Syntax

The commands in Linux have the following syntax:

```
$command options arguments
```

Linux Basic Commands

Let’s start with some simple commands.

1) pwd command

‘pwd’ command prints the absolute path to current working directory.

```
$ pwd
/home/raghu
```

2) cal command

Displays the calendar of the current month.

```
$ cal
July 2012
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 31
```

‘cal ’ will display calendar for the specified month and year.

```
$ cal 08 1991
August 1991
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 31
```

3) echo command

This command will echo whatever you provide it.

```
$ echo "linoxide.com"
linoxide.com
```

The ‘echo’ command is used to display the values of a variable. One such variable is ‘HOME’. To check the value of a variable precede the variable with a $ sign.

```
$ echo $HOME
/home/raghu
```

4) date command

Displays current time and date.

```
$ date
Fri Jul 6 01:07:09 IST 2012
```

If you are interested only in time, you can use 'date +%T' (in hh:mm:ss):

```
$ date +%T
01:13:14
```

6) whoami command

This command reveals the user who is currently logged in.

```
$ whoami
raghu
```

7) id command

This command prints user and groups (UID and GID) of the current user.

```
$ id
uid=1000(raghu) gid=1000(raghu) groups=1000(raghu),4(adm),20(dialout),24(cdrom),46
(plugdev),112(lpadmin),120(admin),122(sambashare)
```

By default, information about the current user is displayed. If another username is provided as an argument, information about that user will be printed:

```
$ id root
uid=0(root) gid=0(root) groups=0(root)
```

8) clear command

This command clears the screen.

Help command

Nobody can remember all the commands. We can use help option from command like

9) help option

With almost every command, '--help' option shows usage summary for that command.

```
$ date --help
Usage: date [OPTION]... [+FORMAT]
or: date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
Display the current time in the given FORMAT, or set the system date.
```

10) whatis command

This command gives a one line description about the command. It can be used as a quick reference for any command.

```
$ whatis date
date (1) - print or set the system date and time
$ whatis whatis
whatis (1) - display manual page descriptions
```

11) Manual Pages

‘--help’ option and ‘whatis’ command do not provide thorough information about the command. For more detailed information, Linux provides man pages and info pages. To see a command's manual page, man command is used.

```
$ man date
```

The man pages are properly documented pages. They have following sections:

NAME: The name and one line description of the command.

SYNOPSIS: The command syntax.

DESCRIPTION: Detailed description about what a command does.

OPTIONS: A list and description of all of the command's options.

EXAMPLES: Examples of command usage.

FILES: Any file associated with the command.

AUTHOR: Author of the man page

REPORTING BUGS: Link of website or mail-id where you can report any bug.

Linux Filesystem commands

12) Changing Directories Command

```
$ cd [path-to-directory]
```

Change the current working directory to the directory provided as argument. If no argument is given to ‘cd’, it changes the directory to the user's home directory. The directory path can be an absolute path or relative to current directory. The absolute path always starts with /. The current directory can be checked with ‘pwd’ command (remember?):

```
$ pwd
/home/raghu
$ cd /usr/share/
$ pwd
/usr/share
$ cd doc
$ pwd
/usr/share/doc
$ cd ..
$ pwd
$ /usr/share
$ cd                    ------ to go to home dir
$ cd -                   --------------to go to pervious working dir
```

In the first ‘cd’ command, absolute path (/usr/share) is used, and with second command, relative path (doc) is used.

14) Listing File And Directories Command

```
$ ls [files-or-directories]
```

List files and/or directories. If no argument is given, the contents of current directory are shown.

```
$ ls
example file1.txt file2.txt file3.txt
```

If a directory is given as an argument, files and directories in that directory are shown.

```
$ ls /usr
bin games include lib lib64 local sbin share src
```

‘ls -l’ displays a long listing of the files.

```
$ ls -l
total 4
drwxr-xr-x 2 raghu raghu 4096 2012-07-06 12:52 example
-rw-r--r-- 1 raghu raghu 0 2012-07-06 12:52 file1.txt

-rw-r--r-- 1 raghu raghu 0 2012-07-06 12:52 file2.txt
-rw-r--r-- 1 raghu raghu 0 2012-07-06 12:52 file3.txt
```

In this long listing, the first character is 'd' or '-'. It distinguishes between file types. The entries with a '-' (dash) are regular files, and ones with 'd' are directories. The next 9 characters are permissions ('rwxr-xr-x' in first listing). The number following the permissions is the link count. Link count follows user and group owner. In the above example, the file owner is 'raghu' and group owner is 'raghu' as well. Next is the size of the file. And then time stamp before the name of file (or directory).

By default, hidden files or directories are not shown, to see hidden files as well, -a option is used. Hidden files in Linux start with a period sign (.). Any file that starts with a period is hidden. So, to hide a file, you just need to rename it (and put a period before it).

```
$ ls -la odesk
total 16
drwxr-xr-x 4 raghu raghu 4096 2012-07-06 13:46 .
drwxr-xr-x 11 raghu raghu 4096 2012-07-06 13:15 ..
drwxr-xr-x 2 raghu raghu 4096 2012-07-06 12:52 example
-rw-r--r-- 1 raghu raghu 0 2012-07-06 12:52 file1.txt
-rw-r--r-- 1 raghu raghu 0 2012-07-06 12:52 file2.txt
-rw-r--r-- 1 raghu raghu 0 2012-07-06 12:52 file3.txt
drwxr-xr-x 2 raghu raghu 4096 2012-07-06 13:46 .hiddendir
-rw-r--r-- 1 raghu raghu 0 2012-07-06 13:46 .hiddenfile1.txt
-rw-r--r-- 1 raghu raghu 0 2012-07-06 13:46 .hiddenfile2.txt
```

If you want to see the properties of a directory instead of the files contained in it, use -d (with -l) option:

```
$ ls -ld odesk/
drwxr-xr-x 4 raghu raghu 4096 2012-07-06 13:46 odesk/
```

Creating files and directories

15) mkdir command

To create a directory, the ‘mkdir’ command is used.

```
$ mkdir example
$ ls -l
total 4
drwxr-xr-x 2 raghu raghu 4096 2012-07-06 14:09 example
```

16) touch command

For creating an empty file, use the touch command.

```
$ touch file1 file2 file3
$ ls -l
total 4
drwxr-xr-x 2 raghu raghu 4096 2012-07-06 14:09 example
-rw-r--r-- 1 raghu raghu 0 2012-07-06 14:20 file1
-rw-r--r-- 1 raghu raghu 0 2012-07-06 14:20 file2
-rw-r--r-- 1 raghu raghu 0 2012-07-06 14:20 file3
```

If a file already exists, touch will update its time stamp. There are a lot of other methods to create a new file, e.g. using a text editor like vi or gedit, or using redirection. Here is an example of creating a file using redirection:

```
$ ls -l /usr > usrlisting
$ ls -l
total 8
drwxr-xr-x 2 raghu raghu 4096 2012-07-06 14:09 example
-rw-r--r-- 1 raghu raghu 0 2012-07-06 14:20 file1
-rw-r--r-- 1 raghu raghu 0 2012-07-06 14:20 file2
-rw-r--r-- 1 raghu raghu 0 2012-07-06 14:20 file3
-rw-r--r-- 1 raghu raghu 491 2012-07-06 14:23 usrlisting
```

A file named usrlisting is created in this example.

Nano command

To create file

```
$ nano f1

$ press ctrl+o to write

Then enter

Then press
```

Vim command

To start using vim, just run the "vim" command on the Linux shell followed by the path of the file that you want to edit.
Example, editing of the file /etc/hosts

```
vim /etc/hosts
```

The result will look like this:

The editor is now in command mode. To start editing the file content, enter:

```
:i[enter]
```

[enter] means to press the return or enter key on your keyboard.
The word --insert-- will appear at the bottom of the editor window to show that you are in insert mode now.

Now you can edit the file by navigating to the line that you want to change with the cursor keys and then start typing the text. When you are finished with editing, press the [esc] key to go back to the command mode.
To save the file and exit the editor, enter:

```
:wq![return]
```

In case you want to quit vim without saving the file, enter:

```
:q![return]
```

Vim Command Reference

save: **:w**
save and exit: **:wq**
exit: **:q**
force: **!** (example **:w! :q!**)
vertical split: open a document and then type :vsplit /path-to-document/document and this will open the specified document and split the screen so you can see both documents.
copy: **y**
copy a line: **yy**
paste: **p**
cut: **d**
cut a line: **dd**

Copy, move and remove commands

17) copy command

```
$cp source destination
```

Copy files and directories. If the source is a file, and the destination (file) name does not exit, then source is copied with new name i.e. with the name provided as the destination.

```
$ cp usrlisting listing_copy.txt
$ ls -l
total 12
drwxr-xr-x 2 raghu raghu 4096 2012-07-06 14:09 example
-rw-r--r-- 1 raghu raghu 0 2012-07-06 14:20 file1
-rw-r--r-- 1 raghu raghu 0 2012-07-06 14:20 file2
-rw-r--r-- 1 raghu raghu 0 2012-07-06 14:20 file3
-rw-r--r-- 1 raghu raghu 491 2012-07-06 16:02 listing_copy.txt
-rw-r--r-- 1 raghu raghu 491 2012-07-06 14:23 usrlisting
```

If the destination is a directory, then the file is copied with its original name in that directory.

```
$ cp listing_copy.txt example/
$ ls -l example/
total 4
-rw-r--r-- 1 raghu raghu 491 2012-07-06 16:07 listing_copy.txt
```

Multiple files can also be copied, but in that case, the last argument will be expected to be a directory where all the files are to be copied. And the rest of the arguments will be treated as file names.

```
$ cp file1 file2 example/
$ ls -l example/
total 4
-rw-r--r-- 1 raghu raghu 0 2012-07-06 16:10 file1
-rw-r--r-- 1 raghu raghu 0 2012-07-06 16:10 file2
-rw-r--r-- 1 raghu raghu 491 2012-07-06 16:07 listing_copy.txt
```

If a directory is to be copied, then it must be copied recursively with the files contained in it. To copy a directory recursively, use -r option with ‘cp’ command:

```
$ cp -r example /tmp/expertslogin/
$ ls -l /tmp/expertslogin
total 4
```

```
drwxr-xr-x 2 raghu raghu 4096 2012-07-06 16:12 example
```

18) move command

```
$ mv source destination
```

Move files or directories. The 'mv' command works like 'cp' command, except that the original file is removed. But, the mv command can be used to rename the files (or directories).

```
$ mv listing_copy.txt usrcopy
$ ls -l
total 12
drwxr-xr-x 2 raghu raghu 4096 2012-07-06 16:10 example
-rw-r--r-- 1 raghu raghu 0 2012-07-06 14:20 file1
-rw-r--r-- 1 raghu raghu 0 2012-07-06 14:20 file2
-rw-r--r-- 1 raghu raghu 0 2012-07-06 14:20 file3
-rw-r--r-- 1 raghu raghu 491 2012-07-06 16:02 usrcopy
-rw-r--r-- 1 raghu raghu 491 2012-07-06 14:23 usrlisting
```

Here, 'listing_copy.txt' is moved with the name 'usrcopy' in the same directory (or you can say that it has been renamed).

19) To remove or Delete

```
$ rmdir
```

'rmdir' command removes any empty directories, but cannot delete a directory if a file is present in it. To use ‘rmdir’ command, you must first remove all the files present in the directory you wish to remove (and possibly directories if any).

To remove files and directories

```
$ rm files|directories
```

A directory must be removed recursively with -r option.

```
$ rm file2
$ rm -r example/
$ ls -l
```

```
total 8
-rw-r--r-- 1 raghu raghu 0 2012-07-06 14:20 file1
-rw-r--r-- 1 raghu raghu 0 2012-07-06 14:20 file3
-rw-r--r-- 1 raghu raghu 491 2012-07-06 16:02 usrcopy
-rw-r--r-- 1 raghu raghu 491 2012-07-06 14:23 usrlisting
```

Here, the file named 'file2' is removed first, and then the directory 'example' is removed recursively. This can be seen in the output of ‘ls -l’ command where these two are no longer present.

Other file commands

20) file command

The file command determines the file type of a given file. For example:

```
$ file /etc/passwd
/etc/passwd: ASCII text
```

You can provide one or more than one file as an argument to the file command.

```
$ file td.c td.out ARP.java Screenshot.png StringTokenizing.class
idl.rar List.pdf
td.c: ASCII C program text, with CRLF line terminators
td.out: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically link
ed (uses shared libs), for GNU/Linux 2.6.15, not stripped
ARP.java: ASCII Java program text, with CRLF line terminators
Screenshot.png: PNG image data, 1366 x 768, 8-bit/color RGB, non-interlaced
StringTokenizing.class: compiled Java class data, version 50.0 (Java 1.6)
idl.rar: RAR archive data, v1d, os: Win32
List.pdf: PDF document, version 1.4
```

22) cat command

The 'cat' command is actually a concatenator but can be used to view the contents of a file.

```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
```

24) head command

Displays the first few lines of a file. By default, the ‘head’ command displays the first 10 lines of a file. But with -n option, the number of lines to be viewed can be specified.

```
$ head /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
```

25) tail command

Similar to ‘head’; the ‘tail’ command shows the last 10 lines by default, and -n option is available as well.

```
$ tail -n 4 /etc/passwd
raghu:x:1000:1000:Raghu Sharma,,,:/home/raghu:/bin/bash
sshd:x:113:65534::/var/run/sshd:/usr/sbin/nologin
dictd:x:114:123:Dictd Server,,,:/var/lib/dictd:/bin/false
mysql:x:115:124:MySQL Server,,,:/nonexistent:/bin/false
```

26) wc command

Word count

This command counts lines, words and letters of the input given to it.

```
$ wc /etc/passwd
35 57 1698 /etc/passwd
```

The /etc/passwd file has 35 lines, 57 words, and 1698 letters present in it.

27) grep command

The ‘grep’ command searches for a pattern in a file (or standard input). It supports regular expressions. It returns a line if it matches the pattern in

that line. So, if we wish to find the lines containing the word ‘nologin’, we use ‘grep’ as follows:

```
$ grep nologin /etc/passwd
sshd:x:113:65534::/var/run/sshd:/usr/sbin/nologin
```

28) ln command

The ln command is used in linux to create links. Links are a kind of shortcuts to other files. The general form of command is:

```
$ ln TARGET LINK_NAME
```

There are two types of links, soft links and hard links. By default, hard links are created. If you want to create soft link, use -s option. In this example, both types of links are created for the file usrlisting.

```
$ ln usrlisting hard_link
$ ln -s usrlisting soft_link
$ ls -l
total 12
-rw-r--r-- 1 raghu raghu 0 2012-07-06 14:20 file1
-rw-r--r-- 1 raghu raghu 0 2012-07-06 14:20 file3
-rw-r--r-- 2 raghu raghu 491 2012-07-06 14:23 hard_link
lrwxrwxrwx 1 raghu raghu 10 2012-07-09 14:00 soft_link -> usrlisting
-rw-r--r-- 1 raghu raghu 491 2012-07-06 16:02 usrcopy
-rw-r--r-- 2 raghu raghu 491 2012-07-06 14:23 usrlisting
```

Useful commands

31) alias command

The ‘alias’ is another name for a command. If no argument is given, it shows current aliases. Aliases can be used for short names of commands. For example, you might use the clear command frequently. You can create an alias for it:

```
$ alias c="clear"
```

Next time you enter 'c ' on command line, your screen will get clear. Current aliases can be checked with 'alias' command:

```
$ alias
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo e
rror)" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\'')"'
alias c='clear'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
```

32) w command

w command is used to check which users are logged in to the system, and what command they are executing at that particular time:

```
$ w
10:06:56 up 57 min, 3 users, load average: 0.04, 0.06, 0.09
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
root tty1 10:06 28.00s 1.02s 0.67s pager -s
raghu tty7 :0 09:19 57:33 1:22 0.20s gnome-session --session=classic-gnome
raghu pts/0 :0.0 09:34 0.00s 0.78s 0.00s w
```

It also shows the uptime, number of users logged in and load average of the system (in the first line of output above).

User management tools

- Command-line

  Useradd

  Usermod

  userdel [-r]

- useradd <username> :- adding new user

useradd redhat

- passwd <username> :- setting password

  # passwd redhat

- passwd -d <username> :- deleting password

  # passwd -d redhat

- userdel <username> :- deleting user

  # userdel redhat

- userdele -r <username> :- deleting from home directory

  # userdel -r redhat

Group management tools

- Groupadd
- groupmod
- groupdel
- Eg :-
- groupadd <groupname> :- adding new group

  # groupadd redhat

- groupdel <groupname> :- removing group

  # groupdel redhat

- Add user in group

groupadd admin

useradd local

usermod -aG admin local

- to remove user from group

  # gpasswd -d user group (remove from group)

- Renaming group
- groupmod -n <new groupname> <old name>

  # groupmod -n sys admin

Permission Types

- Four symbols are used when displaying permissions:

r : permission to read a file or list a directory's contents

w: permission to write to a file or create and remove files from a directory

x: permission to execute a program or change into a directory and do a long listing of the directory

-: no permission (in place of the r, w, or x)

File permissions

- r = read, view
- w = write, update
- x = execute, run

- \- = a permission isn't set

Directory permission

- r = list contents
- w = create/delete contents]
- x = access

\- = a permission isn't set

- File permissions may be viewed using **ls -l file/directory**
- (ls -ld directory)
- Eg :-

ls -l /etc/passwd

-rwxr-xr-x 1 root root 19080 Apr 1 18:26 /etc/passwd

permission field,owner,group,size,date&time,file name

- File type and permissions represented by a 10- character string
- ie., permission are defined for three types of users

Owner	Group	Others

Changing File Ownership

- Only root can change a file's owner
- Ownership is changed with **chown:**

chown *user_name file|directory*

- Group-Ownership is changed with **chgrp:**

chgrp *group_name file|directory*

- Eg :-

  # touch redhat

  # ls -l redhat

  # useradd linux

  # chown linux redhat

  # groupadd rhce

  # chgrp rhce redhat

  # ls -l redhat

Changing Permissions in CLI

Changing permissions are represented in two ways

1. Symbolic method
2. Numeric method

Changing Permissions Symbolic Method

- To change access modes:
- # **chmod WhoWhatWhich file / directory**

Who = **u,g,o** or **a** for user, group other and all

What = **+ ,-** or **=** for grant deny and assig

Which = **r, w** or **x** for read, write and execute

- Eg :-

  # touch redhat

```
# ls -l redhta
  -rw-r--r– redhat
# chmod g+w redhat
# ls -l redhat
  -rw-rw-r-- redhat
# chmod o-r redhat
# ls -li redhat
  -rw-rw---- redhat
# chmod o=w redhat
  -rw-rw--w- redhat
```

Changing Permissions Numeric Method

- Uses a three-digit mode number

  first digit specifies owner's permissions

  second digit specifies group permissions

  third digit represents others' permissions
- Permissions are calculated by adding:

  4 (for read)

  2 (for write)

  1 (for execute)
- Example:

  chmod 640 myfile

Permissions	Symbolic	Numeric
Read	r	4
Write	w	2
Execute	x	1
Full Permission	rwx	7

644	**Owner :** read and write permission **Group :** only read permission **Others:** only read permission
755	**Owner** : full permission **Group** : read and execute permission **Others** : read and execute permission

Default Permissions

- Root :

  directory :- 755

  file :- 644

- User :

  directory :- 775

  file :- 664

UNIX Tools: tar

- The **tar** command is used for creating an archive of a directory hierarchy.
- **tar** archives are a handy way of sending a bunch of files (or a program distribution) across the network or posting them on the internet.
  - Begin by creating a tar archive of the files.
  - Transmit that tar archive over the network or post it online.
  - Untar the files where you want them.
- Usage:
  - Creating a tar archive:

    tar –cvf <archive_name>.tar <files>

  - Viewing the contents of an archive:

    tar –tvf <archive_name>.tar

  - Extracting a tar archive to the current directory:

    tar –xvf <archive_name>.tar

- Create a tar archive of your home directory and place it in your working directory:

  tar –cvf myhome.tar /home

- View the contents of the tar archive:

  tar –tvf myhome.tar

- Extract the tar archive to your current working directory:

  tar –xvf myhome.tar

gzip

- **gzip –** is a compression tool
- Usage:

gzip [options] *file*

- Eg:-

  # gzip a.txt

  # gzip –l a.txt tells compression ratio

gunzip

- **gunzip –** To decompress a file
- Usage:

  gunzip [options] *file*

- Eg:-

  # gunzip a.txt .gz

Zip & unzip

- **zip –** ZIP is a compression and file packaging utility for Unix.
- Usage:

  zip [options] *file.zip f1 f2 f3*

  -d : delete

  -u: add

  -r : directory

- Eg:-

  # zip a.zip a b c

- **uzip –** ZIP is a decompression and file unpackaging utility for Unix.
- Usage:

  unzip [options] *filename*

-d : extract to other dir

-l: to see contain of zip file

Eg:-

unzip a.zip

unzip a.zip –d /dir