LINUX NOTES

By

Cloudblitz Technologies, Nagpur

CONTENT

RHCSA

Chapter 1: Introduction to Linux

Before learning Linux, we should know what operating system is. An Operating System or OS is the program which provides an environment to run other applications. OS interacts with hardware which means it acts as mediator between hardware and user interface. On the basis of their utilization, OS can be classified as Desktop OS and Server OS. Following are some differences between them,

Desktop Operating System	Server Operating System
• Basically, used in Desktop System. • It runs client applications like Photoshop, Office, Games, etc. faster. • Easy to access web services, like email, browsing, searching • rich media services • easy to use for non-expert users • can work on a domain as a member (professional edition only - home edition cannot join a domain) • Rich connectivity support (LAN, wireless, Bluetooth, etc.) • cheaper than the server version • Ex. Windows7, LinuxMint, Ubuntu Desktop, etc.	• Basically, used in server systems. • unlimited user connections • use of large amounts of memory • can act as web server, database server, email server and other server-like roles • optimized for network, instead of local application execution extended management • extended fault tolerance to avoid downtimes • can hold a domain • expensive • Ex. Windows Server 2008, RHEL, Centos, Debian, etc.

History of Linux:

Evolution of Computer

In earlier days, computers were as big as houses or parks. So, you can imagine how difficult it was to operate them. Moreover, every computer has a different operating system which made it completely worse to operate on them. Every software was designed for a specific purpose and was unable to operate on other computer. It were extremely costly and normal people neither can afford it nor can understand it.

Evolution of UNIX

In 1969, a team of developers of Bell Labs started a project to make a common software for all the computers and named it as 'UNIX'. It was simple and elegant, used 'C' language instead of assembly language and its code was recyclable. As it was recyclable, a part of its code now commonly called 'kernel' was used to develop the operating system and other functions and could be used on different systems. Also, its source code was open source.

Initially, UNIX was only found in large organizations like government, university, or larger financial corporations with mainframes and minicomputers (PC is a microcomputer).

UNIX Expansion

In eighties, many organizations like IBM, HP and dozen other companies started creating their own UNIX. It results in a mess of UNIX dialects. Then in 1983, Richard Stallman developed GNU project with the goal to make it freely available UNIX like operating system and to be used by everyone. But his project failed in gaining popularity. Many other UNIX like operating system came into existence but none of them was able to gain popularity.

Evolution of Linux

In 1991, Linus Torvalds a student at the University Of Helsinki, Finland, thought to have a freely available academic version of UNIX started writing its own code. Later this project became the Linux kernel. He wrote this program especially for his own PC as he wanted to use UNIX 386 Intel computer but couldn't afford it. He did it on MINIX using GNU C compiler. GNU C compiler is still the main choice to compile Linux code but other compilers are also used like Intel C compiler.

He started it just for fun but ended up with such a large project. Firstly, he wanted to name it as 'Freax' but later it became 'Linux'.

He published the Linux kernel under his own license and was restricted to use as commercially. Linux uses most of its tools from GNU software and are under GNU copyright. In 1992, he released the kernel under GNU General Public License.

Linux Today

Today, supercomputers, smart phones, desktop, web servers, tablet, laptops and home appliances like washing machines, DVD players, routers, modems, cars, refrigerators, etc. use Linux OS.

Linux	UNIX
The Source Code of Linux is freely available to its Users.	The Source Code of UNIX is not available for the general public.
Linux primarily uses Graphical User Interface with an optional Command Line Interface.	UNIX primarily uses Command Line Interface.
Linux OS is portable and can be executed in different Hard Drives.	UNIX is not portable.
Linux is very flexible and can be installed on most of the Home-Based and Servers.	UNIX has a rigid requirement of the Hardware. Hence, cannot be installed on every other machine.
Different Versions of Linux are: Ubuntu, Debian, OpenSuse, RedHat, Solaris, etc.	Different Versions of UNIX are: AIS, HP-UX, BSD, Iris, etc.
Linux Installation is economical and doesn’t require much specific and high-end hardware.	UNIX Installation is comparatively costlier as it requires more specific hardware circuitry.
The File systems supported by Linux are as follows: xfs, ramfs, nfs, vfat, cramfsm ext3, ext4, ext2, ext1, ufs, autofs, devpts, ntfs	The File systems supported by UNIX are as follows: zfs, js, hfx, gps, xfs, gps, xfs, vxfs.
Linux is development by an active Linux Community worldwide.	UNIX is developed by AT&T Developers

Architecture of Linux:

Linux architecture consist of inner most Hardware layer, kernel, shell, and outer application layer, as shown in fig 1.1. Each layer has different functionalities and different uses.

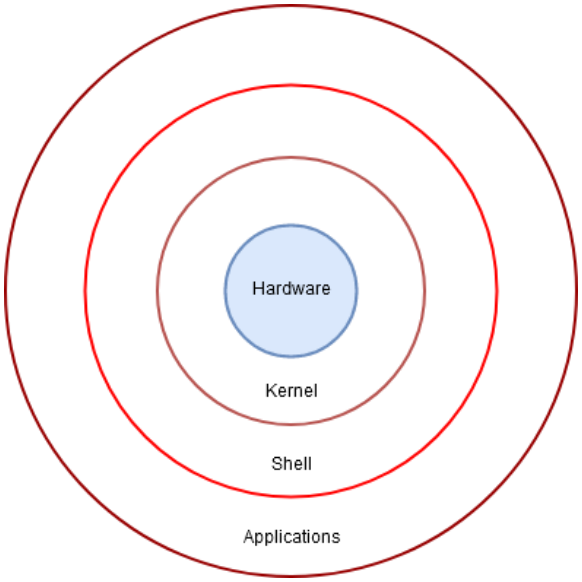


Fig 1.1 Architecture of Linux System

Application Layer: - Users interact with the system through varies applications such as office, games, etc. These applications run in outer layer of architecture.

Shell: - Shell provides environment to run any application. It provides interface to the user to interact with hardware. We can say that it converts higher level language to lower level language.

Kernel: - Kernel is the program which actually communicates with the hardware. Combining shell and kernel forms Operating System.

Hardware: - All the hardware components such as motherboard, CPU, hard disk, etc. are comes under this layer.

Chapter 2: Basic Commands

Commands are nothing but executable programs which perform specific task written in it. These executable programs can be called using their name as per provided syntax.

General Syntax:

 # <commands> -<options> -<argument> <multiple arguments>

Commands- To run the command.

Options - To adjust behavior of the commands.

Arguments - The behavior, file folder name.

BASIC COMMANDS

#tty	This command displays current user terminal number.
#who	Display current logged in all users with their details.
#who am i	Display current user details.
#whoami	Display current user username.
#w	Display all current user details with their time schedule.
#wh <tab> <tab>	To see similar commands or autocomplete cmd.
#clear or **ctrl+l**	It clears screen but not background data.
#cal	It displays current month of calendar.
#cal <year>	It displays specified year calendar.
#cal <month> <year>	It displays specific months calendar in particular year
#cal -3	It displays previous, current and next month's calendar of current year.
#cal -j <year>	It displays Julian days of particular year.
#date	It shows current date and time.
#date -s "DD MM YYYY HH:MM:SS "	It set date and time.

#reboot OR **#init 6**	To restart the system
#power off OR **#init 0**	To power off system.
#shutdown	It shutdown system after 1 min
#shutdown -h 10	It shutdown system after 10 min and broadcast message to all users.
#shutdown -c	It cancels shutdown timer and broadcast message to all users.
#shutdown now OR **shutdown -h**	It immediately shutdown system.
#logout OR **exit** OR **ctrl + d**	Logout from currently logged in user.
#hostname OR **hostname -s**	Shows machine name

COMMAND TO VIEW SYSTEM INFORMATION

#hostnamectl	Show detailed information about system
#uname -a	It displays operating system kernel name.
#uname -r	It display all information regarding system,
#free -h	It display RAM information in human readable form
#lsusb	It lists all available USB devices.
#lspci	It list all PCI devices
#lscpu	It list processor information.
#dmidecode	It display all hardware information.(root)

GETTING HELP FROM COMMANDS

#man <command>	Show manual of mentioned command
#info <command>	Same as that of man command
#whatis <command>	Display one line description of manual page. (#mandb command should use to update database manuals.)
#<command> --help	Show short description of manual page. –help is option thus, some command may not support this option.
#which <command>	To know path of command file.

Chapter 3: Editors in Linux

Editors are used to create new files, and edit or modify the content inside it. Simply editors are used to read and write data in existing or newly created file. Editors can be classified on the basis of interfaces that they use, i.e. Graphical Editors and Command Line Editors.

Graphical editor:

Graphical editor uses graphical user interface. It is easy to use but consumes more memory than command line editors. In Linux, following are some well-known graphical editors,

1. **gedit: -** It is same as that of notepad in windows. You can open gedit graphically in application menu and also using command as **$gedit.**

2. **kedit: -** It is similar to gedit but contain some advance features. Generally, we have to install kedit separately to use it.

3. **Open office: -** Open Office is same as that of MS Office. This Open Office is specially developed for Linux based operating systems.

Command line editor:

Command line editor uses command line interface. These editors are much master and consumes less memory than any graphical editors. Following are some well-known command line editors,

1. **nano: -** nano editor is easy to use since it provides simple features to edit data from files.
   Syntax-
   ```
   # nano <filename>.<extension>
   ```

2. **pico: -** Syntax-
   ```
   # pico <filename>
   ```

3. **vi and vim: -** vi (virtual interface) and vim (virtual interface modified) are most commonly used editors. vi and vim both editors are same where as vim is the advance version of vi editor. Thus, it contains some additional features. These editors works in four different modes,
   - Insert mode
   - Ex-mode
   - Command mode
   - Visual mode

   Syntax-
   ```
   # vim <filename>
   ```

   Cursor Movement,

J	for upward navigation
K	for downward navigation
H	for leftward navigation
L	for rightward navigation

Command Mode, this is default mode. Press esc to exit from any mode and enter into command mode.

dd	Delete current line
<n>dd	Delete n no. of lines from current line
dw	Delete current word
<n>dw	Delete n no. of words from current word
yy	Copy current line
<n>yy	Copy n no. of lines from current line
yw	Copy current word
<n>yw	Copy n no. of words from current word
cc	Cut current line and enter in insert mode
<n>cc	Cut n no. of lines and enter in insert mode
cw	Cut current word and enter in insert mode
<n>cw	Cut n no. of words from current word and enter in insert mode
p	Paste
s	Remove current character and enter in insert mode
S	Remove current line and enter in insert mode
u	Undo
Ctrl+r	Redo
H	Move cursor to the top of screen
M	Move cursor to the middle of screen
L	Move cursor to the bottom of screen
G	Move cursor at the end of file
gg	Move cursor at the beginning
<n>gg	Move cursor at n^{th} line
/<word>	Search particular word/string/character
n	Show next search result
N	Show previous search result

Insert Mode, Following are commands to enter in insert mode

i	insert text at current cursor position
I	insert text at start of the current line
a	insert text just right of the current character
A	insert text at end of the current line
o	insert new line below the current line
O	insert new line above the current line
r	it replaces single character
R	replace multiple characters

Ex-mode, special executing mode,

:q	quit without saving
:q!	quit without saving forcefully
:w	save and stay in file
:wq or **:x**	save and quit
:wq!	save and quit forcefully
:set nu	set line numbers
:<n>	Jump to n^{th} line
:set nonu	remove line numbers
:/<word>	Highlight word/string/character
:nohl	Remove highlight
:%s/<old>/<new>/g	Find and replace old word with new word
:!<command>	Execute any command on terminal without leaving editor

Visual Mode, this mode is used for selection.

v	Select character by character
V	Select line by line
ctrl+v	Select block
y,d,c	For copy, delete, and cut selected area

Chapter 4: Working with Text Files

File Creation:

#touch – touch command is used to create files. Using touch command, multiple files can be created.
Syntax: *#touch <filename>*
Example:
Creating file in current directory

```
[root@server0 /]# touch file1.txt
```

Creating file in desired directory

```
[root@server0 /]# touch /root/file1.txt
```

Creating multiple file at different locations (below example will create two files, one at /root/Desktop and 2nd at /etc directory. You can add more file names along with their path and separate them by space.)

```
[root@server0 /]# touch /root/Desktop/file1.txt /etc/data.mp3
```

Creating multiple files at same location but with different file names (below example will create 3 files. You can add more file names and separate them by coma)

```
[root@server0 /]# touch /root/{data.txt,file.txt,demo.mp3}
```


Creating multiple files having continuous number in their names (below example will create hundred files with name starting from file1 to file100.)

```
[root@server0 /]# touch /root/file{1..100}.txt
```

#mkdir – mkdir command creates directories. Creating multiple directories are also possible using mkdir command
Syntax: *#mkdir <option> <path/directory_name>*
Example,
Creating directory in / directory

```
[root@server0 /]# mkdir /dir1
```

Creating multiple directories

```
[root@server0 /]# mkdir /dir1 /root/Desktop/dir2 /etc/demo
[root@server0 /]# mkdir /root/{demo,data,practice}
[root@server0 /]# mkdir /practical{1..10}
```

Creating parent directory if it is not exist

```
[root@server0 /]# mkdir –p /demo/data/practice
```

File Operations:

READ: - Read operation can be performed to view content of the file. There are five commands which we can use for read operation,

- **cat** – cat command is used to get data of file as output on the terminal. Reading out large file leads to navigate in terminal, which require separate scrolling device (mouse). So, cat command is very useful in reading smaller files with few lines of data in command line.
  Example,

```
[root@server0 /]# cat /root/anaconda-ks.cfg
```

- **more** – more command provides line by line navigation and page by page navigation in downward direction but, upward scrolling not possible.
  Example,

```
[root@server0 /]# more /root/anaconda-ks.cfg
```

- **less** – less command allow navigation keys for scrolling up and down. Thus, it is more useful command than any other four command.
  Example,

```
[root@server0 /]# less /root/anaconda-ks.cfg
```

- **head** – head command show few lines from top of the file. If head command is used without any option, it will show top ten lines by default. *–n* is used to give count of lines to be shown.
  Example,

```
[root@server0 /]# head /root/anaconda-ks.cfg
[root@server0 /]# head –n 5 /root/anaconda-ks.cfg
```

- **tail** – tail command show few lines from bottom of file. If tail command is used without any option, it will show bottom ten lines by default. *–n* is used to give count of lines to be shown.
  Examples,

```
[root@server0 /]# tail /root/anaconda-ks.cfg
[root@server0 /]# tail –n 4 /root/anaconda-ks.cfg
```

SORT: - Sort command will display result in ascending or descending order. Without option, data will be shown in ascending order.

Options,

-r	to show output in reverse order
-k <n>	to show output of arranged by sorting n^{th} column.

Example,

```
[root@server0 /]# sort file1.txt
[root@server0 /]# sort –r file1.txt
```

COPY: - Copy operation use to copy files and directories in Linux, from one location to another. It will copy contents of one file to another. If destination file is not exist in given location then automatically new file will be generated.

Syntax,

#cp <option> <source> <destination>

Options,

-f : forcefully
-v : Verbose/View
-r : recursive (to copy directory)
-a : preserver permissions when copying

Example,

Copy one file content to another file,

```
[root@server0 /]# cp /root/anaconda-ks.cfg ~/Desktop/kickstart.txt
```

Copy file from one location to another,

```
[root@server0 /]# cp /root/anaconda-ks.cfg /mnt/     →copy single file
```

Copy directory and multiple files

```
[root@server0 /]# cp -r /etc /root                   →copy directory etc
[root@server0 /]# cp –r /root/* /media               →copy all files from root
[root@server0 /]# cp –rv /abc.txt /xyz.mp3 /media    →copy multiple files
```

MOVE AND RENAME: - Move and rename both operation can be performed using 'mv' command. It moves files and directories from one location to another. It is possible to move and rename at the same time.

Syntax,

#mv <option> <source> <destination>

Example,

Move files/directories from one location to another

```
[root@server0 /]# mv /root/anaconda-ks.cfg /mnt/          →move single file
[root@server0 /]# mv /media ~/Desktop/                    →move directory
[root@server0 /]# mv /root/* /mnt/                        →move all files
```

Rename file or directory

```
[root@server0 /]# mv fower flower.txt                     → rename file
[root@server0 /]# mv /root/anaconda-ks.cfg /root/kickstart.cfg
[root@server0 /]# mv /opt /demo                           → rename directory
```

Move and rename together,

```
[root@server0 /]# mv /root/anaconda-ks.cfg ~/Desktop/kickstart.txt
```

REMOVE: - 'rmdir' command is use to remove empty directory where as 'rm' command is used to remove files. Directories also can be removed using 'rm' command. (Note: Removing files using rm or rmdir command will delete files permanently and doesn't move in recyclebin or any other place.)

#rmdir – rmdir command only deletes empty directories.
Syntax,

#rmdir <empty_dir_name>

Examples,
Trying to remove non-empty directory using rmdir. It gives error.

```
[root@server0 ~]# rmdir /Demo
rmdir: failed to remove '/Demo': Directory not empty
```

Removing empty directory using rmdir,

```
[root@server0 ~]# rmdir /Demo/data
```

#rm – rm command is use to delete files and directories.
Syntax,

#rm <option> <file_names>

Examples,
Removing files,

```
[root@server0 ~]# rm /Demo/file10.txt
rm: remove regular file '/Demo/file10.txt'? y
```

Remove files without interaction, (**-f** : forcefully, **-v** : verbosely)

```
[root@server0 ~]# rm -vf /Demo/file1.txt
removed '/Demo/file1.txt'
```

Remove all files,

```
[root@server0 ~]# rm –f /Demo/*
```

Remove directories, (**-r** : recursive)

```
[root@server0 ~]# rm –r /Demo
```

Redirectors:

Redirectors are used to write terminal output into file. Output, generated from any command, on terminal can be transferred into existing file. If file does not exist, automatically new file will be created. Following are some redirectors,

Single Redirector (>): Single redirector replace existing data in the file with newly redirected data. It overwrites the contents of existing file.

Double Redirector (>>): Double redirector keeps existing data and newly redirected data will be added at the end of the file. It appends redirected data in existing file.

Syntax,

 # <command_to_generate_output> [> or >>] <new/existing_file>

Example,

```
[root@server0 ~]# echo “Hello World”  >  ~/abc.txt
[root@server0 ~]# cat ~/abc.txt
Hello World
[root@server0 ~]# echo “I am Shubham”  >>  ~/abc.txt
[root@server0 ~]# cat ~/abc.txt
Hello World
I am Shubham
```

Redirectors with cat command:-

Example,

Display content of existing file,

```
[root@server0 ~]# cat /root/anaconda-ks.cfg
```

Create a new file and provide writing into it.

```
[root@server0 ~]# cat > /root/Desktop/flower.txt
Lotus, rose, tulip, lily.                -→ Press [ctrl+d] to save file
[root@server0 ~]# cat /root/Desktop/flower.txt
Lotus, rose, tulip, lily.
```

Overwrite content of existing file.

```
[root@server0 ~]# cat > /root/Desktop/flower.txt
Hello World!
[root@server0 ~]# cat /root/Desktop/flower.txt
Hello World!
```


Add content in existing file, (Append)

```
[root@server0 ~]# cat >> /root/Desktop/flower.txt
I am Shubham
[root@server0 ~]# cat /root/Desktop/flower.txt
Hello World!
I am Shubham
```

View content of files,

```
[root@server0 ~]# cat /etc/redhat-release       → display OS name and Version
[root@server0 ~]# cat /proc/meminfo             → display ram information
[root@server0 ~]# cat /proc/cpuinfo             → display processor information
```

Copy content of one file to another file

```
[root@server0 ~]# cat /etc/redhat-release > /osname.txt
```

Append content of one file to another file,

```
[root@server0 ~]# cat /proc/meminfo >> /osname.txt
```

File content merging,

```
[root@server0 ~]# cat /flower.txt /number.txt /alpha.txt > /newfile.txt
```

Execute Multiple Commands-

Semicolon (;) – Executes two or more commands in same command line argument.
Example,

```
[root@server0 ~]# touch /abc.txt ; mkdir /mydir ; cp –r /mnt /media
```

Pipe (|) – It matches first command output to the second command and execute it.
Example,

```
[root@server0 ~]# dmidecode | less
[root@server0 ~]# ls / | wc
```

Chapter 5: Hierarchy of Linux File Management System

In Linux, files are well managed using file management system. Linux File Management System manage files in hierarchical structure where, "/" (slash) is main directory or root directory (root node in hierarchy). All other directories comes beneath "/" directory.

In RHEL 7.0, there are nineteen default directories created by the system itself. These nineteen directories are present just under "/" directory. Below is the list of all 19 directories, and there uses,

Directory	Description
/root	Home directory of root user. In this directory, root user can store its personal files.
/home	It stores home directories of local users. Home directories are assigned to each user separately and no other user can access home directories of other users (Except root user).
/etc	It stores all configuration files of system and services.
/var	It stores variable data such as mails, logs, messages, etc.
/mnt	Standard directory to mount storage device temporary. In mnt, temporary files available up to 30 days.
/media	All removable devices.
/run	Current Running Devices
/lib	Library files information. This directory is actually a soft link of /usr/lib directory.
/lib64	Same as that of lib directory and stores 64 architecture library file information. It is link of /usr/lib64.
/bin	It stores binary executable files. These binary executable files are nothing but the commands. It is a link of /usr/bin directory. (this directory contains commands that can be used by local users)
/sbin	It stores system binary executable files. It is same as bin directory except that only super user has permission to execute commands from sbin. It is also a link of /usr/sbin directory. (This directory contains commands that only root user can use.)
/usr	User related files. Such as documentary files, manual pages, etc. This directory also contain lib, lib64, bin, and sbin directory whose links are available in main directory i.e. in "/" directory.
/opt	This directory is use for optional add-on service. Sometime path selection for environment variable also done from this directory.
/tmp	It stores temporary files. It stores temporary data for 10 days.
/srv	Service information or data.

/sys	System information.
/proc	Processes information. This directory also stores RAM and CPU related information.
/boot	It stores boot loader program and all other boot related files.
/dev	/dev directory stores device information and their block files.

Chapter 6: User and Group Management

A user is a person who utilizes a computer or network service. Linux is said to be secure because one user cannot access files of other user without its permission. There are three types of user,

1. **Super user:** Super users are those users who has all privileges of Linux system. On all Linux systems, by default there is the user root, also known as the super user. This account is used for managing Linux. Root, for instance, can create other user accounts on the system. For some tasks, root privileges are required. Some examples are installing software, managing users, and creating partitions on disk devices.
2. **System user:** System accounts are used by the services in Linux system. These accounts or users generally created when services are installed in system.
3. **Standard user:** local user accounts or standard user accounts are for the people who need to work on a system and who need limited access to the resources on that system. These user accounts typically have a password that is used for authenticating the user to the system.

Adding New User,

```
[root@ip-172-31-19-5  ~]# useradd shubham
```

Adding new local user means creating user account. User can be added by root user or using root user's privileges. Whenever new user has been added, some files get affected. These files holds user accounts related information. Also whenever new user is created, by default, its home directory and mail account also has been generated. New users are created using some skeleton files located in /etc/skel directory. These files are hidden and copied into home directory of new user.

Files affected by newly added user,

/etc/passwd: It store user profile related information,

```
[root@ip-172-31-19-5 ~]# tail -1 /etc/passwd
shubham:x:1002:1002::/home/shubham/:/bin/bash
```

/etc/shadow: It stores user password policies,

```
[root@ip-172-31-19-5 ~]# tail -1 /etc/shadow
shubham:!!:18206:0:99999:7:::
```

/etc/group: It store group information,

```
[root@ip-172-31-19-5 ~]# tail -1 /etc/group
shubham:x:1002:
```

/etc/gshadow: Stores group password and member's list,

```
[root@ip-172-31-19-5 ~]# tail -1 /etc/gshadow
shubham:!::
```

Home directory and Mail account of new user,

```
[root@ip-172-31-19-5 ~]# ls /home
centos  shubham                                  → Home directories
[root@ip-172-31-19-5 ~]# ls /var/spool/mail/
centos  rpc  shubham                             → Mail Accounts
```

Skeleton files,

.bash_logout: if this file is missing, user will unable to logout from the system.
.bash_profile: If this file is missing, home directory will not be assigned to the new user.
.bashrc: If this file is missing, user will unable to login to the system.

```
[root@ip-172-31-19-5 ~]# ls -a /etc/skel            → skeleton files
.  ..  .bash_logout  .bash_profile  .bashrc
[root@ip-172-31-19-5 ~]# ls -a /home/shubham        → skeleton that copied in
.  ..  .bash_logout  .bash_profile  .bashrc         home directory
```

Switching between users,

su: The 'su' command is use to switch user. It allows user to open a sub shell with different logged in user. Switching user from root to other local user will not require any password. But, switching from local user to any other user will require password of that user for authentication and login.

Syntax, *# su - <user_name>*
Example,
Switching from root user to local user (shubham)

```
[root@ip-172-31-19-5 centos]# su - shubham
Last login: Wed Nov  6 11:53:05 UTC 2019 on pts/0
[shubham@ip-172-31-19-5 ~]$
```

Switching from local user (shubham) to any other user (centos)

```
[shubham@ip-172-31-19-5 ~]$ su - centos
Password:
Last login: Wed Nov  6 11:52:19 UTC 2019 from 49.36.29.134 on pts/0
[centos@ip-172-31-19-5 ~]$
```

(Note: Switching user using 'su' command will open new sub shell with different user login. But, previous user remains logged in. You have to logout user manually.**)**

Password Management,

passwd: Password is the secret phrase that can be used to login to the system. 'passwd' command will be used to assign or change password of any user by root user. Whenever, password assigned to the user, it will stored in /etc/shadow file in encrypted format. Only root user can change password of any user, but local users can change their own password. Password should follow some rules such as,

- Password must be 8 character long
- It should not contain user name
- It cannot accept old password
- Any dictionary name is not allowed
- Password should not be too simplistic

Syntax, *#passwd* → change current user's password
#passwd <user_name> → assign or change other user's password by root user

Example,

Changing root user's password,

```
[root@ip-172-31-19-5 ~]# passwd
Changing password for user root.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
```

Changing local user's password by root user's account,

```
[root@ip-172-31-19-5 centos]# passwd shubham
Changing password for user shubham.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
```

Changing current user's password (local user changing its own password)

```
[shubham@ip-172-31-19-5 ~]$ passwd
Changing password for user shubham.
Changing password for shubham.
(current) UNIX password:
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

Changing other user's password by local user's account, (it generates error because only root user has privilege to change other user's password)

```
[shubham@ip-172-31-19-5 ~]$ passwd centos
passwd: Only root can specify a user name.
```

Password are stored in /etc/shadow file in encrypted format,

```
[root@ip-172-31-19-5 ~]# tail -1 /etc/shadow
shubham:$6$S59rUkc4$iIusUTs6TPb2ueLMty3/2kvShejrTVctesfLYyUwTa78kDQQ/O/f954Euy
omO6nBwwiPyqPt4hAij5OxiQIQ5.:18206:0:99999:7:::
```

/etc/shadow file: /etc/shadow file stores password and password policies of all users. It contains nine fields and each field is separated by colon. Below is the summary of these fields,


```
[root@ip-172-31-19-5 ~]# tail -1 /etc/shadow
shubham:$6$S59rUkc4$iIusUTs6TPb2ueLMty3/2kvShejrTVctesfLYyUwTa78kDQQ/O/f954Euy
omO6nBwwiPyqPt4hAij5OxiQIQ5.:18206:0:99999:7:::
```

①:②:③:④:⑤:⑥:⑦:⑧:⑨

1. **Username:** This is a unique name for the user. User names are important to match a user to his password. On Linux, there can be no spaces in the user name.
2. **Encrypted password:** This field contains all that is needed to store the password in a secure way.
3. **Days since Jan 1, 1970, that the password was last changed:** Many things on Linux refer to this date, which on Linux is considered the beginning of days.
4. **Days before password may be changed**: This is minimum age of password. That means user cannot change password, before the mentioned days, after immediately changing the password. Typically this field is set to the value 0.
5. **Days after which password must be changed:** This field contains the maximal validity period of passwords or maximum age of password. User must have to change their password after mentioned days. By default it is set to 99999 days.
6. **Days before password is to expire that user is warned:** This field is used to warn a user when a forced password change is upcoming. By default it is set to 7 days.
7. **Days after password expires that account is disabled:** Use this field to enforce a password change. After password expiry, users can no longer log in.
8. **Days since Jan 1, 1970, that account is disabled:** An administrator can set this field to disable an account. This is typically a better approach than removing an account, as all associated properties and files of the account will be kept, but it can be used no longer to authenticate on your server.
9. **For future use:** This is reserved field for future use.

View and change password policy,

#**chage –** 'chage' (change age) command use to view or modify password policy of user.

Syntax,

 # chage <option> <parameter> <username>

Options,

-l	**:-**	list / view password policy.
-m	**:-**	min. days between password change.
-M	**:-**	maximum days between password change.
-W	**:-**	number of days of warning.
-I	**:-**	number of inactivation days.
-E	**:-**	Expiry date of user account.
-d	**:-**	force to change password.

Example,

View password policies,

```
[root@ip-172-31-37-64 ]#  chage -l amit
Last password change                                    : May 25, 2019
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
```

Change minimum age,

```
[root@ip-172-31-37-64 ]# chage -m 30 amit
[amit@ip-172-31-37-64 ~]$ chage -l amit
Last password change                                    : May 25, 2019
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 30
Maximum number of days between password change          : 99999
Number of days of warning before password expires       : 7
```

Change maximum age,

```
[root@ip-172-31-37-64 ]# chage -M 45 amit
[amit@ip-172-31-37-64 ~]$ chage -l amit
Last password change                                    : May 25, 2019
Password expires                                        : Jul 09, 2019
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 30
Maximum number of days between password change          : 45
Number of days of warning before password expires       : 7
```

Change warning days,

```
[root@ip-172-31-37-64 ]# chage -W 0 amit
[amit@ip-172-31-37-64 ~]$ chage -l amit
Last password change                                    : May 25, 2019
Password expires                                        : Jul 09, 2019
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 30
Maximum number of days between password change          : 45
Number of days of warning before password expires       : 0
```

Change expiry date of user account

```
[root@ip-172-31-37-64 ]# chage -E "20 OCT 2018" amit
[root@ip-172-31-37-64 ]# su - amit
[amit@ip-172-31-37-64 ~]$ chage -l  amit
Last password change                                    : May 25, 2019
Password expires                                        : Jul 09, 2019
Password inactive                                       : never
Account expires                                         : Oct 20, 2018
Minimum number of days between password change          : 30
Maximum number of days between password change          : 45
Number of days of warning before password expires       : 0
```

Change password immediately,

```
[root@ip-172-31-37-64 amit]# chage -d 0 amit
[amit@ip-172-31-37-64 ~]$ chage -l amit
Last password change                                    : password must be
changed
Password expires                                        : password must be
changed
Password inactive                                       : password must be
changed
Account expires                                         : Oct 20, 2018
Minimum number of days between password change          : 30
Maximum number of days between password change          : 45
Number of days of warning before password expires       : 0
```

Group Administration/Management

Linux users can be a member of two different kinds of groups. First, there is the primary group. Every user must be a member of a primary group and there is only one primary group. When creating files, the primary group becomes group owner of these files.

Users can also access all files their primary group has access to. The user's primary group membership is defined in /etc/passwd; the group itself is stored in the /etc/group configuration file. Besides the mandatory primary group, users can be a member of one or more secondary groups as well. Secondary groups are important to get access to files. If the group a user is a member of has access to specific files, the user will get access to these files also.

#groupadd – 'groupadd' command is use to add secondary or supplementary group in system. Group information are stored in /etc/group file.

Syntax,

 # groupadd <groupname>

Example,

```
[root@ip-172-31-37-64 ~]# groupadd   IBM
[root@ip-172-31-37-64 ~]# tail -1 /etc/group
IBM:x:1005:
```

/etc/group file: This file contain all group's information. The file has four fields and each field is separated by colon (:). Following are fields of shadow file,

①:②:③:④

1. **Group name:** As is suggested by the name of the field, this contains the name of the group.
2. **Redirected group password:** A feature that is hardly used anymore. A group password can be used by users that want to join the group on a temporary basis, so that access to files the group has access to is allowed.
3. **Group id (GID):** A unique numeric group identification number.
4. **List of members:** Here you find the names of users that are a member of this group as a secondary group. Note that it does not show users that are a member of this group as their primary group.

Adding group with customize setting,

Syntax, *# groupadd <option> <parameter> <groupname>*

Options, **-g** :- Group id

-o :- Non unique

-f :- Forcefully

Example,

```
[root@ip-172-31-37-64 ~]# groupadd -g 2005 TCS
[root@ip-172-31-37-64 ~]# tail -1 /etc/group
TCS:x:2005:
```

Modify existing group with customize setting,

Syntax, *# groupmod <option> <parameter> <groupname>*

Options, **-g** **:-** Group id

-n **:-** Group Name

-o **:-** Non unique

Example,

Change group id of existing group,

```
[root@ip-172-31-37-64 ~]#groupmod -g  5903  IBM
[root@ip-172-31-37-64 ~]# tail -1 /etc/group
IBM:x:5903:
```

Changing group name of existing group,

```
[root@ip-172-31-37-64 ~]# groupmod -n TATA TCS
[root@ip-172-31-37-64 ~]# tail -2 /etc/group
TATA:x:2005:
```


User Administration/Management

#useradd – 'useradd' command use to create new user account.

Syntax, *# useradd <username>*

Example,

```
[root@ip-172-31-19-5  ~]# useradd shubham
```

/etc/passwd file: This file stores user profile information. It contain 7 fields as follows,

①:②:③:④:⑤:⑥:⑦

1. **User login name:** This is a unique name for the user. User names are important to match a user to his password, which is stored separately in /etc/shadow. On Linux, there can be no spaces in the user name.
2. **Password link from shadow file:** since /etc/passdw file is readable by all user, for security purpose, the password is stored in /etc/shadow file.
3. **User id (UID):** Each user has a unique user ID (UID). This is a numeric ID. It is the UID that really determines what a user can do. When permissions are set for a user, the UID is stored in the file metadata (and not the user name). UID 0 is reserved for root. The lower UIDs (typically up to 999) are used for system accounts, and the higher UIDs (from 1000 on by default), are reserved for people that need to connect directory to the server. [Note: /etc/login.defs contain default setting for user creation.]

4. **Primary group's id (GID):** On Linux, each user is a member of at least one group. This group is referred to as the *primary*
5. **Comment field:** The Comment field, as you can guess, is used to add comments for user accounts. This field is optional, but it can be used to describe what a user account is created for.
6. **Home Directory Home Directory:** This is the initial directory where the user is placed after logging in, also referred to as the *home directory.* If the user account is used by a person, this is where the person would store his personal files and programs.
7. **Login shell:** This is the program that is started after the user has successfully connected to a server. For most users this will be **/** bin/bash, the default Linux shell. For system user accounts, it will typically be a shell like /sbin/nologin. The **/sbin/nologin** command is a specific command that silently denies access to users.

Adding user with customized setting,

Syntax, *# useradd <options> <parameters> <username>*

Options,

-u	**:-**	User id
-g	**:-**	Primary group
-c	**:-**	Comment
-d	**:-**	Home directory
-s	**:-**	Login shell
-G	**:-**	Secondary group
-r	**:-**	System user
-e	**:-**	Account expiry date
-o	**:-**	Non unique

Example,

Create user with customized user id,

```
[root@ip-172-31-19-5 ~]# useradd -u 2211 amit
[root@ip-172-31-19-5 ~]# tail -1 /etc/passwd
amit:x:2211:2211::/home/amit:/bin/bash
```

Create user with comment,

```
[root@ip-172-31-19-5 ~]# useradd -c "linux user" amit
[root@ip-172-31-19-5 ~]# tail -1 /etc/passwd
amit:x:2212:2212:linux user:/home/amit:/bin/bash
```

Create user with customized home directory,

```
[root@ip-172-31-19-5 ~]# useradd -d /project suresh
[root@ip-172-31-19-5 ~]# tail -1 /etc/passwd
suresh:x:2213:2213::/project:/bin/bash
```

Create user with nologin shell,

```
[root@ip-172-31-19-5 ~]# useradd -s /sbin/nologin mahesh
[root@ip-172-31-19-5 ~]# tail -1 /etc/passwd
mahesh:x:2214:2214::/home/mahesh:/sbin/nologin
```

Create user with customized primary group,

```
[root@ip-172-31-19-5 ~]# useradd -g IBM amit
[root@ip-172-31-19-5 ~]# tail -1 /etc/passwd
amit:x:2216:1003::/home/amit:/bin/bash
```

Create user and assign secondary group,

```
[root@ip-172-31-19-5 ~]# useradd -G IBM shubham
[root@ip-172-31-19-5 ~]# tail -2 /etc/group
IBM:x:1003:shubham
shubham:x:2216:
```

Add user along with its expiry date,

```
[root@ip-172-31-19-5 ~]# useradd -e "10 dec 2019" suresh
[root@ip-172-31-19-5 ~]# chage -l suresh
Last password change                                    : Nov 08, 2019
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : Dec 10, 2019
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
```

Create user with multiple customize options,

```
[root@ip-172-31-19-5 ~]# useradd -u 6001 -g TCS -G IBM -c "admin account" -d
/admin -s /bin/bash -e "31 dec 2030" shubham
[root@ip-172-31-19-5 ~]# tail -1 /etc/passwd
shubham:x:6001:1006:admin account:/admin:/bin/bash
[root@ip-172-31-19-5 ~]# tail -2 /etc/group
IBM:x:1003:shubham
TCS:x:1006:
[root@ip-172-31-19-5 ~]# chage -l shubham
Last password change                                    : Nov 08, 2019
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : Dec 31, 2030
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
```

Modify user with customized setting,

#usermod – 'usermod' command is use to modify existing user's setting.

Syntax,

 # usermod <option> <parameters> <username>

Options,

-u	**:-**	user id
-g	**:-**	primary group
-c	**:-**	Comment
-d	**:-**	home directory
-s	**:-**	login shell
-G	**:-**	secondary group
-l	**:-**	login name
-L	**:-**	lock user password
-U	**:-**	unlock user password
-m	**:-**	modify directories

Example,

Change user id,

```
[root@ip-172-31-37-64 ~]# #usermod    -u  7766    sumit
[root@ip-172-31-37-64 ~]# tail -1 /etc/passwd
sumit:x:6021:6021::/home/sumit:/bin/bash
```

Change comment,

```
[root@ip-172-31-37-64 ~]# usermod  -c  “windows”   sumit
[root@ip-172-31-37-64 ~]# tail -1 /etc/passwd
sumit:x:6021:6021:windows:/home/sumit:/bin/bash
```

Change home directory,

```
[root@ip-172-31-37-64 ~]# usermod -m -d /home/namit  sumit
[root@ip-172-31-37-64 ~]# tail -1 /etc/passwd
sumit:x:6021:6021:windows:/home/namit:/bin/bash
```

Change login shell,

```
[root@ip-172-31-37-64 ~]# usermod -s /bin/bash sumit
[root@ip-172-31-37-64 ~]#  tail -1 /etc/passwd
sumit:x:6021:6021:windows:/home/namit:/bin/bash
```

Assign secondary or supplementary group,

```
[root@ip-172-31-37-64 ~]#usermod -G TCS  suresh
[root@ip-172-31-37-64 ~]# tail -1 /etc/group
TCS:x:6023:suresh
```


Change user login name,

```
[root@ip-172-31-37-64 ~]# usermod -l  sammuel  ritesh
[root@ip-172-31-37-64 ~]# tail -1 /etc/passwd
sammuel:x:2214:2214::/home/ritesh:/bin/bash
```

Change expiry date of account,

```
[root@ip-172-31-37-64 ~]# usermod  -e   '14 dec 2018'   ritesh
[root@ip-172-31-37-64 ~]# chage -l ritesh
Last password change                                    : May 25, 2019
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : Dec 14, 2018
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
```

Lock and Unlock User Password,

```
[root@ip-172-31-19-5 ~]# tail -1 /etc/shadow
amit:$6$lblryj7X$jDt1EWzSANqKVrW/KS4VSEBU/GTFFgUmfCNZJxBjV4kTUJWgirNaNKsHIg79Y
gUIPXO.VT50vQuNnu4ik64ZM/:18209:0:99999:7:::
[root@ip-172-31-19-5 ~]#
[root@ip-172-31-19-5 ~]# usermod -L amit            → Lock Password
[root@ip-172-31-19-5 ~]# tail -1 /etc/shadow
amit:!$6$lblryj7X$jDt1EWzSANqKVrW/KS4VSEBU/GTFFgUmfCNZJxBjV4kTUJWgirNaNKsHIg79
YgUIPXO.VT50vQuNnu4ik64ZM/:18209:0:99999:7:::
[root@ip-172-31-19-5 ~]#
[root@ip-172-31-19-5 ~]# usermod -U amit            → Unlock Password
[root@ip-172-31-19-5 ~]# tail -1 /etc/shadow
amit:$6$lblryj7X$jDt1EWzSANqKVrW/KS4VSEBU/GTFFgUmfCNZJxBjV4kTUJWgirNaNKsHIg79Y
gUIPXO.VT50vQuNnu4ik64ZM/:18209:0:99999:7:::
```

Remove user from system,

#userdel – 'userdel' command is used to remove or delete user account from system.

Syntax,

 # userdel <username>

Example,

Delete user account,

```
[root@ip-172-31-19-5 ~]# userdel amit
```

Delete user account along with its home directory and mail account,

```
[root@ip-172-31-19-5 ~]# userdel -r shubham
```


/etc/gshadow file: This file use to store password of group. It also stores admin and member list of group. It contain four fields,

①:②:③:④

1. Group name
2. Encrypted Password
3. Admin of group
4. Member list

#gpasswd – 'gpasswd' command is use to give password to group. It also can be used to add members and assign admin to the group.

Syntax,

 # gpasswd <option> <parameter> <groupname>

Options,

-a	**:-**	Add members in group
-M	**:-**	Set list of members in group
-A	**:-**	Assign user as group admin

Example,

Assign or change group password,

```
[root@ip-172-31-19-5 ~]# gpasswd TCS
Changing the password for group TCS
New Password:
Re-enter new password:
[root@ip-172-31-19-5 ~]# tail -1 /etc/gshadow
TCS:$6$b4a1sbtiZDy$arjapZjNWW2u.EE2D49ZI2k8VtT7WNZ3zRNkmg0ByFIrJrXbjMZe8fQ0U0R
QfTG/RrXOKAukFC5ganx0k00MO1::
```

Add existing user in group using *gpasswd* command,

```
[root@ip-172-31-19-5 ~]# gpasswd -a mahesh TCS
Adding user mahesh to group TCS
[root@ip-172-31-19-5 ~]# tail -1 /etc/group
TCS:x:2218:mahesh
```

Add existing user in group using *usermod* command,

```
[root@ip-172-31-19-5 ~]# usermod -G TCS suresh
[root@ip-172-31-19-5 ~]# tail -1 /etc/group
TCS:x:2218:mahesh,suresh
```

Add new user in group during creating its account,

```
[root@ip-172-31-19-5 ~]# useradd -G TCS amit
[root@ip-172-31-19-5 ~]# tail -2 /etc/group
TCS:x:2218:mahesh,suresh,amit
amit:x:2219:
```


Set list of members in group, (old list of members will replace with new list.)

```
[root@ip-172-31-19-5 ~]# gpasswd -M atul,shubham TCS
[root@ip-172-31-19-5 ~]# tail -2 /etc/group
TCS:x:2218:atul,shubham
amit:x:2219:
```

Assign user as admin of group

```
[root@ip-172-31-19-5 ~]# gpasswd -A amit TCS
[root@ip-172-31-19-5 ~]# tail -2 /etc/gshadow
TCS:$6$b4a1sbtiZDy$arjapZjNWW2u.EE2D49ZI2k8VtT7WNZ3zRNkmg0ByFIrJrXbjMZe8fQ0U0R
QfTG/RrXOKAukFC5ganx0k00MO1:amit:atul,shubham
amit:!::
```

Chapter 7: Configuring Permissions

Linux File System Security:

Linux File System Security restricts user to access the files and directories. User require permissions to access files or directories. "*ls -l*" or "*ll*" command can be used to check security of any file or directory. Above command will display contents from directory along with its security details. Some of security attributes are shown below,

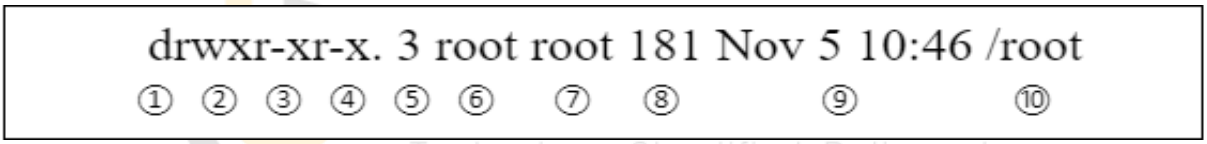
Example,

```
[root@ip-172-31-19-5 ~]# ll /root
total 8
-rw-------. 1 root root 6577 Jan 28  2019 original-ks.cfg
```

To check security of any particular directory,

```
[root@ip-172-31-19-5 ~]# ll -d /root
dr-xr-x---. 3 root root 181 Nov  5 10:46 /root
```

Above example display contents from directory along with its security details. It contains ten fields as mentioned below,



1. File type.
2. Owner Permissions.
3. Group Permissions.
4. Other User Permission.
5. Link Count.
6. Owner of file/directory.
7. Group Owner of File or Directory.
8. File Size.
9. Creation Date and Time.
10. File/Directory Name.

- **File Types in Linux:** There are seven types of file available mentioned as below,

File Type	Symbol	Example
Normal file/Regular File	-	/etc/passwd
Directory	d	/home
Link file	l	/etc/grub.conf
Block Device File	b	/dev/vdb
Character Device File	c	/dev/pts/0
Socket File	s	/dev/log
Normal Pipe File	p	/dev/initctl

Regular File: The regular file is a most common file type found on the Linux system. It governs all different files such us text files, images, binary files, shared libraries, etc.
Directory: Directory is second most common file type found in Linux. Directory can be created with the **mkdir** command
Character Device File: Character and block device files allow users and programs to communicate with hardware peripheral devices.
Block Device File: Block devices are similar to character devices. They mostly govern hardware as hard drives, memory, etc.
Local Domain Socket: Local domain sockets are used for communication between processes. Generally, they are used by services such as X windows, syslog and etc.
Named piped: Similarly, as Local sockets, named pipes allow communication between two local processes. They can be created by the **mknod** command and removed with the **rm** command.
Symbolic Link: With symbolic links an administrator can assign a file or directory multiple identities. Symbolic link can be thought of as a pointer to an original file.

- **Link Count:** Also called as reference count. It shows count of links of file/directory that has been created.

  Default link count of directory is 2 whereas default link count of file is 1. Whenever new directory is created, link count of its just parent directory will increase by 1.

  There are two types of symbolic link,

  a. Hard Link
  b. Soft Link

  Below is the difference between hard and soft links,

Hard Link	**Soft Link**
Inode number of hard link is same as that of inode number of original file.	Inode number of softlink is different than inode number of original file.
Hardlink contain actual data from the file.	Softlink contain path of original file and not the actual data.
Thus, size of hardlink and original file will be same.	Thus, size of softlink depends on path length.
Creating or removing hardlink will increase or decrease link count by 1.	Creating or removing softlink will not affect link count.
Removing original file will only reduce the link count and it will not affect any of its hardlink.	Removing original file will disable soflink, since softlink points to the non-existing file.
Creating hardlink of directory is not possible.	Creating softlink of directory is possible.
Syntax to create hardlink, *# ln <original_file> <hardlink_name>*	Synatx to create softlink, *# ln –s <original_file> <softlink_name>*

Example,
Create Hardlink and softlink

```
[root@ip-172-31-19-5 ~]# ln /root/anaconda-ks.cfg /root/Documents/hardlink
[root@ip-172-31-19-5 ~]# ln –s /root/anaconda-ks.cfg /root/Documents/softlink
[root@ip-172-31-19-5 ~]# ll /root/Documents
Total 4
-rw-------. 2 root root 817 Nov 18 23:06 hardlink
lrwxrwxrwx. 1 root root  21 Dec  4 16:57 softlink -> /root/anaconda-ks.cfg
```

- **User and group ownership:** Basically, by default, user who creates the file/directory is the owner and his primary group acquires group ownerships of that file/directory. User owner and group owner of file will be shown in 6th and 7th field as shown in the figure. "chown" command is use to change owner and group where as using "chgrp" we can change group ownership.

Syntax,

```
# chown <user_name>:<group_name> <file/dir_name>
# chgrp <group_name> <file/dir_name>
```

Example,

Change User Ownership

```
[root@cloudblitz ~]# touch samplefile.txt
[root@cloudblitz ~]# ll
-rw-r--r--. 1 root root 0 May 29 08:56 samplefile.txt
[root@cloudblitz ~]# chown chetan samplefile.txt
[root@cloudblitz ~]# ll
-rw-r--r--. 1 chetan root 0 May 29 08:57 samplefile.txt
```

Change Group Ownership

```
[root@cloudblitz ~]# chgrp nagpur samplefile.txt
[root@cloudblitz ~]# ll
-rw-r--r--. 1 chetan nagpur    0 May 29 08:57 samplefile.txt
```

Change user and group ownership using chown,

```
[root@cloudblitz ~]# chown atul:TCS samplefile.txt
[root@cloudblitz ~]# ll
-rw-r--r--. 1 atul TCS    0 May 29 08:57 samplefile.txt
```

- **Managing Permissions:** Field 2nd, 3rd and 4th represents permissions for owner, group owner and other users. Each of that field contain three basic permissions which allow user to read, write, and executes files. The effect of these permissions differ when applied to file or directory. If applied to a file, the read permission gives user the right to open file for reading. Therefore, user can read it's contain. "chmod" command is use to change these basic permissions.

Permission	Applied to Files	Applied to Directories	Permission (Letter)	Permission (Number)
Read	Open a file	List contents of directory	r	**4**
Write	Change contents of a file	Create and delete files and modify Permissions on files.	w	**2**
Execute	Run a program file	Change to the directory	x	**1**

Change permission using letters -
Syntax,
 # chmod <u,g,o><+,-,=><r,w,x> <file_name>
(Symbols used in syntax are explained in below table)

Symbol	Description	Symbol	Description	Symbol	Description
u	Owner permission	+	add permission	r	Read permission
g	Group permission	-	remove permission	w	Write permission
o	Other user permission	=	assign permission	x	Execute permission

Example,
Give write permission to group,

```
[root@ip-172-31-41-212 ~]# mkdir /nagpur
[root@ip-172-31-41-212 ~]# chmod g+w /nagpur/
[root@ip-172-31-41-212 ~]# ls -ld /nagpur/
drwxrwxr-x. 2 root root 6 May 25 06:14 /nagpur/
```

Remove read and write permission for other users,

```
[root@ip-172-31-41-212 ~]# chmod o-rx /nagpur/
[root@ip-172-31-41-212 ~]# ls -ld /nagpur
drwxrwx---. 2 root root 6 May 25 06:14 /nagpur
```

Assign read only permission for group and other users,

```
[root@ip-172-31-41-212 ~]# chmod go=r /nagpur/
[root@ip-172-31-41-212 ~]# ls -ld /nagpur
drwxr--r--. 2 root root 6 May 25 06:14 /nagpur
```


Give execute permission to owner of file,

```
[root@cloudblitz ~]# touch samplefile3.txt
 [root@cloudblitz ~]# chmod u+rwx samplefile3.txt
[root@cloudblitz ~]# ll
-rwxr--r--. 1 root root 0 May 29 09:29 samplefile3.txt
```

Remove read permission for group and assign read and write permission to other users,

```
[root@ip-172-31-41-212 ~]# chmod g-r,o=rw samplefile3.txt
[root@ip-172-31-41-212 ~]# ls –l samplefile3.txt
-rwx---rw-. 1 root root 0 May 29 09:29 samplefile3.txt
```

Change permission using octal numbers -
Syntax,
 # chmod <permission_in_numbers> <file_name>

OCTAL NO	BINARY	PERMISSION
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

Example,
Changing permission using octal number,

```
[root@ip-172-31-41-212 ~]# mkdir /abhi
[root@ip-172-31-41-212 ~]# chmod 421 /abhi/
[root@ip-172-31-41-212 ~]# ls -ld /abhi/
dr---w---x. 2 root root 6 May 25 07:10 /abhi/
```

```
[root@ip-172-31-41-212 ~]# chmod 732 /abhi/
[root@ip-172-31-41-212 ~]# ls -ld /abhi/
drwx-wx-w-. 2 root root 6 May 25 07:10 /abhi/
```

```
[root@ip-172-31-41-212 ~]# chmod 644 /abhi/
[root@ip-172-31-41-212 ~]# ls -ld /abhi/
drw-r--r--. 2 root root 6 May 25 07:10 /abhi/
```

```
[root@ip-172-31-41-212 ~]# chmod 755 /abhi/
[root@ip-172-31-41-212 ~]# ls -ld /abhi/
drwxr-xr-x. 2 root root 6 May 25 07:10 /abhi/
```

- **Default Permission:** When a user creates a file as a regular user, it's given permission rw-rw-r-- (664) by default. A directory is given the permission rwxrwxr-x (775). For the root user, file and directory permission are rw-r--r-- (644) and rwxr-xr-x (755), respectively. These default values are determined by the value of umask. Type umask to see what your umask value is.

  If you ignore the leading zero for the moment, the umask value masks what is considered to be fully opened permissions for a file 666 or a directory 777. The umask value of 002 results in permission for a directory of 775 (rwxrwxr-x). That same umask results in a file permission of 644 (rw-rw-r--).

Default Permission for root and standard user

User	Directory	File
Root	755	644
Standard User	775	664

Calculating Permission with Umask

USER		Calculation Umask	Umask Permission
Root	Dir	777-022	755
	File	666-022	644
Standard	Dir	777-002	775
	File	666-002	664

Umask for root user and standard user

```
[root@cloudblitz ~]# umask                    → default umask for root user
0022
[student@cloudblitz ~]$ umask                 → umask for standard user
0002
```

Changing umask value temporary

```
[root@cloudblitz ~]# umask 000
[root@cloudblitz ~]# mkdir demo
[root@cloudblitz ~]# ll
drwxrwxrwx. 2 root root 6 May 29 12:34 demo
```

Changing umask value permanent

```
[root@cloudblitz ~]# vim /etc/profile
 59 if [ $UID -gt 199 ] && [ "`/usr/bin/id -gn`" = "`/usr/bin/id -un`" ]; then
 60     umask 002        #change value for standard user line 60
 61 else
 62     umask 000        #change the value for root line 62
 63 fi
:wq

[root@cloudblitz ~]# source /etc/profile
[root@cloudblitz ~]# umask
0000
```

- **Special Permission:** We have three types of special permission, i.e. suid, sgid, and stickey bit.

Name	Numeric Value	Relative Value	On Files	Symbol
SUID	4	u+s	User executes file with permissions of file owner.	No meaning.
SGID	2	g+s	User executes file with permissions of group owner.	Files created in directory get the same group owner.
Sticky Bit	1	o+t	No meaning.	Prevents users from deleting files from other users.

SUID (set user identity): SUID (**S**et owner **U**ser **ID** up on execution) is a special type of file permissions given to a file. Normally in Linux/UNIX when a program runs, it inherits access permissions from the logged in user. SUID is defined as giving temporary permissions to a user to run a program/file with the permissions of the file owner rather that the user who runs it**.** In simple words users will get file owners permissions as well as owner UID and GID when executing a file/program/command.

Syntax,
 # chmod u+s <file_name>
Example,
(As we seen in basic commands, dmidecode command is use to get hardware information of the system. But dmidecode command is owned by root user and present in /sbin directory. That means, no local user can access it. Following example illustrates this scenario, and then set suid permission on command. After applying suid permission, any user can execute this command.)

```
[shubham@ip-172-31-19-21 ~]$ dmidecode
# dmidecode 3.0
/sys/firmware/dmi/tables/smbios_entry_point: Permission denied
Scanning /dev/mem for entry point.
/dev/mem: Permission denied
[shubham@ip-172-31-19-21 ~]$ logout
[root@ip-172-31-19-21 ~]# which dmidecode
/sbin/dmidecode
[root@ip-172-31-19-21 ~]# ll /sbin/dmidecode
-rwxr-xr-x 1 root root 110608 Jul 31  2018 /sbin/dmidecode
[root@ip-172-31-19-21 ~]# chmod u+s /sbin/dmidecode
[root@ip-172-31-19-21 ~]# ll /sbin/dmidecode
-rwsr-xr-x 1 root root 110608 Jul 31  2018 /sbin/dmidecode
[root@ip-172-31-19-21 ~]# su - shubham
Last login: Thu Dec 12 05:38:58 UTC 2019 on pts/0
[shubham@ip-172-31-19-21 ~]$ dmidecode
# dmidecode 3.0
Getting SMBIOS data from sysfs.
SMBIOS 2.7 present.
11 structures occupying 359 bytes.
Table at 0x000EB01F.

Handle 0x0000, DMI type 0, 24 bytes
BIOS Information
        Vendor: Xen
...
```


SGID (set group identity): SGID (Set Group ID up on execution) bit is set on directory if we give SGID permission to particular directory and if file created in that directory by root user or local user that files will get directory group ownership automatically. In other words, when we applied sgid bit to particular directory, it inherit group permission to all files and directories that will create in sgid applied directory.

Syntax,

 # chmod g+s <file_name>

Example,

```
[root@ip-172-31-19-21 ~]# mkdir /demo
[root@ip-172-31-19-21 ~]# chgrp TCS /demo
[root@ip-172-31-19-21 ~]# ll -d /demo
drwxr-xr-x 2 root TCS 6 Dec 12 05:55 /demo
[root@ip-172-31-19-21 ~]# touch /demo/file
[root@ip-172-31-19-21 ~]# ll /demo/file
-rw-r--r-- 1 root root 0 Dec 12 05:56 /demo/file
[root@ip-172-31-19-21 ~]# chmod g+s /demo
[root@ip-172-31-19-21 ~]# ll -d /demo
drwxr-sr-x 2 root shubham 31 Dec 12 05:57 /demo
[root@ip-172-31-19-21 ~]# touch /demo/file1
[root@ip-172-31-19-21 ~]# ll /demo
total 0
-rw-r--r-- 1 root root    0 Dec 12 05:56 file
-rw-r--r-- 1 root TCS     0 Dec 12 05:57 file1
```

Stickey Bit: Sticky Bit is mainly used on folders in order to avoid deletion of a folder and its content by other users though they having write permissions on the folder contents. If Sticky bit is enabled on a folder, the folder contents are deleted by only owner who created them and the root user. No one else can delete other user's data in this folder (Where sticky bit is set). This is a security measure to avoid deletion of critical folders and their content (sub-folders and files), though other users have full permissions.

Syntax,

 # chmod o+t <file_name>

Example,

```
[root@ip-172-31-19-21 ~]# mkdir /demo
[root@ip-172-31-19-21 ~]# chmod 777 /demo
[root@ip-172-31-19-21 ~]# chmod o+t /demo
[root@ip-172-31-19-21 ~]# su - shubham
[shubham@ip-172-31-19-21 ~]$ touch /demo/file.txt
[shubham@ip-172-31-19-21 ~]$ logout
[root@ip-172-31-19-21 ~]# su - chetan
[chetan@ip-172-31-19-21 ~]$ rm –f /demo/file.txt
rm: cannot remove '/demo/file.txt': Operation not permitted
```

- **ACL (access control list):** ACL use to set permission over file and directory to specific user or specific group. We can assign multiple user with different permission on same file or directory. Access control list (ACL) provides an additional, more flexible permission mechanism for file systems. It is designed to assist with UNIX file permissions. ACL allows you to give permissions for any user or group to any disc resource.

  Think of a scenario in which a particular user is not a member of group created by you but still you want to give some read or write access, how can you do it without making user a member of group, here comes in picture Access Control Lists, ACL helps us to do this trick. Basically, ACLs are used to make a flexible permission mechanism in Linux. From Linux man pages, ACLs are used to define more fine-grained discretionary access rights for files and directories.

  setfacl and *getfacl* are used for setting up ACL and showing ACL respectively.

  Syntax, (to apply ACL)

  # setfacl -m u:<user_name>:<permissions> <file_name> → for user

  # setfacl –m g:<grp_name>:<permissions> <file_name> → for group

  Syntax, (to check ACL)

  # getfacl <file_name>

  Examples,

  For user perspective,

```
[root@ip-172-31-19-21 ~]# mkdir /project
[root@ip-172-31-19-21 ~]# ll -d /project
drwxr-xr-x 2 root root 6 Dec 12 05:55 /project
[root@ip-172-31-19-21 ~]# useradd  amit                    → other users
[root@ip-172-31-19-21 ~]# useradd  sumit                   → other users
[root@ip-172-31-19-21 ~]# setfacl -m u:amit:rwx /project   → apply acl
[root@ip-172-31-19-21 ~]# getfacl /project                 → check acl
[root@ip-172-31-19-21 ~]# su - amit
[amit@ip-172-31-19-21 ~]$ touch /project/amit.txt    → successfully create
[amit@ip-172-31-19-21 ~]$ exit
[root@ip-172-31-19-21 ~]# setfacl -m u:sumit:r-- /project
[root@ip-172-31-19-21 ~]# su - sumit
[sumit@ip-172-31-19-21 ~]$ cd /project               → permission denied
```

For group perspective,

```
[root@ip-172-31-19-21 ~]# groupadd TCS
[root@ip-172-31-19-21 ~]# groupadd WIPRO
[root@ip-172-31-19-21 ~]# useradd -G TCS T1
[root@ip-172-31-19-21 ~]# useradd -G TCS T2
[root@ip-172-31-19-21 ~]# useradd -G WIPRO W1
[root@ip-172-31-19-21 ~]# useradd -G WIPRO W2
[root@ip-172-31-19-21 ~]# useradd -G WIPRO W3
[root@ip-172-31-19-21 ~]# ll -ld /project
drwxr-xr-x 2 root root 6 Dec 12 05:55 /project
[root@ip-172-31-19-21 ~]# getfacl /project
[root@ip-172-31-19-21 ~]# setfacl -m g:TCS:rwx /project
[root@ip-172-31-19-21 ~]# setfacl -m g:WIPRO:--- /project
[root@ip-172-31-19-21 ~]# getfacl /project
[root@ip-172-31-19-21 ~]# setfacl –m u:w3:rwx /project
```

In above example, we have no permission of group WIPRO but we can put some permission to the user 'W3' of WIPRO group.

Copy ACL of file/dir to another file/dir

```
[root@ip-172-31-19-21 ~]# touch /abc.txt
[root@ip-172-31-19-21 ~]# getfacl /abc.txt
[root@ip-172-31-19-21 ~]# getfacl /project | setfacl --set-file=- /abc.txt
[root@ip-172-31-19-21 ~]# getfacl /abc.txt
```

Remove ACL from file/dir

```
[root@ip-172-31-19-21 ~]# setfacl -x u:amit /project   → removing single user
[root@ip-172-31-19-21 ~]# setfacl -x u:sumit,u:W3 /project    → multi-user
[root@ip-172-31-19-21 ~]# setfacl -x g:IBM /project    → removing single group
[root@ip-172-31-19-21 ~]# setfacl -x g:WIPRO,g:TCS /project   → multi-group
[root@ip-172-31-19-21 ~]# getfacl /project
[root@ip-172-31-19-21 ~]# setfacl -b /abc.txt    → removing all ACL
```

- **SUDO permission:** *sudo* (**S**uper **U**ser **DO**) command in Linux is generally used as a prefix of some command that only superuser are allowed to run. If you prefix *sudo* with any command, it will run that command with elevated privileges or in other words allow a user with proper permissions to execute a command as another user, such as the superuser. This is the equivalent of "run as administrator" option in Windows. The option of sudo lets us have multiple administrators. To allow user to use sudo command, user must be listed in "/etc/sudoers" file. Or user should belong to wheel group. Wheel group is default group which allow users to use sudo command.
  Syntax,
  # sudo <command_line>
  Examples,

```
[root@ip-172-31-19-21 ~]# su – amit
[amit@ip-172-31-19-21 ~]$ useradd shubham
-bash: /usr/sbin/useradd: Permission denied
[amit@ip-172-31-19-21 ~]$ sudo useradd shubham
```

Chapter 8: Managing Processes

For everything that happens on a Linux server, a process is started. For that reason, process management is among the key skills that an administrator has to master. To do this efficiently, it is important to know which type of process you are dealing with.

A major distinction can be made between two process types:

- **Shell jobs** are commands started from the command line. They are associated with the shell that was current when the process was started. Shell jobs are also referred to as interactive processes.
- **Daemons** are processes that provide services. They normally are started when a computer is booted and often (but certainly not in all cases) they are running with root privileges.

Running Jobs in the Foreground and Background

Command Use,

&	(used at the end of a command line) Starts the command immediately in the background.
Ctrl+Z	Stops the job temporarily so that it can be managed. For instance, it can be moved to the background.
Ctrl+D	Send the End Of File (EOF) character to the current job to indicate that it should stop waiting for further input.
Ctrl+C	Can be used to cancel the current interactive job
# bg	Continues the job that has just been frozen using Ctrl+Z in the background.
# fg	Brings the last job that was moved to background execution back to the foreground.
# jobs	Shows which jobs are currently running from current shell. Displays job numbers that can be used as an argument to the command bg and fg.

Understanding Processes and Threads

Tasks on Linux are typically started as processes. One process can start several worker threads. Working with threads makes sense, because if the process is very busy, the threads can be handled by different CPUs or CPU cores available in the machine. As a Linux administrator, you cannot manage individual threads; you can manage processes, though. It is the programmer of the multithreaded application that has to define how threads relate to one another.

ps command and some of its important options,

# ps -e	standard syntax
# ps aux	system using BSD syntax
# ps axjf	process tree
# ps -eLf	Process threads
# ps -C sshd -o pid=11288	Print only the process IDs of specific id

Monitoring processes using *ps fax,*

```
[root@server2 ~]# ps fax
PID TTY STAT TIME COMMAND
1603 ? Ss 0:00 /usr/sbin/sshd -D
35395 ? Ss 0:00 \_ sshd: root@pts/1
```

Monitoring processes using *ps aux*

```
[root@server1 /]# ps aux
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
root 1 0.0 0.4 52984 4272 ? Ss Feb05 0:03 /usr/lib/
systemd/systemd --switched-root --system --deserialize 23
root 2 0.0 0.0 0 0 ? S Feb05 0:00 [kthreadd]
root 3 0.0 0.0 0 0 ? S Feb05 0:00 [ksoftirqd/0]
root 5 0.0 0.0 0 0 ? S< Feb05 0:00 [kworker/0:0H]
root 7 0.0 0.0 0 0 ? S Feb05 0:00 [migration/0]
root 8 0.0 0.0 0 0 ? S Feb05 0:00 [rcu_bh]
root 9 0.0 0.0 0 0 ? S Feb05 0:00 [rcuob/0]
...
```

Monitoring processes using *ps –ef*

```
[root@server2 ~]# ps -ef
UID PID PPID C STIME TTY TIME CMD
root 874 1 0 May14 ? 00:00:00 /sbin/auditd -n
root 885 874 0 May14 ? 00:00:00 /sbin/audispd
root 889 885 0 May14 ? 00:00:00 /usr/sbin/sedispatch
root 898 1 0 May14 ? 00:00:00 /usr/sbin/alsactl -s -n
19 -c -E ALSA_CONFIG_PATH=/etc/alsa/alsactl.conf --initfile=/lib/alsa/
root 899 1 0 May14 ? 00:00:00 /usr/sbin/bluetoothd -n
libstor+ 903 1 0 May14 ? 00:00:00 /usr/bin/lsmd -d
root 905 1 0 May14 ? 00:02:14 /usr/bin/vmtoolsd
root 908 1 0 May14 ? 00:00:00 /usr/sbin/rsyslogd -n
root 910 1 0 May14 ? 00:00:00 /usr/sbin/abrtd -d -s
...
```

Important columns of ps command's output,

PID	Process identification Number
PPID	Parent Process Identifier
PGID	Process Group Identifier (Process Group Leader)
UID	Process User Identifier (owner of process)
STIME	Time of starting the process
STATE	Process states
CMD	command by which process has started
TTY	Terminal from which process has started
PR	Priority of Process
NI	Nice value of process
LWP	Light Weight process
NLWP	Number of light weight processes

Adjusting Process Priority with nice

When Linux processes are started, they are started with a specific priority. By default, all regular processes are equal and are started with the same priority, which is the priority number 20. In some cases, it is useful to change the default priority that was assigned to the process when it was started. You can do that using the **nice** and **renice** commands. Changing process priority may make sense in two different scenarios.

Example: that you are about to start a backup job that does not necessarily have to finish fast. Typically, backup jobs are rather resource intensive, so you might want to start it in a way that it is not annoying other users too much, by lowering its priority.

Another example is where you are about to start a very important calculation job. To ensure that it is handled as fast as possible, you might want to give it an increased priority, taking away CPU time from other processes. When using **nice** or **renice** to adjust process priority, you can select from values ranging from -20 to 19.

Changing nice value using *renice* command,

```
[root@server2 ~]# sleep 1000 &
[2] 2261
[root@server2 ~]# ps –l
F S UID   PID  PPID C PRI  NI ADDR SZ WCHAN   TTY        TIME  CMD
4 S   0  2061  2054 0  80   0 - 29065 wait    pts/0  00:00:00  bash
0 S   0  2261  2061 0  80   0 - 26973 hrtime  pts/0  00:00:00  sleep
0 R   0  2387  2061 0  80   0 - 30315 -       pts/0  00:00:00  ps
[root@server2 ~]# renice -n -10 -p 2261
[root@server2 ~]# ps –l
F S UID   PID  PPID C PRI  NI ADDR SZ WCHAN   TTY        TIME  CMD
4 S   0  2061  2054 0  80   0 - 29065 wait    pts/0  00:00:00  bash
0 S   0  2261  2061 0  70 -10 - 26973 hrtime  pts/0  00:00:00  sleep
```

TIP: Do not set process priority to -20; it risks blocking other processes from getting served.

Sending Signals to Processes with kill, killall, and pkill

The Linux kernel allows many signals to be sent to processes. Use man 7 signals for a complete overview of all the available signals. Three of these signals work for all processes:

- The signal SIGTERM (15) is used to ask a process to stop.
- The signal SIGKILL (9) is used to force a process to stop.
- The SIGHUP (1) signal is used to hang up a process. The effect is that the process will reread its configuration files, which makes this a useful signal to use after making modifications to a process configuration file.

To send a signal to a process, the `kill` command is used. The most common use is the need to stop a process, which you can do by using the `kill` command followed by the PID of the process. This sends the SIGTERM signal to the process, which normally causes the process to cease its activity.

Sometimes the `kill` command does not work because the process you want to kill is busy. In that case, you can use `kill -9` to send the SIGKILL signal to the process.

Because the SIGKILL signal cannot be ignored, it forces the process to stop, but you also risk losing data while using this command. In general, it is a bad idea to use `kill -9` because,

- You risk losing data.
- Your system may become unstable if other processes depend on the process you have just killed.

Sending signal using `kill` command,

```
[root@server2 ~]# kill –l
1) SIGHUP   2) SIGINT    3) SIGQUIT   4) SIGKILL    5) SIGTRAP
6) SIGABRT  3) SIGBUS    8) SIGFPE    9) SIGKILL   10) SIGUSR1
...
[root@server2 ~]# kill -9 2261
```

TIP: Use `kill -l` to show a list of available signals that can be used with `kill`.

Linux Process States Overview

State	Meaning
Running (R)	The process is currently active and using CPU time, or in the Queue of runnable processes waiting to get services.
Sleeping (S)	The process is waiting for an event to complete.
Uninterruptable sleep (D)	The process is in a sleep state that cannot be stopped. This Usually happens while a process is waiting for I/O .
Stopped (S)	The process has been stopped, which typically has happened to an interactive shell process, using the **Ctrl+Z** key sequence.
Zombie (Z)	The process has been stopped but could not be removed by its parent, which has put it in an unmanageable state.

Monitoring process using top command

top command is used to show the Linux processes. It provides dynamic real-time view of running system. Usually, this command shows the summary information of the system and the list of processes or threads which are currently managed by the Linux Kernel. As soon as you will run this command it will open an interactive command mode where the half portion will contain the statistics of processes and resource usage. And Lower half contains a list of the currently running processes. Pressing q will simply exit the command mode.

```
[root@server2 ~]# top
top - 02:19:11 up 4 days, 10:37,  5 users,  load average: 0.07, 0.13, 0.09
Tasks: 160 total,   1 running, 159 sleeping,   0 stopped,   0 zombie
Cpu(s): 10.7%us,  1.0%sy,  0.0%ni, 88.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    760752k total,   644360k used,   116392k free,     3988k buffers
Swap:  1540088k total,    76648k used,  1463440k free,   196832k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
14401 jhradile  20   0  313m  10m 5732 S  5.6  1.4   6:27.29 gnome-system-mo
 1764 root      20   0  133m  23m 4756 S  5.3  3.2   6:32.66 Xorg
13865 jhradile  20   0 1625m 177m 6628 S  0.7 23.8   0:57.26 java
   20 root      20   0     0    0    0 S  0.3  0.0   4:44.39 ata/0
 2085 root      20   0 40396  348  276 S  0.3  0.0   1:57.13 udisks-daemon
    1 root      20   0 19404  832  604 S  0.0  0.1   0:01.21 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.01 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    4 root      20   0     0    0    0 S  0.0  0.0   0:00.02 ksoftirqd/0
    5 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
```

Controlling Jobs

A job is a process that the shell manages. Each job is assigned a sequential job ID. Because a job is a process, each job has an associated PID. There are three types of job statuses:

- **Foreground**: When you enter a command in a terminal window, the command occupies that terminal window until it completes. This is a foreground job.
- **Background**: When you enter an ampersand (&) symbol at the end of a command line, the command runs without occupying the terminal window. The shell prompt is displayed immediately after you press Return. This is an example of a background job.
- **Stopped**: If you press Control + Z for a foreground job, or enter the stop command for a background job, the job stops. This job is called a stopped job.

Example,
Running job in background

```
[root@localhost ~]# sleep 200 &
[2] 2261
[root@localhost ~]# jobs                                    -→ listing jobs
[1]-  Running                 sleep 1000 &
[2]+  Running                 sleep 200 &
```

Bring background jobs to foreground

```
[root@localhost ~]# fg %1
sleep 1000
^Z                                  → used ctrl+z to stop foreground process
[1]+ Stopped(SIGTSTP)  sleep 1000
```


Killing Processes

Usually, a process terminates on its own when they're done with their task, or when you ask them to quit. However, sometimes a process can hang up or consume a lot of CPU or RAM. In this situation, you would want to manually "kill" the process. In order to kill a process, you should first locate the details of the process. You can do this through following commands: `top`, `ps`, `pidof` and `pgrep`.
(We had already seen how to get details of processes using top and ps command in this chapter.)
Getting process details using pgrep,

`pgrep` command searches for processes currently running on the system, based on a complete or partial process name, or other specified attributes.

```
[root@localhost ~]# pgrep -a sleep            → all processes of name 'sleep'
2261 sleep 1000
2368 sleep 200
[root@localhost ~]# pgrep -a -u root          → all processes of root user
1 /bin/sh /sbin/init
2 kthreadd
3 kworker/0:0
4 kworker/0:0H
5 kworker/u2:0
6 mm_percpu_wq
...
2261 sleep 1000
2368 sleep 200
```

Getting process details using pidof,

`pidof` command searches processes currently running on system. This command is similar to pgrep command but it shows process id only.

```
[root@localhost ~]# pidof sleep
2261 2368
```

Getting process details using pstree and pidstat

`pstree` command is similar to ps command but instead of showing detailed information, pstree command shows processes in tree structure. Pstree command followed by user name will give process tree generated from that particular user.

`pidstat` command is used for monitoring individual tasks currently being managed by the Linux kernel. It writes to standard output activities for every task managed by the Linux kernel. The pidstat command can also be used for monitoring the child processes of selected tasks. The interval parameter specifies the amount of time in seconds between each report.

Important signals that are used to kill or terminate the process:

Signal_no.	**Short_name**	**Signal_no.**	**Short_name**
3	QUIT	9	KILL
15	TERM	18	CONT
19	STOP	20	TSTP

Examples,
Killing process using process id,

```
[root@localhost ~]# pgrep –a sleep
2261 sleep 1000
2368 sleep 200
[root@localhost ~]# kill -9 2261
[1]+ Killed               sleep 1000
```

Killing process using process name (using pkill or killall)

```
[root@localhost ~]# pgrep –a sleep
2261 sleep 1000
2368 sleep 200
[root@localhost ~]# killall sleep
[1]- Terminated           sleep 1000
[2]+ Terminated           sleep 200
```

System Information using `uptime,`

Uptime tell how long the system has been running. It gives a one line display of the following information. The current time, how long the system has been running, how many users are currently logged on, and the system load averages for the past 1, 5, and 15 minutes.

```
[root@localhost ~]# uptime
 14:39:12 up 1:17,  2 users,     load average: 0.00, 0.05, 0.05
```

Chapter 9: Archiving and compression

Archiving is the process of combining multiple files and directories (same or different sizes) into one file. On the other hand, compression is the process of reducing the size of a file or directory. Archiving is usually used as part of a system backup or when moving data from one system to another. One of the oldest and most common command for creating and working with backup archive *tar* command. With *tar* users can gather large amount of data into single unit known as archive.

Syntax:

tar <-options> <compress_fileName>.tar <files_to_be_compressed>

- `-c` -> create an archive.
- `-f` -> file name. (Compulsory option)
- `-v` -> verbose or view.
- `-t` -> list the content from archive.
- `-x` -> execute the content from archive.
- `-P` -> preserve permission when extracting file or directory.
- `-C` -> copy content from an archive to another directory.

Examples,

Archive file using tar command

```
[root@localhost ~]# du –sh /etc              → disk usage or size of file/dir
34M     /etc
[root@localhost ~]# tar -cvf /backup.tar /etc
tar: Removing leading `/' from member names
/etc/
/etc/netconfig
/etc/dracut.conf.d/
/etc/egl/
/etc/egl/egl_external_platform.d/
/etc/rc4.d
...
[root@localhost ~]# du –sh /backup.tar
30M     /backup.tar
```

Listing archive file,

```
[root@localhost ~]# tar -tvf /backup.tar
drwxr-xr-x root/root         0 2018-09-15 16:13 etc/
-rw-r--r-- root/root       767 2018-08-28 01:51 etc/netconfig
drwxr-xr-x root/root         0 2018-07-30 11:07 etc/dracut.conf.d/
drwxr-xr-x root/root         0 2018-09-02 22:17 etc/egl/
...
```

Extraction of archived files,

```
[root@localhost ~]# tar –xvf /backup.tar        → extract at current directory
...
[root@localhost ~]# tar –xvf /backup.tar –C /demo/   → extract at different
...                                                     location
[root@localhost ~]# du –sh /demo/etc
34M     /demo/etc
```

Compression using gzip, bzip2, and xz

Compression is a reduction in the number of bits needed to represent data. Compressing data can save storage capacity, speed up file transfer, and decrease costs for storage hardware and network bandwidth.

GZIP: GZIP is good option for compressing a lot of data as it is quick. Its memory usage is also low. GZIP compression can be used by using "`-z`" option in `tar` command or by using *`gzip`* command. GZIP compressed file can be extracted using *`gunzip`*.

BZIP2: BZIP2 provide better compression ratio compared to GZIP but require more "CPU time" to accomplish it. "`-j`" option in tar or *`bzip2`* command can be used for bzip2 compression. Bunzip2 is command to extract bzip2 compressed file.

XZ: XZ use LZMA algorithm which provide impressive compression ration but in cost of very high CPU and Memory usage. Decompression speed is better but it also consumes a lot of memory. XZ compression can be done by "`-J`" option in tar or *`xz`* command.

METHODS OF AN ARCHIVE

NO.	NAME	OPTION	EXTENSION	UNZIP
1	gzip	-z	<file_name>.tar.gz	gunzip
2	bzip2	-j	<filename>.tar.bz2	Bunzip2
3	xz	-J	<filename>.tar.xz	unxz

Examples,

Compression using `tar` command,

```
[root@localhost ~]# du –sh /etc
34M     /etc
[root@localhost ~]# tar –czvf /backup1.tar.gz /etc          → using gzip method
[root@localhost ~]# du –sh /backup1.tar.gz
8.4M    /backup1.tar.gz
[root@localhost ~]# tar –cjvf /backup2.tar.bz2 /etc         →using bzip2 method
[root@localhost ~]# du –sh /backup2.tar.bz2
7.0M    /backup2.tar.bz2
[root@localhost ~]# tar –cJvf /backup3.tar.xz /etc          → using xz method
[root@localhost ~]# du –sh /backup3.tar.xz
5.7M    /etc.tar.xz
```

Extraction using `tar` command,

```
[root@localhost ~]# mkdir /gz_xtr /bz2_xtr /xz_xtr  →create dir for extraction
[root@localhost ~]# tar –xvzf /backup1.tar.gz –C /gz_xtr          → for gzip
[root@localhost ~]# ls /gz_xtr
etc
[root@localhost ~]# tar –xvjf /packup2.tar.bz2 –C /bz2_xtr        → for bzip2
[root@localhost ~]# ls /bz2_xtr
etc
[root@localhost ~]# tar –xvJf /packup3.tar.xz –C /xz_xtr          → for xz
[root@localhost ~]# ls /xz_xtr
etc
```

Compression using `gzip` command,

```
[root@localhost ~]# du –sh /etc
34M     /etc
[root@localhost ~]# tar -cvf /etc.tar /etc
[root@localhost ~]# ls /
etc.tar
[root@localhost ~]# du -sh /etc.tar
30M     /etc.tar
[root@localhost ~]# gzip /etc.tar
[root@localhost ~]# ls /
etc.tar.gz
[root@localhost ~]# du -sh /etc.tar.gz
8.4M    /etc.tar.gz
```

When we use gzip command for compression, the original archive file get compressed and replaced with new compressed file. This helps you to avoid unnecessary wasting of storage space for both original file and compressed file. Also, new compressed file will get .gz extension automatically. You can clearly see in it above example. Similarly, bzip2 and xz commands also generates compressed file and replace the old file.

Compression using `bzip2` command,

```
[root@localhost ~]# tar -cvf /etc.tar /etc
[root@localhost ~]# ls /
etc.tar
[root@localhost ~]# du -sh /etc.tar
30M     /etc.tar
[root@localhost ~]# bzip2 /etc.tar
[root@localhost ~]# ls /
etc.tar.bz2
[root@localhost ~]# du -sh /etc.tar.bz2
7.5M    /etc.tar.bz2
```

Compression using `xz` command,

```
[root@localhost ~]# tar -cvf /etc.tar /etc
[root@localhost ~]# ls /
etc.tar
[root@localhost ~]# du -sh /etc.tar
30M     /etc.tar
[root@localhost ~]# xz /etc.tar
[root@localhost ~]# ls /
etc.tar.xz
[root@localhost ~]# du -sh /etc.tar.xz
5.8M    /etc.tar.xz
```

Extraction using `gunzip, bunzip2,` and `unxz` command

`gunzip, bunzip2,` and `unxz` commands are used to extract the respective type of compressed files. It replaces the compressed files with new extracted files and also removes extension of compressing method.

```
[root@localhost ~]# gunzip /etc.tar.gz
[root@localhost ~]# bunzip2 /etc.tar.bz2
[root@localhost ~]# unxz /etc.tar.bz2
```

(try it yourself)

Search and Filter utility in Linux

Search utilities are used to search files from the system where as filter utilities filters the output. Following are some filter tools that can we use,

- **cat** – displays the text from file as output.

```
[root@localhost ~]# cat /flower.txt
Rose
Lotus
Lotus
Lily
Daisy
Daisy
Jasmine
Marigold
Tulip
```

- **head** – display top few lines (defaults ten lines) of output

```
[root@localhost ~]# head –n 3 /flower.txt
Rose
Lotus
Lotus
```

- **tail** – displays bottom few lines (defaults ten lines) of output

```
[root@localhost ~]# tail –n 3 /flower.txt
Jasmine
Marigold
Tulip
```

- **sort** – Sorts the lines alphabetically by default but there are many options available to modify the sorting mechanism

```
[root@localhost ~]# sort /flower.txt
Daisy
Daisy
Jasmine
Lily
Lotus
Lotus
Marigold
Rose
Tulip
```

- **uniq** – Removes duplicate lines. uniq has a limitation that it can only remove continuous duplicate lines.

```
[root@localhost ~]# uniq /flower.txt
Rose
Lotus
Lily
Daisy
Jasmine
Marigold
Tulip
```

- **sed** – sed stands for stream editor. It allows us to apply search and replace operation on our data effectively. *sed* is quite an advanced filter and all its options can be seen on its man page.

```
[root@localhost ~]# sed –i s/^/#/ /flower.txt
[root@localhost ~]# cat /flower.txt
#Rose
#Lotus
#Lotus
#Lily
#Daisy
#Daisy
#Jasmine
#Marigold
#Tulip
```

- **wc** – wc command gives the number of lines, words and characters in the data.
  Options,

-l	Show Line Count
-w	Display word count
-m	Show character count

  Example,

```
[root@localhost ~]# wc /flower.txt
9   9   57    /flower.txt
```

- **grep** – grep is used to search a particular information from a text file.
  Options,

-i	Ignore uppercase vs. lowercase.
-v	Invert match.
-c	Output count of matching lines only.
-n	Precede each matching line with a line number.

  Example,

```
[root@localhost ~]# grep “Da” /flower.txt
#Daisy
#Daisy
[root@localhost ~]# grep –c “Da” /flower.txt
2
[root@localhost ~]# grep -v “Da” /flower.txt
#Rose
#Lotus
#Lotus
#Lily
#Jasmine
#Marigold
#Tulip
[root@localhost ~]# grep -c -v “Da” /flower.txt
7
```

Following are some search utilities,

- **locate** – The locate command is used to find files by their filename. The locate command is lightning fast because there is a background process that runs on your system that continuously finds new files and stores them in a database. When you use the locate command, it then searches that database for the filename instead of searching your filesystem while you wait.
  Options,

-i	Ignore uppercase vs. lowercase.
-c	Output count of matching lines only.
-l	Limiting search result

  Example,

```
[root@localhost ~]# locate flower
/flower.txt
/home/shubham/Pictures/flower-123.jpg
```

- **find** – The find Command is one of the most important and frequently used command-line utility in Linux operating systems. Find command is used to search and locate the list of files and directories based on conditions you specify for files that match the arguments. Find can be used in a variety of conditions like you can find files by permissions, users, groups, file type, date, size, and other possible criteria.
  Syntax:
  `#find <search_path> <options> <required-parameters>`
  Options,

Options	Description
-name <file_name>	Search for a file with the specified name
-perm <mode>	File's permission bits are exactly mode (octal or symbolic)
-size <N/+N/-N>	Find files with specific size (size > or size <)
-user <name>	File is owned by user specified
-uid <uid>	Files numeric user id is the same as uid
-group <grp_name>	File is owned by group specified
-gid <gid>	The file belongs to group with the ID n
-amin <n/+n/-n>	The file was last accessed n minutes ago
-mmin <n/+n/-n>	File's data was last modified n minutes ago
-atime <n/+n/-n>	The file was last accessed more n days ago
-cmin <n/+n/-n>	The file was last changed n minutes ago
-ctime <n/+n/-n>	The file was last changed more than n days ago
-mtime <n/+n/-n>	File's data was last modified n days ago
-empty	The file is empty
-executable	The file is executable
-readable	Find files which are readable
-writable	Search for files that can be written to
-type <type>	Search for a particular type (f,d,l,c,b,s,p)
-nouser	Search for a file with no user attached to it
-exec <cmd>	The file being searched which meets the above criteria and executes the provided command

Example,

```
[root@server1 ~]# find /etc -name passwd
/etc/passwd
[root@server1 ~]# find /home/ -perm 644
/home/ec2-user/.bash_logout
[root@server1 ~]# find / -size +100M
/usr/lib/locale/locale-archive
[root@server1 ~]# find / -user cbz
/home/cbz
[root@server1 ~]# find / -uid 1005
/home/cbz
[root@server1 ~]# find / -group admin
/root/demo.txt
[root@server1 ~]# find / -gid 1006
/root/demo.txt
[root@server1 ~]# ll -l /boot/grub/menu.lst
[root@server1 ~]# find /boot/ -amin -1
[root@server1 ~]# vi /etc/hosts
[root@server1 ~]# find /etc -mmin -1
/etc/hosts
[root@server1 home]# find / -name authorized_keys -exec cp -rv {} /home \;
[root@server1 home]# find / -type f -name passwd -exec rm -rf {} \;
```

Chapter 10: Managing Software

Software Management in Linux:

In Linux, software is available in the form of packages (packages are the collection of programs). And installing the packages means simply extracting the files from the archive and put it on the system. Package Management is the method of installing and maintaining the software.

Some package requires shared library, or another package, called dependency. Since there are many families of Linux, different distribution family use a different packaging system. Following are some commonly used package systems:

- Red Hat Packages (*.rpm)
- Debian Packages (*.deb)
- Ubuntu Packages (*.pkg)

There are two types of utilities we can use, low-level tool and high-level tool. Low-level tool manages package files installation, update and uninstallation whereas high-level tool can install the package with their dependencies.

Linux Family	Low-level tool	High-level tool
Red Hat	rpm	yum
Debian	dpkg	apt-get

#rpm: rpm is a RedHat Package Manager use to install package files on RedHat Linux system. It only install the package and not the dependency. It requires full pathname of package for installation.

```
#rpm <option> <package-name>
```

Syntax:

Options:

```
-i    install package file
-v    verbose
-h    show hash bar
-U    Upgrade package
-q    query package
-e    erase package
```

Example:
Installing Package from /package directory

```
#rpm –ivh /package/tree-1.6.0-10.e17.x86_64.rpm
```

Updating Package from remote server

```
#rpm –Uvh http://classroom.com/content/tree-4.6.7-17.e17.x86_64.rpm
```

[Note: Use complete pathname of package before installation. Use only service name of package after installation.]

Query of package

```
#rpm –q tree                                          -→shows the complete name
#rpm –qi tree                                         -→installed package info
#rpm –qip /package/tree-4.6.7-17.e17.x86_64.rpm       -→not-installed package info
#rpm –qa                                             -→list all installed packages
#rpm –ql tree                                      -→list of file extracted location
#rpm –qc tree                                        -→list of configuration files
#rpm –qd tree                                        -→list of documentation files
```

Erase or uninstall the package

```
#rpm –evh tree
```

#yum: yum is an open source CLI as well as GUI tool for rpm based system. It allow user to easily install, update, remove or search packages on system. Yum uses numerous third party repositories to install package automatically. This can also resolves the dependency issue. Yum does not require complete path-name of package for installation.

Because yum uses repositories to get the packages, either you should keep repositories on your system or you should have access to the remote repositories. Repository is nothing but the meta-data of all the packages. Sometimes the packages are not get installed since, its repository is not available. In such case you can create your own repository or you can install the package using rpm but if you use rpm, all the dependencies should be installed manually.

Syntax:

```
#yum <option/action> <package-name> [<-y/-d/-n>]
```

Actions:

```
       install            install package
       update             update package
       list               list all package
       info               package info.
       search             general info. of package
       remove             uninstall package
       history            shows history
       groupinstall       install group of packages
       groupupdate        update group of packages
       grouplist          list all groups of packages
       groupremove        remove installed group of packages
       repolist           list repositories
       clean              clean yum cache
```

Creating own repository:
STEP1: install required packages for creating repository if it is not already installed.

```
#rpm –ivh createrepo
#rpm –ivh deltarpm
```

STEP2: create metadata file of all packages

```
#createrepo –v /Packages
```

STEP3: create new repository file into default repository path i.e. /etc/yum.repo.d/

```
#vim /etc/yum.repo.d/clientdemo.repo

[exampleID]                              -----→repo ID
name=Sampledemo                          -----→repo name
baseurl=file:///Packages                 -----→url of packages(http:// or
ftp:// for remote url)
enabled=1                                -----→ 1 for enable, 0 for disable the
```

STEP4: check repository

```
#yum repolist
#yum clean all                           -----→to clean yum cache
```

Examples:
Install httpd package

```
#yum install httpd
#yum install tree –y                                  ---→ install without
```

Install group of packages

```
#yum grouplist                                        ---→ list all groups
#yum groupinstall “Basic Web Server”
```

Update all packages

```
#yum update –y
#yum update –y --exclude kernel                       ---→ update all but not
kernel packages
```

Show package of specific command or service

```
#yum provide “nmcli”
```

Other yum commands

```
#yum-config-manager –add-repo file:///Project          ---→ to create repository
in shortcut way
```

Chapter 11: Scheduling Task

Job scheduling is a feature that allows a user to submit a command or program for execution at a specified time in the future. On a Linux server, it is important that certain tasks run at certain times the execution of the command or program could be one time or periodically based on a pre-determined time schedule. For example, scheduling system maintenance commands to run during nonworking hours is a good practice, as it does not disrupt normal business activities.

In Linux, we have three methods to schedule a job,

- `at` – single time execution
- `crontab` – periodic execution
- `anacron` – periodic execution

1. **`# at`**

The at command uses atd service for executing jobs. At command queued the task into /var/spool/at and executes them when it is scheduled. After execution, tasks removed from the queue. After writing desired jobs, you can save jobs using shortcut key `"ctrl+d"`

Syntax, `# at "<time> <date>"`

Example,

Scheduling at job,

```
[root@localhost ~]# at "14:30 31 jan 2020"
at> touch /root/file.txt
at> mkdir /root/Practice
at> <EOT>
job 3 at Fri Jan 31 14:30:00 2020
```

Query for queued tasks,

```
[root@localhost ~]# atq
2       Thu Feb 14 10:00:00 2020 a root
3       Fri Jan 31 14:30:00 2020 a root
```

Removing queued jobs,

```
[root@localhost ~]# atrm 2
[root@localhost ~]# atq
3       Fri Jan 31 14:30:00 2020 a root
```

2. **`# crontab`**

Crontab is similar as that of window task scheduler in windows. In Linux, we schedule jobs using `crontab`. Crontab job scheduling technique is very useful for creating backup, scanning system, performing jobs with daily, weekly, monthly basis, etc. A daemon called *crond* runs in the background and check its configuration every minute to examine configuration files in order to execute commands or shell scripts specified in the crontab if the time matches with specified time. Crontab can executes job repeatedly in specified time interval.

Crond executes cron jobs on a regular basis if they comply with the format defined in the `/etc/crontab` file. Crontables for users are located in the `/var/spool/cron` directory. A cron table includes six fields separated by space or tab characters. The first five fields specify the times to run the command, and the sixth field is the absolute pathname to the command to be executed. These fields are mentioned in /etc/crontab file.

/etc/crontab file,

```
# cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,...
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
```

Syntax,

 # crontab <option>

Options,

-e	Edit cron table
-l	List cron table
-r	Remove crontable
-u	Specify user
-e –u	Edit crontable for specific user
-l –u	List crontable of specific user
-r –u	Remove crontable of specific user

Example,

Suppose, we have to schedule following jobs

1. Create a file in /root/Downloads dir with name FLOWER.txt at 10.30 pm on 15 Aug.
2. Display “Welcome to Cloudblitz” massage on terminal at midnight of every Saturday.
3. Display “HELLO” massage on terminal after every hour on 10th of Jan, Feb, and March

```
[root@localhost ~]# crontab –e                → editing current user’s crontab
```

```
30 22 15 aug * /bin/touch /root/Downloads/FLOWER.txt
0 0 * * sat /bin/echo “Welcome to Cloudblitz”
0 * 10 jan,feb,mar * /bin/echo “HELLO”
```

Listing crontable,

```
[root@localhost ~]# crontab –l
30 22 15 aug * /bin/touch /root/Downloads/FLOWER.txt
0 0 * * sat /bin/echo “Welcome to Cloudblitz”
0 * 10 jan,feb,mar * /bin/echo “HELLO”
```

Scheduling a job for Natasha user, creating file in Natasha’s home directory at 10 am on Jan 31st.

```
[root@localhost ~]# crontab –e –u natasha
```

```
0 10 31 jan * /bin/touch ~/demo
```

Chapter 12: Configuring Logging

Understanding System Logging

Most services used on a Linux server write information to log files. This information can be written to different destinations, and there are multiple solutions to find the relevant information in system logs. No less than three different approaches can be used by services to write log information:

- **direct write:** Some services write logging information directly to the log files, even some important services such as the Apache web server and the Samba file server.
- **rsyslogd:** rsyslogd is the enhancement of syslogd, a service that takes care of managing centralized log files. Syslogd has been around for a long time.
- **journald:** With the introduction of systemd to the journald log service, systemd-journald service has been introduced. This service is tightly integrated with systemd, which allows administrators to read detailed information from the journal while monitoring service status using the systemctl status command.

Understanding the Role of rsyslogd and journald

On RHEL 7, journald (which is implemented by the systemd-journald daemon) provides an advanced log management system. journald collects messages from the kernel, the entire boot procedure, and services and writes these messages to an event journal. This event journal is stored in a binary format, and it can be queried using the **journalctl** command.

Because the journal that is written by journald is not persistent between reboots, messages are also forwarded to the rsyslogd service. Rsyslogd writes the messages to different files in the /var/log directory. rsyslogd also offers features that do not exist in journald, such as centralized logging and filtering messages by using modules.

In the current state of Red Hat Enterprise Linux 7, journald is not a replacement for rsyslog; it is just another way of logging information.

Using `systemctl` Status to Show Relevant Log Information

```
[root@ip-172-31-24-16 ~]# systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor
preset: enabled)
   Active: active (running) since Thu 2020-01-30 04:38:47 UTC; 2min 5s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
 Main PID: 3367 (sshd)
   CGroup: /system.slice/sshd.service
           └─3367 /usr/sbin/sshd -D

Jan 30 04:38:47 ip-172-31-24-16.us-east-2.compute.internal systemd[1]:Starting
Op...
Jan 30 04:38:47 ip-172-31-24-16.us-east-2.compute.internal sshd[3367]: Server
list...
Jan 30 04:38:47 ip-172-31-24-16.us-east-2.compute.internal sshd[3367]: Server
list...
Jan 30 04:38:47 ip-172-31-24-16.us-east-2.compute.internal systemd[1]: Started
```

Reading Log Files

Apart from the messages that are written by journald to the journal, and which can be read using the `journalctl` command, on a Linux system you'll also find different log files in the directory /var/log.

Log File	Explanation
`/var/log/messages`	The most commonly used log file, it is the generic log file where most messages are written to.
`/var/log/dmesg`	Contains kernel log messages.
`/var/log/secure`	Contains authentication related messages. Look here to see which authentication errors have occurred on a server.
`/var/log/boot.log`	Look here for messages that are related to system startup.
`/var/log/audit/audit.log`	Contains audit messages. SELinux writes to this file.
`/var/log/maillog`	Look here for mail-related messages.
`/var/log/samba`	Provides log files for the Samba service. Notice that Samba by Default is not managed through rsyslog, but writes directly to the /var/log directory.
`/var/log/sssd`	Contains messages that have been written by the sssd service, which plays an important role in the authentication process.
`/var/log/cups`	Contains log messages that were generated by the print service CUPS.
`/var/log/httpd/`	Directory that contains log files that are written by the Apache web server. Notice that Apache writes messages to these files directly and not through rsyslog.

Understanding Log File Contents

Example, (message logs)

```
[root@ip-172-31-24-16 ~]# tail -5 /var/log/messages
Jan 30 04:43:47 ip-172-31-24-16 amazon-ssm-agent: </body>
Jan 30 04:43:47 ip-172-31-24-16 amazon-ssm-agent: </html>
Jan 30 04:45:08 ip-172-31-24-16 dhclient[3012]: XMT: Solicit on eth0, interval
131720ms.
Jan 30 04:47:19 ip-172-31-24-16 dhclient[3012]: XMT: Solicit on eth0, interval
114430ms.
Jan 30 04:49:14 ip-172-31-24-16 dhclient[3012]: XMT: Solicit on eth0, interval
114820ms.
```

- **Date and time:** Every log message starts with a timestamp. For filtering purposes,
- **Host:** The host the message originated from. This is relevant because rsyslogd can be configured to handle remote logging as well.
- **Service or process name:** The name of the service or process that generated the message.
- **Message content:** The content of the message, which contains the exact message that has been logged.

Live Log File Monitoring,
Syntax,
 # tail –f <logfile>
Example,

```
[root@ip-172-31-24-16 ~]# tail -f /var/log/messages
Jan 30 04:47:19 ip-172-31-24-16 dhclient[3012]: XMT: Solicit on eth0, interval
114430ms.
Jan 30 04:49:14 ip-172-31-24-16 dhclient[3012]: XMT: Solicit on eth0, interval
114820ms.
Jan 30 04:50:01 ip-172-31-24-16 systemd: Created slice User Slice of root.
Jan 30 04:50:01 ip-172-31-24-16 systemd: Starting User Slice of root.
Jan 30 04:50:01 ip-172-31-24-16 systemd: Started Session 3 of user root.
Jan 30 04:50:01 ip-172-31-24-16 systemd: Starting Session 3 of user root.
Jan 30 04:50:01 ip-172-31-24-16 systemd: Removed slice User Slice of root.
Jan 30 04:50:01 ip-172-31-24-16 systemd: Stopping User Slice of root.
```

Using logger,
Most services write information to the log files all by themselves. The logger command enables users to write messages to rsyslog from the command line. User can write logs manually.
Syntax,
logger <option> <Message>
Example, (Writing log with priority option,)

```
[root@ip-172-31-24-16 ~]# logger –p local3.err “Danger”
[root@ip-172-31-32-167 ~]# tail -3 /var/log/messages
Jan 30 06:10:41 ip-172-31-32-167 systemd-logind: New session 5 of user ec2-
user.
Jan 30 06:10:41 ip-172-31-32-167 systemd: Starting Session 5 of user ec2-user.
Jan 30 06:11:14 ip-172-31-32-167 ec2-user: Danger
```

Configuring rsyslogd

To make sure that the information that needs to be logged is written to the location where you want to find it, you can configure the rsyslogd service through the /etc/rsyslog.conf file.

- **Understanding rsyslogd Configuration Files**

  Like many other services on RHEL 7, the configuration for rsyslogd is not defined in just one configuration file. The /etc/rsyslogd.conf file is the central location where rsyslogd is configured. From this file, the content of the directory /etc/rsyslog.d is included. This directory can be populated by installing RPM packages on a server. When looking for specific log configuration, make sure to always consider the contents of this directory also.

  If specific options need to be passed to the rsyslogd service on startup, you can do this by using the /etc/sysconfig/rsyslog file. This file by default contains one line, which reads SYSLOGD_OPTIONS. On this line, you can specify rsyslogd startup parameters. The SYSLOGD_OPTIONS variable is included in the systemd configuration file that starts rsyslogd. Theoretically, you could change startup parameters in this file, as well, but that is not recommended

- **Understanding rsyslog.conf Sections**

The rsyslog.conf file is used to specify what should be logged and where it should be logged. To do this, you'll find different sections in the configuration file:

■ **#### MODULES ####:** rsyslogd is modular. Modules are included to enhance the supported features in rsyslogd.

■ **#### GLOBAL DIRECTIVES ####:** This section is used to specify global parameters, such as the location where auxiliary files are written or the default timestamp format.

■ **#### RULES ####:** This is the most important part of the rsyslog.conf file. It contains the rules that specify what information should be logged to which destination.

- **Understanding Facilities, Priorities, and Log Destinations**

To specify what information should be logged to which destination, rsyslogd uses facilities, priorities, and destinations:

■ a *facility* specifies a category of information that is logged. `rsyslogd` uses a fixed list of facilities, which cannot be extended. This is because of backward compatibility with the legacy syslog service.

■ a *priority* is used to define the severity of the message that needs to be logged. When specifying a priority, by default all messages with that priority and all higher priorities are logged.

■ a *destination* defines where the message should be written to. Typical destinations are files, but rsyslog modules can be used as a destination as well, to allow further processing through an rsyslogd module.

We can see example of RULES section in rsyslog file, i.e. `/etc/rsyslog.conf`. Following are some rsyslog priorities,

Priority	**Used for**
Debug	Debug messages that will give as much information as possible about service operation.
Info	Informational messages about normal service operation.
Notice	Used for informational messages about items that might become an issue later.
warning / warn	Something is suboptimal, but there is no real error yet.
err /error	A noncritical error has occurred.
Crit	A critical error has occurred
Alert	Used when the availability of the service is about to be discontinued.
emerg / panic	Message generated when the availability of the service is discontinued

Rotating Log Files

To prevent syslog messages from filling up your system completely, the log messages can be rotated. That means that when a certain threshold has been reached, the old log file is closed and a new log file is opened. The logrotate utility is started periodically through the crond service to take care of rotating log files. The default settings for log rotation are kept in the file `/etc/logrotate.conf`. Logs can be rotated with customized settings that will kept in `/etc/logrotate.d/` directory.

Sample content for logrotate setting,

```
[root@localhost ~]# cat /etc/logrotate.d/yum
/var/log/yum.log {
    missingok
    notifempty
    size 30k
    yearly
    create 0600 root root
}
```

Logs can be rotated forcefully using `logrotate` command followed by configuration file for logrotate. The logrotate command reads out configuration file and can perform

Working with journald

The systemd-journald service stores log messages in the journal, a binary file that is stored in the file /run/log/journal. This file can be examined using the `journalctl` command.

Understanding journalctl,

- `journalctl,` you'll see the content of the journal since your server last started, starting at the beginning of the journal. The content is shown in less, so you can use common less commands to walk through the file.
- `journalctl --no-pager,` this shows the contents of the journal without using a pager.
- `journalctl –f`, this opens the live view mode of journalctl, which allows you to see new messages scrolling by in real time. Use Ctrl+C to interrupt.
- Type `journalctl` and press the Tab key twice. This shows specific options that can be used for filtering. Ex, `journalctl _UID=0`
- `journalctl -n 20`, It displays the last 20 lines of the journal (just like tail -n 20).
- `journalctl -p err,` this will shows errors only. You can use other priorities with –p option.
- If you want to view journal messages that have been written in a specific time period, you can use the `--since` and `--until` options. Both options take the time parameter in the format `YYYY-MM-DD hh:mm:ss.` Also, you can use yesterday, today, and tomorrow as parameters. Ex, `journalctl --since yesterday,` to show all messages that have been written since yesterday.
- `journalctl` allows you to combine different options, as well. So, if you want to show all messages with a priority err that have been written since yesterday, use journalctl --since yesterday -p err .
- If you need as much detail as possible, use `journalctl -o verbose,` this shows different options that are used when writing to the journal.
- `journalctl _SYSTEMD_ UNIT=sshd.service` to show more information about the sshd systemd unit.

Preserving the systemd Journal

By default, the journal is stored in the file /run/log/journal. The entire /run directory is used for current process status information only, which means that the journal is cleared when the system reboots.

Setting journald Parameters Through /etc/systemd/journald.conf

```
[Journal]
#Storage=auto
#Compress=yes
#Seal=yes
#SplitMode=login
#SyncIntervalSec=5m
#RateLimitInterval=30s
#RateLimitBurst=1000
#SystemMaxUse=
#SystemKeepFree=
#SystemMaxFileSize=
#RuntimeMaxUse=
#RuntimeKeepFree=
#RuntimeMaxFileSize=
#MaxRetentionSec=
#MaxFileSec=1month
#ForwardToSyslog=yes
#ForwardToKMsg=no
#ForwardToConsole=no
#TTYPath=/dev/console
#MaxLevelStore=debug
#MaxLevelSyslog=debug
#MaxLevelKMsg=notice
#MaxLevelConsole=info
```

Making the journald Journal Permanent

STEP 1: Open a root shell and type `mkdir /var/log/journal`.

STEP2: Before journald can write the journal to this directory, you have to set ownership. Type `chown root:systemd-journal /var/log/journal` , followed by `chmod 2755 /var/log/journal .`

STEP 3: Next, you can either reboot your system (restarting the systemd-journald service is not enough) or use the `killall -USR1 systemd-journald` command.

STEP 4: The systemd journal is now persistent across reboots. If you want to see the log messages since last reboot, use `journalctl -b .`

Chapter 13: Managing Partitions

A file system is an organized structure of data-holding files and directories residing on storage device. The process of adding new file system into existing directories is called mounting and the directory is called mount point. Hard disks and other storage device are normally divided into partitions. In Linux, Different types of hard disks has different representations for their partitions.

==➔ Types of Hard disks and there partition representation:

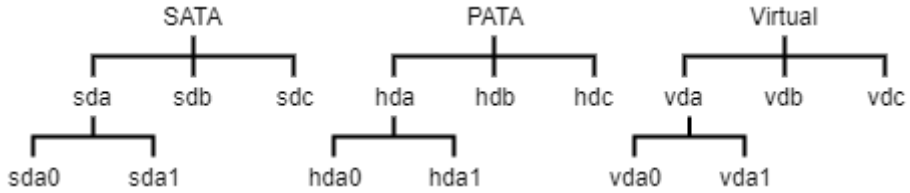


Fig 14.1 HardDisk Representation

These representation are the special files called as block device. The block devices are stored in the **/dev** directory. Disk Partitioning is the method of dividing hard drives into multiple logical storage units referred to as partitions. There are two schemes of disk partitioning: 1. MBR partitioning scheme 2. GPT partitioning scheme.

MBR Partitioning Scheme: Master Boot Record partitioning scheme (MBR) dictates how disks should be partitioned on system. MBR uses the standard BIOS partition table thus it has size limit of 2TB. MBR has size 512 bytes among which 64 bytes are used for partition table info. Each partition creation requires 16bytes thus MBR scheme supports a maximum of four primary partitions (or 3 primary and 1 extended).

GPT Partitioning Scheme: GUID Partition Table (GPT) overcomes from limitations of the MBR partition. It has size limit of 8 ZB and making more than four primary partition is also possible.

==➔ Types of Partitions:

Primary: A primary partition is in which operating system can be installed. In MBR, maximum four primary partition can be create. The primary partition which is use to boot the system is called an active partition. (Active partition is the partition from which operating system is loaded.

Extended: Extended partition breaks the limit of four partition. Using extended partition we can create no. of logical partitions. Extended partition holds logical partition. Only one extended partition is allowed.

Logical: Logical Partitions are created inside the extended partition. RHEL 6 supports maximum of 12 logical partitions whereas RHEL 7 supports maximum of 60 logical partitions. To create logical partition, first you must create extended partition.

MANAGING MBR PARTITION:

For MBR partitioning scheme, fdisk partition editor is used.

Creating Partition

```
[root@localhost Desktop]# fdisk /dev/sdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table
Building a new DOS disklabel with disk identifier 0x56d50362.

Command (m for help):
```

2. Above step opens MBR partition editor, enter m to get help about commands.

```
Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   g   create a new empty GPT partition table
   G   create an IRIX (SGI) partition table
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)

Command (m for help):
```

3. Enter n to request new partition and give values of requesting fields.

```
Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p                                    ---→ select partition type
as per above choice
Partition number (1-4, default 1): 1                     ---→ give partition number
First sector (2048-62914559, default 2048): 2048         ---→ starting sector of
partition
Last sector, +sectors or +size{K,M,G} (2048-62914559, default 62914559): +2G  ---→ size
of disk
Partition 1 of type Linux and of size 2 GiB is set
```

4. Toggle the partition if partition should have type other than Linux. 'L' will show list of partition type.

```
Command (m for help): t
Selected partition 1
Hex code (type L to list all codes): L
```

5. Partition will not be created until the changes has been saved. Give 'w' command to save the changes.

```
Command (m for help): w
```

To view partitioned blocks,

```
[root@localhost Desktop]# lsblk
```

To update partition table,

```
[root@localhost Desktop]# partprobe
```

File Systems

A file system is the way in which files are named, stored, retrieved and organized on the storage device or partition. Following are some commonly used file systems in now a days,

- xfs – A high performance filesystem originally developed by Silicon Graphics that works extremely well with large files. This filesystem is default for RHEL 7. NASA still uses this file system on their 300 TB server.
- ext2 – ext2 filesystem was introduced in 1993 and it was the first default file system in several Linux file system. It overcomes the limitation of legacy ext file system. The maximum supported size is 16GB to 2TB. Journaling feature is not available. ext2 normally is used in flash based storage like pendrives, sd cards, etc.
- ext3 – It was introduced in 2001 with all the features of ext2 and additional journaling feature. It also provides facility to upgrade from ext2 to ext3 without having to backup and restore data.
- ext4 – It is the high anticipated ext3 successor. ext4 was introduced in 2008 with backward capabilities. It supports maximum file size of 16TB. It has option to turn off journaling feature.
- vfat – Microsoft extended FAT filesystem.

Assign file system to partition,

```
[root@localhost Desktop]# mkfs.xfs /dev/sdb1
```

OR

```
[root@localhost Desktop]# mkfs -t ext4 /dev/sdb1
```

List assigned file system disks, (it shows filesystem as well as partition block id)

```
[root@localhost Desktop]# blkid
```

Mounting Partition

Mounting is process of adding partition in to the system am specific directory. The directory is called as mounting point. There are two methods of mounting, temporary mounting mount the partition temporary which get unmounts automatically after the reboot where as permanent mounting allows to mount permanent.

Temporary mount,

```
[root@localhost Desktop]# mount /dev/sdb1 /hello          -->/hello is directory
```

```
[root@localhost Desktop]# vim /etc/fstab

# /etc/fstab
# Created by anaconda on Tue Jun  4 07:23:02 2019
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=c38f98a3-3d9e-4621-99f9-5006264d72cc /                       xfs     defaults
1 1
UUID=757aa47c-0bd7-42a0-9a8e-a379998c0975 /boot                   xfs     defaults
1 2
UUID=f1344dcb-2f8b-4311-9b93-7dc26a16e8fb swap                    swap    defaults
0 0

/dev/sdb1      /hello         ext4           defaults       0 0

<Partition_name>  <Mount_point> <filesystem_type>  <permissions>  <dumps>  <priorities>
```

```
[root@localhost Desktop]# mount -a          -->mount all using file fstab
[root@localhost Desktop]# df –hT            -->to see mounted partitions
```

Unmounts Partition

Temporary unmounting,

```
[root@localhost Desktop]# umount /dev/sdb1
```

Permanent Unmounting can be done by removing the entry from fstab file and perform #mount –a.

Removing Partition

Before removing partition, first unmounts all the partitions that you wants to remove. Then remove partition using fdisk partition editor.

```
[root@localhost Desktop]# fdisk /dev/sdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): d                       ---→ d command is used to delete
partition
Partition number (1,2, default 2): 1          ---→ select partition no.(takes last
partition as default)
Partition 1 is deleted
```

Swap partition

swap space in Linux is used when the amount of physical memory is getting full. If the system needs more memory resource and the RAM is full, all inactive pages in memory moved to swap space. (swap should not be consider a replacement of RAM).

How to create swap space?

Step1: Create primary or logical partition

```
[root@localhost Desktop]# fdisk /dev/sdb          --→ create new partition
```

Step2: Assign swap file system

```
[root@localhost Desktop]# mkswap /dev/sdb1
```

Step3: Mount as swap partition

```
[root@localhost Desktop]# swapon /dev/sdb1        --→ mount swap partition temporary
```

```
[root@localhost Desktop]# vim /etc/fstab          --→ for permanent mount


/dev/sdb2 swap swap defaults 0 0                  -→Entry for swap partition. Mount
point should be swap
```

See swap partitions,

```
[root@localhost Desktop]# swapon
```

unmounts swap partition,

```
[root@localhost Desktop]# swapoff /dev/sdb1
```

Chapter 14: Managing LVM Logical Volumes

Logical Volume Management (LVM):

Logical Volume Management or LVM is another disk partition management tool for the Linux. With LVM, physical disk partitions can be added to pools of space called volume groups. Logical volumes are assigned space from volume groups as needed. Features of LVM,

- More space can be to logical volume from the volume group while the volume is still in use.
- More physical can be added to a volume group if the volume group begins to run out of space. The physical volumes can be from same or different disks.
- Move data from one physical volume to another, so you can remove smaller disks and replace them with larger ones while the filesystem are still in use.

Create Physical Volume

Physical volumes are created from the existing partition,

```
[root@localhost Desktop]# pvcreate /dev/sdb1              --→ sdb1 is the existing
fdisk partition
```

List Physical volumes,

```
[root@localhost Desktop]# pvs                             --→ list of physical volumes
[root@localhost Desktop]# pvdisplay                       --→ show details of physical
```

Create Volume Group

Volume Group is the pool of space using virtual partitions

```
[root@localhost Desktop]# vgcreate vg_demo /dev/sdb1 /dev/sdb2    --→ create volume
                                                                   group using physical
                                                                   volumes
[root@localhost Desktop]# vgs                                     --→ list of volume
groups
[root@localhost Desktop]# vgdisplay                               --→ details of volume
group
```

Logical volumes are created inside volume group,

```
[root@localhost Desktop]# lvcreate –L +300M –n lv1 vg_demo
                           <size_option> <size> <name_option> <logical_vol_name> <vol_grp_name>
```

```
[root@localhost Desktop]# lvs                        --→ list of logical volumes
[root@localhost Desktop]# lvdisplay                  --→ show details of logical volumes
```

Extend volumes using LVM

Extend logical volume

```
[root@localhost Desktop]# lvextend –L +300M /dev/vg_demo/lv1      -→ extend lv size
                                                                  without unmounting it
[root@localhost Desktop]# resize2fs /dev/vg_demo/lv1              -→ resize ext type
                                                                  volumes
```

Extend volume group (extending vg means adding more physical partition in volume group.)

```
[root@localhost Desktop]# vgextend vg_demo /dev/sdb3          -→ sdb3 is physical
                                                              volume added into
                                                              volume group
```

Reduce volumes using LVM

To reduce logical volumes, first unmounts the logical volume and clear processes if any running on that partition. Reduction is possible only in extended type of file system.

```
[root@localhost Desktop]# umount /dev/vg_demo/lv1                  →unmount lv
[root@localhost Desktop]# fsck.ext4 –f /dev/vg_demo/lv1            →clear processes
[root@localhost Desktop]# e2fsck –f /dev/vg_demo/lv1               →if error occur
                                                              for above command
[root@localhost Desktop]# lvreduce –L -50M /dev/vg_demo/lv1        →reduce logical
```

Reducing volume groups means removing physical volumes from the volume group. Only unused physical partitions can be removed.

```
[root@localhost Desktop]# vgreduce vg_demo /dev/sdb3
```

Remove volumes using LVM

Remove LVM partition (first unmounts the volume)

```
[root@localhost Desktop]# lvremove /dev/vg_demo/lv1           →remove logical
                                                              volume
[root@localhost Desktop]# vgremove vg_demo                    →remove volume group
when all lv remove
[root@localhost Desktop]# pvremove /dev/sdb1                  →remove physical vol
when all vg remove
```

Chapter 15: Basic Kernel Management

Install or Update Kernel:

Kernel packages can be upgraded or new version of kernel package can be added into existing system. Below example shows how to update the kernel.

```
[root@desktop0 ~]# rpm –ivh /pkg/kernel-3.10.123.x86_64.rpm      → install
new version kernel
```

Chapter 16: SSH

Note: Assume that
IP of server = 172.25.0.11
IP of client = 172.25.0.10

SSH (Secure Shell):

SSH, also known as Secure Shell or Secure Socket Shell, is a network protocol that gives users, particularly system administrators, a secure way to access a computer over an unsecured network. SSH also refers to the suite of utilities that implement the SSH protocol. Secure Shell provides strong authentication and encrypted data communications between two computers connecting over an open network such as the internet. SSH is widely used by network administrators for managing systems and applications remotely, allowing them to log into another computer over a network, execute commands and move files from one computer to another.

SSH refers both to the cryptographic network protocol and to the suite of utilities that implement that protocol. SSH uses the client-server model, connecting a secure shell client application, the end at which the session is displayed, with an SSH server, the end at which the session runs. SSH implementations often include support for application protocols used for terminal emulation or file transfers. SSH can also be used to create secure tunnels for other application protocols. An SSH server, by default, listens on the standard Transmission Control Protocol (TCP) port 22.

Secure Shell Capabilities,

- Secure remote access to SSH-enabled network systems or devices, for users as well as automated processes;
- secure and interactive file transfer sessions;
- automated and secured file transfers;
- secure issuance of commands on remote devices or systems; and
- Secure management of network infrastructure components.

Default Port: 22
Configuration file : */etc/ssh/sshd_config*
Package required: openssh-server, openssh-client, openssh
Daemon Service: sshd
There are two types of SSH authentication:

1. Password Based Authentication – It uses password for allowing users to access their shell remotely.
2. Key Based Authentication – It uses key pairs (i.e. public key and private key pair) for authentication.

1. Password Based Authentication :

Password-based authentication is the simplest means of authenticating an SSH connection to another machine. The user provides the password for the account at the time of connection.

Syntax:

```
[root@server0 ~]# ssh user@server_ip
```

Example:

```
[root@server0 ~]# ssh root@172.31.19.233
root@172.31.19.233's password:
```

You can enable or disable password based authentication by changing its configuration:

```
[root@server0 ~]# vi /etc/ssh/sshd_config
...
60 # To disable tunneled clear text passwords, change to no here!
61 #PasswordAuthentication yes
62 #PermitEmptyPasswords no
63 PasswordAuthentication yes
...
```

2. Key Based Authentication :

SSH key pairs are two cryptographically secure keys that can be used to authenticate a client to an SSH server. Each key pair consists of a public key and a private key. The private key is retained by the client and should be kept absolutely secret. Any compromise of the private key will allow the attacker to log into servers that are configured with the associated public key without additional authentication. As an additional precaution, the key can be encrypted on disk with a passphrase.

The associated public key can be shared freely without any negative consequences. The public key can be used to encrypt messages that only the private key can decrypt. This property is employed as a way of authenticating using the key pair.

The public key is uploaded to a remote server that you want to be able to log into with SSH. The key is added to a special file within the user account you will be logging into called ~/.ssh/authorized_keys.

When a client attempts to authenticate using SSH keys, the server can test the client on whether they are in possession of the private key. If the client can prove that it owns the private key, a shell session is spawned or the requested command is executed.

- To generate key pair, *ssh-keygen* is used at server:

```
[root@server0 ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):                -→ where
to save keys
Enter passphrase (empty for no passphrase):                             -→
password for key
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:s8E4nAFAMANL2mDLmjYjfYWXLgpEf74dm17DgyFfckk root@server0.example.com
The key's randomart image is:
+---[RSA 2048]----+
|B*o..            |
|**o  o .         |
|o+....=  E       |
|oo  o= =. .      |
|=+. oo*+S+       |
|o.oo .=.@+       |
|  .  . *.=       |
|      . . o      |
|       .         |
+----[SHA256]-----+
```

- Generated keys has extension as .pub for public key and no extension (or .pem) for private key:

```
[root@server0 ~]# ls ~/.ssh/
authorized_keys  id_rsa  id_rsa.pub
                         ──────────►  public key
                ─────────────────────►  private key
```

- Public key must be kept in *~/.ssh/authorized_keys* file

```
[root@server0 ~]# cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

- "600" permission has to be applied on both keys.

```
[root@server0 ~]# chmod 600 ~/.ssh/id_rsa
[root@server0 ~]# chmod 600 ~/.ssh/authorized_keys
[root@server0 ~]# ll ~/.ssh/
total 12
-rw------- 1 root root  975 Oct 21 06:08 authorized_keys
-rw------- 1 root root 1679 Oct 21 05:37 id_rsa
-rw-r--r-- 1 root root  430 Oct 21 05:37 id_rsa.pub
```

- You can delete id_rsa.pub key since it is no longer needed

```
[root@server0 ~]# rm –f ~/.ssh/id_rsa.pub
```

- Transfer private key to the client using any media such as pen drive, mail, ftp, etc. Here we are transferring key using scp:

```
[root@server0 ~]# scp ~/.ssh/id_rsa root@172.25.0.10:/root      -→ ip address
                                                                of client as
                                                                destination
```

- Now, client can access server's shell through ssh using private key,
  Syntax: (To access secure shell using key-based authentication)

```
[root@server0 ~]# ssh –i private_key user@server_ip
```

Example

```
[root@client ~]# ssh –i /root/id_rsa root@172.25.0.11
```

Below are some examples that helps you to understand more about the ssh,

- ➢ You can execute single command remotely without taking whole ssh shell access,

```
[root@client ~]# ssh root@172.25.0.11 mkdir /dir_name2
```

- To get access of graphical applications using secure shell,

```
[root@client ~]# ssh –X root@172.25.0.11
[root@server0 ~]# firefox &                          -→ graphical application
```

- Secure shell access using customized port,

```
[root@server0 ~]# vim /etc/ssh/sshd_config          -→ mention port new number
in config. file
...
     15 # semanage port -a -t ssh_port_t -p tcp #PORTNUMBER
     16 #
     17 #Port 22     -→ remove # symbol and change port number. It will be
like below lines,
...
```

```
...
     15 # semanage port -a -t ssh_port_t -p tcp #PORTNUMBER    -→ execute this
     16 #                                      command by replacing #PORTNUMBER
     17 Port 2020                             with <portnumber> i.e. 2020
...
```

```
[root@server0 ~]# semanage port -a -t ssh_port_t –p 2020          -→ this will
                                                                  allow port
                                                                  from selinux
```

```
[root@server0 ~]# systemctl restart sshd                    -→ restart ssh
```

```
[root@server0 ~]# firewall-cmd --add-port=2020/tcp                -→ open
```

Access remote ssh from client using 2020 port

```
[root@client ~]# ssh root@172.25.0.11 –p 2020
```

Chapter 17: Managing and Understanding Boot Procedure

After pressing the power button, system starts in background till we see the login screen on the display. The boot procedure occurs in six stages of process.

1. **BIOS –** Basic input/output System
2. **MBR –** Master Boot Record
3. **GRUB2 –** Grand Unified Boot loader
4. **Kernel –** Kernel
5. **Systemd –** First process of system
6. **Runlevel-Target –** Run Level Programs

1. BIOS :

BIOS – Basic Input/Output System is a firmware interface that controls not only booting process and also provides all the control of low-level interface to attached peripheral devices. When the system is powered on, it will read all the devices settings and executes the POST (Power ON Self-Test) process to recognize the Hardware devices to test and initialize the system Hardware components. This process is also called as System Integrity Check. After successful POST process, it searches boot loader program in hard drives, CD_ROM, floppy disk, etc. and load boot loader program into ROM. Boot sequence can be changed if any interruption occurs (such as pressing "delete" key). Once boot loader program is detected and loaded into the memory, the control is given to the boot loader program.

2. MBR :

Master boot Record placed in the first sector of the Linux boot Hard Drive and this information pre-loads into ROM (Read Only Memory) by BIOS. The MBR is only 512 bytes in size and it contains the machine code instructions for booting the Operating System, it's called a boot loader, along with the partition table and validation check. Once the BIOS finds and loads the boot loader (GRUB2) program into (ROM) memory or Hard drive, it takes the control of the boot process to it. Simply MBR (Master Boot Record) loads and executes the GRUB2 boot loader.

3. GRUB2:

GRUB stands for Grand Unified Boot loader. GRUB2 is the default boot loader program in all latest version of like Red Hat/CentOS 7 and also Ubuntu from version 9.10. It has been replaced by GRUB boot loader also known as GRUB legacy. GRUB2 configuration file located in /boot/grub2/grub.cfg and it is automatically generated by grub2-mkconfig using templates from /etc/grub.d and settings from /etc/default/grub. The boot loader (GRUB2 for RHEL 7) starts the RHEL 7 kernel and initial RAM disk (initrd). GRUB 2 is installed in the boot sector of your server's hard drive and is configured to load a Linux kernel and the initramfs and the initrd is an initial root file system that will mount prior to the real root file system on Linux system. If you have multiple kernel images installed on your system, you can choose which one to be executed. GRUB displays a splash screen, waits for few seconds, if you don't enter anything, it loads the default kernel image as specified in the grub configuration file. GRUB

has the knowledge of the file. So, in simple terms GRUB just loads and executes Kernel and initrd images.

4. KERNEL

Linux Kernel is the central core of the OS and it is the first program loaded on the system starts up. While system starting kernel loads all the necessary Kernel Modules and Drives from initrd.img to load system first process systemd in Linux 7. It mounts the root file system as specified in the "root=" in grub.conf. Below command can show systemd process id (PID).

```
[root@desktop0 ~]# ps –ef | grep systemd
```

5. SYSTEMD

Systemd process is the first process (PID 1) to run on Linux 7 systems, it initializes the system and launches all the services that were once started by the traditional init (/etc/init.d) process. Systemd process reads the configuration file of /etc/systemd/system/default.target, then its load the OS in targeted runlevel.target. This tells systemd to start everything in the /usr/lib/systemd/system/basic.target before starting the other multi-user services.

6. RUNLEVEL PROGRAMS

Systemd uses 'targets' instead of runlevels. By default, there are two main targets:

multi-user.target: analogous to runlevel 3
graphical.target: analogous to runlevel 5

To view current default target, run:

```
[root@desktop0 ~]# systemctl get-default
```

To set a default target, run: ex: systemctl set-default TARGET.target

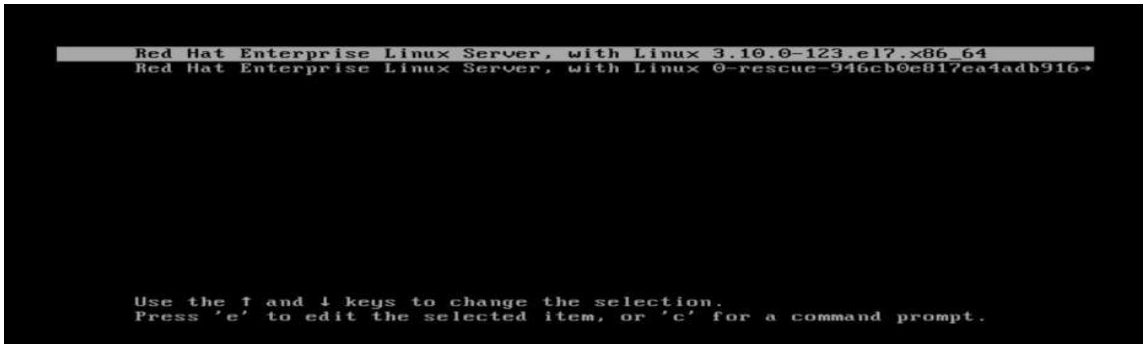
```
[root@desktop0 ~]# systemctl set-default graphical.target
```

Target List in Red Hat Linux 7.

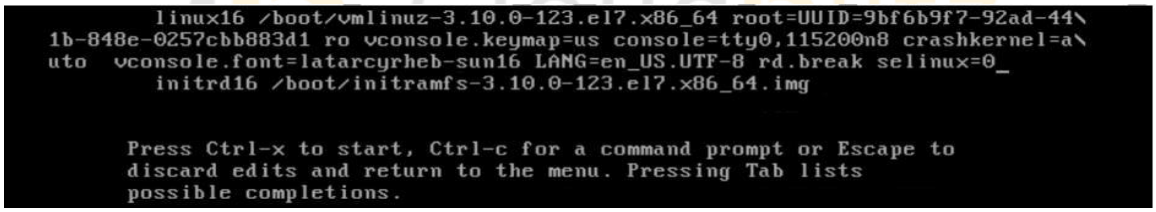
* runlevel0.target -> poweroff.target
* runlevel1.target -> rescue.target
* runlevel2.target -> multi-user.target
* runlevel3.target -> multi-user.target
* runlevel4.target -> multi-user.target
* runlevel5.target -> graphical.target
* runlevel6.target -> reboot.target

Chapter 18: Resetting Root Password

1. Reboot the system.
2. Interrupt the boot loader countdown by pressing any key.

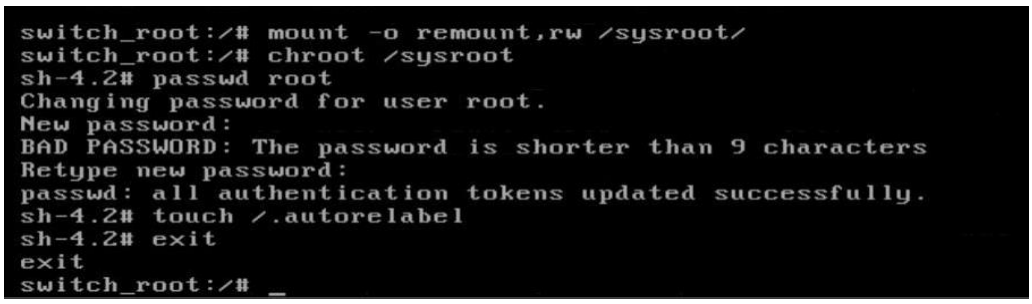


3. Move the cursor to the entry that needs to be booted.
4. Press 'e' to edit the selected entry.
5. Move the cursor to the kernel command line (the line start with linux16).
6. Press End key then give space Append *rd.break* (this will break just before control is handed from the initramfs to the actual system).
7. Press Ctrl+x to boot with changes.



To recover the root password from this point, use the following procedure.

1. Remount /sysroot as read-write.
   switch_root: /# mount –o remount,rw /sysroot
2. Switch into a chroot jail, where /sysroot is treated as the root of the file system tree.
   switch_root:/# chroot /sysroot
3. Set a new root password.
   sh-4.2# passwd root
4. Make sure that all authentication update successfully and all unlabeled file Relabeled during boot.
   sh-4.2# touch /.autorelabel
   As shown in figure…



Chapter 19: Working with LDAP

LDAP, Lightweight Directory Access Protocol, is an Internet protocol that email and other programs use to look up information from a server. LDAP is mostly used by medium-to-large organizations. If you belong to one that has an LDAP server, you can use it to look up contact info and the like. LDAP server mostly used to create centralize user server system so that user can be access their account from any system in the network. Following is the example of client side configuration of LDAP service.

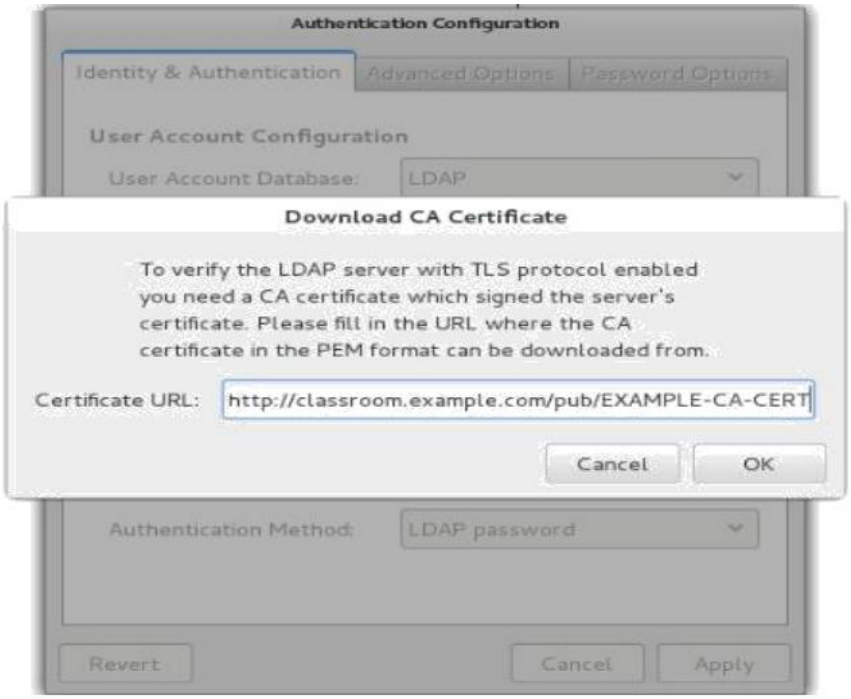
Example,

Authenticating users from LDAP Directory Servers which have:
Server Name: classroom.example.com.
Base DN: dc=example, dc=com.
Download certificate from http://classroom.example.com/pub/EXAMPLECA-CERT
Authenticate with users ldapuserX with have password of "password".

```
[root@server3 ~]# yum install openldap-clients sssd authconfig-gtk -y
[root@server3 ~]# authconfig-gtk
```

Fill all information in below dialog box.

Click on Download CA Certificate the fill certification path in below dialog box.




```
[root@server3 ~]# getent passwd        ---→ Does not show ldapuser
[root@server3 ~]# vim /etc/sssd/sssd.conf
       . . .
       enumerate=true
       [sssd]
:wq

[root@server3 ~]# systemctl restart sssd
[root@server3 ~]# systemctl enable sssd
[root@server3 ~]# getent passwd
ldapuser3:x:1703:1703:LDAP Test User 3:/home/guests/ldapuser3:/bin/bash
```

Configuring autofs such that server's home directory classroom.example.com:/home/guests/ldapuserX are shared by nfs service. Mount this directory on /home/guests/ldapuserX with nfs version 3. Without this home directory, ldapuser cannot login in to the system.

```
[root@server3 ~]# yum install autofs -y
[root@server3 ~]# vim /etc/auto.master
      /misc /etc/auto.misc
      /home/guests /etc/auto.ldap
:wq

[root@server3 ~]# vim /etc/auto.ldap
ldapuser3 -rw,-fstype=nfs,vers=3 classroom.example.com:/home/guests/ldapuser3
:wq

[root@server3 ~]# systemctl stop autofs
[root@server3 ~]# systemctl start autofs
[root@server3 ~]# systemctl enable autofs
[root@server3 ~]# su - ldapuser3
Last login: Mon Oct 16 13:45:13 IST 2017 on pts/0
[ldapuser3@server3 ~]$ pwd
/home/guests/ldapuser3
```

Chapter 20: Managing SELinux

Security Enhanced Linux (SELinux) is a security enhanced module on top of Linux. It provides additional security measures, is includes by default, and set as enforcing mode in RHEL. If SELinux is enabled and nothing else has been configured, all system calls are denied. To specify what exactly is allowed, a policy is used. In this policy, rules define which source domain is allowed to access which target domain. The source domain is the subject that is trying to access something. Typically, these are processes or users. The target domain is the object that is accessed. Typically, these are files, directories, or network ports. To define exactly what is allowed; context labels are used. These labels are the essence of SELinux because these labels are used to define access rules.

SELinux Core Elements:

Element	Use
Policy	A collection of rules that define which source has access to which target.
Source domain	The object that is trying to access a target. Typically a user or a process.
Target domain	The thing that a source domain is trying to access. Typically a file or port.
Context	A security label that is used to categorize objects in SELinux.
Rule	A specific part of the policy that determines which source domain has which access permissions to which target domain.
Labels	Same as context label, defined to determine which source domain has access to which target domain.

Operational Modes:

There are three modes of operating SELinux, i.e. Enforcing, Permissive, and disabled.

Enforcing mode: In Enforcing mode, SELinux is turned on and all the security policy rules are enforced.

Permissive mode: In permissive mode, SELinux is turned on but the security policy rules are not enforced. When a security policy rule should deny admission, access is still allowed. However, a message is sent to a log file denoting that access should have been denied. This mode can be useful for testing new applications, testing new SELinux policy rule and troubleshooting particular property.

Disabled mode: In Disabled mode, SELinux is turned off. Only DAC rules will be applicable for access. No logs will be generated. Disabled mode can be useful for circumstances in which enhanced security is not required.

Set SELinux operational modes:

```
[root@desktop0 ~]# getenforce                    → to check operational mode of
Enforcing                                          SELinux
[root@desktop0 ~]# setenforce 0                      --→ change operational mode
                                                     temporary
                                                     0/permissive = Permissive
                                                     1/enforcing = Enforcing
```

Note: You cannot disable the SELinux using command. You have to disable it from configuration file.

SELinux main configuration file is /etc/sysconfig/selinux. setenforce command will change operational modes temporary. Operational modes can be managed from its main configuration file which will applied permanently.

```
[root@server1 ~]# cat /etc/sysconfig/selinux
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
# enforcing - SELinux security policy is enforced.
# permissive - SELinux prints warnings instead of enforcing.
# disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
# targeted - Targeted processes are protected,
# minimum - Modification of targeted policy. Only selected
processes are protected.
# mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

sestatus command can be used to display selinux status,

```
[root@desktop0 ~]# sestatus                          --→ shows SELinux policies
```

Security Contexts:

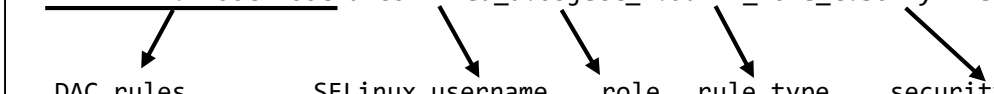
SELinux security context is a method to classify objects (such as files) and subjects (such as processes and users). SELinux context also called as labels allow SELinux policy rules for subject Accessing object. Context settings are an important part of SELinux operations. The context is a label that can be applied to different elements:

- Files and directories
- Ports
- Processes
- Users

Context labels define the nature of the item, and SELinux rules are created to match context labels of source objects to the context labels of target objects. So, setting correct context labels is a very important skill for system administrators.

Monitor context labels of a file/dir,

```
[root@desktop0 ~]# ls –l myfile                      --→ show DAC rules for
myfile
-rw-r--r--. 1 root root 0 Jul 24 15:26 myfile
[root@desktop0 ~]# ls –Z myfile                      --→ show SELinux context for
myfile
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 myfile
```



More examples,

```
[root@desktop0 ~]# ps –eZ | grep bash        → shows SELinux context for
                                             processes (ex. bash)
[root@desktop0 ~]# id                        → shows uid, gid and SELinux
                                             context for user
```

Every context label always consists of three different parts:
■ User: The user can be recognized by _u in the context label; it is set to system_u on most directories. SELinux users are not the same as Linux users.
■ Role: The role can be recognized by _r in the context label. Most objects are labeled with the object_r role. In advanced SELinux management, specific SELinux users can be assigned permissions to specific SELinux roles.
■ Type: The type context can be recognized by _t in the context label.

To view current SELinux security contexts using secon command,

```
[root@desktop0 ~]# secon                → shows current process's all security
                                          context
[root@desktop0 ~]# secon –urt           → shows user, role, type context of
                                          current process
[root@desktop0 ~]# secon –ut –p 1       → shows user and type context of process
                                          with PID 1
[root@desktop0 ~]# secon –sc –f myfile  → shows sensitive & clearance level
                                          context of myfile
```

Setting Context Types:

As an administrator, it is important that you know how to set context types. You can set these context types on files and directories (RHCSA requirement) and other objects such as network ports (RHCE requirement). Because setting context types on files is what you really need to know for RHCSA, let's focus on that task first.
You can use two commands to set context type:
■ **semanage:** This is the command you want to use. The semanage command writes the new context to the SELinux policy, from which it is applied to the file system.
■ **chcon:** This command is for use in specific cases only and normally should be avoided. The **chcon** command writes the new context to the file system and not to the policy. Everything that is applied with chcon is overwritten when the file system is relabeled, or the original context is restored from the policy to the file system. Do *not* use this command!

List labels using semanage,

```
[root@desktop0 ~]# semanage fcontext –l  → list security context for file
[root@desktop0 ~]# semanage port –l      → list security context for port
[root@desktop0 ~]# semanage user –l      → list security context for user
(more list's options can be found in --help option)
```

Add file context, example,

```
[root@desktop0 ~]# semanage fcontext –a –t samba_share_t "/demo(/.*)?"
```

To set context permanently, change the context using *semanage* command and perform *restorecon* command to set the context. *restorecon* command reset correct context and also update running selinux contexts.

```
[root@desktop0 ~]# restorecon –vFR /demo
                     Verbose  forcefully  Recursive   filename
```

Add port number to context, example:

```
[root@desktop0 ~]# semanage port –a –t ssh_port_t –p tcp 2020
```

Note: all security context resets when system is booted. This is done due to /.autorelable file.

Using Boolean Settings to Modify SELinux Settings

In the SELinux policy, there are many rules. Some of these rules allow specific activity, whereas other rules deny that activity. Changing rules is not easy, and that is why SELinux Booleans are provided to easily change the behavior of a rule.

An example of a Boolean is ftpd_anon_write, which by default is set to off. That means that even if you have configured your FTP server to allow anonymous writes, the Boolean will still deny it, and the anonymous user cannot upload any files. If a conflict exists between the setting of a parameter in a service configuration file and in a Boolean, the Boolean always takes precedence. But Booleans are easy to change.

To get a list of Booleans on your system, use **getsebool -a** . If you are looking for Booleans that are set for a specific service, use **grep** to filter down the results.

An alternative way to show current Boolean settings is by using the **semanage boolean -l** command. This command provides some more details, because it shows the current Boolean setting and the default Boolean setting.

Chapter 21: Configuring a Firewall

In computing, a firewall is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules. A firewall typically establishes a barrier between a trusted internal network and untrusted external network, such as the Internet.

Firewalls are often categorized as either hardware firewalls or software firewalls. Hardware firewalls (also called as network firewalls) filter traffic between two or more networks and run on network hardware. Software firewalls (also called as host-based firewall) run on host computers and control network traffic in and out of those machines. Network traffic are controlled by allowing ports.

In Linux, firewalld is frontend controller for firewall-cmd used to implement persistent network traffic rules. It provides command line and graphical interfaces and is available in the repositories of most Linux distributions. FirewallD uses *zones* and *services* to manage traffics dynamically, so that it updates rule without breaking any existing sessions and connections. As default, firewall set all outbound traffic ports allow where as it deny all inbound traffic ports.

Managing traffic using zones:

Zones are pre-constructed rulesets for various trust levels you would likely have for a given location or scenario (e.g. home, public, trusted, etc.). Different zones allow different network services and incoming traffic types while denying everything else. After enabling FirewallD for the first time, Public will be the default zone. Zones can also be applied to different network interfaces. For example, with separate interfaces for both an internal network and the Internet, you can allow DHCP on an internal zone but only HTTP and SSH on external zone. Any interface not explicitly set to a specific zone will be attached to the default zone.

Examples,

To view the default zone:

```
[root@desktop0 ~]# firewall-cmd --get-default-zone
```

To change the default zone:

```
[root@desktop0 ~]# firewall-cmd --set-default-zone=internal
```

To see the zones used by your network interface(s):

```
[root@desktop0 ~]# firewall-cmd --get-active-zones
```

To get all configurations for a specific zone:

```
[root@desktop0 ~]# firewall-cmd --zone=public --list-all
```

To get all configurations for all zones:

```
[root@desktop0 ~]# firewall-cmd --list-all-zones
```

Managing traffic using Services:

FirewallD can allow traffic based on predefined rules for specific network services. You can create your own custom service rules and add them to any zone. The configuration files for the default supported services are located at /usr/lib/firewalld/services and user-created service files would be in /etc/firewalld/services.

To view the default available services:

```
[root@desktop0 ~]# firewall-cmd --get-services
```

As an example, to enable or disable the HTTP service:

```
[root@desktop0 ~]# firewall-cmd --zone=public --add-service=http –permanent
[root@desktop0 ~]# firewall-cmd --zone=public --remove-service=http --permanent
```

Managing traffic using Ports:

Example,

Allowing or Denying an Arbitrary Port

As an example: Allow or disable TCP traffic on port 12345.

```
[root@desktop0 ~]# firewall-cmd --add-port=12345/tcp --permanent
[root@desktop0 ~]# firewall-cmd --remove-port=12345/tcp --permanent
```

Port Forwarding,

The example rule below forwards traffic from port 80 to port 12345 on the same server.

```
[root@desktop0 ~]# firewall-cmd --add-forward-port=port=80:proto=tcp:toport
=12345
```

To forward a port to a different server:

1. Activate masquerade in the desired zone.

```
[root@desktop0 ~]# firewall-cmd --zone=public --add-masquerade
```

2. Add the forward rule. This example forwards traffic from local port 80 to port 8080 on *a remote server* located at the IP address: 198.51.100.0

```
[root@desktop0 ~]# firewall-cmd --zone="public" --add-forward-port=port=80:
proto=tcp:toport=8080:toaddr=198.51.100.0
```

To remove the rules, substitute --add with --remove. For example:

```
[root@desktop0 ~]# firewall-cmd --zone=public --remove-masquerade
```

Chapter 22: Configuring Remote Mounts and FTP

File Transfer Protocol (FTP) is a standard internet protocol for transferring files between computers on the internet over TCP/IP connections. Here we are going to configure FTP using vsftpd (very secure FTP) service and upload/download files over the network.

Configuration Basics:
Package name = *vsftpd*
Default Port = *21 tcp*
Configuration file = */etc/vsftpd/vsftpd.conf*
Default root directory = */var/ftp/*
SELinux context = *public_content_rw_t*
Daemon/Service Name = *vsftpd*

Note: Assume that
IP of server = 172.25.0.11
IP of client = 172.25.0.10

Configuring an FTP Anonymous Drop Box

```
[root@server0 ~]# yum install vsftpd -y
[root@server0 ~]# vim /etc/vsftpd/vsftpd.conf

      (Line no. 12,16,19,29)

      # Allow anonymous FTP?
      anonymous_enable=YES
      . . .
      # Uncoment this to allow local users to login.
      local_enable=YES
      . . .
      # Uncoment this to enable any form of FTP write command
      write_enable=YES
      . . .
      # Uncoment this to allow anonymous user to write data
      anon_upload_enable=YES

[root@server0 ~]# getsebool –a | grep ftp
[root@server0 ~]# setsebool ftpd_anon_write on
[root@server0 ~]# setsebool ftpd_full_access on
[root@server0 ~]# systemctl restart vsftpd
[root@server0 ~]# systemctl enable vsftpd
[root@server0 ~]# firewall-cmd --add-service=ftp
```

The FTP server uses the directory */var/ftp* as the default document root. In this directory, create a subdirectory with the name uploads (i.e. */var/ftp/uploads*). Give permission *730* to */var/ftp/uploads* to set the correct permissions, and set the group owner to the group ftp. On an anonymous drop box, users can write files, but they cannot read them.

Accessing from client,

```
[root@desktop0 ~]# yum install ftp -y
[root@desktop0 ~]# ftp 172.25.0.11
Connected to 172.25.0.11 (172.25.0.11).
220 (vsFTPd 3.0.2)
Name (172.25.0.11:root): ftp
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Listing ftp files,

```
ftp> ls
227 Entering Passive Mode (172,31,49,176,244,22).
150 Here comes the directory listing.
-rw-r--r--    1 0        0               0 May 21 08:48 file
drwxr-xr-x    2 0        0               6 Apr 01 04:55 pub
226 Directory send OK.
```

Download files from ftp,

```
ftp> get file
local: file remote: file
227 Entering Passive Mode (172,31,49,176,87,41).
150 Opening BINARY mode data connection for file (0 bytes).
226 Transfer complete.
ftp> !ls
abc.txt  anaconda-ks.cfg  file  original-ks.cfg
```

("!" symbol is used to execute command on terminal)

Put files on ftp,

```
ftp> cd pub
250 Directory successfully changed.
ftp> put abc.txt
local: abc.txt remote: abc.txt
227 Entering Passive Mode (172,31,49,176,173,212).
150 Ok to send data.
226 Transfer complete.
```

(To upload file in ftp, you should have permission to upload files)

Chapter 23: Configuring Time Services

The network time protocol (NTP) synchronizes the time of a computer client or server to another server or within a few milliseconds of Coordinated Universal Time (UTC). Chrony is a flexible implementation of the Network Time Protocol (NTP). It is used to synchronize the system clock from different NTP servers, reference clocks or via manual input. In Linux OS, we use chrony service to implement NTP. Following is the example to synchronize time of one system with NTP server.

```
[root@server3 ~]# vim /etc/chrony.conf

       server 3.classroom.example.com

:wq

[root@server3 ~]# systemctl restart chronyd.service
[root@server3 ~]# systemctl enable chronyd.service
[root@server3 ~]# ntpdate -b classroom.example.com
16 Oct 00:06:54
ntpdate[2059]: step time server 172.25.254.254 offset -0.928404 sec
```