c** option
- usage: cut -c [numbers delemited by comma or range] <file_name>

```
 $ cut -c 1,2,3-5 students.txt
1 ani
2 das
3 shu
4 sin
```

cut : cutting fields

- With **-f** option you can cut the feilds delemited by some character

```
 $ cut -d" " -f1,4 students.txt
1 Mtech
2 Btech
3 Mtech
```

- **-d** option is used to specify the delimiter and **-f** option used to specify the feild number

paste : pasting side by side

- **paste** command will paste the contents of the file side by side

```
 $ paste cutlist1.txt cutlist2.txt
1 Mtech 1 anil H1
2 Btech 2 dasgupta H4
3 Mtech 3 shukla H7
4 Mtech 4 singhvi H12
5 Btech 5 sumit H13
```

sort : ordering a file

- sort re-orders lines in ASCII collating sequences-whitespaces first, then numerals, uppercase and finally lowercase
- you can sort the file based on a field by using **-t** and **-k** option.

```
 $ sort -t" " -k 2 students.txt
/* sorts the file based on the second field
using the delimiter as space*/
```

grep : searching for a pattern

- grep scans its input for a pattern, and can display the selected pattern, the line numbers or the filename where the pattern occurs.
- usage: **grep** *options pattern filename(s)*