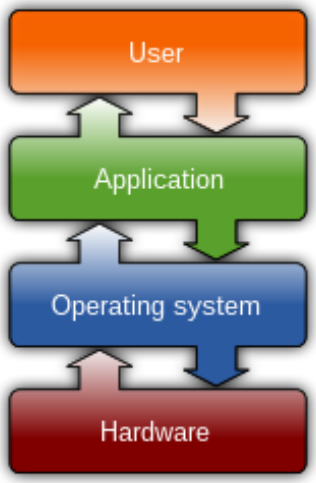
Chapter-1
Introduction to UNIX/LINUX

SOFTWARE

- A set of instructions that directs a computer's hardware to perform a task is called a program, or software program.
- The two main types of software are system software and application software.
- System software controls a computer's internal functioning, chiefly through an operating system, and also controls such peripherals as monitors, printers, and storage devices.
- Application software, by contrast, directs the computer to execute commands given by the user and may be said to include any program that processes data for a user.

Operating system

- An Operating System (OS) is an interface between a computer user and computer hardware.
- An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.



UNIX

UNiplexed Information Computing System.

- The development of Unix started around 1969 at AT&T Bell Labs by Ken Thompson and Dennis Ritchie.
- The Unix operating system is a set of programs that act as a link between the computer and the user.
- C and assembly language.
- Unix was not open-source software, and the Unix source code was licensable via agreements with its owner, AT&T.
- The first known software license was sold to the University of Illinois in 1975.

Unix flavors:

- Berkeley Software Distribution (BSD)
- IBM's AIX (Advanced Interactive eXecutive)
- Sun's Solaris
- Sequent
- Xenix

Features of UNIX

- **Multitasking**– More than one process can be operated at the same time. That is, if a process is functioning, then another process can also be run in the background.
- **Filters and Pipes**– The OS features pipes and filters that can enable the user to create complex programs from much simpler ones.
- **Shell**– A simple interface that allows the user to complete certain tasks.

Meanwhile, the Shell hides all the intricacies of the hardware details from the user.

- **Library is extensive:** The OS has the support of an extensive library that makes the OS very useful

Linux

- **Richard Stallman** was looking to create a truly free and open-source alternative to the proprietary Unix system. He was working on the utilities and programs under the name GNU, a recursive acronym meaning "GNU's not Unix! “.
- **Linus Torvalds** work producing a working and viable kernel that he called Linux that brought the complete operating system to life.
- Because of the free and open source standing of all the Linux components, anyone could create a Linux distribution with a bit of effort, and soon the total number of distros reached into the hundreds.

DISTRIBUTIONS OF LINUX:

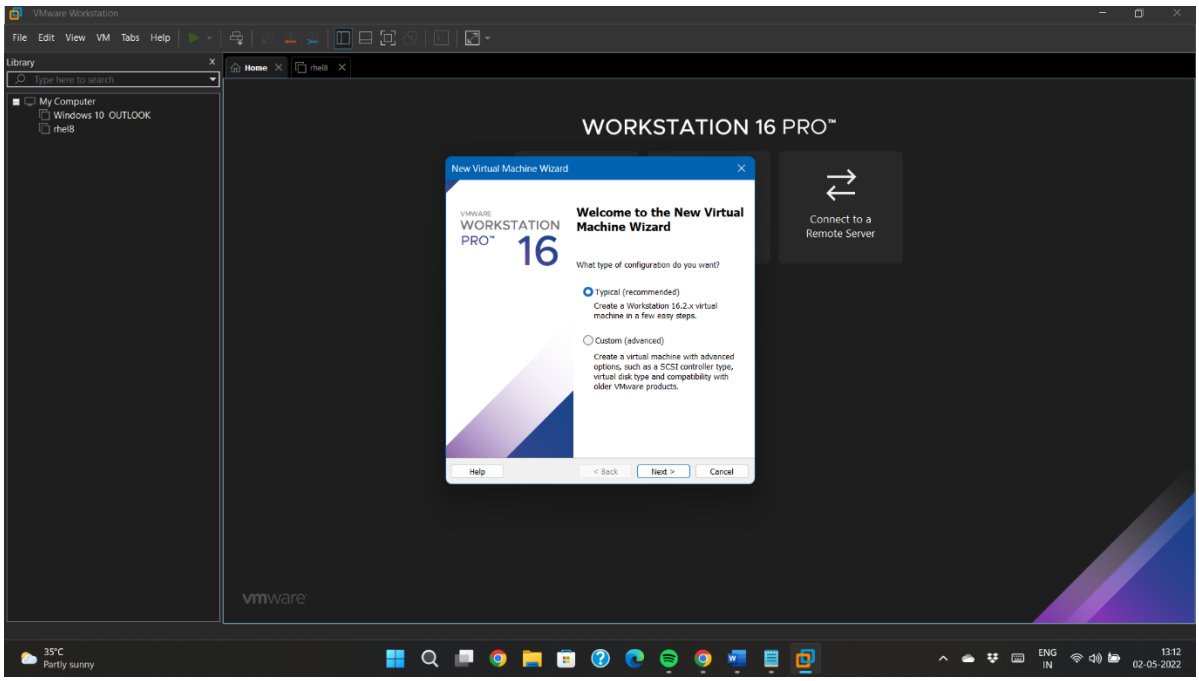
- Debian
- Ubuntu
- Gentoo
- Linux Mint
- Red Hat Enterprise Linux
- CentOS
- Fedora
- Kali Linux
- Arch Linux
- OpenSUSE

Features of Linux

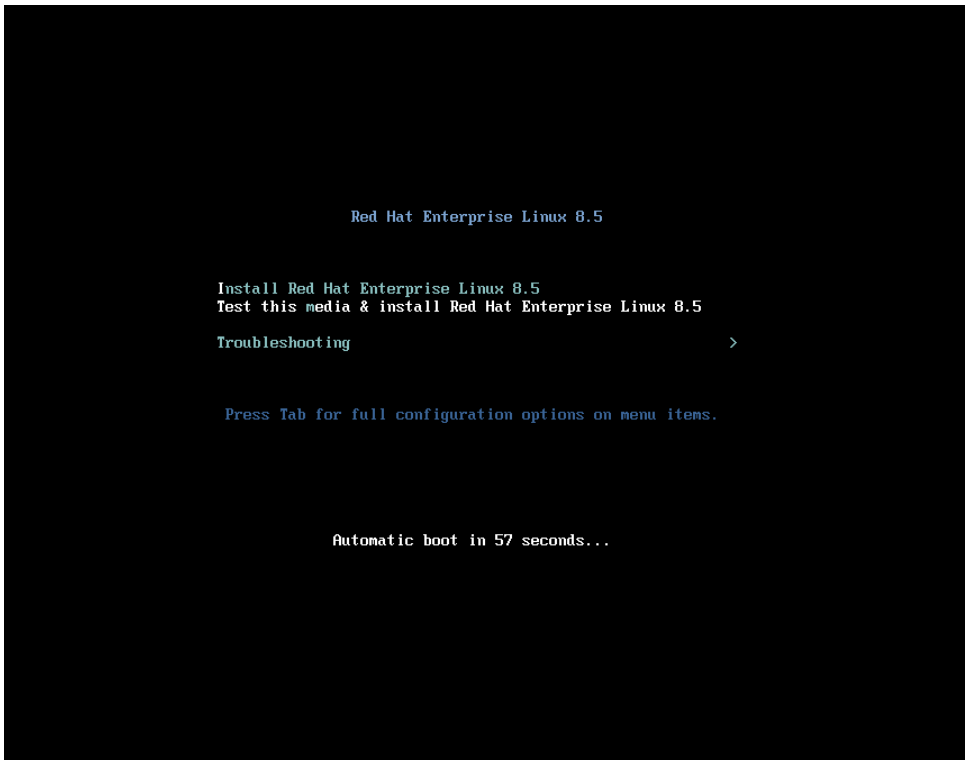
- **Multiuser capability:** Multiple users can access the same system resources like memory, hard disk, etc. But they have to use different terminals to operate.
- **Multitasking:** More than one function can be performed simultaneously by dividing the CPU time intelligently.
- **Portability:** Portability doesn't mean it is smaller in file size or can be carried in pen drives or memory cards. It means that it supports different types of hardware.
- **Security:** It provides security in three ways namely authenticating (by assigning password and login ID), authorization (by assigning permission to read, write and execute) and encryption (converts file into an unreadable format).
- **Graphical User Interface (X Window system):** Linux is command line-based OS but it can be converted to GUI based by installing packages.
- **Support's customized keyboard:** As it is used worldwide, hence supports different languages keyboards.
- **Application support:** It has its own software repository from where users can download and install many applications.
- **File System:** Provides hierarchical file system in which files and directories are arranged.
- **Open Source:** Linux code is freely available to all and is a community-based development project.

INSTALLING REDHAT LINUX

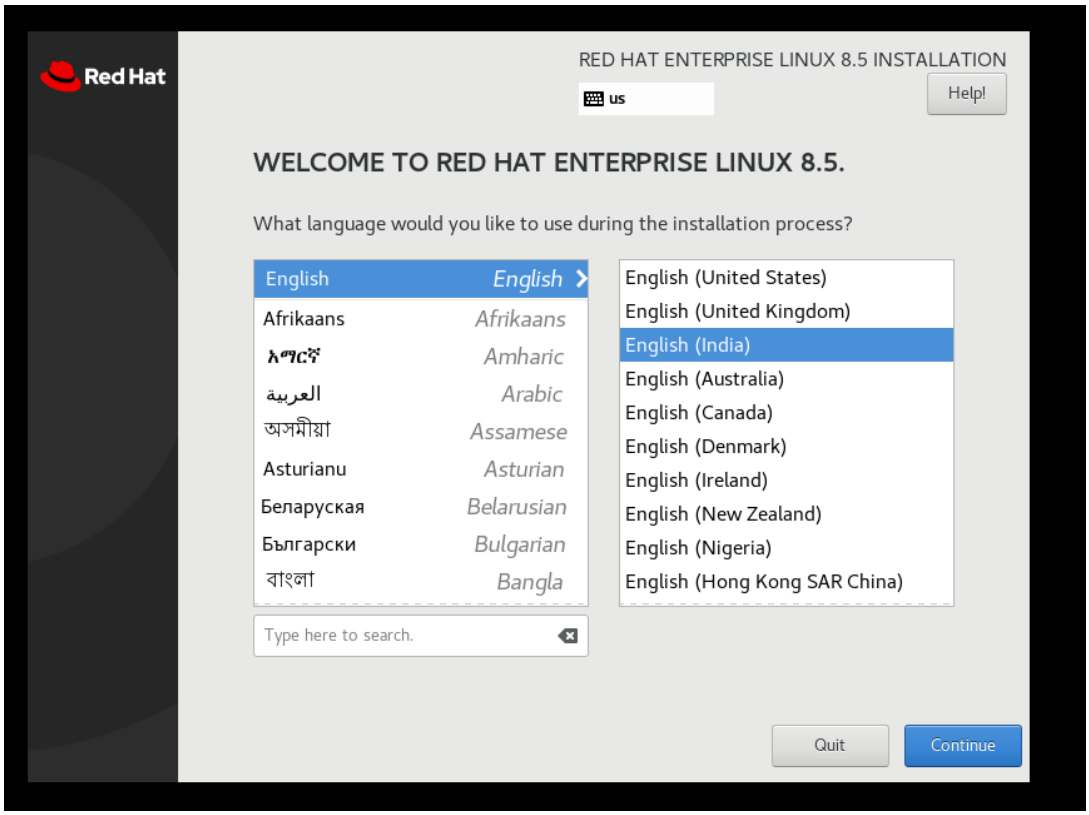
Create a Virtual Machine



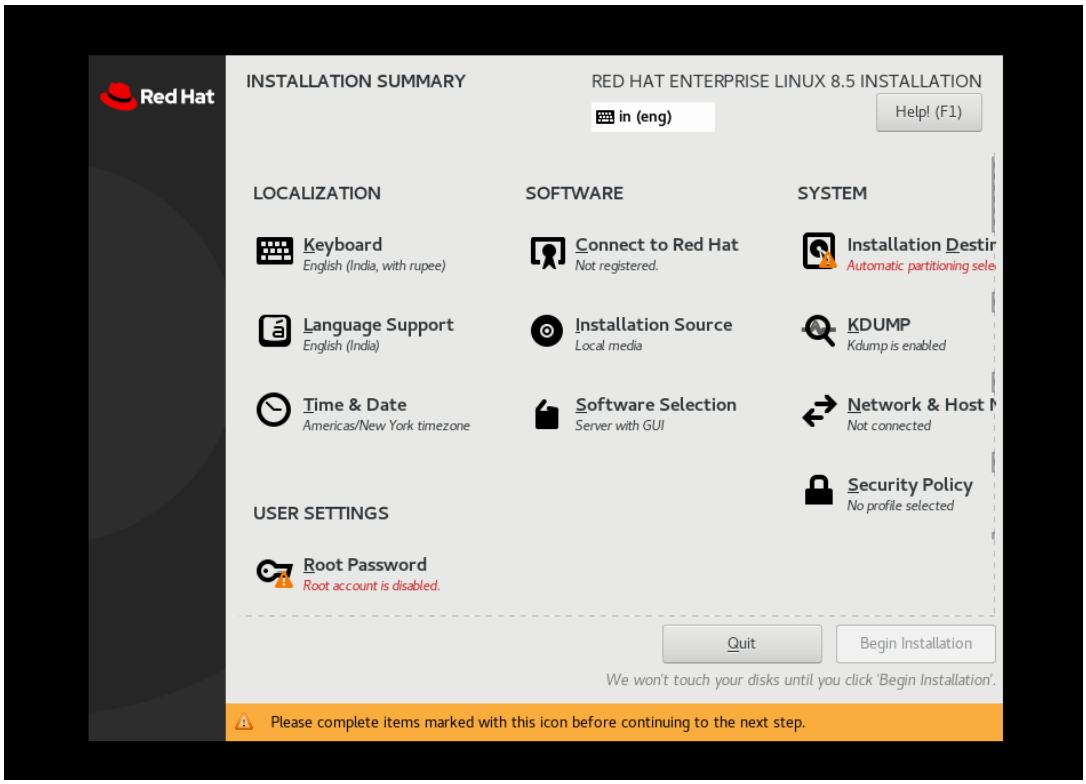
After Placing ISO files, Installation begins.



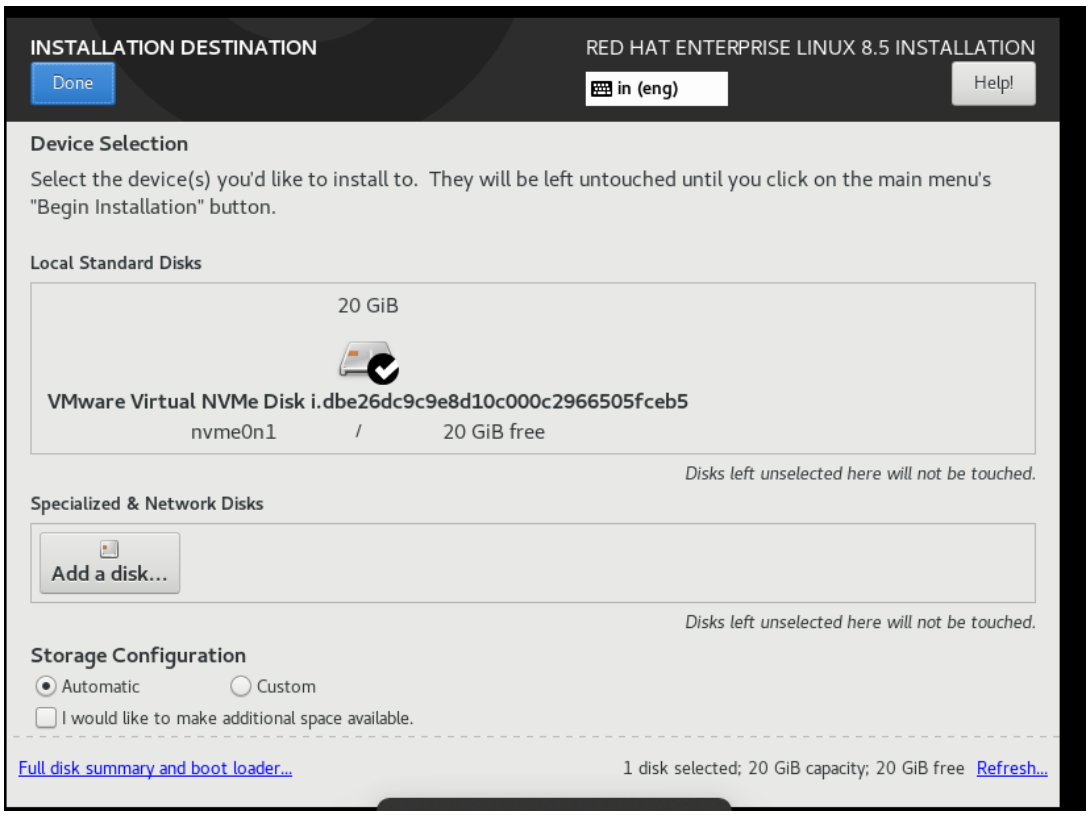
Select Install RedHat Enterprise Linux



Select the language and Country and click on continue

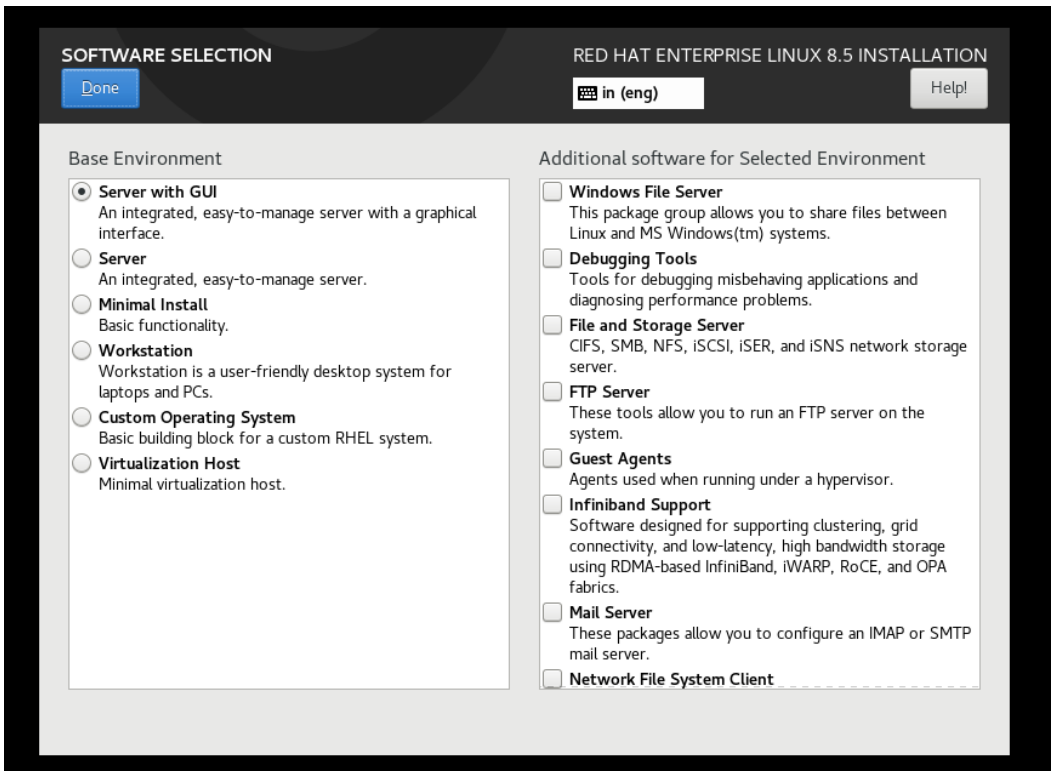


Select the Installation Destination



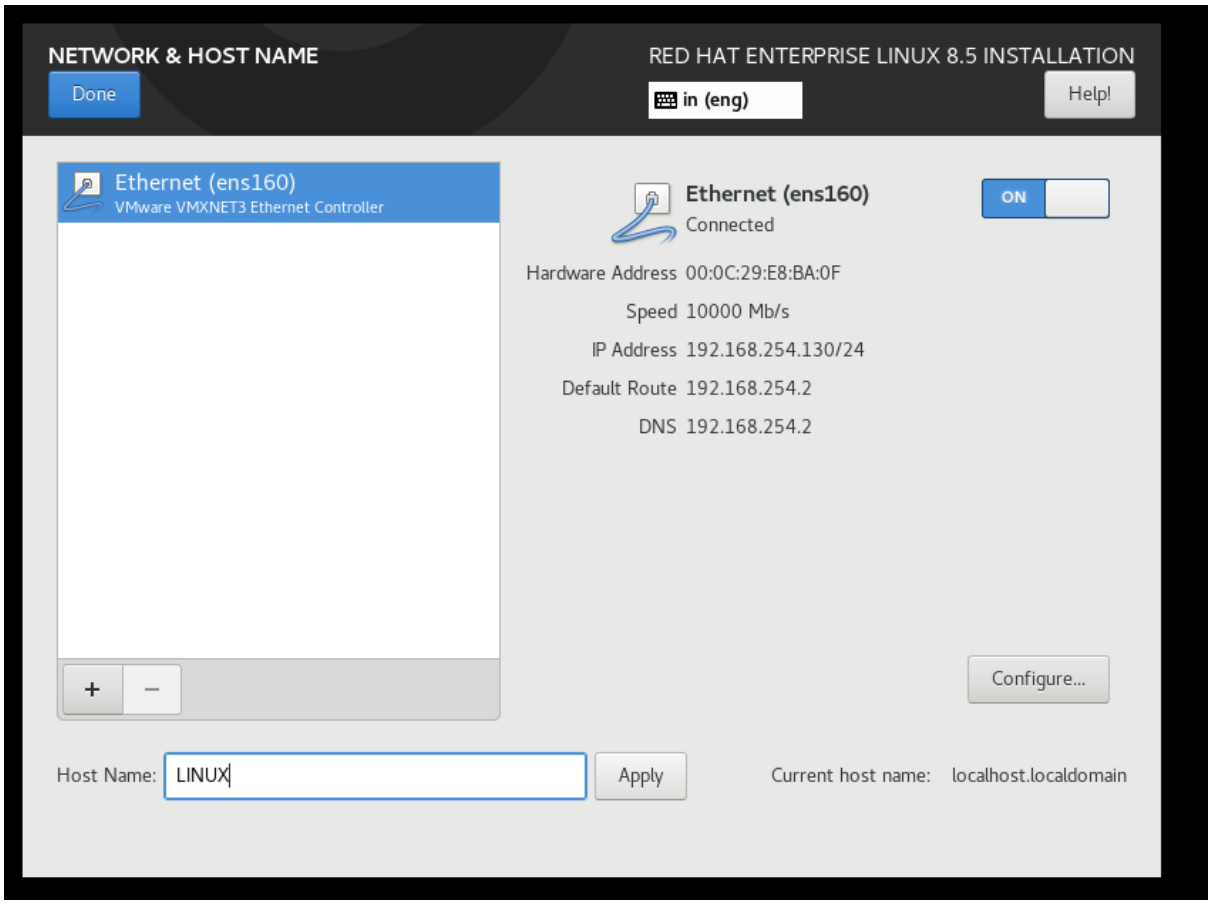
Click on Done twice.

Click on software selection and select the required environment.

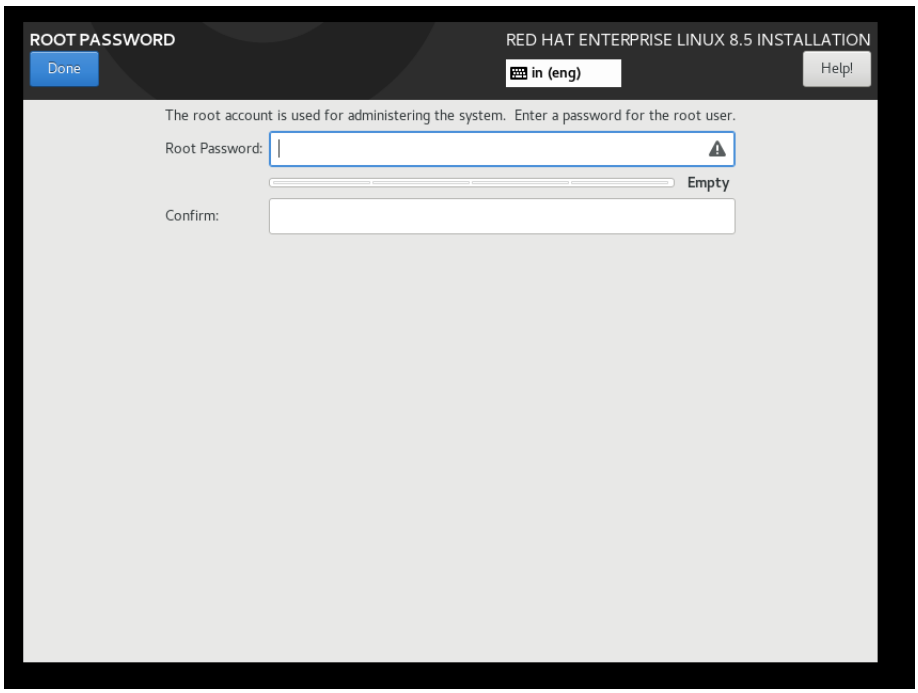


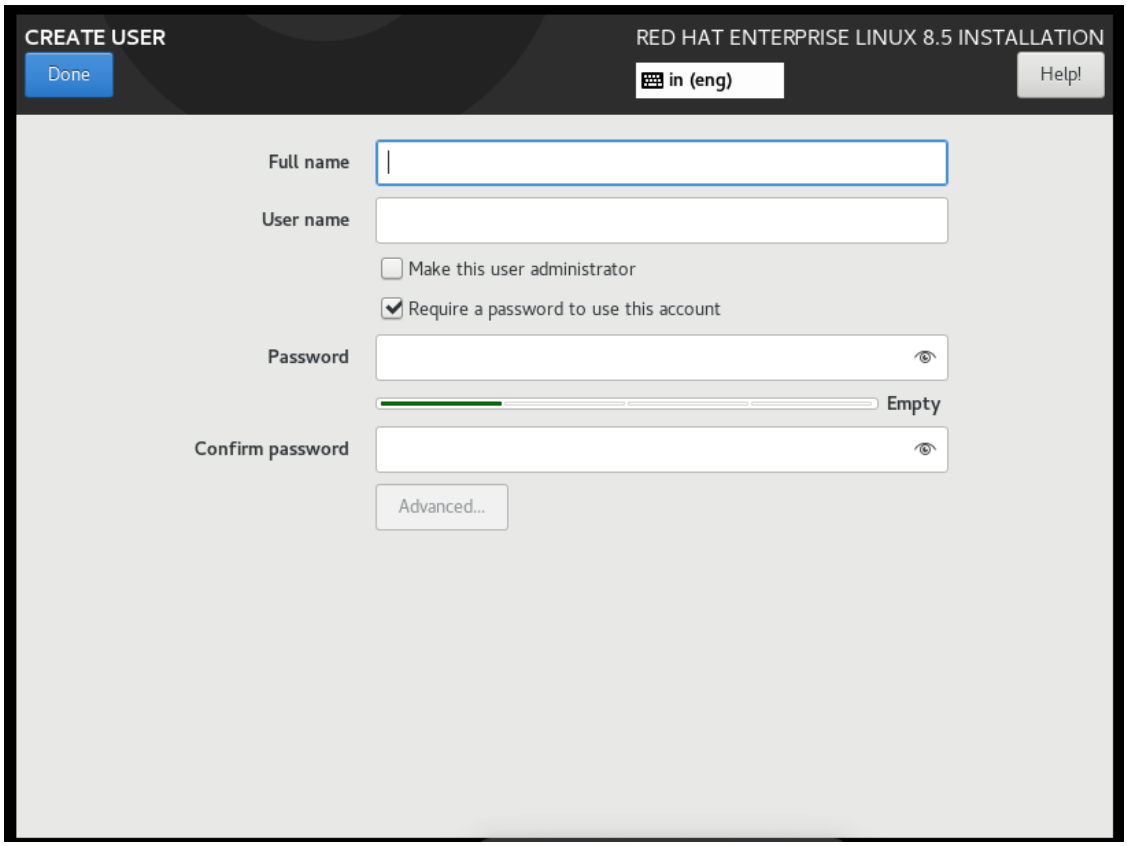
Select network & hostname

Turn on the Ethernet replace the hostname from localhost to any specific name.

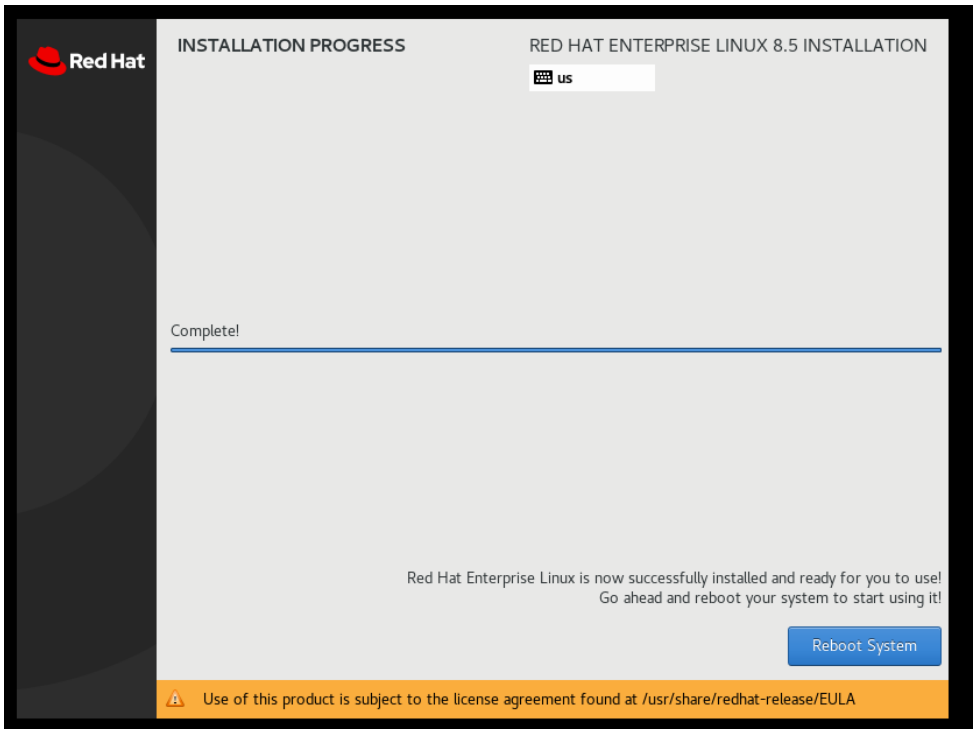


Set the root password and create user account



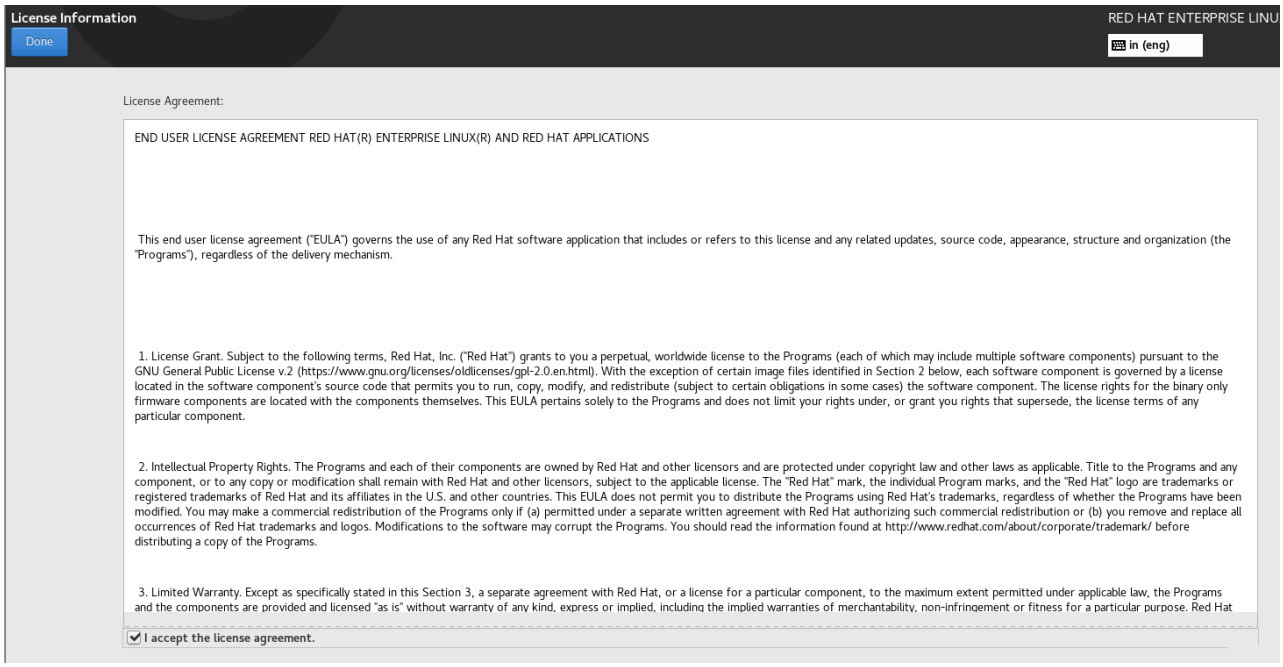


Click on begin installation.

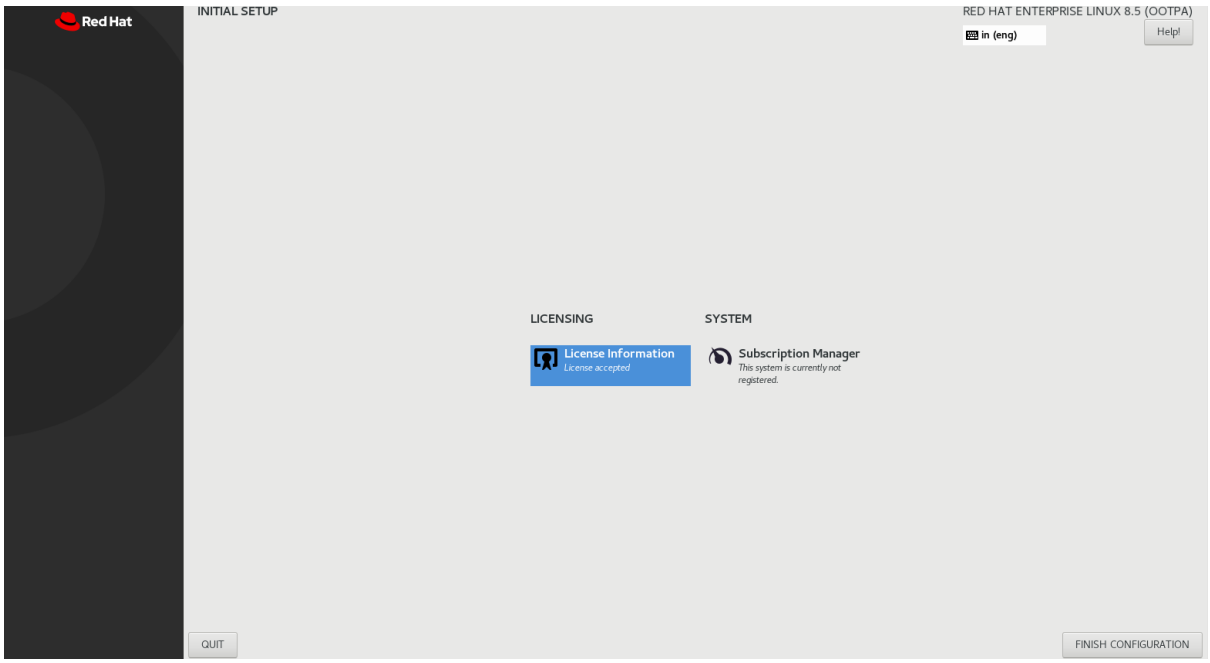


After installation is done. Click on Reboot System.

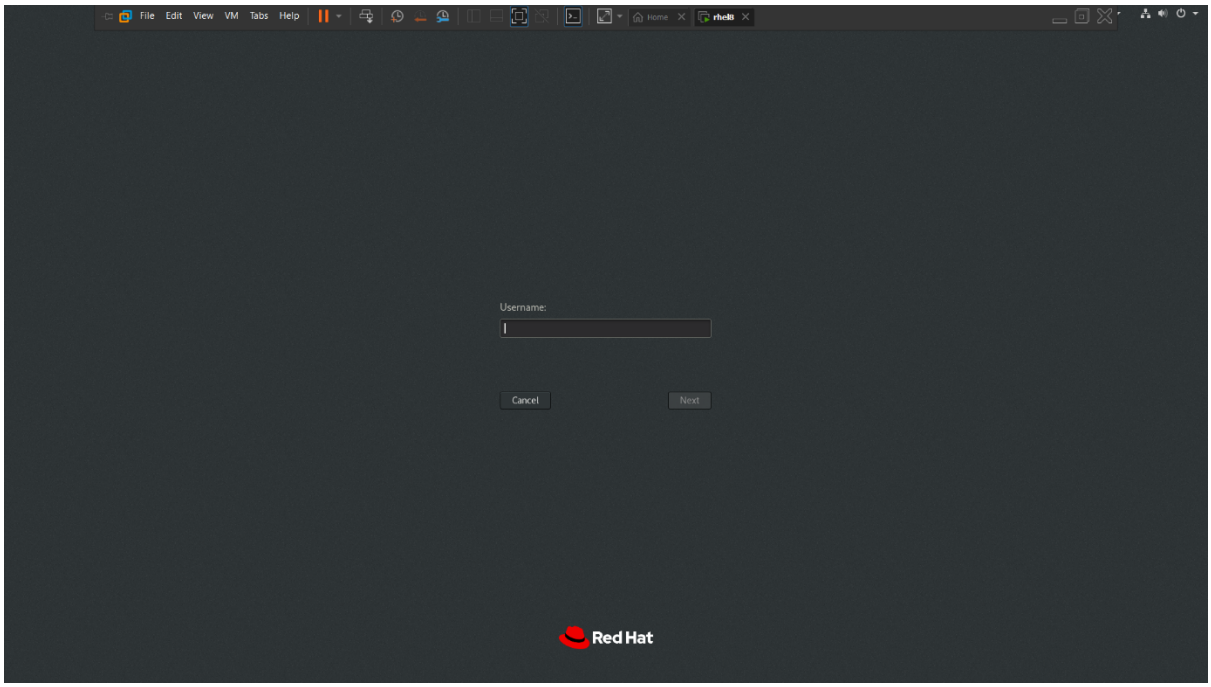
After restarting select the licensing agreement



Double click on the done and click on finish configuration.



Installing of RedHat is done.



Components Of Linux Operating System

An operating system is a collection of software, each designed for a specific function.

Shell:

The terminal contains the shell; it allows us to execute the commands to interact with the system.

We can perform various operations such as store and retrieve data, process information, and various other simple as well as complex tasks.

Linux Bash:

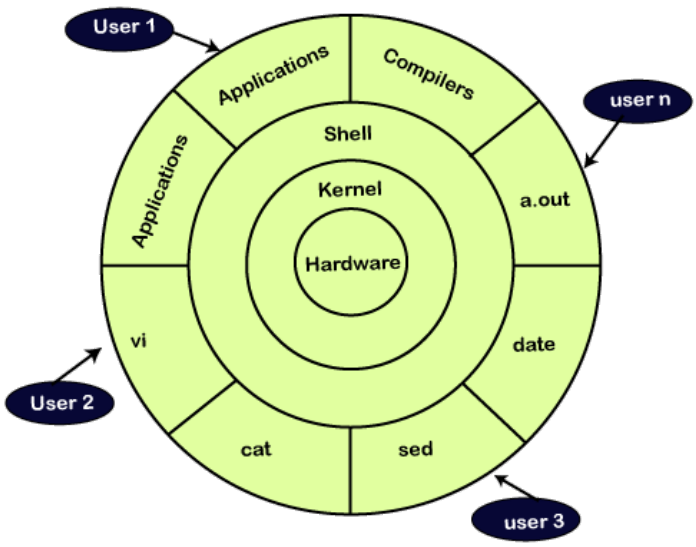
- The Linux Bash is also known as **'Bourne-again Shell**.'
- It is a command language interpreter for the Linux based system. It is a replacement of Bourne shell (sh).
- The Linux/Unix shell allows us to interact with the Linux system through the commands. It let us invoke an executable file to create

a running process

- The Bash is a command language interpreter as well as a programming language. It supports variables, functions, and flow control, like other programming languages.
- It can also read and execute the commands from a file, which is called a **shell script.**

Kernel:

Linux kernel is the core part of the operating system. It establishes communication between devices and software. Moreover, it manages system resources. It has four responsibilities:



- **Device management:** A system has many devices connected to it like CPU, a memory device, sound cards, graphic cards, etc. A kernel stores all the data related to all the devices in the device driver (without this kernel won't be able to control the devices). Thus, kernel knows what a device can do and how to manipulate it to bring out the best performance. It also manages communication between all the devices. The kernel has certain rules that have to be followed by all the devices.
- **Memory management**: Another function that kernel has to manage is the memory management. The kernel keeps track of used and unused memory and makes sure that processes shouldn't manipulate data of each other using virtual memory addresses.

- **Process management:** In the process management kernel assigns enough time and gives priorities to processes before handling CPU to other processes. It also deals with security and ownership information.
- **Handling system calls:** Handling system calls means a programmer can write a query or ask the kernel to perform a task.

Command Syntax:

Linux commands typically have

Command options and arguments

Options:

Modify the way that command works. Usually consists of hyphen or dash followed by single letter. Some commands accept multiple letters options which and be usually grouped together after a single hyphen.

Arguments:

Most commands are used together with one or more arguments. Some commands assume default argument if none is applied. Arguments are optional from some commands and required by others.

Basic Commands of Linux:

pwd :

The ***pwd*** command stands for (print working directory). It displays the current working location or directory of the user.

ls:

The ***ls*** command is used to show the list of a folder. It will list out all the files in the directed folder.

ls -a: In Linux, hidden files start with . (dot) symbol and they are not visible in the regular directory. The (ls -a) command will enlist the whole list of the current directory including the hidden files.

ls -l: It will show the list in a long list format.

ls -lh: This command will show you the file sizes in human readable format.

ls -l --block-size=[SIZE]:
It is used to display the files in a specific size format. Here, in [SIZE] you can assign size according to your requirement.
Example: ls -l --block-size=M

ls -g or ls -lG:
With this you can exclude column of group information and owner.

ls ../ : It give the contents of parent directory.

cd:

The ***cd*** command stands for (change directory). It is used to change to the directory you want to work from the present directory.

mkdir:

With ***mkdir*** command you can create your own directory.

mkdir -p : With the help of mkdir -p command you can create sub-directories of a directory. It will create parent directory first, if it doesn't exist.

mkdir -pv : The command 'mkdir -pv' By combining -p you can create a long list files together like **"office/client/raj/date/day** "with a printed message for each file.

rmdir:

The ***rmdir*** command is used to remove a directory from your system.

cal:

The '***cal*** term stands for calender. It displays current month's calender with current day highlighted.

date:

Linux ***date*** command is used to display date, time, time zone, etc. It is also used to set the date and time of the Linux system.

In RHEL 8 desktop icons are not enabled.

To enable them follow the below procedure and commands.

dnf install gnome-tweaks

yum install gnome-tweaks.noarch -y

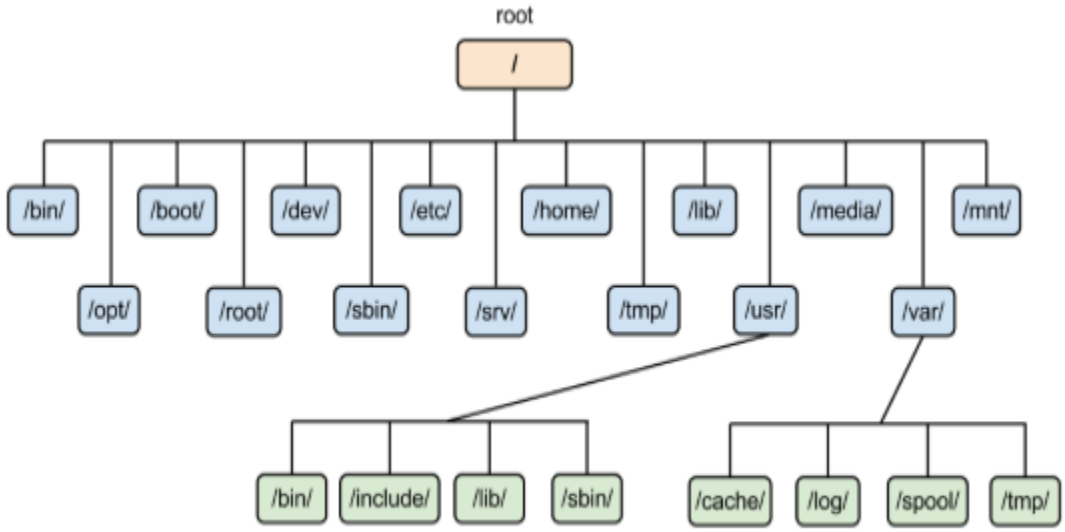
gnome-tweaks

enable the desktop icon option

Chapter-2
Linux Filesystem Hierarchy Standard (FHS)

Filesystem hierarchy standard describes directory structure and its content in Unix like operating system. It explains where files and directories should be located and what it should contain.

If you want to find out information about your system's FHS, enter the command **man hier**. It will display directory structure of your system.



File system hierarchy of Linux

The Root Directory

All the directories in the Linux system comes under the root directory which is represented by a **forward** slash **(/)**. Everything in your system can be found under this root directory even if they are stored in different virtual or physical devices.

/bin

- The '/bin' directory contains user binaries, executable files, Linux

commands that are used in single user mode, and common commands that are used by all the users, like cat, cp, cd, ls, etc.

- Binary files are the files which contain compiled source code (or machine code). They are also called executable files because they can be executed on the computer. The '/bin' directory doesn't contain directories.

/boot

The '/boot' directory contains boot loader files which are essential to boot the system.

/etc

All the machine related configuration files are kept in '/etc'. Almost everything related to the configuration of your system is placed here. It also contains start up and shutdown shell script which is used to start and stop a program.

/dev

The term 'dev' is short for **device**. As you know in Linux operating system everything is a file. It appears to be an ordinary file but doesn't take up disk space.

/sbin

The '/sbin' directory also contains executable files, but unlike '/bin' it only contains system binaries which require root privilege to perform certain tasks and are helpful for system maintenance purpose.

/lib

The '/lib' directory contains **shared libraries** which are often used by the '/bin' and '/sbin' directories. It also contains kernel module. These filenames are identable as ld* or lib*.so.*.

/opt

The term **'opt'** is short for optional. Its main purpose is to store optional application software packages. Add-on applications from individual vendors should be installed in '/opt.

/home

The '/home' directory stores users personnel files. Inside this directory we have our sub-directories like Desktop, Downloads, Documents, pictures, etc.

/root

The '/root' directory is the home directory of the root user.

- Please note that '/root' directory is different from (/) root.

/srv

The term 'srv' is short for **service**. The '/srv' directory contains server specific data for services provided by the system like www, cvs, rysync, ftp, etc.

/media

The '/media' directory acts as a mount point for removable media devices such as CD-Rom, floppy, USB devices, etc.

/mnt

The term 'mnt' stands for **mount**. The '/mnt' directory should be empty and sysadmins can only mount temporary filesystems.

/proc

The term 'proc' is short for process. Same as '/dev', '/proc' also doesn't take up disk space. It contains process information. It is a pseudo filesystem that contains information about running processes.

/usr

Although it is pronounced as user but in actual it stands for **Unix System Resources**. It is also called secondary hierarchy as it contains binaries, libraries, documentation for all the user applications. It only contains shareable read-only data.

/var

The term 'var' is short for **varible**. Files that have an unexpected size and whose content is expected to change continuously (that's why it is named as variable) during normal operation of the system are stored here.

/tmp

The term 'tmp' stands for **temporary**. Data stored in '/tmp' is temporary and may use either disk space or RAM. When system is rebooted, files under this directory is automatically deleted. So it is advisable that never use '/tmp' to store important data.

/sys

The term 'sys' is short for **system**. Basically, it contains kernel information about hardware. It was created for Linux 2.6 kernel. It is a kind of '/proc' and is used for plug and play configuration.

Linux Absolute and Relative Paths

It is very important to know the difference between absolute and relative path. Because correct path will only lead you to your destination directory.

When you define a path starting with a slash ('/') sign, then root of the file is assumed. If you don't put a '/' then the current directory is assumed to be the starting point.

In below example, we're at **/home/sssit**. To go to home directory, we typed **cd /home** instead of cd home.

Below you can see that we're getting an error by typing **cd /sssit** instead of **cd sssit**, because sssit is a sub-directory of home and hence it can't be accessed with a root ('/').

If currently you are in the root directory, then you can give command **home or /home**. Both will lead you to the home directory because you are already on the root directory.

Linux Files:

- In Linux system, everything is a file and if it is not a file, it is a process.

- A file doesn't include only text files, images and compiled programs but also include partitions, hardware device drivers and directories.

Types of Files:

1. **Regular files (-):** It contain programs, executable files and text files.
2. **Directory files (d):** It is shown in blue colour. It contains list of files.
3. **Special files**
    - **Block file (b)**
    - **Character device file (c)**
    - **Symbolic link file (l)**
    - **Socket file (s)**

Block special files(b):

A block special file acts as a direct interface to a block device. A block device is any device which performs data I/O in units of blocks.To know how big a block is on your system, run "blockdev --getbsz device" as root.

sudo blockdev --getbsz /dev/sda1

Character special files(c):

A character special file is similar to a block device, but data is written one character (eight bits, or one byte) at a time.

Symbolic link(l):

A symbolic link may refer as a soft link or symlink, a symbolic link is a file that links to another file or directory using its path. A symbolic link can link to any file or directory on any computer.

Soft Link:

A soft link (also known as Symbolic link) acts as a pointer or a reference to the file name. It does not access the data available in the original file. If the earlier file is deleted, the soft link will be pointing to a file that does not exist anymore.

Soft Link	Hard Link
The inode number of the actual file and the soft link file are different.	The inode number of the actual file and the hard link file are the same.
A soft link can be created across different filesystems.	A hard link can only be created in the same filesystem.
A soft link can link to both regular files and directories.	A hard link doesn't link to directories.
Soft links are not updated if the actual file is deleted. It keeps pointing to a nonexistent file.	Hard links are always updated if the actual file is moved or deleted.

create symbolic links

With Linux and Unix we have ln utility which is used to create symbolic links. The basic syntax to create links would be:

ln [OPTION] TARGET LINK_NAME

TARGET refers to the file or directory for which you wish to create the link

LINK_NAME is the name of the link

Soft link:

using different link name

In this example, I have a file /tmp/orig_file.txt. I wish to create a symlink for this file under /root with a different name orig_link. So our command would look like:

ln -s /tmp/orig_file.txt /root/orig_link

using same link name as source file

In the earlier example we had explicitly given a name for our soft link. If we just provide the path instead of name then a soft link with the same

name as source file will be created.

Let's check this in our example:

 # ln -s /tmp/orig_file.txt /root/

Hard Link:

A hard link acts as a copy (mirrored) of the selected file. It accesses the data available in the original file.If the earlier selected file is deleted, the hard link to the file will still contain the data of that file.

ln TARGET LINK_NAME

using different link name

In this example I have created a dummy file and place some content inside the file

echo "Hello, This is a test file" >> /tmp/source_file

Next I will create a hard link of this file under /root/hard_link_file

 # ln /tmp/source_file /root/hard_link_file

There is no output so this means the execution was successful. Next verify the hard link fileAs you can see, unlike softlink the hard link file is not pointing to it's source file and is acting as a normal file.

I will add some data to the source file

 # echo "new content" >> /tmp/source_file

Socket(s):

A socket is a special file used for inter-process communication, which enables communication between two processes. In addition to sending data, processes can send file descriptors across a Unix domain socket connection using the sendmsg() and recvmsg() system calls.

file command:

file command is used to determine the file type. It does not care about the extension used for file. It simply uses file command and tell us the file type. It has several options.

Syntax:

file <filename>

file -s

Linux file -s command is used for the special files. Let's see a simple example of -s option of file command

Syntax:

file -s <filename>

file *

This command is used to obtain the type of all files of the current directory.

file [range]*

With the help of this command, you can specify a range of the alphabets for the files you want. It will list out only those files which starts from the alphabets present in the range.

Syntax:

file [a-y]*

touch command

touch command is a way to create empty files (there are some other methods also). You can update the modification and access time of each file with the help of touch command.

Syntax:

touch <filename>

Linux rm

The 'rm' means remove. This command is used to remove a file. The command line doesn't have a recycle bin or trash unlike other GUI's to recover the files. Hence, be very much careful while using this command. Once you have deleted a file, it is removed permanently.

Linux File Contents Command

There are many commands which help to look at the contents of a file. Now we'll look at some of the commands like head, tac, cat, less & more and strings.

I/O Redirection

Redirection can be defined as changing the way from where commands read input to where commands sends output. You can redirect input and output of a command.

For redirection, meta characters are used. Redirection can be into a **file** (shell meta characters are angle brackets '<', '>') or a **program** (shell meta characters are **pipe**symbol '|').

Standard Streams In I/O Redirection

The bash shell has three standard streams in I/O redirection:

- **standard input (stdin):** The stdin stream is numbered as stdin (0). The bash shell takes input from stdin. By default, keyboard is used as input.
- **standard output (stdout):** The stdout stream is numbered as stdout (1). The bash shell sends output to stdout. Output goes to display.
- **standard error (stderr):** The stderr stream is numbered as stderr (2). The bash shell sends error message to stderr. Error message goes to display.

Redirection Into A File

Each stream uses redirection commands. Single bracket '>' or double bracket '>>' can be used to redirect standard output. If the target file doesn't exist, a new file with the same name will be created.

Overwrite

Commands with a single bracket **'>' overwrite** existing file content.

- > : standard output
- < : standard input
- 2> : standard error

Note: Writing **'1>'** or **'>'** and **'0<'** or **'<'** is same thing. But for stderr you have to write **'2>'**.

cat > <fileName>

Append

Commands with a double bracket **'>>' do not overwrite** the existing file content.

- >> - standard output
- << - standard input
- 2>> - standard error

cat >> <fileName>

Input Redirection

< stdin

The bash shell uses stdin to take input. In input redirection, a file is made input to the command and this redirection is done with the help of '<' sign.

Output Redirection

Output redirection is used to put output of one command into a file or into another command.

> stdout

The stdout is redirected with a '>' greater than sign. When shell meets the '>' sign, it will clear the file (as you already know).

Error Redirection

2> stderr

Command '2>' redirects the error of an output. It helps us you to keep our display less messy by redirecting error messages.

Cat Command:

The 'cat' command is the most universal and powerful tool. It is considered to be one of the most frequently used commands. It can be used to display the content of a file, copy content from one file to another, concatenate the contents of multiple files, display the line number, display $ at the end of the line, etc.

cat [filename]

The 'cat' command can be used to display the content of a file.

cat > [fileName]

To create a file.

cat [oldfile] > [newfile]

To copy content from older to new file.

cat [file1 file2 and so on] > [new file name]

To concatenate contents of multiple files into one.

cat -n/cat -b [fileName]

To display line numbers

cat -e [fileName]

To display $ character at the end of each line.

head:

The 'head' command displays the starting content of a file. By default, it

displays starting 10 lines of any file.

head <file name>

displays starting 10 lines of any file

head <file name> <file name>
two file names then it will display first ten lines (in this case file has five lines only) of each file separated by a heading.

head -n <file name>
The 'head -n' option displays specified number of lines

head -c <number> <file name>
The 'head -c' command counts the number of bytes of a file.

tail command

Linux tail command is used to display the last ten lines of one or more files. Its main purpose is to read the error message. By default, it displays the last ten lines of a file. Additionally, it is used to monitor the file changes in real-time. It is a complementary command of the head command.

tail <file name>

display the last ten lines

tail -n <number> <file name>
It will display the specified number of lines from the last.

tail -c <number> <file name>
It will display the specified number of bytes.

more command:

As 'cat' command displays the file content. Same way 'more' command also displays the content of a file. Only difference is that, in case of larger files, 'cat' command output will scroll off your screen while 'more' command displays output one screenful at a time.
more <file name>

Following keys are used in 'more' command to scroll the page:

- Enter key: To scroll down page line by line.
- Space bar: To go to next page.
- b key: To go to the backward page.
- / key: Lets you search the string.

less command

The 'less' command is same as 'more' command but include some more features.

It automatically adjust with the width and height of the teminal window, while 'more' command cuts the content as the width of the terminal window get shorter.

less <file name>

cut Command

Linux cut command is useful for selecting a specific column of a file. It is used to cut a specific section by byte position, character, and field and writes them to standard output. It cuts a line and extracts the text data.

cut -d- -f(columnNumber) <fileName>

To cut by using the hyphen (-) as the delimiter.

cut -d ' ' -f(columnNumber) <fileName>

To use space as a delimiter, then we have to quote the space (' ') with the cut command.

cut -b <byte number> <file name>

The '-b' option is used to cut a section of line by byte.

cut -c < characters> <file name>

The '-c' option is used to cut a specific section by character.

sed Command

Linux 'sed' command stands for stream editor. It is used to edit streams

(files) using regular expressions. But this editing is not permanent. It remains only in display, but in actual, file content remains the same.

Removing a Line

The 'd' option will let us remove a complete line from a file. We only need to specify a word from that line with 'd' option, and that line will be deleted.

cat <fileName> | sed '/<Word>/d'

Multiple sed Command

The '-e' option allows us to execute the multiple sed commands at once.

sed -e '<script 1> ; <script 2>' <file name>

sed -n 'np' <filename>

to print the line of a file.' np' n represents the line number.

Printing the Line Numbers

The '=' sign is used to print the line number.

sed '=' <filename>

grep command

Case insensitive search: The -i option enables to search for a string case insensitively in the given file

grep -i "<word>" <filename>

Displaying the count of number of matches: We can find the number of lines that matches the given string/pattern

grep -c "<word>" <filename>

Display the file names that matches the pattern: We can just display the files that contains the given string/pattern.

*grep -l "<word>" **

Checking for the whole words in a file: By default, grep matches the

given string/pattern even if it is found as a substring in a file

grep -w "<word>" <filename>

Displaying only the matched pattern: By default, grep displays the entire line which has the matched string. We can make the grep to display only the matched string by using the -o option.

grep -o "<word>" <filename>

Show line number while displaying the output using grep -n : To show the line number of file with the line matched.

grep -n "<word>" <filename>

Inverting the pattern match: You can display the lines that are not matched with the specified search string pattern using the -v option.

grep -v "<word>" <filename>

comm

The 'comm' command compares two files or streams. By default, 'comm' will always display three columns. First column indicates non-matching items of first file, second column indicates non-matching items of second file, and third column indicates matching items of both the files. Both the files has to be in sorted order for 'comm' command to be executed.

comm <file1> <file2>

uniq Command

Linux uniq command is used to remove all the repeated lines from a file. Also, it can be used to display a count of any word, only repeated lines, ignore characters, and compare specific fields.

Remove repeated lines

To remove repeated lines from a file,

sort <filename> | uniq

Count the number of occurrences of a word

We can count the number of occurrences of a word by using the uniq

command. The '-c' option is used to count the word.

sort <filename> | uniq -c

Display the repeated lines

The '-d' option is used to display only the repeated lines. It will only display the lines that will be more than once in a file and write the output to standard output.

sort <filename> | uniq -d

Display the unique lines

The '-u' option is used to display only the unique lines (which are not repeated). It will only display the lines that occur only once and write the result to standard output.

sort <filename> | uniq -u

find command:

The find command in UNIX is a command line utility for walking a file hierarchy. It can be used to find files and directories and perform subsequent operations on them. It supports searching by file, folder, name, creation date, modification date, owner and permissions.

Search a file with specific name.

$ find ./<dir> -name <file name>

Search a file with pattern.

*$ find ./<dir> -name *.txt*

Search for empty files and directories.

$ find ./<dir> -empty

Search for file with entered permissions.

$ find ./GFG -perm 664

Search text within multiple files.

$ find ./ -type f -name ".txt" -exec grep 'Geek' {} \;*

User Management in Linux

A user is an entity, in a Linux operating system, that can manipulate files and perform several other operations. Each user is assigned an ID that is unique for each user in the operating system. The ID 0 is assigned to the root user and the IDs 1 to 999 (both inclusive) are assigned to the system users and hence the ids for local user begins from 1000 onwards. we can create 60,000 users.

To list out all the users in Linux, use the awk command with -F option. Here, we are accessing a file and printing only first column with the help of print $1 and awk.

awk -F':' '{ print $1}' /etc/passwd

Using id command, you can get the ID of any username. Every user has an id assigned to it and the user is identified with the help of this id. By default, this id is also the group id of the user.

id username

The command to add a user. useradd command adds a new user to the directory. The user is given the ID automatically depending on which

category it falls in. The username of the user will be as provided by us in the command.

useradd

sudo useradd username

To give a home directory path for new user

sudo useradd -d /home/test_user test_user

To create a user with specific user id

sudo useradd -u 1234 test_user

To create a user with specific group id

sudo useradd -g 1000 test_user

To create a user without home directory

sudo useradd -M test_user

To set an unencrypted password for the user

sudo useradd -p test_password test_user

usermod

To change the home directory of a user

sudo usermod -d /home/user

To change user login name

sudo usermod -l account user

To lock a user

sudo usermod -L test_user

To unlock a user

sudo usermod -U test_user

passwd

Using passwd command to assign a password to a user. After using this command we have to enter the new password for the user and then the password gets updated to the new password.

passwd username

Processing in passwd command:

Verify current user password: Once the user enters passwd command, it prompts for current user password, which is verified against the password stored in /etc/shadow file user. The root user can bypass this step and can directly change the password, so as the forgotten passwords may be recovered.

Verify password aging information: In Linux, a user password can be set to expire after a given period of time. Also, a user can be prohibited to change his/her password for a period. This password aging information (and the password itself) is stored in a file /etc/shadow.

Change the password: After authentication, the user is prompted to enter the new password and verify it by retyping the password.

/etc/shadow file: The shadow file is a list of colon separated values with 9 fields, as shown below:

user1:6x8wAJRpP$EWC97sXW5tqac10Q2TQyXkR.1l1jdK4VLK1pkZKmA2mbA6UnSGyo94Pis074viWBA3sVbkCptSZzuP2K.y.an/:17887:0:99999:7:::

field 1: User name.

field 2: Encrypted Password.

field 3: Number of days since January 1, 1970 to when the password was last changed.

field 4: Minimum number of days for which password cannot be changed. (Value 0 means it can be changed anytime).

field 5: Number of days after password must be changed. (Value 99999

means that the password never expires).

field 6: Number of days to warn user for expiring password.

field 7: Number of days after password expires that the account is disabled.

field 8: The number of days from January 1, 1970 to the date when an account was disabled.

field 9: This field is reserved for some possible future use.

passwd options:

-d, –delete: This option deletes the user password and makes the account password-less.

-e, –expire: This option immediately expires the account password and forces the user to change password on their next login.

-i, –inactive INACTIVE_DAYS: This option is followed by an integer, INACTIVE_DAYS, which is the number of days after the password expires that the account will be deactivated.

passwd -i 3 user1

-l, –lock: Lock the password of user. This appends the encrypted password of the user with a character '!', and thus making it unable to match with any of input password combinations. This does not disable the account but prevents the user from logging in using a password. Though other authentication methods like ssh keys can be used to login to the account.

-u, –unlock: Unlock the password of an account.

-w, –warndays WARN_DAYS: This option is used to change the number of days before the password is to expire, to display the warning for expiring password.

-x, –maxdays MAX_DAYS Set the maximum number of days for which the password remains valid. After MAX_DAYS, the password will expire and the user will be forced to change password.

chage command

chage command is used to view and change the user password expiry information. This command is used when the login is to be provided for a user for limited amount of time or when it is necessary to change the login password time to time.

chage [options] LOGIN

-l option : use this option to view the account aging information. In order to view the aging information of the root i am using the keyword sudo.

sudo chage -l <username>

d option : use this option to set the last password change date to your specified date in the command.

chage -d 2018-12-01 <username>

E option : use this option to specify the date when the account should expire.

chage -E 2018-12-01 <username>

-M or -m option : use this option to specify the maximum and minimum number of days between password change.

chage -M 5 <username>

-W option : use this option to give prior warning before the password expires.The input given in the command is the number of days prior to the expiry date when the warning should be given .

chage -w 5 <username>

userdel command

userdel command in Linux system is used to delete a user account and related files. This command basically modifies the system account files, deleting all the entries which refer to the username LOGIN. It is a low-level utility for removing the users.

userdel -f: This option forces the removal of the specified user account. It doesn't matter that the user is still logged in. It also forces the userdel to remove the user's home directory and mail spool, even if another user is using the same home directory or even if the mail spool is not owned

by the specified user.

userdel -f <username>

Groups in Linux

Each group in a Linux system is uniquely identified by a group identification number or GID. All the information listing groups in a system are stored in /etc/group file. Every user has a primary user group and zero or more supplementary groups. On login, the group membership is set to the primary group of user. This can be changed to any other supplementary group using newgrp or chgrp commands.

Description of contents of /etc/group File

This file is readable by any user but only root as read and write permissions for it. This file consists of the following colon separated information about groups in a system:

- Group name field
- Password field
- If this field is empty, no password is needed.
- Group Identification number or GID

Comma separated list of usernames of users that belong to the group.

Description of contents of /etc/gshadow File

This file is readable and writable by only by root user. Each entry of this file contains each group's encrypted password, group membership and administrator information separated by a colon.

- Group name field
- Password field

It contains an encrypted password.The password is used when a user who is not a member of the group wants to gain the permissions of this group. Members can access the group without being prompted for a password.

If no password is set, it is indicated by '!' or '!!'.It implies only members

can access the group. There's no way other users can use the group.'!!' indicates that no password has ever been set on the group.

- Comma separated list of user-names who are group administrators.

  Administrators can change the password or the members of the group.

- Comma separated list of user-names who are group members.

groupadd command is used to create a new user group.

groupadd [option] group_name

-f, –force : This option forces the command to silently abort if the group with given already exists. If this option is used with the -g or –gid option and the group id given already exists, the command forcefully ignores the given group id and creates a new and unique group id.

To add a new user into the group, the group is mentioned using -g option in the command useradd.

useradd -G <group> <new_user>

Configure Passwordless Sudo For A Specific User in Linux

Linux has a tough permission rule. Being a root user you are permitted to do anything however the normal user has only some allowed permissions. To run many other commands, they required to ask permissions from the superuser, using keyword sudo after which they need to enter the password.

This requirement of the password after sudo command can be removed using the following steps:

Edit sudoers file using the following command

nano /etc/sudoers

Find a line which contains includedir /etc/sudoers.d, press Ctrl + F and paste includedir /etc/sudoers.d

Below the line includedir /etc/sudoers.d add the following

username ALL=(ALL) NOPASSWD:ALL

Permissions in Linux

Linux is a multi-user operating system, so it has security to prevent people from accessing each other's confidential files.

When you execute an "ls" command, you are not given any information about the security of the files, because by default "ls" only lists the names of files. You can get more information by using an "option" with the "ls" command. All options start with a '-'. For example, to execute "ls" with the "long listing" option, you would type ls -l .When you do so, each

file will be listed on a separate line in long format.

```
[root@localhost ~]# ls -l
total 28
-rw-r--r--.  1 root root    9 May 15 09:21  3120
-rw-------.  1 root root 1269 Apr 18 19:26  anaconda-ks.cfg
drwxr-xr-x.  4 root root  270 May 16 10:31  Desktop
drwxr-xr-x.  2 root root    6 Apr 18 19:43  Documents
drwxr-xr-x.  2 root root    6 May 11 20:04  Downloads
-rw-r--r--.  1 root root 1606 Apr 18 19:43  initial-setup-ks.cfg
drwx------.  3 1239 1239   78 May 17 09:54  kanada
-rw-r--r--.  1 root root  161 May 16 09:34  kumar
drwxr-xr-x.  2 root root   31 May 11 19:55  move
-rw-r--r--.  1 root root   19 May 15 09:27  move.txt
drwxr-xr-x.  2 root root    6 Apr 18 19:43  Music
drwxrwxr-x. 18 root root 4096 Apr 18 23:19  nagios-4.0.8
drwxr-xr-x. 15 root root 4096 Apr 18 23:35  nagios-plugins-2.3.3
drwxr-xr-x.  2 root root    6 May 11 19:26 'name with space'
drwxr-xr-x.  2 root root    6 Apr 18 19:43  Pictures
```

There's a lot of information in those lines.

1. The first character will almost always be either a '-', which means it's a file, or a 'd', which means it's a directory.
2. The next nine characters (rw-r–r–) show the security; we'll talk about them later.
3. The next column shows the owner of the file. In this case it is me, my userID is "root".
4. The next column shows the group owner of the file. In my case I want to give the "root" group of people special access to these files.
5. The next column shows the size of the file in bytes.
6. The next column shows the date and time the file was last modified.
7. And, of course, the final column gives the filename.

Understanding the security permissions

First, you must think of those nine characters as three sets of three characters (see the box at the bottom). Each of the three "rwx" characters refers to a different operation you can perform on the file.

--- --- ---

rwx rwx rwx

user group other

Read, write, execute and –

The ‘r’ means you can “read” the file’s contents.

The ‘w’ means you can “write”, or modify, the file’s contents.

The ‘x’ means you can “execute” the file. This permission is given only if the file is a program.

If any of the “rwx” characters is replaced by a ‘-‘, then that permission has been revoked.

User, group and others

user – The user permissions apply only the owner of the file or directory, they will not impact the actions of other users.

group – The group permissions apply only to the group that has been assigned to the file or directory, they will not effect the actions of other users.

others – The others permissions apply to all other users on the system, this is the permission group that you want to watch the most.

Changing security permissions

The command you use to change the security permissions on files is called “chmod”, which stands for “change mode”, because the nine security characters are collectively called the security “mode” of the file.

1. The first argument you give to the “chmod” command is ‘u’, ‘g’, ‘o’. We use:

- u for user
- g for group

- for others,

you can also use a combination of them (u,g,o).

This specifies which of the three groups you want to modify.

2. After this use

- a '+' for adding
- a '-' for removing
- a "=" for assigning a permission.

3. Then specify the permission r,w or x you want to change. Here also you can use a combination of r,w,x. This specifies which of the three permissions "rwx" you want to modify
4. use can use commas to modify more permissions
5. Finally, the name of the file whose permission you are changing.

The octal notations

You can also use octal notations like this.

Octal	Binary	File Mode
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

Using the octal notations table instead of 'r', 'w' and 'x'. Each digit octal notation can be used of either of the group 'u','g','o'.

So, the following work the same.

chmod ugo+rwx [file_name]

chmod 777 [file_name]

chown command in Linux

chown, which stands for change owner, is a command in Linux to change user or group ownership of a file, directory, or symbolic link. Every file or directory has a user or group ownership in the Linux systems. The Linux system can have multiple users and groups. They all have unique names and IDs. However, the name and ID of a user and a group can be the same.

Change the owner of a file using chown command

This is a simple chown command that changes the owner/user of a file. It requires a new owner name and file name. The new owner must be a valid user.

$ ***sudo chown new_owner file_name***

Change the group of a file using chown command

chown command allows you to change the group of a file. You must use a colon in front of a new group name. Otherwise, it will be considered as a new owner.

$ sudo chown :new_group file_name

chown command to change the owner using user ID

chown command also changes the owner of a file with numeric user ID. But there should not be any other user having the same name as userID. You can use id -u user_name to print the user ID of a user.

$ sudo chown user_ID file_name

chown command to change owner of multiple files

This is a useful command that allows you to change owner of multiple files simultaneously. You have to separate file names with a space. It will assign same owner to all mentioned files.

$ sudo chown new_owner file_name1 file_name2 file_name3

Change owner and group name at the same time with chown command

chown command allows you to change owner and group name at the same time. You have to provide both new owner and group name in order to assign them in a file.

$ sudo chown new_owner:new_group file_name

chown command to copy owner and group name from one file to another

This command copies the owner and group name from one file and assigns the same ownership to another file in the same directory.

$ sudo chown --reference=souce_file destination_file

Change owner of a file using chown command if only the current owner matches

This command assigns the new owner to a file if only its current owner matches. --from option verifies the current owner.

$ sudo chown --from=current_owner new_owner file_name

Access Control Lists

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.

Use of ACL:

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.

setfacl command in Linux is used to set access control lists (ACLs) of files and directories. ACL helps to create an additional, more flexible

permission mechanism for the file system. It allows us to provide permission for any user or group to any disk resource. getfacl command is used to get file access control lists. For each file, getfacl displays the file name, owner, the group, and the Access Control List (ACL). If a directory has a default ACL, getfacl also displays the default ACL.

getfacl command to display the file access control list

-a or --access options display the file access control list of a file or directory.

getfacl -a file

Display the default access control list with getfacl command

You can view the default access control list with -d or --default option.

$ getfacl -d file

getfacl command to list the ACLs of all files and directories recursively (sub-directories)You can use -R or --recursive options to list the ACLs of all files and directories recursively. It is helpful to view the ACLs of a whole directory, including its sub-directories and files.

$ getfacl -R directory

Display ACLs of files in tabular output with getfacl command

-t or --tabular options tell getfacl to use an alternative tabular output format. The ACL and the default ACL are displayed side by side.

$ getfacl -t file

getfacl command to omit header

-c or --omit-header options are used to hide the output's comment header (the first three lines)

$ getfacl -c file

List numeric user and group IDs with getfacl command

You can use -n or --numeric options to display the numeric user and group IDs in the output.

$ getfacl -n file

setfacl command to modify ACLs of file

The -m or --modify=acl options modify the current ACL of a file or directory. For example, to give read and write permission to user

$ setfacl -m u:user rw file

setfacl command to remove all extended ACL entriesYou can remove all extended ACL entries using -b or --remove-all option. The base ACL entries of the owner, group, and others are retained.

$ setfacl -b file

setfacl command to remove the default ACL

-k or --remove-default options are used to remove the default access control list. If no default ACL exists, no warnings are issued.

$ setfacl -k file

Cockpit

Cockpit is a user-friendly web-based interface for administering servers. It allows monitoring system resources and adjusting configuration with ease.

- Cockpit builds upon existing functionality.
- There is no lock-in. Feel free to use other tools alongside Cockpit. Switch back and forth with ease.
- Cockpit does not need special infrastructure or configuration. Once installed, it is ready to use.
- When not in use, Cockpit uses no memory or CPU on the server.
- Cockpit always updates its data to reflect the current state of the server, within seconds.
- Cockpit stores no data or policy. People keep their system-wide permissions and use the system credentials.
- Optionally take advantage of single sign-on with Kerberos.
- Cockpit itself is not used for configuration management. However, Cockpit can interact with configuration management and custom server tools.

Installing and Enabling Cockpit

A primary Cockpit server is the machine that runs a Cockpit service with the user interface. A secondary server is a machine that is administered using Cockpit. It is possible to add one or more secondary hosts to the primary server.

Setting up a primary Cockpit server involves:

1. Installing the *cockpit* packages.
2. Opening the port for Cockpit.
3. Starting the cockpit service.

After setting up, you can connect to Cockpit in a browser by typing the host name and port of the server. For example, from the primary host you can connect using localhost:9090.

Setting up the primary Cockpit server

To install and enable Cockpit:

Install the cockpit and cockpit-dashboard packages:

$ sudo yum install cockpit cockpit-dashboard

The cockpit-dashboard package provides the "Dashboard" tab in the interface. This package is optional, but is assumed to be installed in this guide.

Allow external connections to port 9090 through the firewall:

 # firewall-cmd --add-port=9090/tcp

 # firewall-cmd --permanent --add-port=9090/tcp

Enable and start the cockpit.socket service:

$ sudo systemctl enable cockpit.socket

$ sudo systemctl start cockpit.socket

Cockpit is now installed and running.

Linux Text Editors

Linux text editors can be used for **editing text files, writing codes, updating user instruction files,** and more. A Linux system supports multiple text editors. There are two types of text editors in Linux, which are given below:

- **Command-line text editors** such as Vi, nano, pico, and more.
- **GUI text editors** such as gedit (for Gnome), Kwrite, and more.

A text editor plays an important role while coding. So, it is important to select the best text editor. A text editor should not only be simple but also functional and should be good to work with.A *text editor with IDE features* is considered as a good text editor.

1.Vi/VIM editor

Vim editor is one of the most used and powerful command-line based editor of the Linux system. By default, it is supported by most Linux distros. It has enhanced functionalities of the old Unix Vi editor. It is a user-friendly editor and provides the same environment for all the Linux distros. It is also termed as **programmer's editor** because most programmers prefer Vi editor.

Vi editor has some special features such as Vi modes and syntax highlighting that makes it powerful than other text editors. Generally, it has two modes:

Command Mode: The command mode allows us to perform actions on files. By default, it starts in command mode. In this mode, all types of words are considered as commands. We can execute commands in this mode.

Insert Mode: The insert mode allows to insert text on files. To switch from command mode to insert mode, press the **Esc** key to exit from active mode and **'i'** key.

To learn more about Vi editor, visit the Vi editor with commands.

To invoke the vi editor, execute the vi command with the file name as follows:

1. vi <file name>

There are following way you can start using vi editor :

Commands and their Description

vi filename: Creates a new file if it already not exist, otherwise opens existing file.

vi -R filename : Opens an existing file in read only mode.

view filename : Opens an existing file in read only mode.

Moving within a File(Navigation):

To move around within a file without affecting text must be in command mode (press Esc twice). Here are some of the commands can be used to move around one character at a time.

Commands and their Description

k : Moves the cursor up one line.

j : Moves the cursor down one line.

h : Moves the cursor to the left one character position.

l : Moves the cursor to the right one character position.

0 or | : Positions cursor at beginning of line.

$: Positions cursor at end of line.

W : Positions cursor to the next word.

B : Positions cursor to previous word.

(: Positions cursor to beginning of current sentence.

) : Positions cursor to beginning of next sentence.

H : Move to top of screen.

M : Move to middle of screen.

L : Move to bottom of screen.

Editing and inserting in Files(Entering and Replacing Text): To edit the file, we need to be in the insert mode. There are many ways to enter insert mode from the command mode.

i : Inserts text before current cursor location.

I : Inserts text at beginning of current line.

a : Inserts text after current cursor location.

A : Inserts text at end of current line.

o : Creates a new line for text entry below cursor location.

O : Creates a new line for text entry above cursor location.

r : Replace single character under the cursor with the next character typed.

R : Replaces text from the cursor to right.

s : Replaces single character under the cursor with any number of characters.

S :Replaces entire line.

Deleting Characters: Here is the list of important commands which can be used to delete characters and lines in an opened file.

X Uppercase: Deletes the character before the cursor location.

x Lowercase : Deletes the character at the cursor location.

Dw : Deletes from the current cursor location to the next word.

d^ : Deletes from current cursor position to the beginning of the line.

d$: Deletes from current cursor position to the end of the line.

Dd : Deletes the line the cursor is on.

Copy and Past Commands: Copy lines or words from one place and paste them on another place by using the following commands.

Yy : Copies the current line.

9yy : Yank current line and 9 lines below.

p : Puts the copied text after the cursor.

P : Puts the yanked text before the cursor.

Save and Exit Commands of the ex Mode : Need to press [Esc] key followed by the colon (:) before typing the following commands:

q : Quit

q! : Quit without saving changes i.e. discard changes.

r fileName : Read data from file called fileName.

wq : Write and quit (save and exit).

w fileName : Write to file called fileName (save as).

w! fileName : Overwrite to file called fileName (save as forcefully).

!cmd : Runs shell commands and returns to Command mode.

Nano editor

Nano is a straight forward editor. It is designed for both beginners and advanced users. It has many customization features.

Some advanced features of a nano text editor are as following:

- It has highly customizable key bindings
- It supports syntax highlighting
- It has undo and redo options
- It provides full line display on the standard output

- It has pager support to read from standard input

To open file with nano editor, execute the command as follows:

1. nano <file name>

To create and open a new file.

$nano new_filename

To save a file

press Ctrl+o

To cut and paste a whole line. Move to the line which you want to cut then press Ctrl+k.To paste it, go to the position where you want to paste and then press Ctrl+u

To search a word in a file. Ctrl+w is used.

Process Management

Process

A process is a program in execution. For example, when we write a program in C or C++ and compile it, the compiler creates binary code. The original code and binary code are both programs. When we actually run the binary code, it becomes a process.

A process is an 'active' entity, instead of a program, which is considered a 'passive' entity. A single program can create many processes when run multiple times; for example, when we open a .exe or binary file multiple times, multiple instances begin (multiple processes are created).

States of a Process in Operating Systems

New (Create) – In this step, the process is about to be created but not yet created, it is the program which is present in secondary memory that will be picked up by OS to create the process.

Ready – New -> Ready to run. After the creation of a process, the process enters the ready state i.e. the process is loaded into the main memory. The process here is ready to run and is waiting to get the CPU time for its execution. Processes that are ready for execution by the CPU are maintained in a queue for ready processes.

Run – The process is chosen by CPU for execution and the instructions within the process are executed by any one of the available CPU cores.

Blocked or wait – Whenever the process requests access to I/O or needs input from the user or needs access to a critical region (the lock

for which is already acquired) it enters the blocked or wait state. The process continues to wait in the main memory and does not require CPU. Once the I/O operation is completed the process goes to the ready state.

Terminated or completed – Process is killed as well as PCB is deleted.

Suspend ready – Process that was initially in the ready state but were swapped out of main memory (refer Virtual Memory topic) and placed onto external storage by scheduler are said to be in suspend ready state. The process will transition back to ready state whenever the process is again brought onto the main memory.

Suspend wait or suspend blocked – Similar to suspend ready but uses the process which was performing I/O operation and lack of main memory caused them to move to secondary memory.

When work is finished, it may go to suspend ready.

Running program is a process. From this process, another process can be created. There is a parent-child relationship between the two processes. This can be achieved using a library function called fork(). fork() function splits the running process into two processes, the existing one is known as parent and the new process is known as a child.

A process means program in execution. It generally takes an input, processes it and gives us the appropriate output.

There are basically 2 types of processes.

Foreground processes: Such kind of processes are also known as interactive processes. These are the processes which are to be executed or initiated by the user or the programmer, they cannot be initialized by system services. Such processes take input from the user and return the output. While these processes are running, we cannot directly initiate a new process from the same terminal.

Background processes: Such kind of processes are also known as non-interactive processes. These are the processes that are to be executed or initiated by the system itself or by users, though they can even be managed by users. These processes have a unique PID or process if assigned to them and we can initiate other processes within the same terminal from which they are initiated.

Example of foreground process.

sleep 5

Stopping a process in between of its execution. To stop a foreground process in between of its execution we may press CTRL+Z to force stop it.

sleep 100

To get the list of jobs that are either running or stopped.

Jobs It will display the stopped processes in this terminal and even the pending ones.

To run all the pending and force stopped jobs in the background.

bg This will start the stopped and pending processes in the background.

To run all the pending and force stopped jobs in the foreground.

fg. This will start the stopped and pending processes in the foreground.

To run a process in the background without getting impacted by the closing of the terminal.

nohup sleep 100 &

To run some processes in the background directly.

sleep 100&. This will run the process in the background and will display the process id of the process.

To get the list of all the running processes on your Linux machine.

Top

Daemons

A **daemon** (also known as background processes) is a Linux or UNIX program that runs in the background. Almost all daemons have names that end with the letter "**d**". For example, httpd the daemon that handles the Apache server, or, sshd which handles SSH remote access connections. Linux often start daemons at boot time. Shell scripts stored in /etc/init.d directory are used to start and stop daemons.

Typical functions of daemons

Open network port (such as port 80) and respond to network requests.
Run scheduled tasks such as cron.

at Command

at command is a command-line utility that is used to schedule a command to be executed at a particular time in the future. Jobs created with at command are executed only once. The at command can be used to execute any program or mail at any time in the future. It executes commands at a particular time and accepts times of the form HH:MM to run a job at a specific time of day. The following expression like noon, midnight, teatime, tomorrow, next week, next Monday, etc. could be used with at command to schedule a job.

Command to list the user's pending jobs

at -l (or) atq

Schedule a job for the coming Monday at a time twenty minutes later than the current time

at Monday +20 minutes

Schedule a job to run at 1:45 Aug 12 2020

at 1:45 081220

Schedule a job to run at 3pm four days from now

at 3pm + 4 days

Schedule a job to shut down the system at 4:30 today:

echo "shutdown -h now" | at -m 4:30

Schedule a job to run five hour from now:

at now +5 hours

cron command

The cron is a software utility, offered by a Linux-like operating system

that automates the scheduled task at a predetermined time. It is a daemon process, which runs as a background process and performs the specified operations at the predefined time when a certain event or condition is triggered without the intervention of a user. Dealing with a repeated task frequently is an intimidating task for the system administrator and thus he can schedule such processes to run automatically in the background at regular intervals of time by creating a list of those commands using cron. It enables the users to execute the scheduled task on a regular basis unobtrusively like doing the backup every day at midnight, scheduling updates on a weekly basis, synchronizing the files at some regular interval. Cron checks for the scheduled job recurrently and when the scheduled time fields match the current time fields, the scheduled commands are executed. It is started automatically from /etc/init.d on entering multi-user run levels.

The crontab (abbreviation for “cron table”) is list of commands to execute the scheduled tasks at specific time. It allows the user to add, remove or modify the scheduled tasks. The crontab command syntax has six fields separated by space where the first five represent the time to run the task and the last one is for the command.

1. Minute (holds a value between 0-59)
2. Hour (holds value between 0-23)
3. Day of Month (holds value between 1-31)
4. Month of the year (holds a value between 1-12 or Jan-Dec, the first three letters of the month’s name shall be used)
5. Day of the week (holds a value between 0-6 or Sun-Sat, here also first three letters of the day shall be used)

Command

The rules which govern the format of date and time field as follows:

When any of the first five fields are set to an asterisk(*), it stands for all the values of the field. For instance, to execute a command daily, we can put an asterisk(*) in the week’s field.

One can also use a range of numbers, separated with a hyphen(-) in the time and date field to include more than one contiguous value but not all the values of the field. For example, we can use the 7-10 to run a command from July to October.

The comma (,) operator is used to include a list of numbers which may or may not be consecutive. For example, “1, 3, 5” in the weeks’ field signifies execution of a command every Monday, Wednesday, and Friday.

A slash character(/) is included to skip given number of values. For instance, “*/4” in the hour’s field specifies ‘every 4 hours’ which is equivalent to 0, 4, 8, 12, 16, 20.

Permitting users to run cron jobs:

The user must be listed in this file to be able to run cron jobs if the file exists.

/etc/cron.allow

If the cron.allow file doesn’t exist but the cron.deny file exists, then a user must not be listed in this file to be able to run the cron job.

/etc/cron.deny

Run /home/folder/gfg-code.sh every hour, from 9:00 AM to 6:00 PM, everyday.

00 09-18 * * * /home/folder/gfg-code.sh

Run /usr/local/bin/backup at 11:30 PM, every weekday.

30 23 * * Mon, Tue, Wed, Thu, Fri /usr/local/bin/backup

Run sample-command.sh at 07:30, 09:30, 13:30 and 15:30.

30 07, 09, 13, 15 * * * sample-command.sh

The following points should be remembered while working with cron:

Have a source version control to track and maintain the changes to the cron expressions.

Organize the scheduled jobs based on their importance or the frequency and group them by their action or the time range.

Test the scheduled job by having a high frequency initially.

Do not write complex code or several pipings and redirection in the cron expression directly. Instead, write them to a script and schedule the script to the cron tab.

Use aliases when the same set of commands are frequently repeated.

Avoid running commands or scripts through cron as a root user.

Systemctl:

systemctl is used to examine and control the state of “systemd” system and service manager. systemd is system and service manager for Unix like operating systems(most of the distributions, not all). As the system boots up, the first process created, i.e. init process with PID = 1, is systemd system that initiates the userspace services.

tar command:

The Linux ‘tar’ stands for tape archive, is used to create Archive and extract the Archive files. tar command in Linux is one of the important command which provides archiving functionality in Linux. We can use Linux tar command to create compressed or uncompressed Archive files and also maintain and modify them.

An Archive file is a file that is composed of one or more files along with metadata. Archive files are used to collect multiple data files together into a single file for easier portability and storage, or simply to compress files to use less storage space.

Creating an uncompressed tar Archive using option -cvf

tar cvf <file.tar> <filename>

Gzip Command:

gzip command compresses files. Each single file is compressed into a

single file. The compressed file consists of a GNU zip header and deflated data.

If given a file as an argument, gzip compresses the file, adds a ".gz" suffix, and deletes the original file. With no arguments, gzip compresses the standard input and writes the compressed file to standard output.

To create a compressed gzip archive file we use the option as z.

tar cvzf <img.tar.gz> /home/MyImages

-f option: Sometimes a file cannot be compressed. Perhaps you are trying to compress a file called "myfile1" but there is already a file called "myfile1.gz". In this instance, the "gzip" command won't ordinarily work.

To force the "gzip" command to do its stuff simply use -f option:

gzip -f <filename>

-k option: By default when you compress a file using the "gzip" command you end up with a new file with the extension ".gz".If you want to compress the file and keep the original file you have to run the gzip command with -k option:

$ gzip -k mydoc.txt

-L option: This option displays the gzip license.

$ gzip -L filename.gz

-[1-9] option: It allows to change the compression level.A file can be compressed in different ways. For instance, you can go for a smaller compression which will work faster or you can go for maximum compression which has the tradeoff of taking longer to run.The speed and compression level can vary by levels using numbers between 1 and 9.

$ gzip -1 mydoc.txt

-d option: This option allows to decompress a file using the "gzip" command.

$ gzip -d mydoc.txt.gz

Bzip2 Command:

The bz2 feature compresses and creates an archive file less than the size of the gzip. The bz2 compression takes more time to compress and decompress files than gzip, which takes less time.To create a highly compressed tar file we use the option j.

Create tar.bz2 Archive File in Linux

tar cvfj <filename>.tar.bz2 /home/php

-z : This option forces compression. It is an opposite command of decompression i.e. -d Option.

$ bzip2 -z input.txt

This option deletes the original file also.

-v: Verbose mode show the compression ratio for each file processed. It also increases the verbosity level, spewing out lots of information which is primarily of interest for diagnostic purposes.

$ bzip2 -v input.txt

-k: This option does compression but does not delete the original file.

$ bzip2 -k input.txt

-d : This option is used for decompression of compressed files.

$ bzip2 -d input.txt.bz2

XZ:

xz is a general-purpose data compression and decompression command-line tool similar to gzip and bzip2. It can be used to compress and decompress the files. The native file format of xz is .xz. But it can also support other various formats to compress or decompress files.xz gives us complete control over the compression and decompression of files. In Linux, xz tool is come by default with the system you don't need to install it. Now let's see how to use the xz tool to compress the files.

Compressing files using XZ:

There are two ways in which we can compress the files using the xz on way is just mention file name followed by xz command like:

xz filename

Another way is by using the -z or –compress option:

xz -z filename

Here in place of filename mention the file name or folder name. But while compressing files using xz make sure that there is no compressed file of the same file which we are going to compress.

Networking

A computer network is a system that connects numerous independent computers in order to share information (data) and resources. The integration of computers and other different devices allows users to communicate more easily.A computer network is a collection of two or more computer systems that are linked together. A network connection can be established using either cable or wireless media. Hardware and software are used to connect computers and tools in any network. A computer network consists of various kinds of nodes. Servers, networking hardware, personal computers, and other specialized or general-purpose hosts can all be nodes in a computer network. Hostnames and network addresses are used to identify them.

Goal Of Networking:

1. Programs do not have to execute on a single system because of resource and load sharing.
2. Reduced costs – Multiple machines can share printers, tape drives, and other peripherals.
3. Reliability – If one machine fails, another can take its place.
4. Scalability (it's simple to add more processors or computers)
5. Communication and mail (people living apart can work together)
6. Information Access (remote information access, access to the internet, e-mail, video conferencing, and online shopping)
7. Entertainment that is interactive (online games, videos, etc.)
8. Social Networking

Types of Networks

1. Division based on the communication medium

Wired Network: As we all know, "wired" refers to any physical medium made up of cables. Copper wire, twisted pair, or fiber optic cables are all options. A wired network employs wires to link devices to the Internet or another network, such as laptops or desktop PCs.

Wireless Network: "Wireless" means without wire, media that is made up of electromagnetic waves (EM Waves) or infrared waves. Antennas or sensors will be present on all wireless devices. Cellular phones, wireless sensors, TV remotes, satellite disc receivers, and laptops with WLAN cards are all examples of wireless devices. For data or voice communication, a wireless network uses radio frequency waves rather than wires.

2. Division based on area covered

Local Area Network (LAN): A LAN is a network that covers an area of around 10 kilometers. For example, a college network or an office network.

Metropolitan Area Network (MAN): MAN refers to a network that covers an entire city. For example: consider the cable television network.

Wide Area Network (WAN): WAN refers to a network that connects countries or continents. For example, the Internet allows users to access a distributed system called www from anywhere in the globe.

3. Based on types of communication

Point To Point networks: Point-to-Point networking is a type of data networking that establishes a direct link between two networking nodes.A direct link between two devices, such as a computer and a printer, is known as a point-to-point connection.

Broadcast networks: In broadcast networks, a signal method in which numerous parties can hear a single sender. Radio stations are an excellent illustration of the “Broadcast Network” in everyday life. The radio station is a sender of data/signal in this scenario, and data is only intended to travel in one direction. Away from the radio transmission tower, to be precise.

4. Based on type of architecture

P2P Networks: Computers with similar capabilities and configurations are referred to as peers.

“Peer to Peer” is the abbreviation for “peer to peer.” The “peers” in a peer -to-peer network are computer systems that are connected to each other over the Internet. Without the use of a central server, files can be shared directly between systems on the network.

Client – Server Networks: Each computer or process on the network is either a client or a server in a client-server architecture (client/server). The client asks services from the server, which the server provides. Servers are high-performance computers or processes that manage disc drives (file servers), printers (print servers), or network traffic (network servers)

OSI Model

OSI stands for **Open System Interconnection** is a reference model that describes how information from a software application in one computer moves through a physical medium to the software application in another computer.

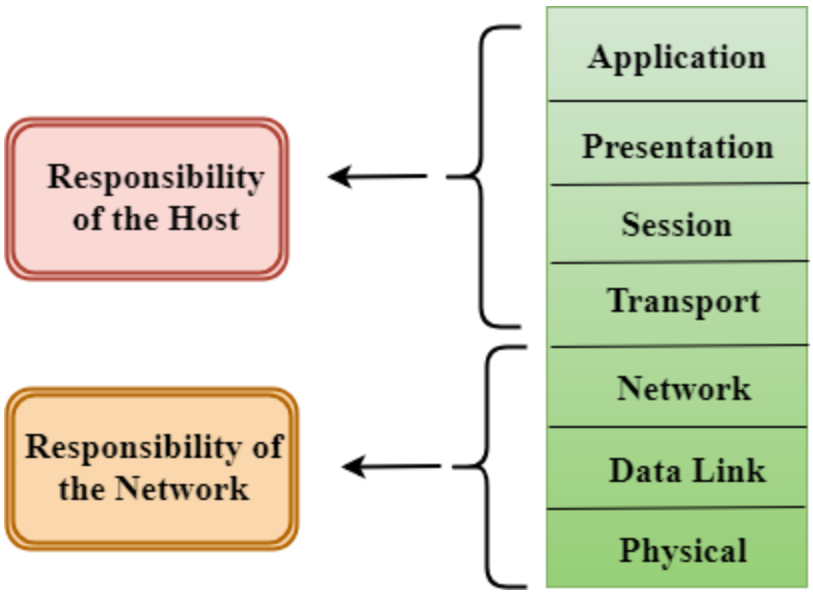
OSI consists of seven layers, and each layer performs a particular network function.

OSI model was developed by the International Organization for Standardization (ISO) in 1984, and it is now considered as an architectural model for the inter-computer communications.

OSI model divides the whole task into seven smaller and manageable tasks. Each layer is assigned a particular task.

Each layer is self-contained, so that task assigned to each layer can be performed independently.

Characteristics of OSI Model:



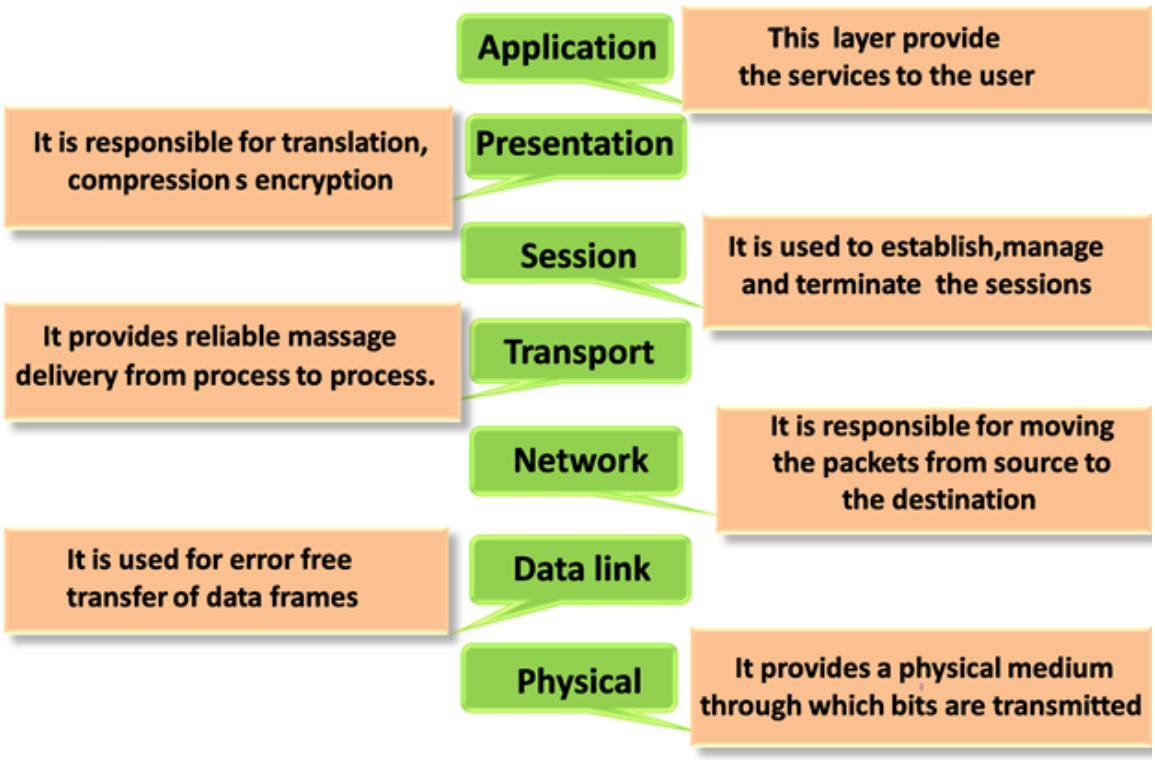
- The OSI model is divided into two layers: upper layers and lower layers.
- The upper layer of the OSI model mainly deals with the application related issues, and they are implemented only in the software. The application layer is closest to the end user. Both the end user and the application layer interact with the software applications. An upper layer refers to the layer just above another layer.
- The lower layer of the OSI model deals with the data transport issues. The data link layer and the physical layer are implemented in hardware and software. The physical layer is the lowest layer of the OSI model and is closest to the physical medium. The physical layer is mainly responsible for placing the information on the

physical medium.

Functions of the OSI Layers

There are the seven OSI layers. Each layer has different functions. A list of seven layers are given below:

1. Physical Layer
2. Data-Link Layer
3. Network Layer
4. Transport Layer
5. Session Layer
6. Presentation Layer
7. Application Layer



Physical layer

- The main functionality of the physical layer is to transmit the individual bits from one node to another node.
- It is the lowest layer of the OSI model.

- It establishes, maintains and deactivates the physical connection.
- It specifies the mechanical, electrical and procedural network interface specifications.

Functions of a Physical layer:

- **Line Configuration:** It defines the way how two or more devices can be connected physically.
- **Data Transmission:** It defines the transmission mode whether it is simplex, half-duplex or full-duplex mode between the two devices on the network.
- **Topology:** It defines the way how network devices are arranged.
- **Signals:** It determines the type of the signal used for transmitting the information.

Data-Link Layer

- This layer is responsible for the error-free transfer of data frames.
- It defines the format of the data on the network.
- It provides a reliable and efficient communication between two or more devices.
- It is mainly responsible for the unique identification of each device that resides on a local network.
- It contains two sub-layers:
    - **Logical Link Control Layer**
        - It is responsible for transferring the packets to the Network layer of the receiver that is receiving.
        - It identifies the address of the network layer protocol from the header.
        - It also provides flow control.
    - **Media Access Control Layer**
        - A Media access control layer is a link between the Logical Link Control layer and the network's physical layer.

- It is used for transferring the packets over the network.

Functions of the Data-link layer

- **Framing:** The data link layer translates the physical's raw bit stream into packets known as Frames. The Data link layer adds the header and trailer to the frame. The header which is added to the frame contains the hardware destination and source address.
- **Physical Addressing:** The Data link layer adds a header to the frame that contains a destination address. The frame is transmitted to the destination address mentioned in the header.
- **Flow Control:** Flow control is the main functionality of the Data-link layer. It is the technique through which the constant data rate is maintained on both the sides so that no data get corrupted. It ensures that the transmitting station such as a server with higher processing speed does not exceed the receiving station, with lower processing speed.
- **Error Control:** Error control is achieved by adding a calculated value CRC (Cyclic Redundancy Check) that is placed to the Data link layer's trailer which is added to the message frame before it is sent to the physical layer. If any error seems to occurr, then the receiver sends the acknowledgment for the retransmission of the corrupted frames.
- **Access Control:** When two or more devices are connected to the same communication channel, then the data link layer protocols are used to determine which device has control over the link at a given time.

Network Layer

- It is a layer 3 that manages device addressing, tracks the location of devices on the network.
- It determines the best path to move data from source to the destination based on the network conditions, the priority of service, and other factors.
- The Data link layer is responsible for routing and forwarding the packets.

- Routers are the layer 3 devices, they are specified in this layer and used to provide the routing services within an internetwork.
- The protocols used to route the network traffic are known as Network layer protocols. Examples of protocols are IP and Ipv6.

Functions of Network Layer:

- **Internetworking:** An internetworking is the main responsibility of the network layer. It provides a logical connection between different devices.
- **Addressing:** A Network layer adds the source and destination address to the header of the frame. Addressing is used to identify the device on the internet.
- **Routing:** Routing is the major component of the network layer, and it determines the best optimal path out of the multiple paths from source to the destination.
- **Packetizing:** A Network Layer receives the packets from the upper layer and converts them into packets. This process is known as Packetizing. It is achieved by internet protocol (IP).

Transport Layer

- The Transport layer is a Layer 4 ensures that messages are transmitted in the order in which they are sent and there is no duplication of data.
- The main responsibility of the transport layer is to transfer the data completely.
- It receives the data from the upper layer and converts them into smaller units known as segments.
- This layer can be termed as an end-to-end layer as it provides a point-to-point connection between source and destination to deliver the data reliably.

The two protocols used in this layer are:

- **Transmission Control Protocol**
  - It is a standard protocol that allows the systems to communicate over the internet.

- It establishes and maintains a connection between hosts.
- When data is sent over the TCP connection, then the TCP protocol divides the data into smaller units known as segments. Each segment travels over the internet using multiple routes, and they arrive in different orders at the destination. The transmission control protocol reorders the packets in the correct order at the receiving end.

- **User Datagram Protocol**
  - User Datagram Protocol is a transport layer protocol.
  - It is an unreliable transport protocol as in this case receiver does not send any acknowledgment when the packet is received, the sender does not wait for any acknowledgment. Therefore, this makes a protocol unreliable.

Functions of Transport Layer:

- **Service-point addressing:** Computers run several programs simultaneously due to this reason, the transmission of data from source to the destination not only from one computer to another computer but also from one process to another process. The transport layer adds the header that contains the address known as a service-point address or port address. The responsibility of the network layer is to transmit the data from one computer to another computer and the responsibility of the transport layer is to transmit the message to the correct process.
- **Segmentation and reassembly:** When the transport layer receives the message from the upper layer, it divides the message into multiple segments, and each segment is assigned with a sequence number that uniquely identifies each segment. When the message has arrived at the destination, then the transport layer reassembles the message based on their sequence numbers.
- **Connection control:** Transport layer provides two services Connection-oriented service and connectionless service. A connectionless service treats each segment as an individual packet, and they all travel in different routes to reach the destination. A connection-oriented service makes a connection

with the transport layer at the destination machine before delivering the packets. In connection-oriented service, all the packets travel in the single route.

- **Flow control:** The transport layer also responsible for flow control but it is performed end-to-end rather than across a single link.
- **Error control:** The transport layer is also responsible for Error control. Error control is performed end-to-end rather than across the single link. The sender transport layer ensures that message reach at the destination without any error.

Session Layer

- It is a layer 5 in the OSI model.
- The Session layer is used to establish, maintain and synchronizes the interaction between communicating devices.

Functions of Session layer:

- **Dialog control:** Session layer acts as a dialog controller that creates a dialog between two processes or we can say that it allows the communication between two processes which can be either half-duplex or full-duplex.
- **Synchronization:** Session layer adds some checkpoints when transmitting the data in a sequence. If some error occurs in the middle of the transmission of data, then the transmission will take place again from the checkpoint. This process is known as Synchronization and recovery.

Presentation Layer

- A Presentation layer is mainly concerned with the syntax and semantics of the information exchanged between the two systems.
- It acts as a data translator for a network.
- This layer is a part of the operating system that converts the data from one presentation format to another format.
- The Presentation layer is also known as the syntax layer.

Functions of Presentation layer:

- **Translation:** The processes in two systems exchange the information in the form of character strings, numbers and so on. Different computers use different encoding methods, the presentation layer handles the interoperability between the different encoding methods. It converts the data from sender-dependent format into a common format and changes the common format into receiver-dependent format at the receiving end.
- **Encryption:** Encryption is needed to maintain privacy. Encryption is a process of converting the sender-transmitted information into another form and sends the resulting message over the network.
- **Compression:** Data compression is a process of compressing the data, i.e., it reduces the number of bits to be transmitted. Data compression is very important in multimedia such as text, audio, video.

Application Layer

- An application layer serves as a window for users and application processes to access network service.
- It handles issues such as network transparency, resource allocation, etc.
- An application layer is not an application, but it performs the application layer functions.
- This layer provides the network services to the end-users.

Functions of Application layer:

- **File transfer, access, and management (FTAM):** An application layer allows a user to access the files in a remote computer, to retrieve the files from a computer and to manage the files in a remote computer.
- **Mail services:** An application layer provides the facility for email forwarding and storage.
- Directory services: An application provides the distributed database sources and is used to provide that global information about various objects.

IP address

An IP address is the identifier that enables your device to send or receive data packets across the internet. It holds information related to your location and therefore making devices available for two-way communication. The internet requires a process to distinguish between different networks, routers, and websites. Therefore, IP addresses provide the mechanism of doing so, and it forms an indispensable part in the working of the internet.

An IP address is represented by a series of numbers segregated by periods(.). They are expressed in the form of four pairs - an example address might be 255.255.255.255 wherein each set can range from 0 to 255.IP addresses are not produced randomly. They are generated mathematically and are further assigned by the IANA (Internet Assigned Numbers Authority), a department of the ICANN.ICANN stands for Internet Corporation for Assigned Names and Numbers. It is a non-profit corporation founded in the US back in 1998 with an aim to manage Internet security and enable it to be available by all.

The process of IP address works in the following way:

1. Your computer, smartphone, or any other Wi-Fi-enabled device firstly connects to a network that is further connected to the internet. The network is responsible for giving your device access to the internet.
2. While working from home, your device would be probably using that network provided by your Internet Service Provider (ISP). In a professional environment, your device uses your company network.
3. Your ISP is responsible to generate the IP address for your device.
4. Your internet request penetrates through the ISP, and they place

the requested data back to your device using your IP address. Since they provide you access to the internet, ISP's are responsible for allocating an IP address to your computer or respective device.

5. Your IP address is never consistent and can change if there occurs any changes in its internal environment. For instance, if you turn your modem or router on or off, it will change your IP address. Or the user can also connect the ISP to change their IP address.
6. When you are out of your home or office, mainly if you travel and carry your device with you, your computer won't be accessing your home IP address anymore. This is because you will be accessing the different networks (your phone hotspot, Wi-Fi at a cafe, resort, or airport, etc.) to connect the device with the internet. Therefore, your device will be allocated a different (temporary) IP address by the ISP of the hotel or cafe.

Types of IP addresses

There are various classifications of IP addresses, and each category further contains some types.

Consumer IP addresses

Every individual or firm with an active internet service system pursues two types of IP addresses, i.e., Private IP (Internet Protocol) addresses and public IP (Internet Protocol) addresses. The public and private correlate to the network area. Therefore, a private IP address is practiced inside a network, whereas the other (public IP address) is practiced outside a network.

1. Private IP addresses

All the devices that are linked with your internet network are allocated a private IP address. It holds computers, desktops, laptops, smartphones, tablets, or even Wi-Fi-enabled gadgets such as speakers, printers, or smart Televisions. With the expansion of IoT (internet of things), the demand for private IP addresses at individual homes is also seemingly growing. However, the router requires a method to identify these things distinctly. Therefore, your router produces unique private IP addresses that act as an identifier for every device using your internet network. Thus, differentiating them from one another on the network.

2. Public IP addresses

A public IP address or primary address represents the whole network of devices associated with it. Every device included within with your primary address contains their own private IP address. ISP is responsible to provide your public IP address to your router. Typically, ISPs contains the bulk stock of IP addresses that they dispense to their clients. Your public IP address is practiced by every device to identify your network that is residing outside your internet network.

Public IP addresses are further classified into two categories- dynamic and static.

DynamicIPaddresses
As the name suggests, Dynamic IP addresses change automatically and frequently. With these types of IP address, ISPs already purchase a bulk stock of IP addresses and allocate them in some order to their customers. Periodically, they re-allocate the IP addresses and place the used ones back into the IP addresses pool so they can be used later for another client. The foundation for this method is to make cost savings profits for the ISP.

- **StaticIPaddresses**
  In comparison to dynamic IP addresses, static addresses are constant in nature. The network assigns the IP address to the device only once and, it remains consistent. Though most firms or individuals do not prefer to have a static IP address, it is essential to have a static IP address for an organization that wants to host its network server. It protects websites and email addresses linked with it with a constant IP address.

ifconfig command:

ifconfig(interface configuration) command is used to configure the kernel-resident network interfaces. It is used at the boot time to set up the interfaces as necessary. After that, it is usually used when needed during debugging or when you need system tuning. Also, this command is used to assign the IP address and netmask to an interface or to enable or disable a given interface.

-a : This option is used to display all the interfaces available, even if they are down.

Syntax:

ifconfig -a

-s : Display a short list, instead of details.

Syntax:

ifconfig -s

up/down : This option is used to activate/deactivate the driver for the given interface.

Syntax:

ifconfig interface up/down

nmcli command

nmcli is a command-line tool which is used for controlling NetworkManager. nmcli command can also be used to display network device status, create, edit, activate/deactivate, and delete network connections.

Typical Uses:

Scripts: Instead of manually managing the network connections it utilize NetworkMaager via nmcli.

Servers, headless machine and terminals: Can be used to control NetworkManager with no GUI and control system-wide connections.

Syntax:

nmcli [OPTIONS] OBJECT { COMMAND | help }

Where the OBJECT can be any one of the following:

nm: NetworkManager’s status.

connection/cn: NetworkManager’s connection.

device: devices managed by NetworkManager.

To check the device status using nmcli command.

nmcli dev status

To check active connection on the device.

nmcli connection

What is UUID in network?

A universally unique identifier (UUID) is a 128-bit number used to identify information in computer systems.

Configure Static IP Address:

To set ip address

nmcli connection modify <interface_name> ipv4.address <ip/prefix>

example:

nmcli con mod enp0s3 ipv4.addresses 192.168.1.4/24

nmcli con mod enp0s3 ipv4.gateway 192.168.1.1

nmcli con mod enp0s3 ipv4.method manual

nmcli con mod enp0s3 ipv4.dns "8.8.8.8"

nmcli con up enp0s3

cat /etc/sysconfig/network-scripts/ifcfg-enp0s3

ip addr show enp0s3

systemctl restart NetworkManager

Configure Static IP Address using ‘nmtui’ utility

nmtui is a text based user interface for controlling network manager, when we execute nmtui, it will open a text base user interface through which we can add, modify and delete connections. Apart from this nmtui can also be used to set hostname of your system.

Create NIC Teaming

NIC teaming is the aggregation or bonding of two or more network links into a single logical link to provide redundancy and high availability. The logical interface/link is known as a team interface. In the event that the active physical link goes down, one of the backup or reserved links automatically kicks and ensures an uninterrupted connection to the server.

Before we roll our sleeves, it's crucial to familiarize yourself with the following terminologies:

- **Teamd** – This is the nic teaming daemon that uses the **libteam** library to communicate with team devices via the Linux kernel.
- **Teamdctl**– This is a utility that allows users to control an instance of **teamd**. You can check and change the port status, as well as switch between backup and active states.
- **Runner** – These are units of code written in **JSON** and are used for the implementation of various NIC teaming concepts. Examples of runner modes include Round robbin, load balancing, broadcast, and active backup.

Install the teamd Daemon

Teamd is the daemon that is responsible for creating a network team that will act as the logical interface during runtime. By default, it comes installed with CentOS/RHEL 8.

dnf install teamd

Once installed verify that teamd is installed by running the rpm command:

rpm -qi teamd

Configure NIC Teaming:

To configure NIC teaming we will use the handy nmcli tool that can be

used for the management of NetworkManager service. In my system, I have 2 NIC cards that I'm going to bond or combine to create a logical team interface: ens160 and ens192. This may be different in your case.

To confirm the active network interfaces run:

nmcli device status

nmcli connection show

nmcli connection delete <UUID>

nmcli device status

nmcli connection add type team con-name team0 ifname team0 config '{"runner": {"name": "activebackup"}}'

nmcli connection show team0

nmcli connection show

$ nmcli con mod team0 ipv4.addresses 192.168.2.100/24

$ nmcli con mod team0 ipv4.gateway 192.168.2.1

$ nmcli con mod team0 ipv4.dns 8.8.8.8

$ nmcli con mod team0 ipv4.method manual

$ nmcli con mod team0 connection.autoconnect yes

$ nmcli con add type team-slave con-name team0-slave0 ifname enp0s3 master team0

$ nmcli con add type team-slave con-name team0-slave1 ifname enp0s8 master team0

$nmcli connection show

$ nmcli connection down team0 && nmcli connection up team0

$ ip addr show dev team0

$ sudo teamdctl team0 state

Testing Network Teaming Redundancy

To test our active-backup teaming mode, we will disconnect the currently active link – enp0s3 – and check whether the other link kicks in.

$ nmcli device disconnect enp0s3

$ sudo teamdctl team0 state

Deleting a Network Teaming Interface

If you wish to delete the teaming interface/link and revert to default network settings, first bring down the teaming link:

$ nmcli connection down team0

$ nmcli connection delete team0-slave0 team0-slave1

$ nmcli connection delete team0

$ sudo ifconfig enp0s3 up

$ sudo ifconfig enp0s8 up

$ sudo systemctl restart NetworkManager

NIC teaming offers an excellent solution for network redundancy. With 2 or more network interfaces, you can configure a teaming interface in any runner mode to ensure high availability in the event one link goes down accidentally.

hostnamectl command:

hostnamectl command provides a proper API used to control Linux system hostname and change its related settings. The command also helps to change the hostname without actually locating and editing the /etc/hostname file on a given system.

Types of hostname:

Static: Assigned by system admin and it is used to initialize the kernel hostname during boot time.

Dynamic or Transient: Assigned by mDNS server or DHCP server during run time.

Pretty: It's a high-level hostname assigned by system admin or end-user.

Running hostnamectl command to check the current host names. We can either execute hostnamectl or hostnamectl status, the result will be same as status option is automatically assumed if no option is given.

To change static host name to linux. It may require root permission.

hostnamectl set-hostname linux

FTP: File Transfer Protocol

FTP (File Transfer Protocol) is a network protocol for transmitting files between computers over Transmission Control Protocol/Internet Protocol (TCP/IP) connections. Within the TCP/IP suite, FTP is considered an application layer protocol.

In an FTP transaction, the end user's computer is typically called the *local host*. The second computer involved in FTP is a *remote host*, which is usually a server. Both computers need to be connected via a network and configured properly to transfer files via FTP. Servers must be set up to run FTP services, and the client must have FTP software

installed to access these services.

Although many file transfers can be conducted using Hypertext Transfer Protocol (HTTP) -- another protocol in the TCP/IP suite -- FTP is still commonly used to transfer files behind the scenes for other applications, such as banking services. It is also sometimes used to download new applications via web browsers.

How does FTP work?

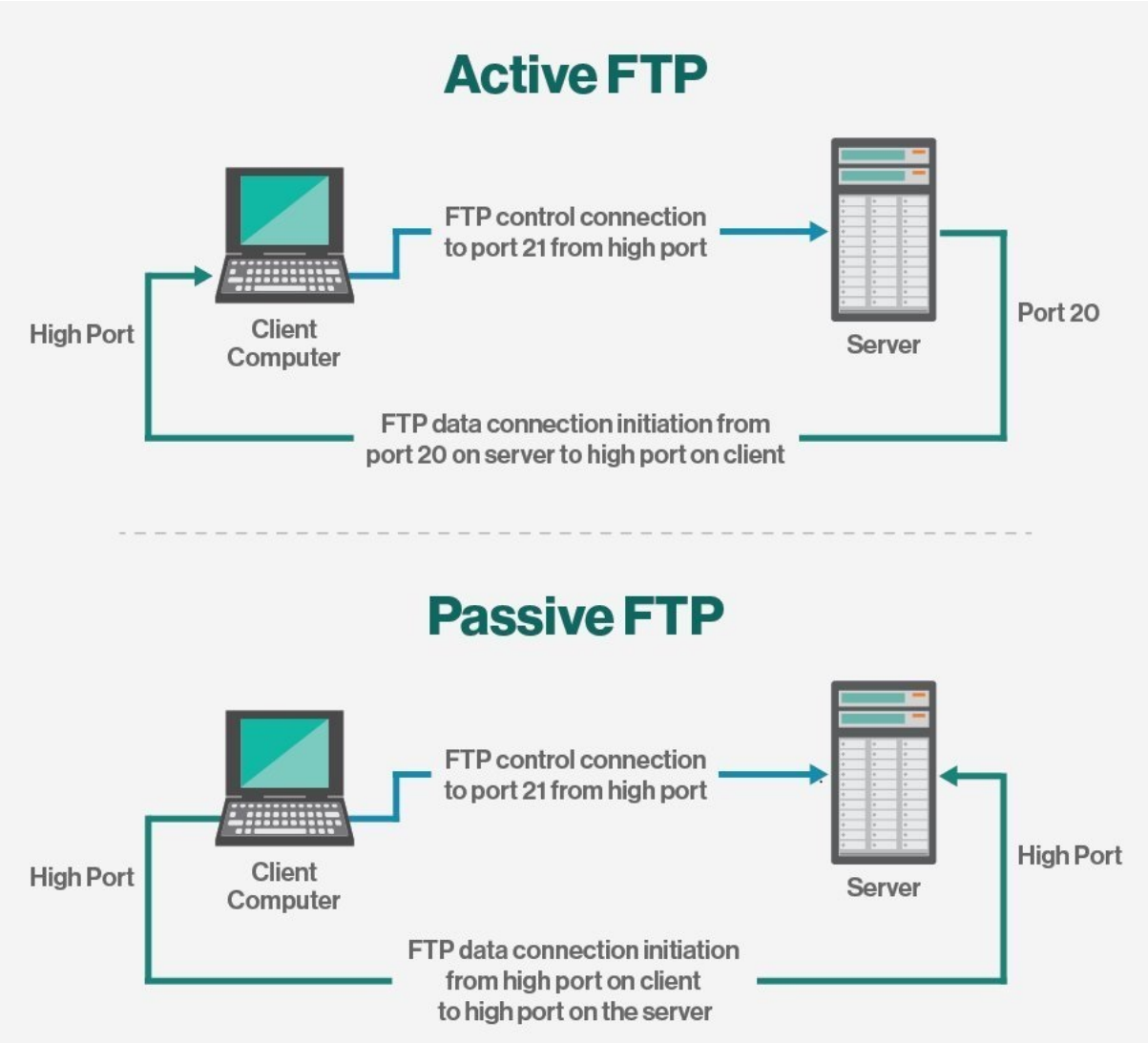
FTP is a client-server protocol that relies on two communications channels between the client and server: a command channel for controlling the conversation and a data channel for transmitting file content.

Here is how a typical FTP transfer works:

1. A user typically needs to log on to the FTP server, although some servers make some or all of their content available without a login, a model known as anonymous FTP.
2. The client initiates a conversation with the server when the user requests to download a file.
3. Using FTP, a client can upload, download, delete, rename, move and copy files on a server.

FTP sessions work in active or passive modes:

- Active mode. After a client initiates a session via a command channel request, the server creates a data connection back to the client and begins transferring data.
- Passive mode. The server uses the command channel to send the client the information it needs to open a data channel. Because passive mode has the client initiating all connections, it works well across firewalls and network address translation gateways.



FTP is a standard network protocol that can enable expansive file transfer capabilities across IP networks. Without FTP, file and data transfer can be managed with other mechanisms -- such as email or an HTTP web service -- but those other options lack the clarity of focus, precision and control that FTP enables.

FTP is used for file transfers between one system and another, and it has several common use cases, including the following:

- **Backup.** FTP can be used by backup services or individual users to backup data from one location to a secured backup server

running FTP services.

- **Replication.** Similar to backup, replication involves duplication of data from one system to another but takes a more comprehensive approach to provide higher availability and resilience. FTP can also be used to facilitate this.
- **Access and data loading.** FTP is also commonly used to access shared web hosting and cloud services as a mechanism to load data onto a remote system.

FTP types

There are several different ways an FTP server and client software can conduct a file transfer using FTP:

- **Anonymous FTP.** This is the most basic form of FTP. It provides support for data transfers without encrypting data or using a username and password. It's most commonly used for download of material that is allowed for unrestricted distribution. It works on port
- **Password-protected FTP.** This is also a basic FTP service, but it requires the use of a username and password, though the service might not be encrypted or secure. It also works on port 21.
- **FTP Secure (FTPS).** Sometimes referred to as FTP Secure Sockets Layer (FTP-SSL), this approach enables implicit Transport Layer Security (TLS) as soon as an FTP connection is established. FTPS was initially used to help enable a more secure form of FTP data transfer. It typically defaults to using port 990.
- **FTP over explicit SSL/TLS (FTPES).** This approach enables explicit TLS support by upgrading an FTP connection over port 21 to an encrypted connection. This is a commonly used approach by web and file sharing services to enable secure file transfers.

- **Secure FTP (SFTP).** This is technically not an FTP protocol, but it functions similarly. Rather, SFTP is a subset of the Secure Shell (SSH) protocol that runs over port 22. SSH is commonly used by systems administrators to remotely and securely access systems and applications, and SFTP provides a mechanism within SSH for secure file transfer.

FTP clients

FTP clients are used to upload, download and manage files on a server. FTP clients include the following:

- **FileZilla.** This is a free FTP client for Windows, macOS and Linux that supports FTP, FTPS and SFTP.
- **Transmit.** This is an FTP client for macOS that supports FTP and SSH.
- **WinSCP.** This is a Windows FTP client that supports FTP, SSH and SFTP.
- **WS_FTP.** This is another Windows FTP client that supports SSH.

INSTALLING FTP SERVICE:

Yum install vsftpd*

Provide access to a folder edit the file /etc/vsftpd/vsftpd.conf

Add the folder into the file

local_root=<dir>

Provide access to only a specific directory edit the file /etc/vsftpd/vsftpd.conf

Find chroot_local_user=yes remove the #

Save the file and reload the service

Systemctl restart vsftpd

Provide ftp access on a specific port edit the file /etc/vsftpd/vsftpd.conf

Add listen_port=<port number>

Restart the service

Secure Shell (SSH)

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 password authentication and public key authentication, as well as encrypted data communications between two computers connecting over an open network, such as the internet.

In addition to providing strong encryption, SSH is widely used by network administrators to manage systems and applications remotely, enabling them to log in to 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, which is the end where the session is displayed, with an SSH server, which is the end where 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, for example, to securely run X Window System graphical sessions remotely. An SSH server, by default, listens on the standard Transmission Control Protocol (TCP) port 22.

How does SSH work?

Secure Shell was created to replace insecure terminal emulation or login programs, such as Telnet, rlogin (remote login) and rsh (remote shell). SSH enables the same functions -- logging in to and running terminal sessions on remote systems. SSH also replaces file transfer programs, such as File Transfer Protocol (FTP) and rcp (remote copy).

The most basic use of SSH is to connect to a remote host for a terminal session. The form of that command is the following:

```
ssh UserName@SSHserver.example.com
```

This command will cause the client to attempt to connect to the server

named *server.example.com*, using the user ID *UserName*. If this is the first time negotiating a connection between the local host and the server, the user will be prompted with the remote host's public key fingerprint and prompted to connect, despite there having been no prior connection:

```
The authenticity of host 'sample.ssh.com' cannot be established.
 DSA key fingerprint is 01:23:45:67:89:ab:cd:ef:ff:fe:dc:ba:98:76:54:32:10.
 Are you sure you want to continue connecting (yes/no)?
```

Answering *yes* to the prompt will cause the session to continue, and the host key is stored in the local system's known_hosts file. This is a hidden file, stored by default in a hidden directory, called /.ssh/known_hosts, in the user's home directory. Once the host key has been stored in the known_hosts file, the client system can connect directly to that server again without need for any approvals; the host key authenticates the connection.

What is SSH used for?

Present in all data centers, SSH ships by default with every Unix, Linux and Mac server. SSH connections have been used to secure many different types of communications between a local machine and a remote host, including secure remote access to resources, remote execution of commands, delivery of software patches, and updates and other administrative or management tasks.

In addition to creating a secure channel between local and remote computers, SSH is used to manage routers, server hardware, virtualization platforms, operating systems (OSes), and inside systems management and file transfer applications.

Secure Shell is used to connect to servers, make changes, perform uploads and exit, either using tools or directly through the terminal. SSH keys can be employed to automate access to servers and often are used in scripts, backup systems and configuration management tools.

Designed to be convenient and work across organizational boundaries,

SSH keys provide single sign-on (SSO) so that users can move between their accounts without typing a password each time.

While playing pivotal roles in identity management and access management, SSH does more than authenticate over an encrypted connection. All SSH traffic is encrypted. Whether users are transferring a file, browsing the web or running a command, their actions are private.

While it is possible to use SSH with an ordinary user ID and password as credentials, SSH relies more often on public key pairs to authenticate hosts to each other. Individual users must still employ their user ID and password -- or other authentication methods -- to connect to the remote host itself, but the local machine and the remote machine authenticate separately to each other. This is accomplished by generating a unique public key pair for each host in the communication. A single session requires two public key pairs: one public key pair to authenticate the remote machine to the local machine and a second public key pair to authenticate the local machine to the remote machine.

Secure Shell capabilities

Functions that SSH enables include the following:

- 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.

SSH can be used interactively to enable terminal sessions and should be used instead of the less secure Telnet program. SSH is also commonly used in scripts and other software to enable programs and systems to

remotely and securely access data and other resources.

Secure Shell security issues

Enterprises using SSH should consider finding ways to manage host keys stored on client systems. These keys can accumulate over time, especially for information technology (IT) staff that needs to be able to access remote hosts for management purposes.

Because the data stored in an SSH known_hosts file can be used to gain authenticated access to remote systems, organizations should be aware of the existence of these files and should have a standard process for retaining control over the files, even after a system is taken out of commission, as the hard drives may have this data stored in plaintext.

Developers should be careful when incorporating SSH commands or functions in a script or other type of program. While it is possible to issue an SSH command that includes a user ID and password to authenticate the user of the local machine to an account on the remote host, doing so may expose the credentials to an attacker with access to the source code.

Shellshock, a security hole in the Bash command processor, can be executed over SSH but is a vulnerability in Bash, not in SSH.

The biggest threat to SSH is poor key management. Without the proper centralized creation, rotation and removal of SSH keys, organizations can lose control over who has access to which resources and when, particularly when SSH is used in automated application-to-application processes.

SSH vs. Telnet

Telnet was one of the first internet application protocols -- the other is FTP. It is used to initiate and maintain a terminal emulation session on a remote host.

SSH and Telnet are functionally similar, with the primary difference being that the SSH protocol uses public key cryptography to authenticate endpoints when setting up a terminal session, as well as for encrypting session commands and output.

While Telnet is primarily used for terminal emulation, SSH can be used to do terminal emulation -- similar to the rlogin command -- as well as for issuing commands remotely as with rsh, transferring files using SSH File Transfer Protocol (SFTP) and tunneling other applications.

SSH vs. SSL/TLS

The Transport Layer Security (TLS) protocol, which updates the Secure Sockets Layer (SSL) protocol, was designed to provide security for network transmissions at the transport layer. The SSH protocol also operates at or just above the transport layer, but there are important differences between the two protocols.

While both rely on public/private key pairs to authenticate hosts, only the server is authenticated with a key pair under TLS. SSH uses a separate key pair to authenticate each connection: one key pair for a connection from a local machine to a remote machine and a second key pair to authenticate the connection from the remote machine to the local machine.

Another difference between SSH and TLS is that TLS enables connections to be encrypted without authentication or authenticated without encryption. SSH encrypts and authenticates all connections.

SSH provides IT and information security (infosec) professionals with a secure mechanism to manage SSH clients remotely. Rather than requiring password authentication to initialize a connection between an SSH client and server, SSH authenticates the devices themselves. This enables IT staff to connect with remote systems and modify SSH configurations, including adding or removing host key pairs in the known_hosts file.

SSH implementations

SSH is an open protocol. It has been implemented for most computing platforms. The open source OpenSSH implementation is the one most commonly found on Linux, Unix and other OSes based on Berkeley Software Distribution (BSD), including Apple's macOS.

OpenSSH was ported to run in Windows PowerShell starting in 2015. In 2018, optional OpenSSH support was added to Windows 10. While SSH is directly accessible by default in most Unix-like OSes, Microsoft's ported version of OpenSSH must be explicitly enabled in the Windows Settings app.

PuTTY is another open source implementation of SSH. While it currently is available for Windows, macOS and Unix/BSD, PuTTY was originally written to run on Windows. It has long been one of the top options for using SSH on a Windows system.

Most implementations of the SSH suite comprise three utilities:

1. slogin (secure login)
2. ssh
3. scp (secure copy)

These are secure versions of the earlier insecure Unix utilities: rlogin, rsh and rcp.

SSH uses public key cryptography to authenticate the remote computer and enables the remote computer to authenticate the user, if necessary.

There are currently dozens of SSH implementations available for various platforms and under a variety of open source and proprietary licenses.

SSH commands

While there are graphical implementations of SSH, the program is

usually invoked at the command line or executed as part of a script. Running the ssh command on its own, with no arguments such as a destination host or user ID, returns a list of SSH command parameters and options.

The most basic form of SSH command is to invoke the program and the destination host name or Internet Protocol (IP) address:

```
ssh server.example.org
```

This will connect to the destination, server.example.org. The destination host will respond by prompting for a password for the user ID of the account under which the client is running. In other words, if the user ID in use is *jsmith*, then the remote host will ask for a password associated with the account *jsmith* on the remote host.

In many cases, the user ID for the remote host will be different, in which case the command should be issued with the remote host user ID, like this:

```
ssh remote_host_userID@server.example.org
```

SSH can also be used from the command line to issue a single command on the remote host and then exit -- for example:

```
ssh example.org ls
```

This command executes the Unix ls command, which lists all contents of the current directory on the remote host. While this example is trivial, it demonstrates that SSH can be used to execute more interesting commands on a remote host. For example, a command can be crafted that initializes a server instance that will give a remote machine access to a single file -- or other resource -- and then terminate the server after the file is accessed by the specified remote host.

In addition to the ssh executable, SSH has other executable commands

used at the command line for additional functions, including the following:

- sshd initiates the SSH server, which waits for incoming SSH connection requests and enables authorized systems to connect to the local host.
- ssh-keygen is a program to create a new authentication key pair for SSH, which can be used to automate logins, to implement SSO and to authenticate hosts.
- ssh-copy-id is a program used to copy, install and configure an SSH key on a server to automate passwordless logins and SSO.
- ssh-agent is a helper program that tracks identity keys and their passphrases -- from which SSH derives an encryption key -- and enables the user to use the identity keys to log in to different servers without the need to reenter passwords or passphrases.
- ssh-add is used to add a key to the SSH authentication agent and is used with ssh-agent to implement SSO using SSH.
- scp is a program used for copying files from one computer to another and is an SSH-secured version of rcp.
- sftp is a program used to copy files from one computer to another and is an SSH-secured version of ftp, the original File Transfer Protocol. SFTP has become the preferred mechanism for file sharing over the internet, replacing both FTP and FTP/S (FTP Secure), which is a protocol for using FTP over an SSL/TLS tunnel.

By default openssh server is installed in the linux server

change the ssh port on linux server

Edit the file ***/etc/ssh/sshd_config***

Remove the comment of port:22 and provide the custom port. Example:

Port:2222, save the file and restart the service

Systemctl restart sshd

Restrict root access for ssh:

Edit the file */etc/ssh/sshd_config*

comment the line43 permitrootlogin yes

#permitrootlogin yes

Authentication with keys:

#ssh-keygen

 # ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.254.145

Linux File System

A Linux file system is a structured collection of files on a disk drive or a partition. A partition is a segment of memory and contains some specific data. A file system is designed in a way so that it can manage and provide space for non-volatile storage data. All file systems required a namespace that is a naming and organizational methodology. The namespace defines the naming process, length of the file name, or a subset of characters that can be used for the file name.

Types of Linux File System:

When we install the Linux operating system, Linux offers many file systems such as

Ext, Ext2, Ext3 and Ext4 file system

The file system Ext stands for **Extended File System**. It was primarily developed for **MINIX OS**. The Ext file system is an older version, and is no longer used due to some limitations.

Ext4 file system is the faster file system among all the Ext file systems. It is a very compatible option for the SSD (solid-state drive) disks, and it is the default file system in Linux distribution.

JFS File System

JFS stands for **Journaled File System**, and it is developed by **IBM for AIX Unix**. It is an alternative to the Ext file system. It can also be used in place of Ext4, where stability is needed with few resources. It is a handy file system when CPU power is limited.

XFS File System

XFS file system was considered as high-speed JFS, which is developed for parallel I/O processing. NASA still using this file system with its high storage server.

SGI released XFS to the open source community in 1999. The community subsequently merged XFS into the kernel of the Linux OS, making the file system available as an option for Linux distributions. XFS supports large files and large file systems. For a 64-bit implementation, XFS can handle file systems of up to 18 exabytes, with a maximum file size of 9 exabytes. There is no limit on the number of files.

XFS is a journaling file system and, as such, keeps track of changes in a log before committing the changes to the main file system. The advantage is guaranteed consistency of the file system and expedited recovery in the event of power failures or system crashes.

Scan a New Disk in CentOS/RHEL Online (Without Reboot)

Sometimes the new disks are not scanned automatically and we need to troubleshoot the same.

'fdisk –l | grep –i disk' command to check whether the new disk detected or not.

```
fdisk –l | grep –i disk
```

The new disk is not visible to the system. Since it is not visible to the system you need to rescan the scsi bus to get the new disk. Rescan can be issued using this command

```
echo "- - -" > /sys/class/scsi_host/host#/scan
```

Replace '**host#**' with actual value such as **host0** or **host1** like this. You can find scsi_host value using the following command:

```
ls /sys/class/scsi_host
```

Now you might be thinking how do I find the right scsi_hostvalue that you need to use with 'echo "- - -" > /sys/class/scsi_host/host#/scan' command. The below command help you to find which is the correct scsi_host value you need to use.

```
grep mpt /sys/class/scsi_host/host?/proc_name
```

Once you have scsi_host value with you now you are ready for scanning the scsi bus.

echo "- - -" > /sys/class/scsi_host/host2/scan

Creating Linux Partitions

The next step is to create one or more Linux partitions on the new disk drive. This is achieved using the fdisk utility which takes as a command-line argument the device to be partitioned:

fdisk /dev/sdb

Welcome to fdisk (util-linux 2.32.1).

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.

Created a new DOS disklabel with disk identifier 0xbd09c991.

Command (m for help):

In order to view the current partitions on the disk enter the p command:

Command (m for help): p

Disk /dev/sdb: 8 GiB, 8589934592 bytes, 16777216 sectors

Units: sectors of 1 * 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disklabel type: dos

Disk identifier: 0xbd09c991

As we can see from the above fdisk output, the disk currently has no partitions because it is a previously unused disk. The next step is to create a new partition on the disk, a task which is performed by entering n (for new partition) and p (for primary partition):

Command (m for help): n

Partition type

p primary (0 primary, 0 extended, 4 free)

e extended (container for logical partitions)

Select (default p): p

Partition number (1-4, default 1):

In this example we only plan to create one partition which will be partition 1. Next we need to specify where the partition will begin and end. Since this is the first partition we need it to start at the first available sector and since we want to use the entire disk we specify the last sector as the end. Note that if you wish to create multiple partitions you can specify the size of each partition by sectors, bytes, kilobytes or megabytes.

Partition number (1-4, default 1): 1

First sector (2048-16777215, default 2048):

Last sector, +sectors or +size{K,M,G,T,P} (2048-16777215, default 16777215):

Created a new partition 1 of type ‘Linux’ and of size 8 GiB.

Command (m for help):

Now that we have specified the partition, we need to write it to the disk using the w command:

Command (m for help): w

The partition table has been altered.

Calling ioctl() to re-read partition table.

Syncing disks.

If we now look at the devices again we will see that the new partition is visible as /dev/sdb1:

ls /dev/sd*

/dev/sda /dev/sda1 /dev/sda2 /dev/sdb /dev/sdb1

The next step is to create a file system on our new partition.

Creating a File System on a RHEL 8 Disk Partition

We now have a new disk installed, it is visible to RHEL 8 and we have configured a Linux partition on the disk. The next step is to create a Linux file system on the partition so that the operating system can use it to store files and data. The easiest way to create a file system on a partition is to use the mkfs.xfs utility:

 # mkfs.xfs /dev/sdb1

In this case we have created an XFS file system. XFS is a high performance file system which is the default filesystem type on RHEL 8 and includes a number of advantages in terms of parallel I/O performance and the use of journaling.

A journaling filesystem keeps a journal or log of the changes that are being made to the filesystem during disk writing that can be used to rapidly reconstruct corruptions that may occur due to events such as a system crash or power outage.

There are a number of advantages to using a journaling file system. Both the size and volume of data stored on disk drives has grown exponentially over the years. The problem with a non-journaled file system is that following a crash the *fsck* (filesystem consistency check) utility has to be run. The *fsck* utility will scan the entire filesystem validating all entries and making sure that blocks are allocated and referenced correctly. If it finds a corrupt entry it will attempt to fix the problem. The issues here are two-fold. First, the *fsck* utility will not always be able to repair damage and you will end up with data in the *lost+found* directory. This is data that was being used by an application but the system no longer knows where it was referenced from. The other problem is the issue of time. It can take a very long time to complete the *fsck* process on a large file system, potentially leading to unacceptable down time.

A journaled file system, on the other hand, records information in a log area on a disk (the journal and log do not need to be on the same device)

during each write. This is a essentially an “intent to commit” data to the filesystem. The amount of information logged is configurable and ranges from not logging anything, to logging what is known as the “metadata” (i.e. ownership, date stamp information etc), to logging the “metadata” and the data blocks that are to be written to the file. Once the log is updated the system then writes the actual data to the appropriate areas of the filesystem and marks an entry in the log to say the data is committed.

After a crash the filesystem can very quickly be brought back on-line using the journal log, thereby reducing what could take minutes using *fsck* to seconds with the added advantage that there is considerably less chance of data loss or corruption.

Mounting a File System:

Now that we have created a new file system on the Linux partition of our new disk drive we need to mount it so that it is accessible and usable. In order to do this we need to create a mount point. A mount point is simply a directory or folder into which the file system will be mounted. For the purposes of this example we will create a /backup directory to match our file system label (although it is not necessary that these values match):

mkdir /backup

The file system may then be manually mounted using the mount command:

mount /dev/sdb1 /backup

Configuring RHEL 8 to Automatically Mount a File System

In order to set up the system so that the new file system is automatically mounted at boot time an entry needs to be added to the /etc/fstab file. The format for an fstab entry is as follows:

<device> <dir> <type> <options> <dump> <fsck>

These entries can be summarized as follows: * <device> - The device on which the filesystem is to be mounted.

* <dir> - The directory that is to act as the mount point for the filesystem.

* <type> - The filesystem type (xfs, ext4 etc.)

* <options> - Additional filesystem mount options, for example making

the filesystem read-only or controlling whether the filesystem can be mounted by any user. Run man mount to review a full list of options. Setting this value to defaults will use the default settings for the filesystem (rw, suid, dev, exec, auto, nouser, async).

* <dump> - Dictates whether the content of the filesystem is to be included in any backups performed by the dump utility. This setting is rarely used and can be disabled with 0 value.

* <fsck> - Whether the filesystem is checked by fsck after a system crash and the order in which filesystems are to be checked. For journaled filesystems such as XFS this should be set to 0 to indicate that the check is not required.

Logical Volume Management

LVM allows for very flexible disk space management. It provides features like the ability to add disk space to a logical volume and its filesystem while that filesystem is mounted and active and it allows for the collection of multiple physical hard drives and partitions into a single volume group which can then be divided into logical volumes.

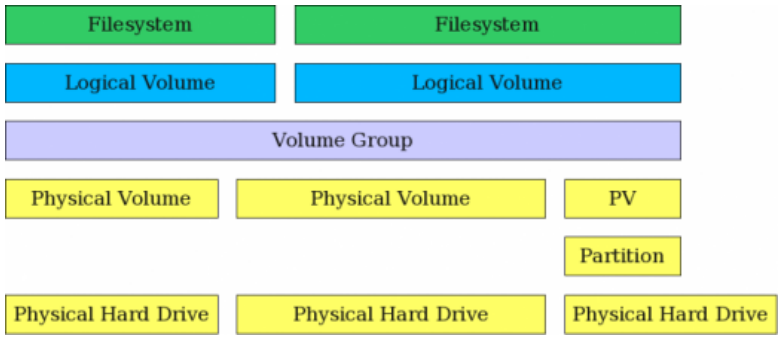
The volume manager also allows reducing the amount of disk space allocated to a logical volume, but there are a couple requirements. First, the volume must be unmounted. Second, the filesystem itself must be reduced in size before the volume on which it resides can be reduced.

It is important to note that the filesystem itself must allow resizing for this feature to work. The EXT2, 3, and 4 filesystems all allow both offline (unmounted) and online (mounted) resizing when increasing the size of a filesystem, and offline resizing when reducing the size. You should check the details of the filesystems you intend to use in order to verify whether they can be resized at all and especially whether they can be resized while online.

LVM Structure

The structure of a Logical Volume Manager disk environment is illustrated by Figure 1, below. Logical Volume Management enables the combining of multiple individual hard drives and/or disk partitions into a single volume group (VG). That volume group can then be subdivided into logical volumes (LV) or used as a single large volume. Regular file systems, such as EXT3 or EXT4, can then be created on a logical volume.

In Figure 1, two complete physical hard drives and one partition from a third hard drive have been combined into a single volume group. Two logical volumes have been created from the space in the volume group, and a filesystem, such as an EXT3 or EXT4 filesystem has been created on each of the two logical volumes.



Adding disk space to a host is fairly straightforward but, in my experience, is done relatively infrequently. The basic steps needed are listed below. You can either create an entirely new volume group or you can add the new space to an existing volume group and either expand an existing logical volume or create a new one.

Adding a new logical volume

There are times when it is necessary to add a new logical volume to a host.

The basic steps for adding a new logical volume are as follows.

1. If necessary, install a new hard drive.
2. Optional: Create a partition on the hard drive.
3. Create a physical volume (PV) of the complete hard drive or a partition on the hard drive.
4. Assign the new physical volume to an existing volume group (VG) or create a new volume group.
5. Create a new logical volumes (LV) from the space in the volume group.
6. Create a filesystem on the new logical volume.
7. Add appropriate entries to /etc/fstab for mounting the filesystem.
8. Mount the filesystem.

Now for the details. The following sequence is taken from an example I used as a lab project when teaching about Linux filesystems.

To create physical volumes on top of **/dev/sdb**, **/dev/sdc**, and **/dev/sdd**, do:

pvcreate /dev/sdb /dev/sdc /dev/sdd

You can list the newly created PVs with:

 # pvs

and get detailed information about each PV with:

*# **pvdisplay /dev/sdX X= sdb,sdc,sdd,sde***

To create a volume group named vg00 using /dev/sdb and /dev/sdc

vgcreate vg00 /dev/sdb /dev/sdc

As it was the case with physical volumes, you can also view information about this volume group by issuing:

 # vgdisplay vg00

When it comes to creating logical volumes, the distribution of space must take into consideration both current and future needs. It is considered good practice to name each logical volume according to its intended use.

For example, let's create two LVs named vol_projects (10 GB) and vol_backups (remaining space), which we can use later to store project documentation and system backups, respectively.

The -n option is used to indicate a name for the LV, whereas -L sets a fixed size and -l (lowercase L) is used to indicate a percentage of the remaining space in the container VG.

 # lvcreate -n vol_projects -L 10G vg00

 # lvcreate -n vol_backups -l 100%FREE vg00

As before, you can view the list of LVs and basic information with:

 # lvs

and detailed information with

 # lvdisplay

To view information about a single LV, use lvdisplay with the VG and LV as parameters, as follows:

 # lvdisplay vg00/vol_projects

In the image above we can see that the LVs were created as storage devices (refer to the LV Path line). Before each logical volume can be used, we need to create a filesystem on top of it.

We'll use ext4 as an example here since it allows us both to increase and reduce the size of each LV (as opposed to xfs that only allows to increase the size):

 # mkfs.ext4 /dev/vg00/vol_projects

 # mkfs.ext4 /dev/vg00/vol_backups

Resizing Logical Volumes and Extending Volume Groups

Now picture the following scenario. You are starting to run out of space in vol_backups, while you have plenty of space available in vol_projects. Due to the nature of LVM, we can easily reduce the size of the latter (say 2.5 GB) and allocate it for the former, while resizing each filesystem at the same time.

Fortunately, this is as easy as doing:

 # lvreduce -L -2.5G -r /dev/vg00/vol_projects

 # lvextend -l +100%FREE -r /dev/vg00/vol_backups

It is important to include the minus (-) or plus (+) signs while resizing a logical volume. Otherwise, you're setting a fixed size for the LV instead of resizing it.

It can happen that you arrive at a point when resizing logical volumes cannot solve your storage needs anymore and you need to buy an extra storage device. Keeping it simple, you will need another disk. We are going to simulate this situation by adding the remaining PV from our initial setup (/dev/sdd).

To add /dev/sdd to vg00, do

 # vgextend vg00 /dev/sdd

If you run vgdisplay vg00 before and after the previous command, you will see the increase in the size of the VG:

#***vgdisplay vg00***

Now you can use the newly added space to resize the existing LVs according to your needs, or to create additional ones as needed.

Mounting Logical Volumes on Boot and on Demand

Of course there would be no point in creating logical volumes if we are not going to actually use them! To better identify a logical volume we will need to find out what its UUID (a non-changing attribute that uniquely identifies a formatted storage device) is.

To do that, use blkid followed by the path to each device:

 # blkid /dev/vg00/vol_projects

 # blkid /dev/vg00/vol_backups

Create mount points for each LV:

 # mkdir /home/projects

 # mkdir /home/backups

and insert the corresponding entries in /etc/fstab (make sure to use the UUIDs obtained before):

Then save the changes and mount the LVs:

 # mount -a

 # mount | grep home

When you deal with LVM you need to be extremely careful while performing deletion/removal operations. There will be changes when you no longer require a logical volume and want to remove the same.

Things that needs to be taken care of before proceeding with removal:

Make sure its a non-root partition

Please take a backup of the data before proceeding

Make sure the volume is unmounted

Step 1: Delete entry from /etc/fstab

cat /etc/fstab

Step 2: unmount the partition

umount /data

Step 3: Disable LVM

lvchange -an /dev/CVOL/workspace

Step 4: Delete LVM volume

lvremove /dev/CVOL/workspace

Step 5: Disable volume group

vgremove CVOL

Step 6: Delete physical volumes used for volume group “vg”

pvremove /dev/sda4

Network File System (NFS)

Network File System (**NFS**) also known as client/server file system is a popular, cross-platform and distributed file system protocol used to export local file systems over the network so that clients can share directories and files with others over a network and interact with them as though they are mounted locally.

Setting Up NFS Server:

First, start by installing the required packages on the **NFS** server. The packages are **nfs-utils** which provides a daemon for the kernel NFS server and related tools such as the contains the **showmount** program. Run the following command to install the package on the **NFS** server.
dnf install nfs-utils

Once the installation is complete, start the nfs-server service, enable it to automatically start at system boot, and then verify its status using the systemctl commands.

 # systemctl start nfs-server.service

 # systemctl enable nfs-server.service

 # systemctl status nfs-server.service

Note that the other services that are required for running an NFS server or mounting NFS shares such as nfsd, nfs-idmapd, rpcbind, rpc.mountd, lockd, rpc.statd, rpc.rquotad, and rpc.idmapd will be automatically started.

The configuration files for the NFS server are:

/etc/nfs.conf – main configuration file for the NFS daemons and tools.

/etc/nfsmount.conf – an NFS mount configuration file.

Next, create the file systems to export or share on the **NFS** server. For this guide, we will create four file systems, three of which are used by staff from three departments: human resource, finance and marketing to share files and one is for root user backups.

 # mkdir -p /mnt/nfs_shares/{Human_Resource,Finance,Marketing}

 # mkdir -p /mnt/backups

 # ls -l /mnt/nfs_shares/

Then export the above file systems in the NFS server /etc/exports configuration file to determine local physical file systems that are accessible to NFS clients.

/mnt/nfs_shares/Human_Resource 192.168.254.0/24(rw,sync)

/mnt/nfs_shares/Finance 192.168.254.0/24(rw,sync)

/mnt/nfs_shares/Marketing 192.168.254.0/24(rw,sync)

/mnt/backups <client ip>/24(rw,sync,no_all_squash,root_squash)

Here are some of the exports options (read man exports for more information and export options):

rw – allows both read and write access on the file system.

sync – tells the NFS server to write operations (writing information to the disk) when requested (applies by default).

all_squash – maps all UIDs and GIDs from client requests to the anonymous user.

no_all_squash – used to map all UIDs and GIDs from client requests to identical UIDs and GIDs on the NFS server.

root_squash – maps requests from root user or UID/GID 0 from the client to the anonymous UID/GID.

To export the above file system, run the exportfs command with the -a flag means export or unexport all directories, -r means reexport all directories, synchronizing /var/lib/nfs/etab with /etc/exports and files under /etc/exports.d, and -v enables verbose output.

exportfs -arv

To display the current export list, run the following command. Note that the exports table also applies some of the default exports options that are not explicitly defined as shown in the following screenshot.

 # exportfs -s

Next, if you have the firewalld service running, you need to allow traffic to the necessary NFS services (mountd, nfs, rpc-bind) via the firewall, then reload the firewall rules to apply the changes, as follows.

 # firewall-cmd --permanent --add-service=nfs

 # firewall-cmd --permanent --add-service=rpc-bind

 # firewall-cmd --permanent --add-service=mountd

 # firewall-cmd –reload

Setting Up NFS Client on Client Systems

Now on the client node(s), install the necessary packages to access NFS shares on the client systems. Run the appropriate command for your distribution:

 # dnf install nfs-utils nfs4-acl-tools

Then run the showmount command to show mount information for the NFS server. The command should output the exported file system on the client as shown in the screenshot.

 # showmount -e <server ip>

Next, create a local file system/directory for mounting the remote NFS file system and mount it as an ntf file system.

 # mkdir -p /mnt/backups

 # mount -t nfs -o sync 192.168.254.154:/mnt/backups /mnt/backups

Then confirm that the remote file system has been mounted by running the mount command and filter nfs mounts.

 # mount | grep nfs

To enable the mount to persistent even after a system reboot, run the following command to enter the appropriate entry in the /etc/fstab.

 # echo "192.168.254.154:/mnt/backups /mnt/backups nfs defaults 0 0">>/etc/fstab

 # cat /etc/fstab

Lastly, test if NFS setup is working fine by creating a file on the server and check if the file can be seen in the client.

 # touch /mnt/backups/file_created_on_server.text

Install Samba4 on RHEL 8 for File Sharing on Windows

Samba is an open source, fast, secure, stable and widely-used network file system that provides file sharing and print services for all clients using the **SMB/CIFS** protocol, such as Linux, all versions of DOS and Windows, OS/2, and so many other operating systems.

To install the **Samba 4** along with its dependencies use the DNF package manager as shown.

 # dnf install samba samba-client samba-common

Once the installation is complete, start the Sambe service, enable it to auto-start at system boot time and verify that service using the systemctl commands as follows.

systemctl enable smb

systemctl enable nmb

systemctl start smb

systemctl start nmb

systemctl status nmb

systemctl status smb

Next, if you have a firewalld configured, you need to add the Samba service in the firewall configuration to allow access to shared directories and files through system.

sudo firewall-cmd --permanent --add-service=samba

sudo firewall-cmd –reload

In this section, the first step is to create the shared directory which will store files on the server. Then define the appropriate permissions on the directory as shown.

mkdir /etc/samba/linuxserver

useradd sambaser -s /sbin/nologin

smbpasswd -a sambaser

Now open the configuration file using your favorite text-based file editor to configure the anonymous unsecured file sharing on a shared directory.

 # nano /etc/samba/smb.conf

Modify the following global parameters and add a section for the Anonymous share. Note that you can set your own values where necessary (read man smb.conf for more information).

[share]

comment = share for windows client

path = /etc/samba/linuxserver

browseable = yes

vaild users = sambaser

writable = yes

create mask = 0664

directory mask = 0775

Save the changes in the file and close.

Change the ownership for the user

chown sambaser:root /etc/samba/linuxserver/

Finally, test if the share is working fine, log into your Windows machine, open the Windows Explorer, click on Network, then click on the RHEL host, or use the server IP address to access it (running ip add command on the server can help you to view the IP address).