Linux Commands

Linux is effectively is a command based OS.

The system administrator uses the commands to carry out even the most advanced tasks.

The are commands for every task that the administrator may want to achieve. This includes commands for:

1. File System Managements
2. Disk Subsystem Managements
3. User and Group Managements
4. Basic Computer Security Management.
5. Network installation and Configuration Managements
6. Server Installation and Management
7. Network Security Managements

Commands Review.

- Linux Shells
- Command, Options and Arguments,
- Redirections >, >>, < and pipes

Basic General Commands

cal, history, date, wall, echo, export, bc, expr .

File Management Commands

cat, cut, cp, rm, mkdir, mv, touch,find, grep, wc, sort, less, head,tail, ls, pwd, rmdir, sed, vim.

Linux "Touch" Command

In **Linux** every single file is associated with timestamps, and every file stores the information of last access time, last modification time and last change time. So, whenever we create new file, access or modify an existing file, the timestamps of that file automatically updated.

The **touch command** is a standard program for **Unix/Linux** operating systems, that is used to create, change and modify timestamps of a file. It has the following options.

Touch Command Options

1. **-a**, change the access time only
2. **-c**, if the file does not exist, do not create it
3. **-d**, update the access and modification times
4. **-m**, change the modification time only

5. **-r**, use the access and modification times of file
6. **-t**, creates a file using a specified time

Touch Command Examples

`1. Create an Empty File`	`touch James`	creates an empty (zero byte) new file called **James**.
`2. Create Multiple Files`	`touch James Jane John`	
`3. Change File Access and Modification Time`	`touch -a  James`	sets the current time and date on a file. If the **James** file does not exist, it will create the new empty file with the name.
4. Avoid Creating New File	`touch -c  James`	create a file called **James** if it does not exists.
5. Change File Modification Time	`touch -m  James`	it will only updates the last modification times (not the access times) of the file.
6. Explicitly Set the Access and Modification times `# touch -c -t YYDDHHMM filename`	`touch -c -t 12101730 expenses`	Sets the access and modification date and time to a file **expenses** as **17:30** (**17:30 p.m**.) **December 10** of the current year (**2012**).
7. Use the time stamp of another File	`# touch -r` **`expenses james`**	touch command with **-r** option, will update the time-stamp of file **james** with the time-stamp of **expenses** file. So, both the file holds the same time stamp.
8. File using a specified time `touch -t YYMMDDHHMM.SS tecmint`	`# touch -t 201212101830.55 tecmint`	will gives the **tecmint** file a time stamp of **18:30:55 p.m**. on **December 10**, **2012**.

Note

The most popular Linux commands such as **find** command and **ls** command uses timestamps for listing and finding files.

Next verify the access and modification time of file **expenses**, with **ls -l** command.

```
# ls -l

total 2
-rw-r--r--.  1 root    root   0 Dec 10 17:30  expenses
```

WC Command & Options

The **wc** (**word count**) command in Unix/Linux operating systems is used to find out number of **newline count**, **word count**, **byte and characters** count in a files specified by the file arguments. The syntax of **wc** command as shown below.

```
[root@tecmint ~]# wc --help

Usage: wc [OPTION]... [FILE]...
  or:  wc [OPTION]... --files0-from=F
Print newline, word, and byte counts for each FILE, and a total line if
more than one FILE is specified.  With no FILE, or when FILE is -,
read standard input.
  -c, --bytes            print the byte counts
  -m, --chars            print the character counts
  -l, --lines            print the newline counts
  -L, --max-line-length  print the length of the longest line
  -w, --words            print the word counts
      --help                    display this help and exit
      --version                 output version information and exit
```

Create a file ‘distros.**txt**‘ for testing the commands with the following content. Use it to practice the various options of wc command.

```
Red Hat
CentOS
Fedora
Debian
Scientific Linux
OpenSuse
Ubuntu
Xubuntu
Linux Mint
Pearl Linux
Slackware
Mandriva
```

The History Command & Options

When you are using Linux command line frequently, using the history effectively can be a major productivity boost.

1. Display timestamp using HISTTIMEFORMAT

Typically when you type history from command line, it displays the command number and the command. For auditing purpose, it may be beneficial to display the timepstamp along with the command as shown below.

```
# export HISTTIMEFORMAT='%F %T '
# history | more
1  2008-08-05 19:02:39 service network restart
2  2008-08-05 19:02:39 exit
3  2008-08-05 19:02:39 id
4  2008-08-05 19:02:39 cat /etc/redhat-release
```

2. Search the history using Control+R

When you’ve already executed a very long command, you can simply search history using a keyword and re-execute the same command without having to type it fully.

Press Control+R and type the keyword. In the following example, I searched for **red**, which displayed the previous command “**cat /etc/redhat-release**” in the history that contained the word red.

```
# [Press Ctrl+R from the command prompt,
which will display the reverse-i-search prompt]
(reverse-i-search)`red': cat /etc/redhat-release
[Note: Press enter when you see your command,
which will execute the command from the history]
# cat /etc/redhat-release
Fedora release 9 (Sulphur)
```

Sometimes you want to edit a command from history before executing it. For e.g. you can search for **httpd**, which will display **service httpd stop** from the command history, select this command and **change the stop to start** and re-execute it again as shown below.

```
# [Press Ctrl+R from the command prompt,
which will display the reverse-i-search prompt]
(reverse-i-search)`httpd': service httpd stop
[Note: Press either left arrow or right arrow key when you see your
command, which will display the command for you to edit, before executing it]
# service httpd start
```

3. Repeat previous command quickly using 4 different methods

Sometime you may end up repeating the previous commands for various reasons. Following are the 4 different ways to repeat the last executed command.

1. Use the **up arrow** to view the previous command and press enter to execute it.
2. Type **!!** and press enter from the command line
3. Type **!-1** and press enter from the command line.
4. Press **Control+P** will display the previous command, press enter to execute it

4. Execute a specific command from history

In the following example, If you want to repeat the command #4, you can do **!4** as shown below.

```
# history | more
1  service network restart
2  exit
3  id
4  cat /etc/redhat-release

# !4
cat /etc/redhat-release
Fedora release 9 (Sulphur)
```

5. Execute previous command that starts with a specific word

Type ! followed by the starting few letters of the command that you would like to re-execute. In the following example, typing !ps and enter, executed the previous command starting with ps, which is ‘ps aux | grep yp’.

```
# !ps
ps aux | grep yp
root     16947  0.0  0.1  36516  1264 ?        Sl   13:10   0:00 ypbind
root     17503  0.0  0.0   4124   740 pts/0    S+   19:19   0:00 grep yp
```

6. Control the total number of lines in the history using HISTSIZE

Append the following two lines to the .bash_profile and relogin to the bash shell again to see the change. In this example, only 450 command will be stored in the bash history.

```
# vi ~/.bash_profile
HISTSIZE=450
HISTFILESIZE=450
```

7. Change the history file name using HISTFILE

By default, history is stored in **~/.bash_history** file. Add the following line to the .bash_profile and relogin to the bash shell, to store the history command in .commandline_warrior file instead of .bash_history file. I’m yet to figure out a practical use for this. I can see this getting used when you want to track commands executed from different terminals using different history file name.

```
# vi ~/.bash_profile
HISTFILE=/root/.commandline_warrior
```

If you have a good reason to change the name of the history file, please share it with me, as I’m interested in finding out how you are using this feature.

8. Eliminate the continuous repeated entry from history using HISTCONTROL

In the following example pwd was typed three times, when you do history, you can see all the 3 continuous occurrences of it. To eliminate duplicates, set HISTCONTROL to ignoredups as shown below.

```
# pwd
# pwd
# pwd
# history | tail -4
44  pwd
45  pwd
46  pwd [Note that there are three pwd commands in history, after
executing pwd 3 times as shown above]
47  history | tail -4

# export HISTCONTROL=ignoredups
# pwd
# pwd
# pwd
# history | tail -3
56  export HISTCONTROL=ignoredups
57  pwd [Note that there is only one pwd command in the history, even after
executing pwd 3 times as shown above]
```

```
58  history | tail -4
```

9. Erase duplicates across the whole history using HISTCONTROL

The ignoredups shown above removes duplicates only if they are consecutive commands. To eliminate duplicates across the whole history, set the HISTCONTROL to erasedups as shown below.

```
# export HISTCONTROL=erasedups
# pwd
# service httpd stop
# history | tail -3
38  pwd
39  service httpd stop
40  history | tail -3

# ls -ltr
# service httpd stop
# history | tail -6
35  export HISTCONTROL=erasedups
36  pwd
37  history | tail -3
38  ls -ltr
39  service httpd stop
[Note that the previous service httpd stop after pwd got erased]
40  history | tail -6
```

10. Force history not to remember a particular command using HISTCONTROL

When you execute a command, you can instruct history to ignore the command by setting HISTCONTROL to ignorespace AND typing a space in front of the command as shown below. I can see lot of junior sysadmins getting excited about this, as they can hide a command from the history. It is good to understand how ignorespace works. But, as a best practice, don't hide purposefully anything from history.

```
# export HISTCONTROL=ignorespace
# ls -ltr
# pwd
#  service httpd stop [Note that there is a space at the beginning of service,
to ignore this command from history]
# history | tail -3
67  ls -ltr
68  pwd
69  history | tail -3
```

11. Clear all the previous history using option -c

Sometime you may want to clear all the previous history, but want to keep the history moving forward.

```
# history -c
```

12. Subtitute words from history commands

When you are searching through history, you may want to execute a different command but use the same parameter from the command that you’ve just searched.

In the example below, the **!!:$** next to the vi command gets the argument from the previous command to the current command.

```
# ls  distros.txt
# cat !!:$   ( Same as cat  distros.txt)
```

In the example below, the **!^** next to the vi command gets the first argument from the previous command (i.e cp command) to the current command (i.e vi command).

```
# cp distros.txt  distros.txt.bak
# cat  !^   ( same as cat  distros.txt)
```

13. Substitute a specific argument for a specific command.

In the example below, **!cp:2** searches for the previous command in history that starts with cp and takes the second argument of cp and substitutes it for the ls -l command as shown below.

```
# cp ~/distros.txt /really/a/very/long/path/long-filename.txt
# ls -l !cp:2
ls -l /really/a/very/long/path/long-filename.txt
```

In the example below, **!cp:$** searches for the previous command in history that starts with cp and takes the last argument (in this case, which is also the second argument as shown above) of cp and substitutes it for the ls -l command as shown below.

```
# ls -l !cp:$
ls -l /really/a/very/long/path/long-filename.txt
```

14. Disable the usage of history using HISTSIZE

If you want to disable history all together and don’t want bash shell to remember the commands you’ve typed, set the HISTSIZE to 0 as shown below.

```
# export HISTSIZE=0
# history
# [Note that history did not display anything]
```

15. Ignore specific commands from the history using HISTIGNORE

Sometimes you may not want to clutter your history with basic commands such as pwd and ls. Use HISTIGNORE to specify all the commands that you want to ignore from the history. Please note that adding ls to the HISTIGNORE ignores only ls and not ls -l. So, you have to provide the exact command that you would like to ignore from the history.

```
# export HISTIGNORE="pwd:ls:ls -ltr:"
# pwd
# ls
# ls -ltr
# service httpd stop
# history | tail -3
79  export HISTIGNORE="pwd:ls:ls -ltr:"
80  service httpd stop
81  history
```

The LS Command & Options

ls – Unix users and sysadmins cannot live without this two letter command. Whether you use it 10 times a day or 100 times a day, knowing the power of ls command can make your command line journey enjoyable.

We examine some Examples of the mighty **ls** command.

1. Open Last Edited File Using ls -t

To open the last edited file in the current directory use the combination of ls, head and vi commands as shown below.

ls -t sorts the file by modification time, showing the last edited file first. **head -1** picks up this first file.

```
$ vi first-long-file.txt
$ vi second-long-file.txt

$ vi `ls -t | head -1`
[Note: This will open the last file you edited (i.e second-long-file.txt)]
```

2. Display One File Per Line Using ls -1

To show single entry per line, use -1 option as shown below.

```
$ ls -1
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
```

3. Display All Information About Files/Directories Using ls -l

To show long listing information about the file/directory.

```
$ ls -l
-rw-r----- 1 ramesh team-dev 9275204 Jun 13 15:27 mthesaur.txt.gz
```

- **1st Character – File Type:** First character specifies the type of the file.
  In the example above the hyphen (-) in the 1st character indicates that this is a normal file.
  Following are the possible file type options in the 1st character of the ls -l output.
    - **Field Explanation**

    - – normal file
    - d directory
    - s socket file
    - l link file
- **Field 1 – File Permissions:** Next 9 character specifies the files permission. Each 3 characters refers to the read, write, execute permissions for user, group and world In this example, -rw-r—– indicates read-write permission for user, read permission for group, and no permission for others.
- **Field 2 – Number of links:** Second field specifies the number of links for that file. In this example, 1 indicates only one link to this file.
- **Field 3 – Owner:** Third field specifies owner of the file. In this example, this file is owned by username 'ramesh'.
- **Field 4 – Group:** Fourth field specifies the group of the file. In this example, this file belongs to "team-dev' group.
- **Field 5 – Size:** Fifth field specifies the size of file. In this example, '9275204' indicates the file size.
- **Field 6 – Last modified date & time:** Sixth field specifies the date and time of the last modification of the file. In this example, 'Jun 13 15:27′ specifies the last modification time of the file.
- **Field 7 – File name:** The last field is the name of the file. In this example, the file name is mthesaur.txt.gz.

4. Display File Size in Human Readable Format Using ls -lh

Use **ls -lh** (h stands for human readable form), to display file size in easy to read format. i.e M for MB, K for KB, G for GB.

```
$ ls -l
-rw-r----- 1 ramesh team-dev 9275204 Jun 12 15:27 arch-linux.txt.gz*

$ ls -lh
-rw-r----- 1 ramesh team-dev 8.9M Jun 12 15:27 arch-linux.txt.gz
```

5. Display Directory Information Using ls -ld

When you use "ls -l" you will get the details of directories content. But if you want the details of directory then you can use -d option as., For example, if you use ls -l /etc will display all the files under etc directory. But, if you want to display the information about the /etc/ directory, use -ld option as shown below.

```
$ ls -l /etc
total 3344
-rw-r--r--   1 root root   15276 Oct  5  2004 a2ps.cfg
-rw-r--r--   1 root root    2562 Oct  5  2004 a2ps-site.cfg
drwxr-xr-x   4 root root    4096 Feb  2  2007 acpi
-rw-r--r--   1 root root      48 Feb  8  2008 adjtime
drwxr-xr-x   4 root root    4096 Feb  2  2007 alchemist

$ ls -ld /etc
drwxr-xr-x 21 root root 4096 Jun 15 07:02 /etc
```

6. Order Files Based on Last Modified Time Using ls -lt

To sort the file names displayed in the order of last modification time use the -t option. You will be finding it handy to use it in combination with -l option.

```
$ ls -lt
total 76
drwxrwxrwt  14 root root  4096 Jun 22 07:36 tmp
drwxr-xr-x 121 root root  4096 Jun 22 07:05 etc
drwxr-xr-x  13 root root 13780 Jun 22 07:04 dev
drwxr-xr-x  13 root root  4096 Jun 20 23:12 root
drwxr-xr-x  12 root root  4096 Jun 18 08:31 home
drwxr-xr-x   2 root root  4096 May 17 21:21 sbin
lrwxrwxrwx   1 root root    11 May 17 20:29 cdrom -> media/cdrom
drwx------   2 root root 16384 May 17 20:29 lost+found
drwxr-xr-x  15 root root  4096 Jul  2  2008 var
```

7. Order Files Based on Last Modified Time (In Reverse Order) Using ls -ltr

To sort the file names in the last modification time in reverse order. This will be showing the last edited file in the last line which will be handy when the listing goes beyond a page. This is my default ls usage. Anytime I do ls, I always use ls -ltr as I find this very convenient.

```
$ ls -ltr

total 76
drwxr-xr-x  15 root root  4096 Jul  2  2008 var
drwx------   2 root root 16384 May 17 20:29 lost+found
lrwxrwxrwx   1 root root    11 May 17 20:29 cdrom -> media/cdrom
drwxr-xr-x   2 root root  4096 May 17 21:21 sbin
drwxr-xr-x  12 root root  4096 Jun 18 08:31 home
drwxr-xr-x  13 root root  4096 Jun 20 23:12 root
drwxr-xr-x  13 root root 13780 Jun 22 07:04 dev
drwxr-xr-x 121 root root  4096 Jun 22 07:05 etc
drwxrwxrwt  14 root root  4096 Jun 22 07:36 tmp
```

8. Display Hidden Files Using ls -a (or) ls -A

To show all the hidden files in the directory, use ‘-a option’. Hidden files in Unix starts with ‘.’ in its file name.

```
$ ls -a
[rnatarajan@asp-dev ~]$ ls -a
.                             Debian-Info.txt
..                            CentOS-Info.txt
.bash_history                 Fedora-Info.txt
.bash_logout                  .lftp
.bash_profile                 libiconv-1.11.tar.tar
.bashrc                       libssh2-0.12-1.2.el4.rf.i386.rpm
```

It will show all the files including the ‘.’ (current directory) and ‘..’ (parent directory). To show the hidden files, but not the ‘.’ (current directory) and ‘..’ (parent directory), use option -A.

```
$ ls -A
Debian-Info.txt               Fedora-Info.txt
CentOS-Info.txt               Red-Hat-Info.txt
```

```
.bash_history                    SUSE-Info.txt
.bash_logout                     .lftp
.bash_profile                    libiconv-1.11.tar.tar
.bashrc                          libssh2-0.12-1.2.el4.rf.i386.rpm
[Note: . and .. are not displayed here]
```

9. Display Files Recursively Using ls -R

```
$ ls  /etc/sysconfig/networking
devices  profiles

$ ls  -R /etc/sysconfig/networking
/etc/sysconfig/networking:
devices  profiles

/etc/sysconfig/networking/devices:

/etc/sysconfig/networking/profiles:
default

/etc/sysconfig/networking/profiles/default:
```

To show all the files recursively, use -R option. When you do this from /, it shows all the unhidden files in the whole file system recursively.

10. Display File Inode Number Using ls -i

Sometimes you may want to know the inone number of a file for internal maintenance. Use -i option as shown below to display inone number. Using inode number you can remove files that has special characters in it’s name as explained in the example#6 of the find command article.

```
$ ls -i /etc/xinetd.d/
279694 chargen      279724 cups-lpd  279697 daytime-udp
279695 chargen-udp  279696 daytime   279698 echo
```

11. Hide Control Characters Using ls -q

To print question mark instead of the non graphics control characters use the -q option.

```
ls -q
```

12. Display File UID and GID Using ls -n

Lists the output like -l, but shows the uid and gid in numeric format instead of names.

```
$ ls -l ~/.bash_profile
-rw-r--r--  1 ramesh ramesh 909 Feb  8 11:48 /home/ramesh/.bash_profile
$ ls -n ~/.bash_profile
-rw-r--r--  1 511 511 909 Feb  8 11:48 /home/ramesh/.bash_profile

[Note: This display 511 for uid and 511 for gid]
```

13. Visual Classification of Files With Special Characters Using ls -F

Instead of doing the ‘ls -l’ and then the checking for the first character to determine the type of file. You can use -F which classifies the file with different special character for different kind of files.

```
$ ls -F
Desktop/  Documents/  Ubuntu-App@  firstfile  Music/  Public/  Templates/
```

Thus in the above output,

- / – directory.
- nothing – normal file.
- @ – link file.
- * – Executable file

14. Visual Classification of Files With Colors Using ls -F

Recognizing the file type by the color in which it gets displayed is an another kind in classification of file. In the above output directories get displayed in blue, soft links get displayed in green, and ordinary files gets displayed in default color.

```
$ ls --color=auto
Desktop  Documents Examples firstfile Music  Pictures  Public  Templates  Videos
```

15. Useful ls Command Aliases

You can take some required ls options in the above, and make it as aliases. We suggest the following.

- Long list the file with size in human understandable form.

  ```
  alias ll="ls -lh"
  ```

- Classify the file type by appending special characters.

  ```
  alias lv="ls -F"
  ```

- Classify the file type by both color and special character.

  ```
  alias ls="ls -F --color=auto"
  ```

16 List Files Based on Last Access Time ls -ltu

Listing of files in directory based on last access time, i.e. based on time the file was last accessed, not modified.

```
# ls -ltu

total 3084272
drwxr-xr-x  2 tecmint tecmint       4096 Jan 19 15:24 Music
drwxr-xr-x  2 tecmint tecmint       4096 Jan 19 15:22 Linux-ISO
drwxr-xr-x  2 tecmint tecmint       4096 Jan 19 15:22 Music-Player
drwx------  3 tecmint tecmint       4096 Jan 19 15:22 tor-browser_en-US
drwxr-xr-x  2 tecmint tecmint       4096 Jan 19 15:22 bin
drwxr-xr-x 11 tecmint tecmint       4096 Jan 19 15:22 Android Games
drwxr-xr-x  2 tecmint tecmint       4096 Jan 19 15:22 Songs
```

```
drwxr-xr-x  2 tecmint tecmint       4096 Jan 19 15:22 renamefiles
drwxr-xr-x  2 tecmint tecmint       4096 Jan 19 15:22 katoolin-master
drwxr-xr-x  2 tecmint tecmint       4096 Jan 19 15:22 Tricks
drwxr-xr-x  3 tecmint tecmint       4096 Jan 19 15:22 Linux-Tricks
drwxr-xr-x  6 tecmint tecmint       4096 Jan 19 15:22 tuptime
drwxr-xr-x  4 tecmint tecmint       4096 Jan 19 15:22 xdm
drwxr-xr-x  2 tecmint tecmint      20480 Jan 19 15:22 ffmpeg usage
drwxr-xr-x  2 tecmint tecmint       4096 Jan 19 15:22 xdm-helper
```

3. List Files Based on Last Modification Time

Listing of files in directory based on last modification time of file's status information, or the `'ctime'`. This command would list that file first whose any status information like: owner, group, permissions, size etc has been recently changed.

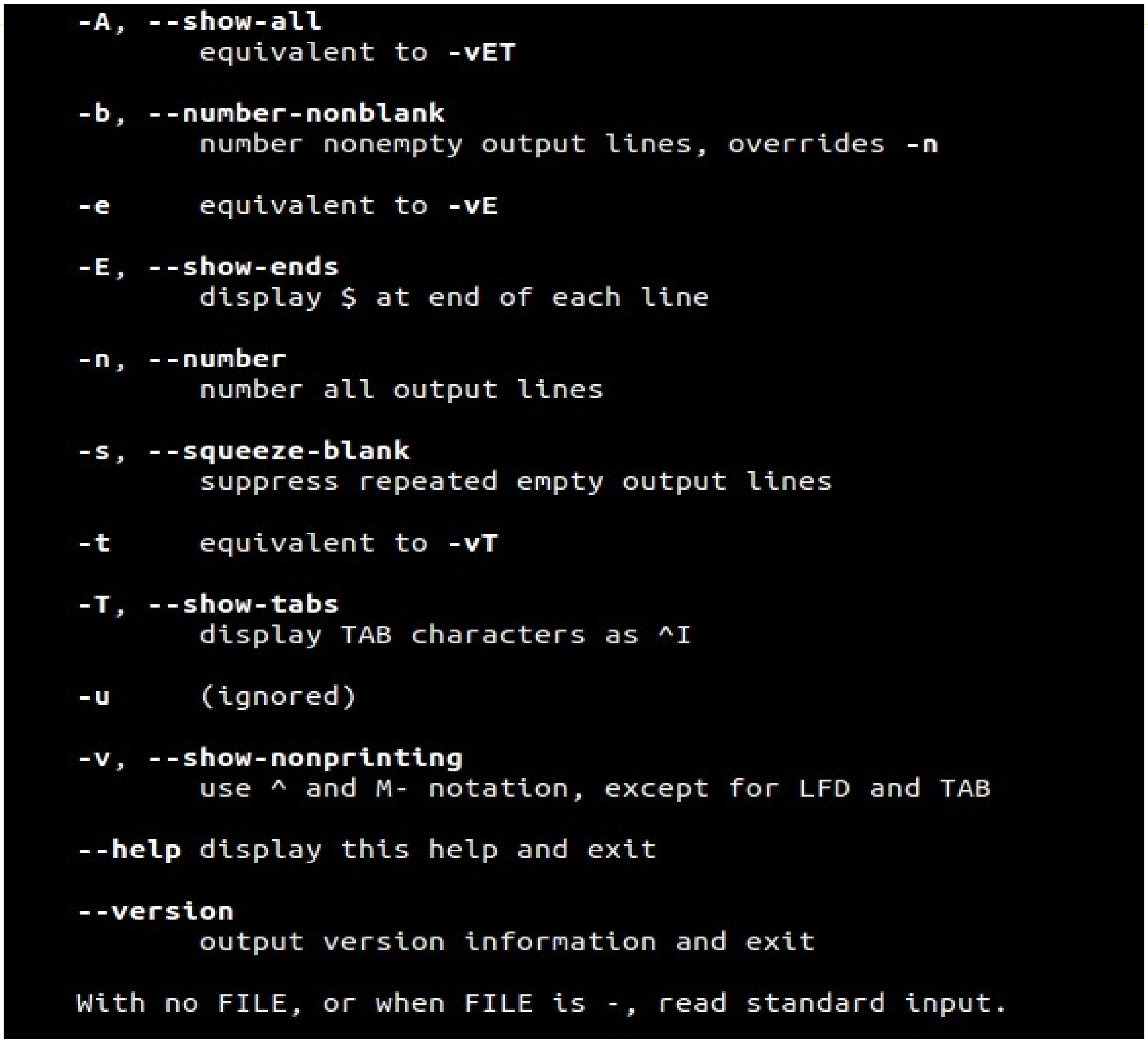
`# ls -ltc`

```
total 3084272
drwxr-xr-x  2 tecmint tecmint       4096 Jan 19 15:24 Music
drwxr-xr-x  2 tecmint tecmint       4096 Jan 19 13:05 img
-rw-------  1 tecmint tecmint     262191 Jan 19 12:15 tecmint.jpeg
drwxr-xr-x  5 tecmint tecmint       4096 Jan 19 10:57 Desktop
drwxr-xr-x  7 tecmint tecmint      12288 Jan 18 16:00 Downloads
drwxr-xr-x 13 tecmint tecmint       4096 Jan 18 15:36 VirtualBox VMs
-rwxr-xr-x  1 tecmint tecmint        691 Jan 13 14:57 special.sh
-rw-r--r--  1 tecmint tecmint     654325 Jan  4 16:55 powertop-2.7.tar.gz.save
-rw-r--r--  1 tecmint tecmint     654329 Jan  4 11:17 filename.tar.gz
drwxr-xr-x  3 tecmint tecmint       4096 Jan  4 11:04 powertop-2.7
-rw-r--r--  1 tecmint tecmint     447795 Dec 31 14:22 Happy-New-Year-2016.jpg
-rw-r--r--  1 tecmint tecmint         12 Dec 18 18:46 ravi
-rw-r--r--  1 tecmint tecmint       1823 Dec 16 12:45 setuid.txt
...
```

If `'-a'` switch is used with above commands, they can list and sort even the hidden files in current directory, and `'-r'` switch lists the output in reverse order.

The cat command & Options

cat stands for Concatenate. Cat is the basic command when we start learning Linux/Unix, as the name suggest it is used to create new file ,concatenate files and display the output of files on the standard output.



```
-A, --show-all
       equivalent to -vET

-b, --number-nonblank
       number nonempty output lines, overrides -n

-e     equivalent to -vE

-E, --show-ends
       display $ at end of each line

-n, --number
       number all output lines

-s, --squeeze-blank
       suppress repeated empty output lines

-t     equivalent to -vT

-T, --show-tabs
       display TAB characters as ^I

-u     (ignored)

-v, --show-nonprinting
       use ^ and M- notation, except for LFD and TAB

--help display this help and exit

--version
       output version information and exit

With no FILE, or when FILE is -, read standard input.
```

Example:1 Create a new file using ‘cat > {file_name}’

Let’s suppose i want to create a new file with name ‘linux_world’. Type the following cat command followed by the text you want in to insert in the file. Make sure you type ‘Ctrl-d’ at the end to save the file.

[root@linuxtechi ~]# cat > linux_world

```
Hi this is my first file in linux.
```

```
Linux always rocks
Thanks
[root@linuxtechi ~]#
```

Example:2 View the Contents of a File.

To display or view the contents of a file using cat command use the below syntax

# cat {file_name}

Let's display the contents of linux_world file.

```
[root@linuxtechi ~]# cat linux_world
```

Example:3 View the Contents of Multiple Files

```
[root@linuxtechi ~]# cat linux_world linux_distributions /etc/fstab
```

Above command will display output of three files on the terminal.

Example:4 Display the output of a file using page wise.

For example if we have a big file whose contents can't be display at once on the screen , in that case we can use more and less command with cat to view the contents page wise.

```
[root@linuxtechi ~]# cat /etc/passwd | more
[root@linuxtechi ~]# cat /etc/passwd | less
```

Example:Using cat command without filename arguments

if we don't specify any arguments in the cat command then it will read the inputs from the keyboard attached to the system. Type some text after entering the cat command.

```
[root@linuxtechi ~]# cat
Ubuntu Linux Rocks at desktop Level
```

Now press '**Ctrl-d**' to inform cat that it has reached end of file (EOF). In this case it will display the line of text twice because it copies std input to std output.

```
[root@linuxtechi ~]# cat
Ubuntu Linux Rocks at desktop Level
Ubuntu Linux Rocks at desktop Level
[root@linuxtechi ~]#
```

Example:6 Display the contents of a file with Line Numbers

```
[root@linuxtechi ~]# cat -n linux_world
1 Hi this is my first file in linux.
2 Linux always rocks
3 Thanks
[root@linuxtechi ~]#
```

In case if your file has blank lines , then above command will also display the number of blank lines as well, so to remove the numbering of blank lines , we can use '**-b**' option in place of '-n' in the above command.

Example:Copy the contents of One file to Another file.

Using greater than **‘>‘** symbol in cat command we can copy the contents of one file to another , example is shown below :

```
[root@linuxtechi ~]# cat linux_world > linux_text
[root@linuxtechi ~]#
```

Example:8Appending the contents of one file to another.

Using double greater than symbol ‘>>‘ in cat command we can append the contents of one file to another. Example is shown below :

```
[root@linuxtechi ~]# cat /etc/passwd >> linux_text
[root@linuxtechi ~]#
```

Above Command will append the contents of /etc/passwd file to linux_text file at the end. Now we can verify the contents of linux_text file.

Example:Redirecting the output of multiple files into a Single File.

```
[root@linuxtechi ~]# cat linux_world linux_distributions /etc/fstab >
linux_merge_text
```

Above command will merge the output of 3 files into a single file ‘linux_merge_text’.

Example:Getting input using standard input operator.

```
[root@linuxtechi ~]# cat < distros.txt
CentOS
Fedora
Ubuntu
SuSE
Linux Mint
[root@linuxtechi ~]#
```

Above cat command is getting input from the file using std input operator ‘<‘

Example:Sorting the output of multiple files into a single file

```
[root@linuxtechi ~]# cat linux_text linux_distributions /etc/passwd | sort >
linux_sort
```

By default sorting will done on the alphabetic order, if you want the sorting on basis of number then use ‘**-n’** option in the sort command.

Example: Insert $ at end of each line using -E option

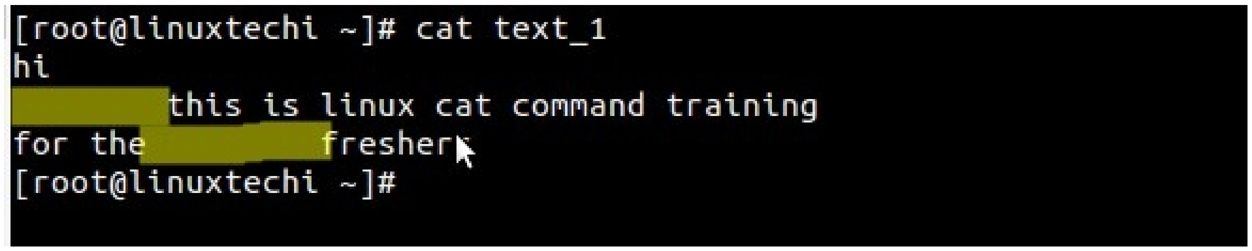
```
[root@linuxtechi ~]# cat -E linux_world
Hi this is my first file in linux.$
```

```
Linux always rocks$
Thanks$
[root@linuxtechi ~]#
```

Above command will insert '$' at the end of each line in the output.

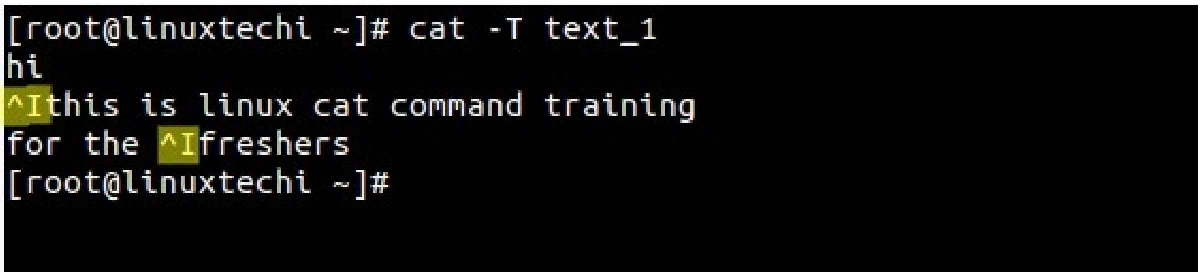
Example:Show the tab space in the file as '^I' using -T option.

Let's create a file with some tab spaces.



```
[root@linuxtechi ~]# cat text_1
hi
        this is linux cat command training
for the         fresher
[root@linuxtechi ~]#
```

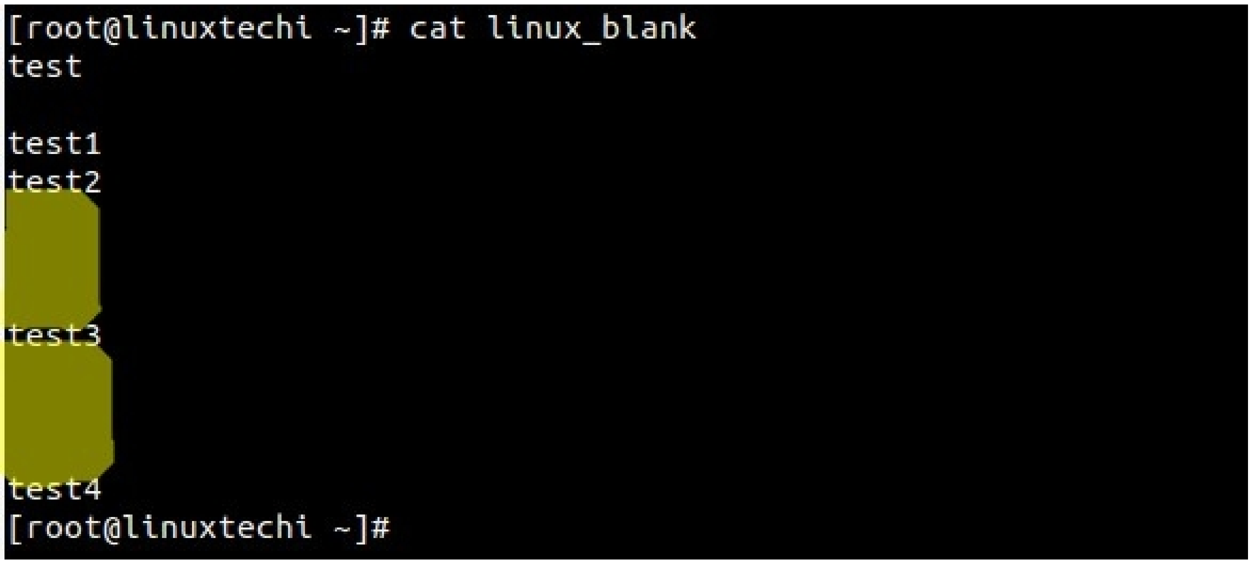
Now display these tab spaces as ^I



```
[root@linuxtechi ~]# cat -T text_1
hi
^Ithis is linux cat command training
for the ^Ifreshers
[root@linuxtechi ~]#
```

Example:Squeeze blank repeated lines using -s option

Let's take am example of file 'linux_blank' , which consists of multiple repeated blank lines.



```
[root@linuxtechi ~]# cat linux_blank
test

test1
test2



test3



test4
[root@linuxtechi ~]#
```

Now remove the blank repeated lines in the output using below command.

```
[root@linuxtechi ~]# cat -s linux_blank
test

test1
test2

test3

test4
[root@linuxtechi ~]#
```

Example:View the Contents in Reverse Order (using the tac command)

tac is the reverse of cat command. tac will display the output in revers order example is shown below

```
[root@linuxtechi ~]# tac linux_world
Thanks
Linux always rocks
Hi this is my first file in linux.
[root@linuxtechi ~]#
```

Example:Display non-printing characters using -v option.

-v option in the cat command is used to show the non-printing characters in the output. This option become useful when we are suspecting the CRLF ending lines, in that case it will show ^M at the end of each line.

```
[root@linuxtechi tmp]# cat test_file
hi there

[root@linuxtechi tmp]# cat -v test_file
hi there^M
[root@linuxtechi tmp]#
```

The find Command & Options

The find command is one of the most essential commands on the linux terminal, that enables searching of files very easy. Its a must of all system administrators.

We have divided the section into **Five** parts from basic to advance usage of find command.

1. Basic Find Commands for Finding Files with Names
2. Find Files Based on their Permissions
3. Search Files Based On Owners and Groups
4. Find Files and Directories Based on Date and Time
5. Find Files and Directories Based on Size

Basic Find Commands for Finding Files with Names

1. Find Files Using Name in Current Directory

Find all the files whose name is **tecmint.txt** in a current working directory.

```
# find . -name tecmint.txt

./tecmint.txt
```

2. Find Files Under Home Directory

Find all the files under **/home** directory with name **tecmint.txt**.

```
# find /home -name tecmint.txt

/home/tecmint.txt
```

3. Find Files Using Name and Ignoring Case

Find all the files whose name is **tecmint.txt** and contains both capital and small letters in **/home** directory.

```
# find /home -iname tecmint.txt

./tecmint.txt
./Tecmint.txt
```

4. Find Directories Using Name

Find all directories whose name is **Tecmint** in **/** directory.

```
# find / -type d -name Tecmint

/Tecmint
```

5. Find PHP Files Using Name

Find all **php** files whose name is **tecmint.php** in a current working directory.

```
# find . -type f -name tecmint.php

./tecmint.php
```

6. Find all PHP Files in Directory

Find all **php** files in a directory.

```
# find . -type f -name "*.php"

./tecmint.php
./login.php
./index.php
```

Find Files Based on their Permissions

7. Find Files With 777 Permissions

Find all the files whose permissions are **777**.

```
# find . -type f -perm 0777 -print
```

8. Find Files Without 777 Permissions

Find all the files without permission **777**.

```
# find / -type f ! -perm 777
```

9. Find SGID Files with 644 Permissions

Find all the **SGID bit** files whose permissions set to **644**.

```
# find / -perm 2644
```

10. Find Sticky Bit Files with 551 Permissions

Find all the **Sticky Bit** set files whose permission are **551**.

```
# find / -perm 1551
```

11. Find SUID Files

Find all **SUID** set files.

```
# find / -perm /u=s
```

12. Find SGID Files

Find all **SGID** set files.

```
# find / -perm /g+s
```

13. Find Read Only Files

Find all **Read Only** files.

```
# find / -perm /u=r
```

14. Find Executable Files

Find all **Executable** files.

```
# find / -perm /a=x
```

15. Find Files with 777 Permissions and Chmod to 644

Find all **777** permission files and use **chmod** command to set permissions to **644**.

```
# find / -type f -perm 0777 -print -exec chmod 644 {} \;
```

16. Find Directories with 777 Permissions and Chmod to 755

Find all **777** permission directories and use **chmod** command to set permissions to **755**.

```
# find / -type d -perm 777 -print -exec chmod 755 {} \;
```

17. Find and remove single File

To find a single file called **tecmint.txt** and remove it.

```
# find . -type f -name "tecmint.txt" -exec rm -f {} \;
```

18. Find and remove Multiple File

To find and remove multiple files such as **.mp3** or **.txt**, then use.

```
# find . -type f -name "*.txt" -exec rm -f {} \;

OR

# find . -type f -name "*.mp3" -exec rm -f {} \;
```

19. Find all Empty Files

To file all empty files under certain path.

```
# find /tmp -type f -empty
```

20. Find all Empty Directories

To file all empty directories under certain path.

```
# find /tmp -type d -empty
```

21. File all Hidden Files

To find all hidden files, use below command.

```
# find /tmp -type f -name ".*"
```

Search Files Based On Owners and Groups

22. Find Single File Based on User

To find all or single file called **tecmint.txt** under / root directory of owner root.

```
# find / -user root -name tecmint.txt
```

23. Find all Files Based on User

To find all files that belongs to user **Tecmint** under **/home** directory.

```
# find /home -user tecmint
```

24. Find all Files Based on Group

To find all files that belongs to group **Developer** under **/home** directory.

```
# find /home -group developer
```

25. Find Particular Files of User

To find all **.txt** files of user **Tecmint** under **/home** directory.

```
# find /home -user tecmint -iname "*.txt"
```

Find Files and Directories Based on Date and Time

26. Find Last 50 Days Modified Files

To find all the files which are modified **50** days back.

```
# find / -mtime 50
```

27. Find Last 50 Days Accessed Files

To find all the files which are accessed **50** days back.

```
# find / -atime 50
```

28. Find Last 50-100 Days Modified Files

To find all the files which are modified more than **50** days back and less than **100** days.

```
# find / -mtime +50 –mtime -100
```

29. Find Changed Files in Last 1 Hour

To find all the files which are changed in last **1 hour**.

```
# find / -cmin -60
```

30. Find Modified Files in Last 1 Hour

To find all the files which are modified in last **1 hour**.

```
# find / -mmin -60
```

31. Find Accessed Files in Last 1 Hour

To find all the files which are accessed in last **1 hour**.

```
# find / -amin -60
```

Find Files and Directories Based on Size

32. Find 50MB Files

To find all **50MB** files, use.

```
# find / -size 50M
```

33. Find Size between 50MB – 100MB

To find all the files which are greater than **50MB** and less than **100MB**.

```
# find / -size +50M -size -100M
```

34. Find and Delete 100MB Files

To find all **100MB** files and delete them using one single command.

```
# find / -size +100M -exec rm -rf {} \;
```

35. Find Specific Files and Delete

Find all **.mp3** files with more than **10MB** and delete them using one single command.

```
# find / -type f -name *.mp3 -size +10M -exec rm {} \;
```

Below commands show usage of `sort` with `find` command to sort the list of files based on **Date** and **Time**.

36. Sorting Files based on Month

Here, we use `find` command to find all files in root `(‘/’)` directory and then print the result as: **Month** in which file was accessed and then filename. Of that complete result, here we list out top **11** entries.

```
# find / -type f -printf "\n%Ab %p" | head -n 11
```

```
Dec /usr/lib/nvidia/pre-install
Dec /usr/lib/libcpufreq.so.0.0.0
Apr /usr/lib/libchromeXvMCPro.so.1.0.0
Apr /usr/lib/libt1.so.5.1.2
Apr /usr/lib/libchromeXvMC.so.1.0.0
Apr /usr/lib/libcdr-0.0.so.0.0.15
Dec /usr/lib/msttcorefonts/update-ms-fonts
Nov /usr/lib/ldscripts/elf32_x86_64.xr
Nov /usr/lib/ldscripts/elf_i386.xbn
Nov /usr/lib/ldscripts/i386linux.xn
```

The below command sorts the output using key as first field, specified by `'-k1'` and then it sorts on Month as specified by `'M'` ahead of it.

```
# find / -type f -printf "\n%Ab %p" | head -n 11 | sort -k1M

Apr /usr/lib/libcdr-0.0.so.0.0.15
Apr /usr/lib/libchromeXvMCPro.so.1.0.0
Apr /usr/lib/libchromeXvMC.so.1.0.0
Apr /usr/lib/libt1.so.5.1.2
Nov /usr/lib/ldscripts/elf32_x86_64.xr
Nov /usr/lib/ldscripts/elf_i386.xbn
Nov /usr/lib/ldscripts/i386linux.xn
Dec /usr/lib/libcpufreq.so.0.0.0
Dec /usr/lib/msttcorefonts/update-ms-fonts
Dec /usr/lib/nvidia/pre-install
```

36. Sort Files Based on Date

Here, again we use `find` command to find all the files in root directory, but now we will print the result as: **last date** the file was accessed, **last time** the file was accessed and then filename. Of that we take out top 11 entries.

```
# find / -type f -printf "\n%AD %AT %p" | head -n 11

12/08/15 11:30:38.0000000000 /usr/lib/nvidia/pre-install
12/07/15 10:34:45.2694776230 /usr/lib/libcpufreq.so.0.0.0
04/11/15 06:08:34.9819910430 /usr/lib/libchromeXvMCPro.so.1.0.0
04/11/15 06:08:34.9939910430 /usr/lib/libt1.so.5.1.2
04/11/15 06:08:35.0099910420 /usr/lib/libchromeXvMC.so.1.0.0
04/11/15 06:08:35.0099910420 /usr/lib/libcdr-0.0.so.0.0.15
12/18/15 11:19:25.2656728990 /usr/lib/msttcorefonts/update-ms-fonts
11/12/15 12:56:34.0000000000 /usr/lib/ldscripts/elf32_x86_64.xr
11/12/15 12:56:34.0000000000 /usr/lib/ldscripts/elf_i386.xbn
11/12/15 12:56:34.0000000000 /usr/lib/ldscripts/i386linux.xn
```

The below sort command first sorts on basis of last digit of the year, then sorts on basis of last digit of month in reverse order and finally sorts on basis of first field. Here, '**1.8**' means 8th column of first field and **'n'** ahead of it means numerical sort, while **'r'** indicates reverse order sorting.

```
# find / -type f -printf "\n%AD %AT %p" | head -n 11 | sort -k1.8n -k1.1nr -k1

12/07/15 10:34:45.2694776230 /usr/lib/libcpufreq.so.0.0.0
12/08/15 11:30:38.0000000000 /usr/lib/nvidia/pre-install
12/18/15 11:19:25.2656728990 /usr/lib/msttcorefonts/update-ms-fonts
11/12/15 12:56:34.0000000000 /usr/lib/ldscripts/elf32_x86_64.xr
```

```
11/12/15 12:56:34.0000000000 /usr/lib/ldscripts/elf_i386.xbn
11/12/15 12:56:34.0000000000 /usr/lib/ldscripts/i386linux.xn
04/11/15 06:08:34.9819910430 /usr/lib/libchromeXvMCPro.so.1.0.0
04/11/15 06:08:34.9939910430 /usr/lib/libt1.so.5.1.2
04/11/15 06:08:35.0099910420 /usr/lib/libcdr-0.0.so.0.0.15
04/11/15 06:08:35.0099910420 /usr/lib/libchromeXvMC.so.1.0.0
```

37. Sorting Files Based on Time

Here, again we use `find` command to list out top 11 files in root directory and print the result in format: last time file was accessed and then filename.

```
# find / -type f -printf "\n%AT %p" | head -n 11

11:30:38.0000000000 /usr/lib/nvidia/pre-install
10:34:45.2694776230 /usr/lib/libcpufreq.so.0.0.0
06:08:34.9819910430 /usr/lib/libchromeXvMCPro.so.1.0.0
06:08:34.9939910430 /usr/lib/libt1.so.5.1.2
06:08:35.0099910420 /usr/lib/libchromeXvMC.so.1.0.0
06:08:35.0099910420 /usr/lib/libcdr-0.0.so.0.0.15
11:19:25.2656728990 /usr/lib/msttcorefonts/update-ms-fonts
12:56:34.0000000000 /usr/lib/ldscripts/elf32_x86_64.xr
12:56:34.0000000000 /usr/lib/ldscripts/elf_i386.xbn
12:56:34.0000000000 /usr/lib/ldscripts/i386linux.xn
```

The below command sorts the output based on first column of the first field of the output which is first digit of hour.

```
# find / -type f -printf "\n%AT %p" | head -n 11 | sort -k1.1n

06:08:34.9819910430 /usr/lib/libchromeXvMCPro.so.1.0.0
06:08:34.9939910430 /usr/lib/libt1.so.5.1.2
06:08:35.0099910420 /usr/lib/libcdr-0.0.so.0.0.15
06:08:35.0099910420 /usr/lib/libchromeXvMC.so.1.0.0
10:34:45.2694776230 /usr/lib/libcpufreq.so.0.0.0
11:19:25.2656728990 /usr/lib/msttcorefonts/update-ms-fonts
11:30:38.0000000000 /usr/lib/nvidia/pre-install
12:56:34.0000000000 /usr/lib/ldscripts/elf32_x86_64.xr
12:56:34.0000000000 /usr/lib/ldscripts/elf_i386.xbn
12:56:34.0000000000 /usr/lib/ldscripts/i386linux.xn
```

38. Sorting Ouptut of ls -l based on Date

This command sorts the output of `'ls -l'` command based on 6th field month wise, then based on 7th field which is date, numerically.

```
# ls -l | sort -k6M -k7n

total 116
-rw-r--r-- 1 root root     0 Oct  1 19:51 backup.tgz
drwxr-xr-x 2 root root  4096 Oct  7 15:27 Desktop
-rw-r--r-- 1 root root 15853 Oct  7 15:19 powertop_report.csv
-rw-r--r-- 1 root root 79112 Oct  7 15:25 powertop.html
-rw-r--r-- 1 root root     0 Oct 16 15:26 file3
-rw-r--r-- 1 root root    13 Oct 16 15:17 B
```

```
-rw-r--r-- 1 root root    21 Oct 16 15:16 A
-rw-r--r-- 1 root root    64 Oct 16 15:38 C
```

Some advanced operations

The find command not only finds files based on a certain criteria, it can also act upon those files using any linux command. For example, we might want to delete some files.

Here are some quick examples

39. List out the found files

Lets say we found files using find command, and now want to list them out as the ls command would have done. This is very easy.

```
$ find . -exec ls -ld {} \;
```

```
drwxrwxr-x 4 enlightened enlightened 4096 Aug 11 19:01 .
-rw-rw-r-- 1 enlightened enlightened 0 Aug 11 16:25 ./abc.txt
drwxrwxr-x 2 enlightened enlightened 4096 Aug 11 16:48 ./abc
drwxrwxr-x 2 enlightened enlightened 4096 Aug 11 16:26 ./subdir
-rw-rw-r-- 1 enlightened enlightened 0 Aug 11 16:26 ./subdir/how.php
-rw-rw-r-- 1 enlightened enlightened 29 Aug 11 19:13 ./abc.php
-rw-rw-r-- 1 enlightened enlightened 0 Aug 11 16:25 ./cool.php
```

40 Delete all matching files or directories

The following command will remove all text files in the tmp directory.

```
$ find /tmp -type f -name "*.txt" -exec rm -f {} \;
```

The same operating can be carried out with directories, just put type d, instead of type f.

Lets take another example where we want to delete files larger than 100MB

```
$ find /home/bob/dir -type f -name *.log -size +10M -exec rm -f {} \;
```

The Cut Command & Options

Linux command cut is used for text processing. You can use this command to extract portion of text from a file by selecting columns.

Below are some practical examples of cut command.

For most of the example, we’ll be using the following test file.

```
$ cat test.txt
cat command for file oriented operations.
cp command for copy files or directories.
ls command to list out files and directories with its attributes.
```

1. Select Column of Characters

To extract only a desired column from a file use -c option. The following example displays 2nd character from each line of a file test.txt

```
$ cut -c2 test.txt
a
p
s
```

As seen above, the characters a, p, s are the second character from each line of the test.txt file.

2. Select Column of Characters using Range

Range of characters can also be extracted from a file by specifying start and end position delimited with -. The following example extracts first 3 characters of each line from a file called test.txt. The first character index is 1.

```
$ cut -c1-3 test.txt
cat
cp
ls
```

3. Select Column of Characters using either Start or End Position

Either start position or end position can be passed to cut command with -c option.

The following specifies only the start position before the '-'. This example extracts from 3rd character to end of each line from test.txt file.

```
$ cut -c3- test.txt
t command for file oriented operations.
 command for copy files or directories.
 command to list out files and directories with its attributes.
```

The following specifies only the end position after the '-'. This example extracts 8 characters from the beginning of each line from test.txt file.

```
$ cut -c-8 test.txt
cat comm
cp comma
ls comma
```

The entire line would get printed when you don't specify a number before or after the '-' as shown below.

```
$ cut -c- test.txt
cat command for file oriented operations.
cp command for copy files or directories.
ls command to list out files and directories with its attributes.
```

4. Select a Specific Field from a File

Instead of selecting x number of characters, if you like to extract a whole field, you can combine option

-f and -d. The option -f specifies which field you want to extract, and the option -d specifies what is the field delimiter that is used in the input file.

The following example displays only first field of each lines from /etc/passwd file using the field delimiter : (colon). In this case, the 1st field is the username. The file

```
$ cut -d':' -f1 /etc/passwd
root
daemon
bin
sys
sync
games
bala
```

5. Select Multiple Fields from a File

You can also extract more than one fields from a file or stdout. Below example displays username and home directory of users who has the login shell as “/bin/bash”.

```
$ grep "/bin/bash" /etc/passwd | cut -d':' -f1,6
root:/root
bala:/home/bala
```

To display the range of fields specify start field and end field as shown below. In this example, we are selecting field 1 through 4, 6 and 7

```
$ grep "/bin/bash" /etc/passwd | cut -d':' -f1-4,6,7
root:x:0:0:/root:/bin/bash
bala:x:1000:1000:/home/bala:/bin/bash
```

6. Select Fields Only When a Line Contains the Delimiter

In our /etc/passwd example, if you pass a different delimiter other than : (colon), cut will just display the whole line.

In the following example, we’ve specified the delimiter as | (pipe), and cut command simply displays the whole line, even when it doesn’t find any line that has | (pipe) as delimiter.

```
$ grep "/bin/bash" /etc/passwd | cut -d'|'  -f1
root:x:0:0:root:/root:/bin/bash
bala:x:1000:1000:bala,,,:/home/bala:/bin/bash
```

But, it is possible to filter and display only the lines that contains the specified delimiter using -s option.

The following example doesn’t display any output, as the cut command didn’t find any lines that has | (pipe) as delimiter in the /etc/passwd file.

```
$ grep "/bin/bash" /etc/passwd | cut -d'|' -s -f1
```

7. Select All Fields Except the Specified Fields

In order to complement the selection field list use option –complement.

The following example displays all the fields from /etc/passwd file except field 7

```
$ grep "/bin/bash" /etc/passwd | cut -d':' --complement -s -f7
root:x:0:0:root:/root
bala:x:1000:1000:bala,,,:/home/bala
```

8. Change Output Delimiter for Display

By default the output delimiter is same as input delimiter that we specify in the cut -d option.

To change the output delimiter use the option –output-delimiter as shown below. In this example, the input delimiter is : (colon), but the output delimiter is # (hash).

```
$ grep "/bin/bash" /etc/passwd | cut -d':'  -s -f1,6,7 --output-delimiter='#'
root#/root#/bin/bash
bala#/home/bala#/bin/bash
```

9. Change Output Delimiter to Newline

In this example, each and every field of the cut command output is displayed in a separate line. We still used –output-delimiter, but the value is $'\n' which indicates that we should add a newline as the output delimiter.

```
$ grep bala /etc/passwd | cut -d':' -f1,6,7 --output-delimiter=$'\n'
bala
/home/bala
/bin/bash
```

Once you master the basic usage of cut command as explained above, you can wisely use cut command to solve lot of your text manipulation requirements.

The TR Command Examples

tr is an UNIX utility for translating, or deleting, or squeezing repeated characters. It will read from STDIN and write to STDOUT.

tr stands for translate.

Syntax

The syntax of tr command is:

```
$ tr [OPTION] SET1 [SET2]

 -c, -C, --complement    use the complement of SET1
 -d, --delete           delete characters in SET1, do not translate
 -s, --squeeze-repeats  replace each input sequence of a repeated character
                         that is listed in SET1 with a single occurrence of that
                        character
 -t, --truncate-set1     first truncate SET1 to length of SET2
      --help     display this help and exit
      --version  output version information and exit
SETs are specified as strings of characters.  Most represent themselves.
```

```
Interpreted sequences are:

  \NNN            character with octal value NNN (1 to 3 octal digits)
  \\              backslash
  \a              audible BEL
  \b              backspace
  \f              form feed
  \n              new line
  \r              return
  \t              horizontal tab
  \v              vertical tab
  CHAR1-CHAR2     all characters from CHAR1 to CHAR2 in ascending order
  [CHAR*]         in SET2, copies of CHAR until length of SET1
  [CHAR*REPEAT]   REPEAT copies of CHAR, REPEAT octal if starting with 0
  [:alnum:]       all letters and digits
  [:alpha:]       all letters
  [:blank:]       all horizontal whitespace
  [:cntrl:]       all control characters
  [:digit:]       all digits
  [:graph:]       all printable characters, not including space
  [:lower:]       all lower case letters
  [:print:]       all printable characters, including space
  [:punct:]       all punctuation characters
  [:space:]       all horizontal or vertical whitespace
  [:upper:]       all upper case letters
  [:xdigit:]      all hexadecimal digits
  [=CHAR=]        all characters which are equivalent to CHAR
```

Translation

If both the SET1 and SET2 are specified and '-d' OPTION is not specified, then tr command will replace each characters in SET1 with each character in same position in SET2.

1. Convert lower case to upper case

The following tr command is used to convert the lower case to upper case

```
$ tr abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ
thegeekstuff
THEGEEKSTUFF
```

The following command will also convert lower case to upper case

```
$ tr [:lower:] [:upper:]
thegeekstuff
THEGEEKSTUFF
```

You can also use ranges in tr. The following command uses ranges to convert lower to upper case.

```
$ tr a-z A-Z
thegeekstuff
```

```
THEGEEKSTUFF
```

2. Translate braces into parenthesis

You can also translate from and to a file. In this example we will translate braces in a file with parenthesis.

```
$ tr '{}' '()' < inputfile > outputfile
```

The above command will read each character from "inputfile", translate if it is a brace, and write the output in "outputfile".

3. Translate white-space to tabs

The following command will translate all the white-space to tabs

```
$ echo "This is for testing" | tr [:space:] '\t'
This    is      for     testing
```

4. Squeeze repetition of characters using -s

In Example 3, we see how to translate space with tabs. But if there are two are more spaces present continuously, then the previous command will translate each spaces to a tab as follows.

```
$ echo "This   is   for testing" | tr [:space:] '\t'
This                    is                      for     testing
```

We can use -s option to squeeze the repetition of characters.

```
$ echo "This   is   for testing" | tr -s [:space:] '\t'
This    is      for     testing
```

Similarly you can convert multiple continuous spaces with a single space

```
$ echo "This  is  for testing" | tr -s [:space:] ' '
This is for testing
```

5. Delete specified characters using -d option

tr can also be used to remove particular characters using -d option. From the file test.txt

```
cat <test.txt | tr -d 't'
```

To remove all the digits from the string, use

```
$ echo "my username is 432234" | tr -d [:digit:]
my username is
```

6. Complement the sets using -c option

You can complement the SET1 using -c option. For example, to remove all characters except digits, you can use the following.

```
$ echo "my username is 432234" | tr -cd [:digit:]
432234
```

7. Remove all non-printable character from a file

The following command can be used to remove all non-printable characters from a file.

```
$ tr -cd [:print:] < file.txt
```

8. Join all the lines in a file into a single line

The below command will translate all newlines into spaces and make the result as a single line.

```
$ tr -s '\n' ' ' < file.txt
```

The Grep Command Examples

First create the following demo_file that will be used in the examples below to demonstrate grep command.

```
$ cat demo_file
THIS LINE IS THE 1ST UPPER CASE LINE IN THIS FILE.
this line is the 1st lower case line in this file.
This Line Has All Its First Character Of The Word With Upper Case.

Two lines above this line is empty.
And this is the last line.
```

1. Search for the given string in a single file

The basic usage of grep command is to search for a specific string in the specified file as shown below.

```
Syntax:
grep "literal_string" filename
$ grep "this" demo_file
this line is the 1st lower case line in this file.
Two lines above this line is empty.
And this is the last line.
```

2. Checking for the given string in multiple files.

```
Syntax:
grep "string" FILE_PATTERN
```

This is also a basic usage of grep command. For this example, let us copy the demo_file to demo_file1. The grep output will also include the file name in front of the line that matched the specific pattern as shown below. When the Linux shell sees the meta character, it does the expansion and gives all the files as input to grep.

```
$ cp demo_file demo_file1

$ grep "this" demo_*
demo_file:this line is the 1st lower case line in this file.
demo_file:Two lines above this line is empty.
demo_file:And this is the last line.
demo_file1:this line is the 1st lower case line in this file.
demo_file1:Two lines above this line is empty.
```

```
demo_file1:And this is the last line.
```

3. Case insensitive search using grep -i

```
Syntax:
grep -i "string" FILE
```

This is also a basic usage of the grep. This searches for the given string/pattern case insensitively. So it matches all the words such as “the”, “THE” and “The” case insensitively as shown below.

```
$ grep -i "the" demo_file

THIS LINE IS THE 1ST UPPER CASE LINE IN THIS FILE.
this line is the 1st lower case line in this file.
This Line Has All Its First Character Of The Word With Upper Case.
And this is the last line.
```

4. Match regular expression in files

```
Syntax:
grep "REGEX" filename
```

This is a very powerful feature, if you can use use regular expression effectively. In the following example, it searches for all the pattern that starts with “lines” and ends with “empty” with anything in-between. i.e To search “lines[anything in-between]empty” in the demo_file.

```
$ grep "lines.*empty" demo_file

Two lines above this line is empty.
```

From documentation of grep: A regular expression may be followed by one of several repetition operators:

- ? The preceding item is optional and matched at most once.
- * The preceding item will be matched zero or more times.
- + The preceding item will be matched one or more times.
- {n} The preceding item is matched exactly n times.
- {n,} The preceding item is matched n or more times.
- {,m} The preceding item is matched at most m times.
- {n,m} The preceding item is matched at least n times, but not more than m times.

5. Checking for full words, not for sub-strings using grep -w

If you want to search for a word, and to avoid it to match the substrings use -w option. Just doing out a normal search will show out all the lines.

The following example is the regular grep where it is searching for “is”. When you search for “is”, without any option it will show out “is”, “his”, “this” and everything which has the substring “is”.

```
$ grep -i "is" demo_file

THIS LINE IS THE 1ST UPPER CASE LINE IN THIS FILE.
this line is the 1st lower case line in this file.
This Line Has All Its First Character Of The Word With Upper Case.
Two lines above this line is empty.
And this is the last line.
```

The following example is the WORD grep where it is searching only for the word “is”. Please note that this output does not contain the line “This Line Has All Its First Character Of The Word With Upper Case”, even though “is” is there in the “This”, as the following is looking only for the word “is” and not for “this”.

```
$ grep -iw "is" demo_file

THIS LINE IS THE 1ST UPPER CASE LINE IN THIS FILE.
this line is the 1st lower case line in this file.
Two lines above this line is empty.
And this is the last line.
```

6. Displaying lines before/after/around the match using grep -A, -B and -C

When doing a grep on a huge file, it may be useful to see some lines after the match. You might feel handy if grep can show you not only the matching lines but also the lines after/before/around the match.

Please create the following demo_text file for this example.

```
$ cat demo_text

4. Vim Word Navigation

You may want to do several navigation in relation to the words, such as:

 * e - go to the end of the current word.
 * E - go to the end of the current WORD.
 * b - go to the previous (before) word.
 * B - go to the previous (before) WORD.
 * w - go to the next word.
 * W - go to the next WORD.

WORD - WORD consists of a sequence of non-blank characters, separated with white
space.
word - word consists of a sequence of letters, digits and underscores.

Example to show the difference between WORD and word

 * 192.168.1.1 - single WORD
 * 192.168.1.1 - seven words.
```

7.Display N lines after match

-A is the option which prints the specified N lines after the match as shown below.

```
Syntax:
grep -A <N> "string" FILENAME
```

The following example prints the matched line, along with the 3 lines after it.

```
$ grep -A 3 -i "example" demo_text

Example to show the difference between WORD and word

* 192.168.1.1 - single WORD
* 192.168.1.1 - seven words.
```

8 Display N lines before match

-B is the option which prints the specified N lines before the match.

```
Syntax:
grep -B <N> "string" FILENAME
```

When you had option to show the N lines after match, you have the -B option for the opposite.

```
$ grep -B 2 "single WORD" demo_text
Example to show the difference between WORD and word

* 192.168.1.1 - single WORD
```

9 Display N lines around match

-C is the option which prints the specified N lines before the match. In some occasion you might want the match to be appeared with the lines from both the side. This options shows N lines in both the side(before & after) of match.

```
$ grep -C 2 "Example" demo_text

word - word consists of a sequence of letters, digits and underscores.

Example to show the difference between WORD and word

* 192.168.1.1 - single WORD
```

10 Highlighting the search using GREP_OPTIONS

As grep prints out lines from the file by the pattern / string you had given, if you wanted it to highlight which part matches the line, then you need to follow the following way.

When you do the following export you will get the highlighting of the matched searches. In the following example, it will highlight all the this when you set the GREP_OPTIONS environment variable as shown below.

```
$ export GREP_OPTIONS='--color=auto' GREP_COLOR='100;8'

$ grep this demo_file

this line is the 1st lower case line in this file.
Two lines above this line is empty.
And this is the last line.
```

11. Searching in all files recursively using grep -r

When you want to search in all the files under the current directory and its sub directory. -r option is the one which you need to use. The following example will look for the string “ramesh” in all the files in the current directory and all it’s subdirectory.

```
$ grep -r "ramesh" *
```

12. Invert match using grep -v

You had different options to show the lines matched, to show the lines before match, and to show the lines after match, and to highlight match. So definitely You’d also want the option -v to do invert match.

When you want to display the lines which does not matches the given string/pattern, use the option -v as shown below. This example will display all the lines that did not match the word “go”.

```
$ grep -v "go" demo_text
4. Vim Word Navigation

You may want to do several navigation in relation to the words, such as:

WORD - WORD consists of a sequence of non-blank characters, separated with white
space.
word - word consists of a sequence of letters, digits and underscores.

Example to show the difference between WORD and word

* 192.168.1.1 - single WORD
* 192.168.1.1 - seven words.
```

13. display the lines which does not matches all the given pattern.

```
Syntax:
grep -v -e "pattern" -e "pattern"
$ cat test-file.txt
a
b
c
d

$ grep -v -e "a" -e "b" -e "c" test-file.txt
d
```

14. Counting the number of matches using grep -c

When you want to count that how many lines matches the given pattern/string, then use the option -c.

```
Syntax:
grep -c "pattern" filename

$ grep -c "go" demo_text
6
```

When you want do find out how many lines matches the pattern

```
$ grep -c this demo_file
3
```

When you want do find out how many lines that does not match the pattern

```
$ grep -v -c this demo_file
4
```

15. Display only the file names which matches the given pattern using grep -l

If you want the grep to show out only the file names which matched the given pattern, use the -l (lower-case L) option.

When you give multiple files to the grep as input, it displays the names of file which contains the text that matches the pattern, will be very handy when you try to find some notes in your whole directory structure.

```
$ grep -l this demo_*
demo_file
demo_file1
```

16. Show only the matched string

By default grep will show the line which matches the given pattern/string, but if you want the grep to show out only the matched string of the pattern then use the -o option.

It might not be that much useful when you give the string straight forward. But it becomes very useful when you give a regex pattern and trying to see what it matches as

```
$ grep -o "is.*line" demo_file
is line is the 1st lower case line
is line
is is the last line
```

17. Show the position of match in the line

When you want grep to show the position where it matches the pattern in the file, use the following options as

```
Syntax:
grep -o -b "pattern" file

$ cat temp-file.txt
12345
12345

$ grep -o -b "3" temp-file.txt
2:3
8:3
```

Note: The output of the grep command above is not the position in the line, it is byte offset of the whole file.

18. Show line number while displaying the output using grep -n

To show the line number of file with the line matched. It does 1-based line numbering for each file. Use -n option to utilize this feature.

```
$ grep -n "go" demo_text
5: * e - go to the end of the current word.
6: * E - go to the end of the current WORD.
7: * b - go to the previous (before) word.
8: * B - go to the previous (before) WORD.
9: * w - go to the next word.
10: * W - go to the next WORD.
```

Grep OR, Grep AND, Grep NOT Operator

Question: Can you explain how to use OR, AND and NOT operators in Unix grep command with some examples?

Answer: In grep, we have options equivalent to OR and NOT operators. There is no grep AND opearator. But, you can simulate AND using patterns. The examples mentioned below will help you to understand how to use OR, AND and NOT in Linux grep command.

The following employee.txt file is used in the following examples.

```
$ cat employee.txt
100  Thomas  Manager    Sales       $5,000
200  Jason   Developer  Technology  $5,500
300  Raj     Sysadmin   Technology  $7,000
400  Nisha   Manager    Marketing   $9,500
500  Randy   Manager    Sales       $6,000
```

You already knew that grep is extremely powerful based on these grep command examples.

Grep OR Operator

Use any one of the following 4 methods for grep OR. I prefer method number 3 mentioned below for grep OR operator.

19 Grep OR Using \|

If you use the grep command without any option, you need to use \| to separate multiple patterns for the or condition.

```
grep 'pattern1\|pattern2' filename
```

For example, grep either Tech or Sales from the employee.txt file. Without the back slash in front of the pipe, the following will not work.

```
$ grep 'Tech\|Sales' employee.txt
100  Thomas  Manager    Sales       $5,000
200  Jason   Developer  Technology  $5,500
300  Raj     Sysadmin   Technology  $7,000
500  Randy   Manager    Sales       $6,000
```

20 Grep OR Using -E

grep -E option is for extended regexp. If you use the grep command with -E option, you just need to use | to separate multiple patterns for the or condition.

```
grep -E 'pattern1|pattern2' filename
```

For example, grep either Tech or Sales from the employee.txt file. Just use the | to separate multiple OR patterns.

```
$ grep -E 'Tech|Sales' employee.txt
100  Thomas  Manager    Sales       $5,000
200  Jason   Developer  Technology  $5,500
300  Raj     Sysadmin   Technology  $7,000
500  Randy   Manager    Sales       $6,000
```

21 Grep OR Using egrep

egrep is exactly same as ‘grep -E’. So, use egrep (without any option) and separate multiple patterns for the or condition.

```
egrep 'pattern1|pattern2' filename
```

For example, grep either Tech or Sales from the employee.txt file. Just use the | to separate multiple OR patterns.

```
$ egrep 'Tech|Sales' employee.txt
100  Thomas  Manager    Sales       $5,000
200  Jason   Developer  Technology  $5,500
300  Raj     Sysadmin   Technology  $7,000
500  Randy   Manager    Sales       $6,000
```

22 Grep OR Using grep -e

Using grep -e option you can pass only one parameter. Use multiple -e option in a single command to use multiple patterns for the or condition.

```
grep -e pattern1 -e pattern2 filename
```

For example, grep either Tech or Sales from the employee.txt file. Use multiple -e option with grep for the multiple OR patterns.

```
$ grep -e Tech -e Sales employee.txt
100  Thomas  Manager    Sales       $5,000
200  Jason   Developer  Technology  $5,500
300  Raj     Sysadmin   Technology  $7,000
500  Randy   Manager    Sales       $6,000
```

Grep AND

23 Grep AND using -E ‘pattern1.*pattern2′

There is no AND operator in grep. But, you can simulate AND using grep -E option.

```
grep -E 'pattern1.*pattern2' filename
grep -E 'pattern1.*pattern2|pattern2.*pattern1' filename
```

The following example will grep all the lines that contain both “Dev” and “Tech” in it (in the same order).

```
$ grep -E 'Dev.*Tech' employee.txt
200  Jason   Developer  Technology  $5,500
```

The following example will grep all the lines that contain both “Manager” and “Sales” in it (in any order).

```
$ grep -E 'Manager.*Sales|Sales.*Manager' employee.txt
```

Note: Using regular expressions in grep is very powerful if you know how to use it effectively.

24 Grep AND using Multiple grep command

You can also use multiple grep command separated by pipe to simulate AND scenario.

```
grep -E 'pattern1' filename | grep -E 'pattern2'
```

The following example will grep all the lines that contain both “Manager” and “Sales” in the same line.

```
$ grep Manager employee.txt | grep Sales
100  Thomas  Manager    Sales       $5,000
500  Randy   Manager    Sales       $6,000
```

Grep NOT

26. Grep NOT using grep -v

Using grep -v you can simulate the NOT conditions. -v option is for invert match. i.e It matches all the lines except the given pattern.

```
grep -v 'pattern1' filename
```

For example, display all the lines except those that contains the keyword "Sales".

```
$ grep -v Sales employee.txt
200  Jason   Developer  Technology  $5,500
300  Raj     Sysadmin   Technology  $7,000
400  Nisha   Manager    Marketing   $9,500
```

You can also combine NOT with other operator to get some powerful combinations.

For example, the following will display either Manager or Developer (bot ignore Sales).

```
$ egrep 'Manager|Developer' employee.txt | grep -v Sales
200  Jason   Developer  Technology  $5,500
400  Nisha   Manager    Marketing   $9,500
```

The Date Command & Options

Date command is helpful to display date in several formats. It also allows you to set systems date and time.

We examine few examples on how to use date command with practical examples.

When you execute date command without any option, it will display the current date and time as shown below.

```
$ date
Mon May 20 22:02:24 PDT 2013
```

1. Display Date from a String Value using –date Option

If you have a static date or time value in a string, you can use -d or –date option to convert the input string into date format as shown below.

Please note that this doesn't use the current date and time value. Instead is uses the date and time value that you pass as string.

The following examples takes an input date only string, and displays the output in date format. If you don't specify time, it uses 00:00:00 for time.

```
$ date --date="12/2/2014"
Tue Dec  2 00:00:00 PST 2014

$ date --date="2 Feb 2014"
Sun Feb  2 00:00:00 PST 2014

$ date --date="Feb 2 2014"
Sun Feb  2 00:00:00 PST 2014
```

The following example takes an input date and time string, and displays the output in date format.

```
$ date --date="Feb 2 2014 13:12:10"
Sun Feb  2 13:12:10 PST 2014
```

2. Read Date Patterns from a file using –file option

This is similar to the -d or –date option that we discussed above. But, you can do it for multiple date strings. If you have a file that contains various static date strings, you can use -f or –file option as shown below.

In this example, we can see that datefile contained 2 date strings. Each line of datefile is parsed by date command and date is outputted for each line.

```
$ cat datefile
Sept 9 1986
Aug 23 1987

$ date --file=datefile
Tue Sep  9 00:00:00 EAT 1986
Sun Aug 23 00:00:00 EAT 1987
```

3. Get Relative Date Using –date option

You can also use date command to get a future date using relative values.

For example, the following examples gets date of next Monday.

```
$ date --date="next mon"
Mon Feb  1 00:00:00 EAT 2016
```

If string=@is given to date command, then date command convert seconds since the epoch (1970-01-01 UTC) to a date.

It displays date in which 5 seconds are elapsed since epoch 1970-01-01 UTC:

```
$ date --date=@5
Thu Jan  1 03:00:05 EAT 1970
```

It displays date in which 10 seconds are elapsed since epoch 1970-01-01 UTC:

```
$ date --date=@10
Thu Jan  1 03:00:10 EAT 1970
```

It displays date in which 1 minute (i.e. 60 seconds) is elapsed since epoch 1970-01-01 UTC:

```
$ date --date=@60
Thu Jan  1 03:01:00 EAT 1970
```

4. Display Past Date

You can display a past date using the -date command. Few possibilities are shown below.

```
$ date --date='3 seconds ago'
Sun Jan 31 17:49:39 EAT 2016

$ date --date="1 day ago"
Sun May 19 21:59:36 PDT 2013

$ date --date="yesterday"
Sat Jan 30 17:50:26 EAT 2016
```

```
$ date --date="1 month ago"
Thu Dec 31 17:50:53 EAT 2015

$ date --date="1 year ago"
Sun May 20 22:00:09 EAT 2014
```

5. Set Date and Time using –set option

You can set date and time of your system using -s or –set option as shown below..

In this example, initially it displayed the time as 20:09:31. We then used date command to change it to 21:00:00.

```
$ date
Sun May 20 20:09:31 EAT 2013

$ date -s "Sun May 20 21:00:00 PDT 2013"
Sun May 20 21:00:00 EAT 2013

$ date
Sun May 20 21:00:05 EAT 2013
```

5. Display Universal Time using -u option

You can display date in UTC format using -u, or –utc, or –universal option as shown below.

```
$ date
Mon May 20 22:07:53 EAT 2013

$ date -u
Tue May 21 05:07:55 EAT 2013
```

6. Display Last Modification Time using -r option

In this example, the current time is 20:25:48

```
$ date
Sun May 20 20:25:48 EAT 2013
```

The timestamp of datefile is changed using touch command. This was done few seconds after the above date command's output.

```
$ touch datefile
```

The current time after the above touch command is 20:26:12

```
$ date
Sun May 20 20:26:12 EAT 2013
```

Finally, use the date command -r option to display the last modified timestamp of a file as shown below. In this example, it displays last modified time of datefile as 20:25:57. It is somewhere between 20:25:48 and 20:26:12 (which is when we execute the above touch command to modify the timestamp).

```
$ date -r datefile
Sun May 20 20:25:57 EAT 2013
```

7. Various Date Command Formats

You can use formatting option to display date command in various formats using the following syntax:

```
$ date +%<format-option>
```

The following table displays various date command formatting options.

Format options	Purpose of Option	Output
date +%a	Displays Weekday name in short (like Mon, Tue, Wed)	Thu
date +%A	Displays Weekday name in full short (like Monday, Tuesday)	Thursday
date +%b	Displays Month name in short (like Jan, Feb, Mar)	Feb
date +%B	Displays Month name in full short (like January, February)	February
date +%d	Displays Day of month (e.g., 01)	07
date +%D	Displays Current Date; shown in MM/DD/YY	02/07/13
date +%F	Displays Date; shown in YYYY-MM-DD	2013-02-07
date +%H	Displays hour in (00..23) format	23
date +%I	Displays hour (01..12) format	11
date +%j	Displays day of year (001..366)	038
date +%m	Displays month (01..12)	02
date +%M	Displays minute (00..59)	44
date +%S	Displays second (00..60)	17
date +%N	Displays nanoseconds (000000000..999999999)	573587606
date +%T	Displays time; shown as HH:MM:SS Note: Hours in 24 Format	23:44:17
date +%u	Displays day of week (1..7); 1 is Monday	4
date +%U	Displays week number of year, with Sunday as first day of week (00..53)	05
date +%Y	Displays full year i.e. YYYY	2013
date +%Z	alphabetic time zone abbreviation (e.g., EDT)	IS

The sort command & Options

sort command is used to sort a file, arranging the records in a particular order. By default, the sort command sorts file assuming the contents are ascii. Using options in sort command, it can also be used to sort numerically. Let us discuss it with some examples:

File with Ascii data:

Let us consider a file with the following contents:

```
$ cat file
Unix
Linux
Solaris
AIX
Linux
```

HPUX

1. sort simply sorts the file in alphabetical order:

```
$ sort file
AIX
HPUX
Linux
Linux
Solaris
Unix
```

All records are sorted alphabetically.

2. sort removes the duplicates using the -u option:

```
$ sort -u file
AIX
HPUX
Linux
Solaris
Unix
```

The duplicate 'Linux' record got removed. '-u' option removes all the duplicate records in the file. Even if the file have had 10 'Linux' records, with -u option, only the first record is retained.

File with numbers:

Let us consider a file with numbers:

```
$ cat file
20
19
5
49
200
```

3. The default sort 'might' give incorrect result on a file containing numbers:

```
$ sort file
19
20
200
49
5
```

In the above result, 200 got placed immediately below 20, not at the end which is incorrect. This is because the sort did ASCII sort. If the file had not contained '200', the default sort would have given proper result. However, it is incorrect to sort a numerical file in this way since the sorting logic is incorrect.

4. To sort a file numericallly:

```
$ sort -n file
5
19
```

```
20
49
200
```

-n option can sort the decimal numbers as well.

5. sort file numerically in reverse order:

```
$ sort -nr file
200
49
20
19
5
```

'r' option does a reverse sort.

Multiple Files:
Let us consider examples with multiple files, say file1 and file2, containing numbers:

```
$ cat file1
20
19
5
49
200

$ cat file2
25
18
5
48
200
```

6. sort can sort multiple files as well.

```
$ sort -n file1 file2
5
5
18
19
20
25
48
49
200
200
```

The result of sort with multiple files will be a sorted and merged output of the multiple files.

7. Sort, merge and remove duplicates:

```
$ sort -nu file1 file2
5
18
```

```
19
20
25
48
49
200
```

-u option becomes more handy in case of multiple files. With this, the output is now sorted, merged and without duplicate records.

Files with multiple fields and delimiter:
Let us consider a file with multiple fields:

```
$ cat file
Linux,20
Unix,30
AIX,25
Linux,25
Solaris,10
HPUX,100
```

8. sorting a file containing multiple fields:

```
$ sort file
AIX,25
HPUX,100
Linux,20
Linux,25
Solaris,10
Unix,30
```

As shown above, the file got sorted on the 1st field, by default.

9. sort file on the basis of 1st field:

```
$ sort -t"," -k1,1 file
AIX,25
HPUX,100
Linux,20
Linux,25
Solaris,10
Unix,30
```

This is being more explicit. '-t' option is used to provide the delimiter in case of files with delimiter. '-k' is used to specify the keys on the basis of which the sorting has to be done. The format of '-k' is : '-km,n' where *m* is the starting key and *n* is the ending key. In other words, sort can be used to sort on a range of fields just like how the group by in sql does. In our case, since the sorting is on the 1st field alone, we speciy '1,1'. Similarly, if the sorting is to be done on the basis of first 3 fields, it will be: '-k 1,3'.

Note: For a file which has fields delimited by a space or a tab, there is no need to specify the "-t" option since the white space is the delimiter by default in sort.

10. sorting file on the basis of the 2nd field:

```
$ sort -t"," -k2,2 file
Solaris,10
HPUX,100
Linux,20
AIX,25
Linux,25
Unix,30
```

11. sorting file on the basis of 2nd field , numerically:

```
$ sort -t"," -k2n,2 file
Solaris,10
Linux,20
AIX,25
Linux,25
Unix,30
HPUX,100
```

12. Remove duplicates from the file based on 1st field:

```
$ sort -t"," -k1,1 -u file
AIX,25
HPUX,100
Linux,20
Solaris,10
Unix,30
```

The duplicate Linux record got removed. Keep in mind, the command "sort -u file" would not have worked here becuase both the 'Linux' records are not same, the values were different. However, in the above, sort is told to remove the duplicates based on the 1st key, and hence the duplicate 'Linux' record got removed. According to sort, in case of a group of similar records, except the first one, the rest are considered duplicate.

13. Sort the file numerically on the 2nd field in reverse order:

```
$ sort -t"," -k2nr,2 file
HPUX,100
Unix,30
AIX,25
Linux,25
Linux,20
Solaris,10
```

14. sort the file alphabetically on the 1st field, numerically on the 2nd field:

```
$ sort -t"," -k1,1 -k2n,2 file
AIX,25
HPUX,100
Linux,20
Linux,25
Solaris,10
Unix,30
```

15. sort a file based on the 1st and 2nd field, and numerically on 3rd field on a file containing 5 columns:

```
$ sort -t"," -k1,2 -k3n,3 file
```

File Security Commands

1. **chmod command**
2. **chown command**
3. **chgrp command**

The Chmod Command

Following are the symbolic representation of three different roles:

- u is for user,
- g is for group,
- and o is for others.

Following are the symbolic representation of three different permissions:

- r is for read permission,
- w is for write permission,
- x is for execute permission.

Following are few examples on how to use the symbolic representation on chmod.

1. Add single permission to a file/directory

Changing permission to a single set. + symbol means adding permission. For example, do the following to give execute permission for the user irrespective of anything else:

```
$ chmod u+x filename
```

2. Add multiple permission to a file/directory

Use comma to separate the multiple permission sets as shown below.

```
$ chmod u+r,g+x filename
```

3. Remove permission from a file/directory

Following example removes read and write permission for the user.

```
$ chmod u-rx filename
```

4. Change permission for all roles on a file/directory

Following example assigns execute privilege to user, group and others (basically anybody can execute this file).

```
$ chmod a+x filename
```

5. Make permission for a file same as another file (using reference)

If you want to change a file permission same as another file, use the reference option as shown below. In this example, file2's permission will be set exactly same as file1's permission.

```
$ chmod --reference=file1 file2
```

6. Apply the permission to all the files under a directory recursively

Use option -R to change the permission recursively as shown below.

```
$ chmod -R 755 directory-name/
```

7. Change execute permission only on the directories (files are not affected)

On a particular directory if you have multiple sub-directories and files, the following command will assign execute permission only to all the sub-directories in the current directory (not the files in the current directory).

```
$ chmod u+X *
```

Note: If the files has execute permission already for either the group or others, the above command will assign the execute permission to the user

The Chown Command

The concept of owner and groups for files is fundamental to Linux. Every file is associated with an owner and a group. You can use chown and chgrp commands to change the owner or the group of a particular file or directory.

In this article, we will discuss the 'chown' command as it covers most part of the 'chgrp' command also.

Even if you already know this command, probably one of the examples mentioned below might be new to you.

1. Change the owner of a file

```
# ls -lart tmpfile
-rw-r--r-- 1 himanshu family 0 2012-05-22 20:03 tmpfile

# chown root tmpfile

# ls -l tmpfile
-rw-r--r-- 1 root family 0 2012-05-22 20:03 tmpfile
```

So we see that the owner of the file was changed from 'himanshu' to 'root'.

2. Change the group of a file

Through the chown command, the group (that a file belongs to) can also be changed.

```
# ls -l tmpfile
```

```
-rw-r--r-- 1 himanshu family 0 2012-05-22 20:03 tmpfile

# chown :friends tmpfile

# ls -l tmpfile
-rw-r--r-- 1 himanshu friends 0 2012-05-22 20:03 tmpfile
```

If you observe closely, the group of the file changed from ‘family’ to ‘friends’. So we see that by just adding a ‘:’ followed by the new group name, the group of the file can be changed.

3. Change both owner and the group

```
# ls -l tmpfile
-rw-r--r-- 1 root family 0 2012-05-22 20:03 tmpfile

# chown himanshu:friends tmpfile

# ls -l tmpfile
-rw-r--r-- 1 himanshu friends 0 2012-05-22 20:03 tmpfile
```

So we see that using the syntax ‘<newOwner>:<newGroup>’, the owner as well as group can be changed in one go.

4. Using chown command on symbolic link file

Here is a symbolic link :

```
 # ls -l tmpfile_symlnk
lrwxrwxrwx 1 himanshu family 7 2012-05-22 20:03 tmpfile_symlnk -> tmpfile
```

So we see that the symbolic link ‘tmpfile_symlink’ links to the file ‘tmpfile’.

Lets see what happens if chown command is issued on a symbolic link:

```
# chown root:friends tmpfile_symlnk

# ls -l tmpfile_symlnk
lrwxrwxrwx 1 himanshu family 7 2012-05-22 20:03 tmpfile_symlnk -> tmpfile

# ls -l tmpfile
-rw-r--r-- 1 root friends 0 2012-05-22 20:03 tmpfile
```

When the chown command was issued on symbolic link to change the owner as well as the group then its the referent of the symbolic link ie ‘tmpfile’ whose owner and group got changed. This is the default behavior of the chown command. Also, there exists a flag ‘–dereference’ for the same.

5. Using chown command to forcefully change the owner/group of symbolic file.

Using flag ‘-h’, you can forcefully change the owner or group of a symbolic link as shown below.

```
# ls -l tmpfile_symlnk
lrwxrwxrwx 1 himanshu family 7 2012-05-22 20:03 tmpfile_symlnk -> tmpfile

# chown -h root:friends tmpfile_symlnk
# ls -l tmpfile_symlnk
lrwxrwxrwx 1 root friends 7 2012-05-22 20:03 tmpfile_symlnk -> tmpfile
```

6. Change owner only if a file is owned by a particular user

Using chown "–from" flag, you can change the owner of a file, only if that file is already owned by a particular owner.

```
# ls -l tmpfile
-rw-r--r-- 1 root friends 0 2012-05-22 20:03 tmpfile

# chown --from=guest himanshu tmpfile

# ls -l tmpfile
-rw-r--r-- 1 root friends 0 2012-05-22 20:03 tmpfile

# chown --from=root himanshu tmpfile

# ls -l tmpfile
-rw-r--r-- 1 himanshu friends 0 2012-05-22 20:03 tmpfile
```

- In the example above, we verified that the original owner/group of the file 'tmpfile' was root/friends.
- Next we used the '–from' flag to change the owner to 'himanshu' but only if the existing owner is 'guest'.
- Now, as the existing owner was not 'guest'. So, the command failed to change the owner of the file.
- Next we tried to change the owner if the existing owner is 'root' (which was true) and this time command was successful and the owner was changed to 'himanshu'.

7. Change group only if a file already belongs to a certain group

Here also the flag '–from' is used but in the following way:

```
# ls -l tmpfile
-rw-r--r-- 1 himanshu friends 0 2012-05-22 20:03 tmpfile

# chown --from=:friends :family tmpfile

# ls -l tmpfile
-rw-r--r-- 1 himanshu family 0 2012-05-22 20:03 tmpfile
```

Since the file 'tmpfile' actually belonged to group 'friends' so the condition was correct and the command was successful.

So we see that by using the flag '–from=:<conditional-group-name>' we can change the group under a particular condition.

NOTE: By following the template '–from=<conditional-owner-name>:<conditional-group-name>', condition on both the owner and group can be applied.

8. Copy the owner/group settings from one file to another

This is possible by using the '–reference' flag.

```
# ls -l file
-rwxr-xr-x 1 himanshu family 8968 2012-04-09 07:10 file

# ls -l tmpfile
-rw-r--r-- 1 root friends 0 2012-05-22 20:03 tmpfile

# chown --reference=file tmpfile

# ls -l tmpfile
-rw-r--r-- 1 himanshu family 0 2012-05-22 20:03 tmpfile
```

In the above example, we first checked the owner/group of the reference-file 'file' and then checked the owner/group of the target-file 'tmpfile'. Both were different. Then we used the chown command with the '–reference' option to apply the owner/group settings from the reference file to the target file. The command was successful and the owner/group settings of 'tmpfile' were made similar to the 'file'.

9. Change the owner/group of the files by traveling the directories recursively

This is made possible by the '-R' option.

```
# ls -l linux/linuxKernel
-rw-r--r-- 1 root friends 0 2012-05-22 21:52 linux/linuxKernel

# ls -l linux/ubuntu/ub10
-rw-r--r-- 1 root friends 0 2012-05-22 21:52 linux/ubuntu/ub10

# ls -l linux/redhat/rh7
-rw-r--r-- 1 root friends 0 2012-05-22 21:52 linux/redhat/rh7

# chown -R himanshu:family linux/

# ls -l linux/redhat/rh7
-rw-r--r-- 1 himanshu family 0 2012-05-22 21:52 linux/redhat/rh7

# ls -l linux/ubuntu/ub10
-rw-r--r-- 1 himanshu family 0 2012-05-22 21:52 linux/ubuntu/ub10

# ls -l linux/linuxKernel
-rw-r--r-- 1 himanshu family 0 2012-05-22 21:52 linux/linuxKernel
```

So we see that after checking the owner/group of all the files in the directory 'linux' and its two sub-directories 'ubuntu' and 'redhat'. We issued the chown command with the '-R' option to change both the owner and group. The command was successful and owner/group of all the files was changed successfully.

10. Using chown command on a symbolic link directory

Lets see what happens if we issue the 'chown' command to recursively change the owner/group of files in a directory that is a symbolic link to some other directory.

Here is a symbolic link directory 'linux_symlnk' that links to the directory 'linux' (already used in example '9' above) :

```
$ ls -l linux_symlnk
lrwxrwxrwx 1 himanshu family 6 2012-05-22 22:02 linux_symlnk -> linux/
```

Now, lets change the owner (from himanshu to root) of this symbolic link directory recursively :

```
# chown -R root:friends linux_symlnk

# ls -l linux_symlnk/
-rw-r--r-- 1 himanshu friends    0 2012-05-22 21:52 linuxKernel
drwxr-xr-x 2 himanshu friends 4096 2012-05-22 21:52 redhat
drwxr-xr-x 2 himanshu friends 4096 2012-05-22 21:52 ubuntu
```

In the ouput above we see that the owner of the files and directories was not changed. This is because by default the 'chown' command cannot traverse a symbolic link. This is the default behavior but there is also a flag '-P' for this.

11. Using chown to forcefully change the owner/group of a symbolic link directory recursively

This can be achieved by using the flag -H

```
# chown -R -H guest:family linux_symlnk

# ls -l linux_symlnk/
total 8
-rw-r--r-- 1 guest family    0 2012-05-22 21:52 linuxKernel
drwxr-xr-x 2 guest family 4096 2012-05-22 21:52 redhat
drwxr-xr-x 2 guest family 4096 2012-05-22 21:52 ubuntu
```

So we see that by using the -H flag, the owner/group of all the files/folder were changed.

12. List all the changes made by the chown command

Use the verbose option -v, which will display whether the ownership of the file was changed or retained as shown below.

```
# chown -v -R guest:friends linux
changed ownership of `linux/redhat/rh7' to guest:friends
changed ownership of `linux/redhat' retained to guest:friends
ownership of `linux/redhat_sym' retained as guest:friends
ownership of `linux/ubuntu_sym' retained as guest:friends
changed ownership of `linux/linuxKernel' to guest:friends
changed ownership of `linux/ubuntu/ub10' to guest:friends
ownership of `linux/ubuntu' retained as guest:friends
ownership of `linux' retained as guest:friends
```

class="title single-title"

chgrp command & Options

CHGRP(CHange GRouP) is one more command which is useful to change group associated to a file/folder from one group to other in a Linux box. This is sister command to **chown** which is used to change owner of the file/folder as well as group name associated with that file.

Learn chgrp with examples:

Example1: Change group name:sales of a file to other group name:hrgroup.

```
chgrp hrgroup file1
```

Example2: Give access permissions to a command so that the command can be executed by all users belonging to apache-admins

```
chgrp apache-admins /etc/init.d/httpd
```

Example3: Change group ownership all the files located in /var/apache to group:apache

```
chgrp -R apache /var/apache
```

Example4: Change group ownership forcefully

```
chgrp -f apache /var/apache
```

Difference between chown and chgrp

1) **chown** command is used to change ownership as well as group name associated to different one, where as **chgrp** can change only group associated to it.

2) Many people say that regular user only able to use **chgrp** to change the group if the user belongs to them. But it's not true a user can use **chown** and **chgrp** irrespective to change group to one of their group because chown is located in /bin folder so every can use it with some limited access.

Usages of chgrp command:

1)Used to change group ownership from one group to other group for a file/folder

2)As a security measure if you want to give permissions to a command to some group you can use this command.

User and Group Management Commands

The passwd command

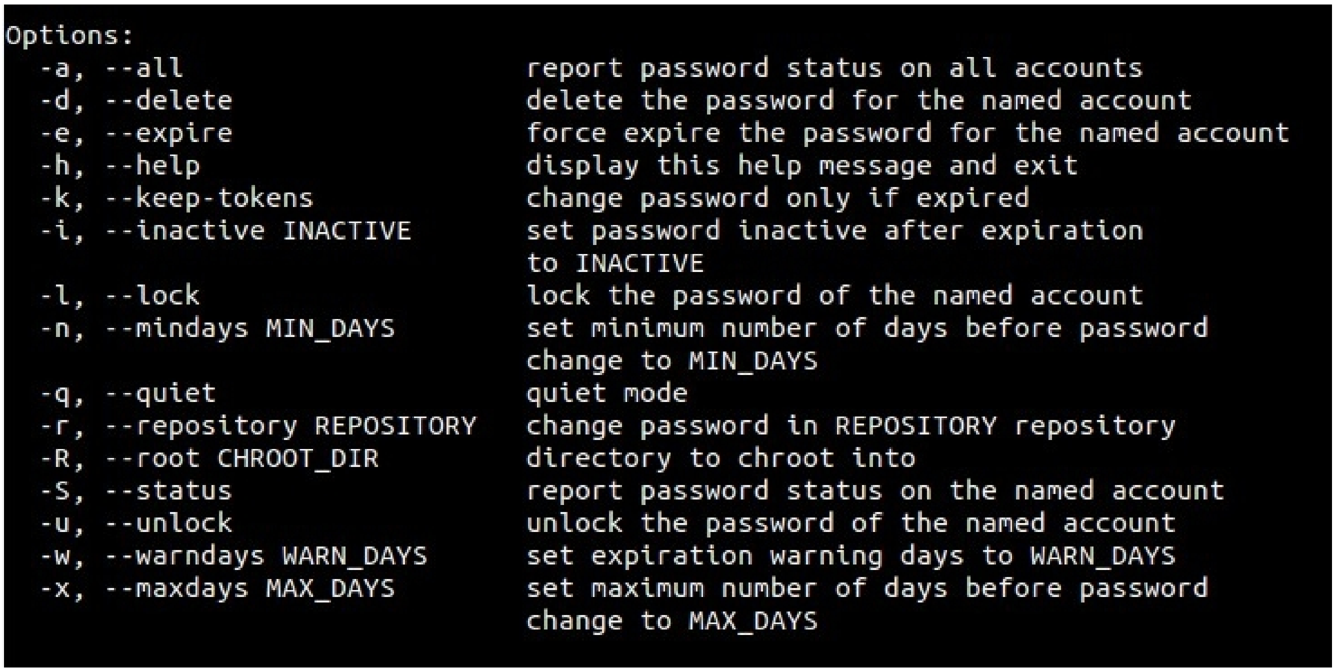
As the name suggest **passwd** command is used to change the password of system users. If the passwd command is executed by non-root user then it will ask for the current password and then set the new password of a user who invoked the command. When this command is executed by super user or root then it can reset the password for any user including root without knowing the current password.

In this post we will discuss passwd command with practical examples.

Syntax :

```
# passwd {options} {user_name}
```

Different options that can be used in passwd command are listed below :



```
Options:
  -a, --all                     report password status on all accounts
  -d, --delete                  delete the password for the named account
  -e, --expire                  force expire the password for the named account
  -h, --help                    display this help message and exit
  -k, --keep-tokens             change password only if expired
  -i, --inactive INACTIVE       set password inactive after expiration
                                to INACTIVE
  -l, --lock                    lock the password of the named account
  -n, --mindays MIN_DAYS        set minimum number of days before password
                                change to MIN_DAYS
  -q, --quiet                   quiet mode
  -r, --repository REPOSITORY   change password in REPOSITORY repository
  -R, --root CHROOT_DIR         directory to chroot into
  -S, --status                  report password status on the named account
  -u, --unlock                  unlock the password of the named account
  -w, --warndays WARN_DAYS      set expiration warning days to WARN_DAYS
  -x, --maxdays MAX_DAYS        set maximum number of days before password
                                change to MAX_DAYS
```

Example:1 Change Password of System Users

When you logged in as non-root user like ‘linuxtechi’ in my case and run passwd command then it will reset password of logged in user.

```
[linuxtechi@linuxworld ~]$ passwd
Changing password for user linuxtechi.
Changing password for linuxtechi.
(current) UNIX password:
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[linuxtechi@linuxworld ~]$
```

When you logged in as root user and run **passwd** command then it will reset the root password by default and if you specify the user-name after passwd command then it will change the password of that user.

```
[root@linuxworld ~]# passwd
[root@linuxworld ~]# passwd linuxtechi
```

Note : System user’s password is stored in an encrypted form in /etc/shadow file.

Example:2 Display Password Status Information.

To display password status information of a user , use **-S** option in passwd command.

```
[root@linuxworld ~]# passwd -S linuxtechi
linuxtechi PS 2015-09-20 0 99999 7 -1 (Password set, SHA512 crypt.)
[root@linuxworld ~]#
```

In the above output first field shows the user name and second field shows Password status (**PS** = **Password Set** , **LK** = **Password locked** , **NP** = **No Password**), third field shows when the password was changed and last & fourth field shows minimum age, maximum age, warning period, and inactivity period for the password

Example:3 Display Password Status info for all the accounts

To display password status info for all the accounts use “**-aS**” option in passwd command, example is shown below :

```
root@localhost:~# passwd -Sa
```

Example:4 Removing Password of a User using -d option

In my case i am removing/ deleting the password of ‘**linuxtechi**‘ user.

```
[root@linuxworld ~]# passwd -d linuxtechi
Removing password for user linuxtechi.
passwd: Success
[root@linuxworld ~]#
[root@linuxworld ~]# passwd -S linuxtechi
linuxtechi NP 2015-09-20 0 99999 7 -1 (Empty password.)
[root@linuxworld ~]#
```

“**-d**” option will make user’s password empty and will disable user’s account.

Example:5 Set Password Expiry Immediately

Use **'-e'** option in passwd command to expire user's password immediately , this will force the user to change the password in the next login.

```
[root@linuxworld ~]# passwd -e linuxtechi
Expiring password for user linuxtechi.
passwd: Success
[root@linuxworld ~]# passwd -S linuxtechi
linuxtechi PS 1970-01-01 0 99999 7 -1 (Password set, SHA512 crypt.)
[root@linuxworld ~]#
```

Now Try to ssh machine using linuxtechi user.

Example:6 Lock the password of System User

Use '**-l**' option in passwd command to lock a user's password, it will add "**!**" at starting of user's password. A User can't Change it's password when his/her password is locked.

```
[root@linuxworld ~]# passwd -l linuxtechi
Locking password for user linuxtechi.
passwd: Success
[root@linuxworld ~]# passwd -S linuxtechi
linuxtechi LK 2015-09-20 0 99999 7 -1 (Password locked.)
[root@linuxworld ~]#
```

Example:7 Unlock User's Password using -u option

```
[root@linuxworld ~]# passwd -u linuxtechi
Unlocking password for user linuxtechi.
passwd: Success
[root@linuxworld ~]#
```

Example:8 Setting inactive days using -i option

-i option in passwd command is used to set inactive days for a system user. This will come into the picture when password of user (in my case linuxtechi) expired and user didn't change its password in '**n**' number of days (i.e 10 days in my case) then after that user will not able to login.

```
[root@linuxworld ~]# passwd -i 10 linuxtechi
Adjusting aging data for user linuxtechi.
passwd: Success
[root@linuxworld ~]#
[root@linuxworld ~]# passwd -S linuxtechi
linuxtechi PS 2015-09-20 0 99999 7 10 (Password set, SHA512 crypt.)
[root@linuxworld ~]#
```

Example:9 Set Minimum Days to Change Password using -n option.

In the below example linuxtechi user has to change the password in 90 days. A value of zero shows that user can change it's password in any time.

```
[root@linuxworld ~]# passwd -n 90 linuxtechi
```

```
Adjusting aging data for user linuxtechi.
passwd: Success
[root@linuxworld ~]# passwd -S linuxtechi
linuxtechi PS 2015-09-20 90 99999 7 10 (Password set, SHA512 crypt.)
[root@linuxworld ~]#
```

Example:10 Set Warning days before password expire using -w option

'-w' option in passwd command is used to set warning days for a user. It means a user will be warned for n number of days that his/her password is going to expire.

```
[root@linuxworld ~]# passwd -w 12 linuxtechi
Adjusting aging data for user linuxtechi.
passwd: Success
[root@linuxworld ~]# passwd -S linuxtechi
linuxtechi PS 2015-09-20 90 99999 12 10 (Password set, SHA512 crypt.)
[root@linuxworld ~]#
```

The USERADD command in linux

Example 1.The most used useradd command:

```
$ sudo useradd -m -d /home/mike1 -s /bin/bash -c "the mike1 user" -U
mike1
```

Explanation:

- **-m -d** /home/mike1 : the -m argument creates the /home/mike1 homedirectory, specified by the -d argument
- **-s** /bin/bash : the -s is used for specifing the user's default shell, /bin/bash in this case
- **-c** "message" : extra information about the user

Example 2:

```
$ sudo useradd -m -d /home/geeks/mike2 -s /bin/zsh -c "the mike2
user" -u 1099 -g 1050 mike2
```

Explanation:

- **-m -d** /home/geeks/mike2 : the -m argument creates the /home/geeks/mike2 homedirectory, specified by the -d argument . as you can notice, the homedir can be different that /home/user_name
- **-s** /bin/zsh : the -s is used for specifing the user's default shell, /bin/zsh in the case
- **-c** "the mike2 user" : extra information about the user
- **-u** 1099 : the new user's UID, in this case 1099
- **-g** 1050 : the user belongs to the group with the 1050 GID

Example 3:

$ sudo useradd -m -d /home/mike3 -s /usr/sbin/nologin -c "nologin user" -u 1098 mike3

Explanation:

- **-m -d** /home/mike3 : the -m argument creates the /home/mike3 homedirectory, specified by the

-d argument
- **-s** /usr/sbin/nologin : the -s is used for specifing the user's default shell, in this case /usr/sbin/nologin . mike3 cannot login to the system with su, but can login by ssh.
- **-c** "nologin user" : extra information about the user
- **-u** 1098 : the new user's UID, in this case 1098

Example 4:

```
$ sudo useradd -m -d /home/mike4 -k /etc/custom.skell -s /bin/tcsh -c "mike4 user" -u 1097 mike4
```

Explanation:

- **-m -d** /home/mike4 : the -m argument creates the /home/mike4 homedirectory, specified by the -d argument
- **-s** /bin/tcsh : the -s is used for specifing the user's default shell, /bin/tcsh in this case
- **-k** /etc/custom.skel : another skeleton directory, /etc/custom.skell in this case, different than the default skeleton directory /etc/skel
- **-c** "mike4 user" : extra information about the user
- **-u** 1097 : the new user's UID, in this case 1097

Example 5:

```
$ sudo useradd -M -N -r -s /bin/false -c "system user" sys_user
```

Explanation:

- **-M** : the -M argument tells the system not to create a home directory
- **-N** : the -N argument tells the system not to create a group having the user's name
- **-r** : the -r arguments is for creating a system user
- **-s** /bin/false : the -s is used for specifing the user's default shell, /bin/false in this case. The users having /bin/false as the default shell, cannot login to the system.
- **-c** "system user" : extra information about the user

Q: How do you know what default values would be assigned to a user when created using useradd

A: These are the two files which contain the default values to be assigned to a user when created using useradd

```
# less /etc/default/useradd
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/href="javascript:void(0);" style="color:#000FFF;text-decoration:underline"
id="Y1697543S6"bin/bash
SKEL=/etc/skel
CREATE_MAIL_SPOOL=yes
```

You can also view the **default parameters** set for new user to be created using

```
# useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
```

```
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
CREATE_MAIL_SPOOL=yes
```

The second file containing values used by useradd command for UID, GID, password encryption method and expiry related information

```
# less /etc/login.defs
MAIL_DIR        /var/spool/mail

PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_MIN_LEN    5
PASS_WARN_AGE   7

UID_MIN                   500
UID_MAX                 60000

GID_MIN                   500
GID_MAX                 60000

CREATE_HOME     yes
UMASK           077

USERGROUPS_ENAB yes
ENCRYPT_METHOD SHA512
```

1. How to change default values of useradd command?

Either you can open `/etc/default/useradd` file and edit the file or you can also do the same using CLI as shown below

To change the **default home directory** location for all new users

```
# useradd -D -b /opt/users
# useradd -D | grep HOME
HOME=/opt/users
```

To change the **default login shell**

```
# useradd -D -s /bin/sh
# useradd -D | grep -i shell
SHELL=/bin/sh
```

Now what if you want to add custom arguments to your user while creating them. let us discuss in detail the different options which you can use along with useradd command

2. Create multiple users with same UID

```
# useradd -o deepak -u 501
# useradd -o deep -u 501
# useradd -o user -u 501



# grep 501 /etc/passwd
deepak:x:501:501::/home/deepak:/bin/sh
deep:x:501:504::/home/deep:/bin/sh
```

```
user:x:501:505::/home/user:/bin/sh
```

3. Manually assign a UID to the user

By default a user automatically gets any free uid more than 500 when you run the useradd command. But what if you manually want to assign a uid to your user

```
# useradd -u 550 deepak
```

Let us verify the assigned uid to deepak

```
# id deepak
uid=550(deepak) gid=550(deepak) groups=550(deepak)
```

4. Create user without home directory

```
# useradd -M test
# su - test
su: warning: cannot change directory to /home/test: No such file or directory

-bash-4.1$ pwd
/root
```

5. Create user with custom defined home directory

```
# useradd -d /home/users/test test
# su - test

$ pwd
/home/users/test
```

6. Add user to different primary group

By default when you run useradd command, a group with the same name is created inside `/etc/group` but what if you donot want a group to be created with the same name instead add the user to some different already existing group.

Here we will create a user "`deep`" and add him to group "`admin`" without creating another "`deep`" group

```
# useradd -g admin deep
```

Verify the groups of user "deep"

```
# groups deep
deep : admin
```

7. Add user to different secondary group

In the above command you saw if we are mentioning a different primary group while using useradd command then a default group with the name of user is NOT created. Now what if you want a group with username's to be created but instead you want the user to add some secondary group.

Here user `deepak` is created along with group `deepak` but also in the same command we are adding `deepak` to `dba` group

```
# useradd  -G dba deepak
```

Verify groups of deepak

```
# groups deepak
deepak : deepak dba
```

8. Add user to multiple groups

You can add the user to multiple secondary groups using single command

```
# useradd -G admin,dba deepak
```

Verify

```
# groups deepak
deepak : deepak admin dba
```

9. Manually assign a shell to user

Be default when you create a user in Red Hat Linux the user gets /bin/bash shell. But in case you want to give them some other shell for login use the below command

```
# useradd -s /bin/sh  deepak
# su - deepak

-sh-4.1$ echo $SHELL
/bin/sh
-sh-4.1$
```

10. Creating a user along with encrypted password

Now you can create a user with pre-defined password, but the condition is the password used should be href="javascript:void(0);" style="color:#000FFF;text-decoration:underline" id="Y1697543S8"encrypted which you can do with various methods. Here I will show you one method to do so

Encrypt your password using below command

```
# openssl passwd -crypt mypassw0rd
Warning: truncating password to 8 characters
TuUFdiN1KaCHQ
```

Now you can use the encrypted password for your new user

```
# useradd -p TuUFdiN1KaCHQ deepak
```

Try to login to the user, for which the password would be "**mypassw0rd**"

Usermod Command & Options

Linux command "usermod" is used to modify a user's information. The files that may be affected during this operation are **/etc/passwd** (user account information), **/etc/shadow** (secure account information) and **/etc/group** (group information).

Only root/super user can use this command.

The basic syntax of this command is as follows.

usermod [-c comment] [-d home_dir [-m]] [-e expire_date] [-f inactive_time][-g initial_group] [-

G group [,...]] [-l login_name] [-p passwd][-s shell] [-u uid [-o]] [-L|-U]login

In this article, we will go through some example usages of “usermod” command which will help you to learn these options in detail.

First we can create a user “test” using useradd. In order to view user information, we can use the “id command.

```
# id test
uid=501(test) gid=501(test) groups=501(test)
```

Example 1: Changing the home directory of user “test”

Suppose the current home directory of the user “test” is /home/test and you want to change it to the existing directory /home/testnew without copying the contents of /home/test, you can use the following command.

```
#usermod –d /home/testnew test
```

If you want to move the contents of /home/test also (if the new directory doesn’t exist, it will create and move), you need to use the option “-m”.

```
#usermod –d /home/testnew –m test
```

Example 2: Adding groups to a user

When a user is added using “useradd” command without specifying group, then a group with the same name as that of user will be created. This is the primary group of the user. You can add as many groups to a user using the option “-G” as follows.

Suppose, you need to add a group “developer” to the user “test”, you can add it as follows.

```
#usermod –G developer test
```

Please note that, if you added the user to any other groups earlier (other than the primary group), that will get removed by the above command.

So, if you want to preserve the current groups of the user and add one more group you need to use the option –aG as follows.

```
#usermod –aG developer test
# id test
uid=501(test) gid=501(test) groups=501(test),506(pros),508(developer)
```

Example 3: Changing the primary group of a user

If you want to add a group as the primary group of the user, you can do it as follows.

```
# usermod –g developer test
# id test
uid=501(test) gid=508(developer) groups=508(developer), 506(pros)
```

Example 4: Locking and Unlocking users

In some cases, you may need to temporarily lock the account. This can be done with the “-L” option. This puts a '!' in front of the encrypted password, effectively disabling the password.

```
# usermod –L test
```

User can be unlocked as follows which will remove the '!' in front of the encrypted password.

```
# usermod –U test
```

Example 5: Changing the expiry data of an account

You can use the following command to disable the account “test” on 2012-12-01.

```
usermod -e 2012-12-01 test
```

Example 6: Changing login and password using usermod

You can change the login name itself using the –l switch.

```
# usermod –l newtest test

# id test
Id: test: No such user
# id newtest
uid=501(newtest) gid=508(developer) groups=508(developer), 506(pros)
```

You can change the password as follows.

```
# usermod –p newpass newtest
```

Example 7: Changing shell of a user

The “shell” provided to a user can be changed as follows. This will change the shell of “newtest” user to “/bin/bash”.

```
# usermod –s /bin/bash newtest
```

Userdel Command & Options

Maintaining users on the server means add them, modify them and delete them. When a user is no longer need on the system for any reasons, we should delete it to avoid security breach. On Linux system we have **userdel** command to delete a user

What is userdel

Userdel is a low level utility for removing users. On Debian, we should usually use deluser command. Userdel will look the system account files such as **/etc/password** and **/etc/group**. Then it will deleting all entries related to the user name. The user name must exist before we can delete it.

How to use Userdel

Since userdel will modify system account files, we **need root privilege** to run it. Otherwise we will have an error message that saying *"only root can do that"* or similar. After we gain root privilege, we can delete a user by typing userdel from your console. Here's a sample of default usage of userdel

```
$ sudo userdel pasadena
```

or

```
# userdel pasadena
```

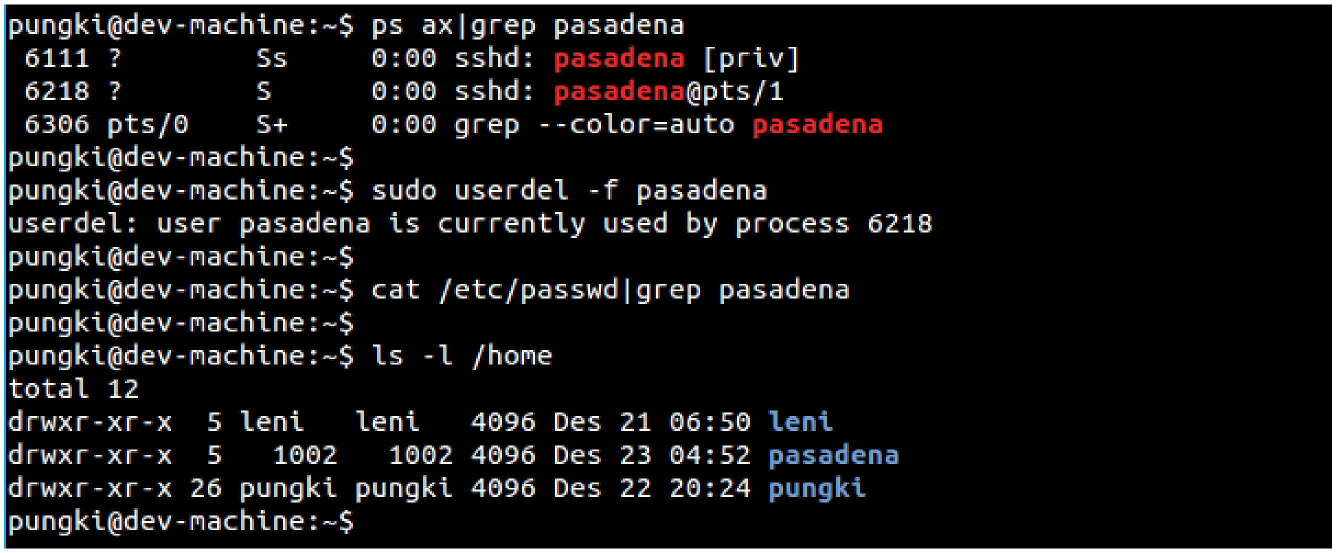
As you see above, we can't delete a user with name pasadena without root privilege. When we have it, system give us no error which mean that user is deleted successfully.

Completely remove user home directory

- Using userdel without options, will only delete the user. User home directory will still remain at /home folder.
- When we go into /home folder, we still seeing pasadena folder which owned by 1002. Created user will usually have a same group name with user name. 1002 was the UID and GID of pasadena user name and pasadena group name.
- To completely remove the home user along user deletion, we can use **-r** option. This option will also delete user's mail spool if exist.

Force delete a user

Userdel provide **-f** option to force user deletion. This option will delete a user even the user still log in into Linux system. Please take a look a sample screenshot.



```
pungki@dev-machine:~$ ps ax|grep pasadena
 6111 ?        Ss     0:00 sshd: pasadena [priv]
 6218 ?        S      0:00 sshd: pasadena@pts/1
 6306 pts/0    S+     0:00 grep --color=auto pasadena
pungki@dev-machine:~$
pungki@dev-machine:~$ sudo userdel -f pasadena
userdel: user pasadena is currently used by process 6218
pungki@dev-machine:~$
pungki@dev-machine:~$ cat /etc/passwd|grep pasadena
pungki@dev-machine:~$
pungki@dev-machine:~$ ls -l /home
total 12
drwxr-xr-x  5 leni   leni   4096 Des 21 06:50 leni
drwxr-xr-x  5   1002   1002 4096 Des 23 04:52 pasadena
drwxr-xr-x 26 pungki pungki 4096 Des 22 20:24 pungki
pungki@dev-machine:~$
```

Screenshot above show us that pasadena user is logged in to Linux system. It is marked by process **6218** which is SSHD process. But when we do ***"userdel -f pasadena"*** the command only show us the information that the user is logged in. The command itself was succeed. If we see the content of **/etc/passwd** file using cat command, we don't see pasadena user there. It's home directory still exist but the owner is changed.

One thing that we must know that, userdel with **-f** option **did not broke** the SSH connection. So the user actually still logged in and active even the user is not exist. But when the user log off, the the user can not log in anymore because that user has been deleted.

So **this options is dangerous to use** since it can lead your system into inconsistent state.

Chage Command & Options

The command name 'chage' is an acronym for 'change age'. This command is used to change the user password aging / expiry information. Any user can execute this command with '-l' option to view their password aging information. No users can view the password aging/expiry information of other users. As root, you can execute this command to modify the users aging information.

Chage command syntax

```
chage [-m mindays] [-M maxdays] [-d lastday] [-I inactive] [-E expiredate] [-W warndays]
user
```

We can go through some examples to get a better understanding of this command.

TASK 1: Use chage command to list the password aging information of a user

```
chage –l testuser
Output:
Last password change : May 01, 2012
Password expires : never
Password inactive : never
Account expires : never
Minimum number of days between password change : 0
Maximum number of days between password change : 99999
Number of days of warning before password expires : 7
```

As you can see, password expiration is disabled for this user.

TASK 2: Disable password aging for a user

```
chage -I -1 -m 0 -M 99999 -E -1 testuser
```

• -I -1 : This will set the "Password inactive" to never

• -m 0 : This will set the minimum number of days between password change to 0

• -M 99999 : This will set the maximum number of days between password change to 99999

• -E -1 : This will set "Account expires" to never.

This will disable the password expiry of a user if it is already enabled.

TASK 3: Enable password expiry date of a user

In most cases, as an administrator, you need to set password expiry for all users for better security. Once you enable password expiry date for a user, the user will be forced to change their password from the next login after the expiry date.

```
chage -M 20 testuser
Output
Last password change : May 01, 2012
Password expires : May 21, 2012
Password inactive : never
Account expires : never
Minimum number of days between password change : 0
Maximum number of days between password change : 20
Number of days of warning before password expires : 7
```

TASK 4 : Set the Account expiry date in the format 'YYYY-MM-DD'

```
chage –E "2012-05-28"

Output
Last password change : May 01, 2012
Password expires : May 21, 2012
Password inactive : never
Account expires : May 28, 2012
Minimum number of days between password change : 0
Maximum number of days between password change : 20
Number of days of warning before password expires : 7
```

TASK 5: Set the password expiry warning message

By default, this value is set to 7. So, when a user logins prior to 7 days of expiry, he will start to get warning about password expiry. If you want it to change to 10 days, you can do it as follows.

```
chage –W 10 testuser
```

TASK 6: Forcing the users to change the password on next logon

When you create a new user account, you can set it to force the user to change the password when he login for the first time as follows,

```
chage –d 0 testuser
```

This will reset “Last Password Change” to “Password must be changed”

How to create groups in Linux with: groupadd

```
$ sudo groupadd mikegr
$ < /etc/group grep mikegr
mikegr:x:1022:
```

With groupadd and no arguments, a new group will be with the default setting. The GID will be the first one available, bigger than 1000 (because from 1 to 999 the GIDs are reserved).

You can also create a group with a customized GID:

```
$ sudo groupadd -g 1033 starwarsmovie
$ < /etc/group grep starwarsmovie starwarsmovie:x:1033:
```

It is not recommended to create a password protected group with groupadd -p because you're password will be stored in **/etc/gshadow** in clear format, not encrypted. (unless you give groupadd -p a crypted password as an argument)

```
$ sudo groupadd -p "linuxg.net" geekster
$ < /etc/group grep geekster geekster:x:1035: $ sudo < /etc/gshadow
grep geekster geekster:linuxg.net::
```

Deleting a group is very simple with groupdel:

```
$ sudo groupdel geekster
```

We did not get any output, so the group has been succesfully deleted.
If you need to change the group settings, use **groupmod**:

```
$ groupmod
Usage: groupmod [options] GROUP

Options:
-g, --gid GID change the group ID to GID
-h, --help display this help message and exit
-n, --new-name NEW_GROUP change the name to NEW_GROUP
```

```
-o, --non-unique allow to use a duplicate (non-unique) GID
-p, --password PASSWORD change the password to this (encrypted)
PASSWORD
```

How to create users with custom primary and secondary groups:

I will create the groups and set them as primary and secondary groups for the new created user:

Creating the groups jim, michael and johnson:

```
$ sudo groupadd jim
$ sudo groupadd michael
$ sudo groupadd johnson
```

Verifing if the groups have been created, and finding out their GIDs:

```
$ tail -3 /etc/group
jim:x:1039:
michael:x:1040:
johnson:x:1041:
```

Creating the test3 user with jim as primary group and michael and johnson as secondary groups with useradd:
the -g parameter is for adding the user to the primary group, and -G for the secondary groups. Only one group can be set as primary group.

```
$ sudo useradd -g jim -G michael,johnson test3
$ id test3
uid=1015(test3) gid=1039(jim)
groups=1039(jim),1040(michael),1041(johnson)
```

The GID displayed in the id commands's output is the ID of the primary group.

Creating the test4 user with the same primary and secondary groups as test3, by using the GIDs:

```
$ sudo useradd -g 1039 -G 1040,1041 test4
$ id test4
uid=1016(test4) gid=1039(jim)
groups=1039(jim),1040(michael),1041(johnson)
```

How to add an existing user to primary and secondary groups:

I will create the user test5 and add him in primary and secondary groups, by using the names and the GIDs.

```
$ sudo usermod -g group3 -G group4,group5 test5
$ sudo -g 3000 -G 4000,5000 test6
```

Usermod is used for changing the user account information. The usermod command's parameters are the same as the useradd parameters: -g for primary group and -G for secondary groups.

How to remove users from secondary groups:

The gpasswd command is used for working with groups.

How to remove a user from a group with gpasswd: **gpasswd -d username groupname**.

```
$ id test4
uid=1016(test4) gid=1039(jim)
groups=1039(jim),1040(michael),1041(johnson)
$ sudo gpasswd -d test4 johnson
Removing user test4 from group johnson
$ id test4
uid=1016(test4) gid=1039(jim) groups=1039(jim),1040(michael)
```

To remove a user's primary group, set a new group as primary for that user and after that, remove the user from the old primary group.

The chgrp Command Examples

This tutorial explains Linux "chgrp" command, options and its usage with examples.

"chgrp" command is used to change group ownership of a file/directory. This post describes "chgrp" command used in Linux along with usage examples and/or output.

All files in linux belong to an owner, and a group. The owner is set by the "chown" command, and the group by the "chgrp" command.

Usage:
chgrp [OPTION]… GROUP FILE…
chgrp [OPTION]… –reference=RFILE FILE…

Here's the listing of example usage of command. :

Note:
Before using "chgrp" command, You should be aware of the users existing in your system, otherwise you will get "Invalid user name" Error.
To see the list of the users, You may use(cat /etc/group). But if you still want to add a particular user , then run in the terminal following command and then use "chgrp" command accordingly:

```
  sudo addgroup user_name_to_add
```

1. To change the group name of a file(chgrp group_name file_name):

```
sanfoundry-> touch 1.txt
sanfoundry-> ls -l 1.txt
-rw-rw-r-- 1 himanshu himanshu 0 Jun 12 18:57 1.txt
sanfoundry-> sudo chgrp user 1.txt
sanfoundry-> ls -l 1.txt
-rw-rw-r-- 1 himanshu user 0 Jun 12 18:57 1.txt
sanfoundry->
```

As you can see, Here user for the 1.txt was "himanshu" and after executing the "chgrp" command, user_name has been changed to "user".

2. To change the group name of a folder(chgrp group_name folder_name):

```
sanfoundry-> mkdir sample_folder
sanfoundry-> ls -l
```

```
drwxrwxr-x 2 himanshu himanshu  4096 Jun 12 19:04 sample_folder

sanfoundry-> sudo chgrp user sample_folder/
sanfoundry-> ls -l
drwxrwxr-x 2 himanshu user      4096 Jun 12 19:04 sample_folder
```

As you can see, Here user for the folder sample_folder was “himanshu” and after executing the “chgrp” command, user_name has been changed to “user”.

3. To change the group name of all contents in a folder(chgrp -R group_name folder_name): “-R” option is for the recursive changes in the folder.

```
sanfoundry-> mkdir sample_folder
sanfoundry-> cd sample_folder/
sanfoundry-> touch x y
sanfoundry-> mkdir folder1
sanfoundry-> cd ..
sanfoundry-> ls -l sample_folder/
total 4
drwxrwxr-x 2 himanshu himanshu 4096 Jun 12 19:09 folder1
-rw-rw-r-- 1 himanshu himanshu    0 Jun 12 19:09 x
-rw-rw-r-- 1 himanshu himanshu    0 Jun 12 19:09 y
sanfoundry-> sudo chgrp -R user sample_folder/
sanfoundry-> ls -l sample_folder/
total 4
drwxrwxr-x 2 himanshu user 4096 Jun 12 19:09 folder1
-rw-rw-r-- 1 himanshu user    0 Jun 12 19:09 x
-rw-rw-r-- 1 himanshu user    0 Jun 12 19:09 y
```

4. To print the verbose messages when changes made in the ownership(chgrp -c group_name file/folder name):

```
sanfoundry-> touch 1.txt
sanfoundry-> sudo chgrp -c user 1.txt
changed group of `1.txt' from himanshu to user
```

5. To use group name of a file instead of specifying a particular group_name(chgrp -–reference=file1 file2):

```
sanfoundry-> touch 1.txt 2.txt
sanfoundry-> sudo chgrp user 1.txt
sanfoundry-> ls -l 1.txt 2.txt
-rw-rw-r-- 1 himanshu user     0 Jun 12 19:19 1.txt
-rw-rw-r-- 1 himanshu himanshu 0 Jun 12 19:19 2.txt
sanfoundry-> sudo chgrp --reference=1.txt 2.txt
sanfoundry-> ls -l 1.txt 2.txt
-rw-rw-r-- 1 himanshu user 0 Jun 12 19:19 1.txt
-rw-rw-r-- 1 himanshu user 0 Jun 12 19:19 2.txt
```

As you can see from the above example, 1.txt has group name “user” and with –reference option that has been given to the file “2.txt” and now both shares a common group name,”user”.

6. output a diagnostic for every file processed(chgrp -v group_name file_name):

```
sanfoundry-> touch 1.txt
sanfoundry-> sudo chgrp -v user 1.txt
changed group of `1.txt' from himanshu to user
```

7. To change the group name of link files(chgrp –dereference group_name … file_name):
To create a symbolic link for a file, say 1.txt, You have to use ln -s command:

```
sanfoundry-> touch 1.txt
sanfoundry-> ln -s 1.txt symbolic_link
```

advertisements

undefined

Here file symbolic_link is the link_name for file 1.txt.
* with "-dereference" option the group name of actual file pointed by symbolic_link gets changed.
* with "-no-dereference" option the group name of symbolic_link gets changed itself.

```
sanfoundry-> sudo  chgrp --dereference user symbolic_link
sanfoundry-> ls -l symbolic_link 1.txt
-rw-rw-r-- 1 himanshu user     0 Jun 13 00:43 1.txt
lrwxrwxrwx 1 himanshu himanshu 5 Jun 13 00:43 symbolic_link -> 1.txt

sanfoundry-> sudo  chgrp --no-dereference hello symbolic_link
sanfoundry-> ls -l symbolic_link
lrwxrwxrwx 1 himanshu hello 5 Jun 13 00:43 symbolic_link -> 1.txt
```

From the above example, You can see that with "–dereference" option the group name of the linked file 1.txt has been changed and group name of symbolic_link remains the same. But with "–no-dereference" option the group name of the symbolic link has been changed itself.

8. To change the group name of symbolic link itself without "–no-dereference" option(chgrp -h group_name link_name):

```
sanfoundry-> sudo  hello -h user symbolic_link
sanfoundry-> ls -l symbolic_link
lrwxrwxrwx 1 himanshu hello 5 Jun 13 00:43 symbolic_link -> 1.txt
```

Some more options for chgrp
* "-f" To suppress most of the error messages
* "-RL" To traverse every symbolic link to a directory encountered

Note:
* As a security measure if you want to give permissions to a command to some group you can use this command.
* "chown" command is used to change ownership as well as group name associated to different one, where as "chgrp" can change only group associated to it.