

➤ **Introduction to Linux:**

Linux is a Unix-like computer operating system assembled under the model of free and open source software development and distribution. The defining component of Linux is the Linux kernel, an operating system kernel first released 5 October 1991 by Linus Torvalds.

Linux was originally developed as a free operating system for Intel x86-based personal computers. It has since been ported to more computer hardware platforms than any other operating system. It is a leading operating system on servers and other big iron systems such as mainframe computers and supercomputers more than 90% of today's 500 fastest supercomputers run some variant of Linux, including the 10 fastest. Linux also runs on embedded systems (devices where the operating system is typically built into the firmware and highly tailored to the system) such as mobile phones, tablet computers, network routers, televisions and video game consoles; the Android system in wide use on mobile devices is built on the Linux kernel.

➤ **Basic Features :**

Following are some of the important features of Linux Operating System.

- **Portable** - Portability means software's can work on different types of hardware's in the same way. Linux kernel and application programs support their installation on any kind of hardware platform.
- **Open Source** - Linux source code is freely available and it is a community based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.

- **Multi-User** - Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
- **Multiprogramming** - Linux is a multiprogramming system means multiple applications can run at same time.
- **Hierarchical File System** - Linux provides a standard file structure in which system files/ user files are arranged.
- **Shell** - Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs etc.
- **Security** - Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

➤ **Linux Advantages :**

1. **Low cost:** You don't need to spend time and money to obtain licenses since Linux and much of its software come with the GNU General Public License. You can start to work immediately without worrying that your software may stop working anytime because the free trial version expires. Additionally, there are large repositories from which you can freely download high quality software for almost any task you can think of.
2. **Stability:** Linux doesn't need to be rebooted periodically to maintain performance levels. It doesn't freeze up or slow down over time due to memory leaks and such. Continuous up- times of hundreds of days (up to a year or more) are not uncommon.
3. **Performance:** Linux provides persistent high performance on workstations and on networks. It can handle unusually large numbers of

users simultaneously, and can make old computers sufficiently responsive to be useful again.

4. Network friendliness: Linux was developed by a group of programmers over the Internet and has therefore strong support for network functionality; client and server systems can be easily set up on any computer running Linux. It can perform tasks such as network backups faster and more reliably than alternative systems.

5. Flexibility: Linux can be used for high performance server applications, desktop applications, and embedded systems. You can save disk space by only installing the components needed for a particular use. You can restrict the use of specific computers by installing for example only selected office applications instead of the whole suite.

6. Compatibility: It runs all common UNIX software packages and can process all common file formats.

7. Choice: The large number of Linux distributions gives you a choice. Each distribution is developed and supported by a different organization. You can pick the one you like best; the core functionalities are the same; most software runs on most distributions.

8. Fast and easy installation: Most Linux distributions come with user-friendly installation and setup programs. Popular Linux distributions come with tools that make installation of additional software very user friendly as well.

9. Full use of hard disk: Linux continues work well even when the hard disk is almost full.

10. Multi-tasking: Linux is designed to do many things at the same time; e.g., a large printing job in the background won't slow down your other work.

11. Security: Linux is one of the most secure operating systems. File access permission systems prevent access by unwanted visitors or viruses. Linux users have to option to select and safely download software, free of charge, from online repositories containing thousands of high quality packages. No purchase transactions requiring credit card numbers or other sensitive personal information are necessary.

12. Open Source: If you develop software that requires knowledge or modification of the operating system code, LINUX's source code is at your fingertips. Most Linux applications are Open Source as well.

➤ **Difference between UNIX and LINUX :**

Comparison	Linux	Unix
Definition	It is an open-source operating system which is <i>freely available to everyone</i> .	It is an operating system which <i>can be only used by its copyrighters</i> .
Examples	It has different distros like Ubuntu, Redhat, Fedora, etc	IBM AIX, HP-UX and Sun Solaris.
Users	Nowadays, Linux is in great demand. Anyone can use Linux whether a home user, developer or a student.	It was developed mainly for servers, workstations and mainframes.
Usage	Linux is used everywhere from servers, PC, smartphones, tablets to mainframes and supercomputers.	It is used in servers, workstations and PCs.
Cost	Linux is freely distributed,downloaded, and distributed through magazines also. And priced distros of Linux are also cheaper than Windows.	Unix copyright vendors decide different costs for their respective Unix Operating systems.
Development	As it is open source, it is developed by sharing and collaboration of codes by world-wide developers.	Unix was developed by AT&T Labs, various commercial vendors and non-profit organizations.
Manufacturer	Linux kernel is developed by the community of developers from different parts of the world. Although the father of Linux, Linus Torvalds oversees things.	Unix has three distributions IBM AIX, HP-UX and Sun Solaris. Apple also uses Unix to make OSX operating system.
GUI	Linux is command based but some distros provide GUI based Linux. Gnome and KDE are mostly used GUI.	Initially it was command based OS, but later Common Desktop Environment was created. Most Unix distributions use Gnome.
Interface	The default interface is BASH (Bourne Again SHell). But some distros have developed their own interfaces.	It originally used Bourne shell. But is also compatible with other GUIs.
File system support	Linux supports more file system than Unix.	It also supports file system but lesser than Linux.
Coding	Linux is a Unix clone,behaves like Unix but doesn't contain its code.	Unix contain a completely different coding developed by AT&T Labs.
Operating system	Linux is just the kernel.	Unix is a complete package of Operating system.
Security	It provides higher security. Linux has about 60-100 viruses listed till date.	Unix is also highly secured. It has about 85-120 viruses listed till date

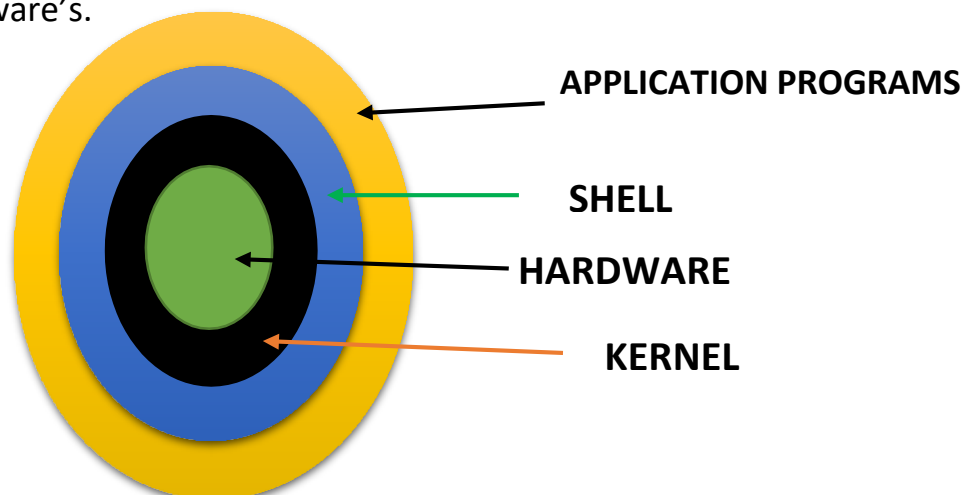
➤ **Advantages of Linux**

- **Open source:-** Linux is an Open source operating systems. You can easily get the source code for Linux and edit it to develop your personal operating system. Today, Linux is widely used for both basic home and office uses. It is the main operating system used for high performance business and in web servers. Linux has made a high impact in this world.
- **Low cost:-** There is no need to spend time and huge amount 'money to obtain licenses since Linux add much of its software come with the GNU General Public License. There is no need to worry about any software's that you use in Linux.
- **Stability:-** Linux has high stability compared with other operating systems. There is no need to reboot the Linux system to maintain performance levels. Rarely it freeze up or slow down. It has continuous up-times of hundreds of days or more.
- **Performance:-** Linux provides high performance on various networks. It has the ability to handle large numbers of users simultaneously.
- **Flexibility:-**Linux is very flexible, Linux can be used for high performance server applications, desktop applications, and embedded systems. You can install only the needed components for a particular use. You can also restrict the use of specific computers.
- **Compatibility:-** It runs all common Unix software packages and can process all common file formats.
- **Security:** Linux is one of the most secure operating systems. File ownership and permissions make Linux more secure.

- **Networking:-** Linux provides a strong support for network functionality; client and server systems can be easily set up on any computer running Linux. It can perform tasks like network backup faster than other operating systems.
- **Multitasking:-** Linux is a multitasking operating system. It can handle many things at the same time.
- **Fast and easy installation:-** Linux distributions come with user-friendly installation.
- **Better use of hard disk:-** Linux uses its resources well enough even when the hard disk is almost full.
- **Wider Choice:-** There are a large number of Linux distributions which gives you a wider choice. Each organization develops and support different distribution.

Architecture of Linux operating system

The architecture of Linux is composed of kernel, shell and application programs that is software's.



Application programs : An application, or application program, is a software program that runs on your computer. It is excited by user. Some inbuilt application programs in Linux are terminal, Firefox browser, Libre office

HARDWARE: physical parts of a computer, such as central processing unit (CPU), monitor, mouse, keyboard, hard disk and other connected devices to CPU.

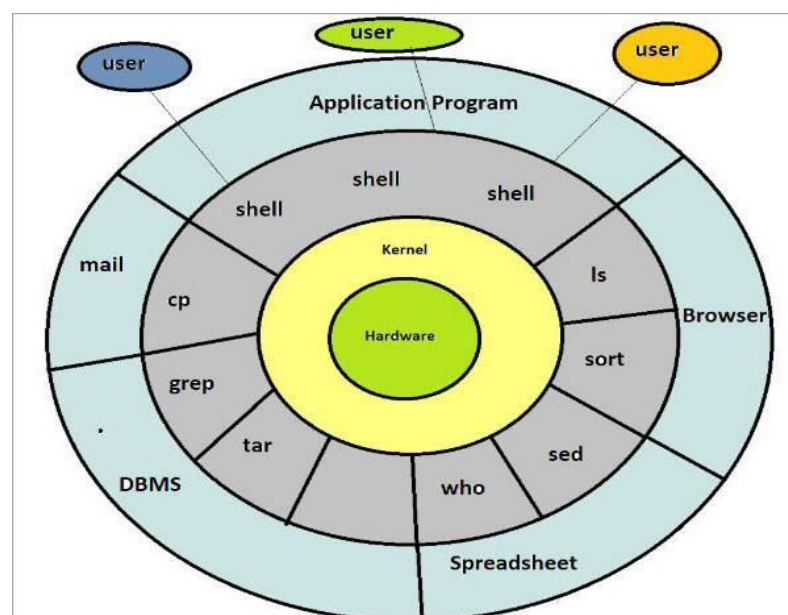
KERNEL: A kernel is a computer program and is the central, core part of an operating system. It manages the operations of the computer and the hardware, most notably memory and CPU time. It is an integral part of any operating system.

SHELL: Shell is an environment in which we can run our commands, programs, and shell scripts. It is a user interface for access to an operating system's services. (User interface program execution, file system manipulation, input/output operations, communication, resource allocation, error detection, security and protection).

The diagram of kernel shell user relationship is as below:

THE KERNEL

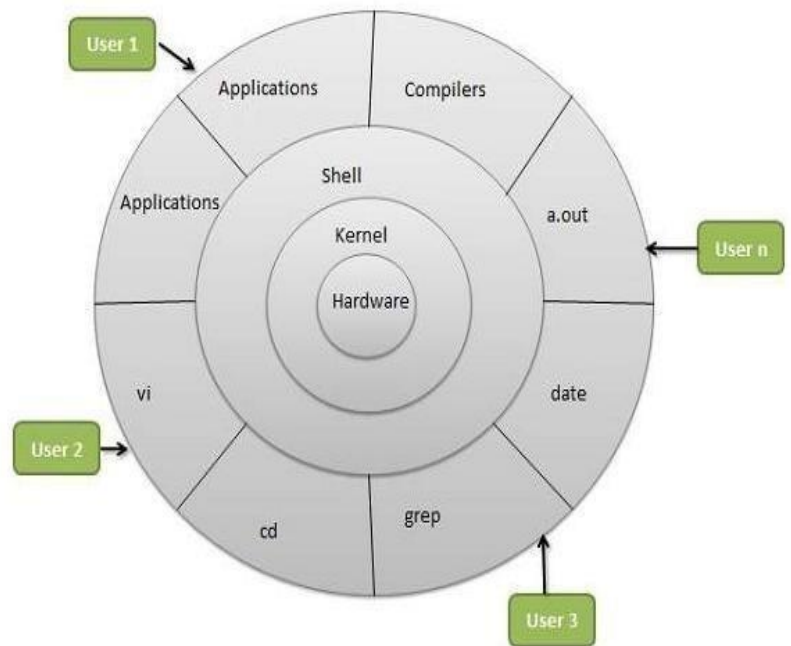
1. Kernel is core (main) part of Linux operating system.
2. It is collection of routine communicate with hardware



directly.

3. It loads into memory when Linux is booted.
4. Kernel provides support to user programs through system call.
5. Kernel manages Computer memory, schedules processes, decides priorities of processes and performs other tasks.
6. Kernel does lot of work even if no application software is running.
7. Hence kernel often called as application software gateway to the computer resources.
8. Kernel is represented by /boot/vmlinuz.

Architecture



SHELL:

1. Shell is interface between user and kernel.
2. It is outer part of operating system.
3. A shell is a user interface for access to an operating system's services
Shell is an environment in which we can run our commands, programs, software's and shell scripts.
4. Computers do not have any inherent (मूळची) capability (क्षमता) of translating commands into actions, it is done by Shell.
5. There can be many shells in action - one shell for each user who logged in

How Shell works?

When we enter commands through the keyboard, it gathers input from you and executes programs based on that input. When a program finishes executing, it displays that program's output. **OR** the shell thoroughly examines the keyboard input for special characters. If it finds any, it rebuilds a simplified command-line, and finally communicate with the Kernel to see that the command is executed.

To know the running shell, use echo \$SHELL command in terminal.

Linux Distribution (Operating System) Names :

1. Redhat Enterprise Linux
2. Fedora Linux
3. Debian Linux
4. Suse Enterprise Linux
5. Ubuntu Linux

Common things between Linux & UNIX :

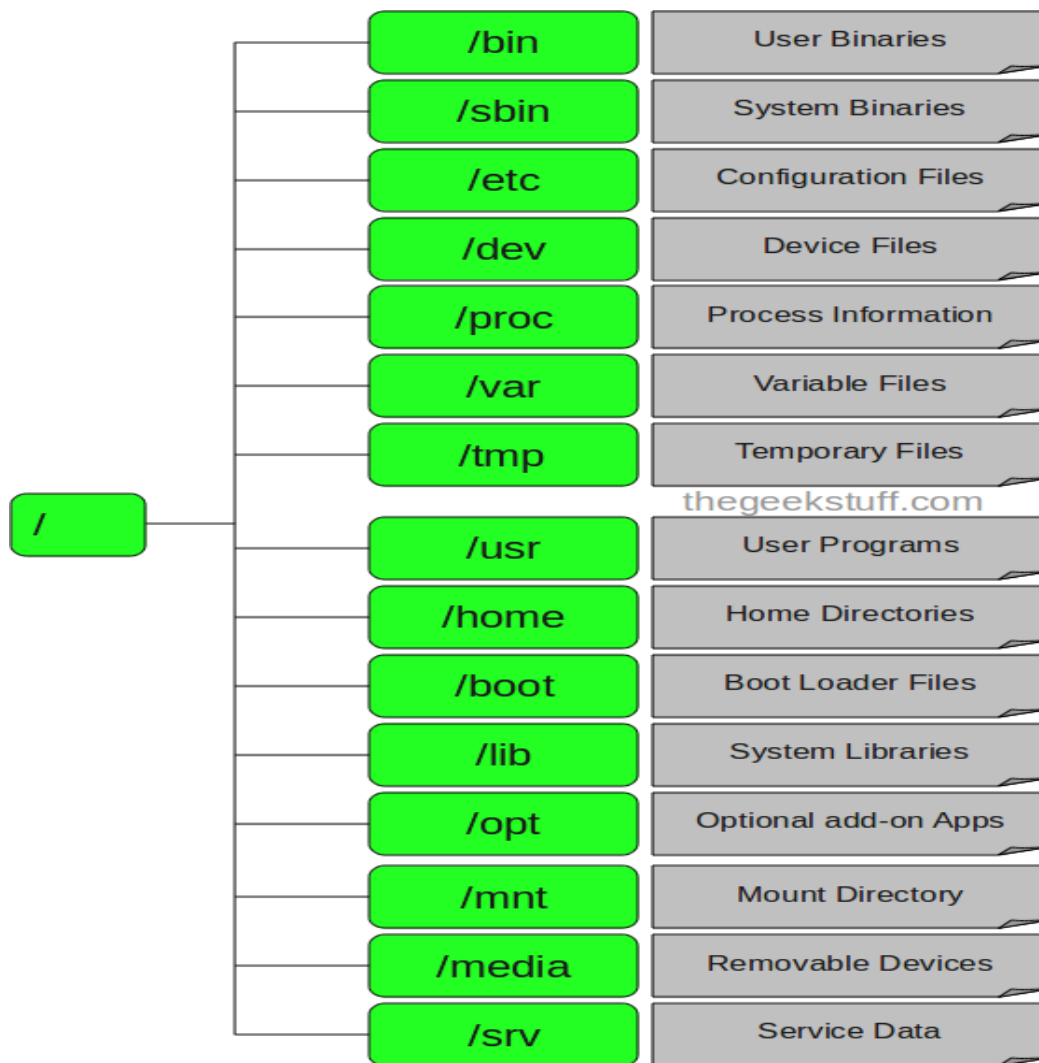
Both share many common applications such as:

1. GUI, file, and windows managers (KDE, Gnome)
2. Shells (ksh, csh, bash)
3. Various office applications such as OpenOffice.org
4. Development tools (perl, php, python, GNU c/c++ compilers)
5. Posix interface

➤ LINUX File system:

Linux file structure files are grouped according to purpose.

Ex: commands, data files, documentation. Parts of a Unix directory tree are listed below. All directories are grouped under the root entry "/". That part of the directory tree is left out of the below diagram.



1. / - Root :

Every single file and directory starts from the root directory.

Only root user has write privilege under this directory.

Please note that /root is root user's home directory, which is not same as /.

2. /bin - User Binaries :

Contains binary executables.

Common linux commands you need to use in single-user modes are located under this directory.

Commands used by all the users of the system are located here.

For example: ps, ls, ping, grep, cp.

3. /sbin - System Binaries :

Just like /bin, /sbin also contains binary executables.

But, the linux commands located under this directory are used typically by system administrator, for system maintenance purpose.

For example: iptables, reboot, fdisk, ifconfig, swapon

4. /etc – Configuration Files :

Contains configuration files required by all programs.

This also contains startup and shutdown shell scripts used to start/stop individual programs.

For example: /etc/resolv.conf, /etc/logrotate.conf

5. /dev – Device Files :

Contains device files.

These include terminal devices, usb, or any device attached to the system.

For example: /dev/tty1, /dev/usbmon0

6. /proc – Process Information

Contains information about system process.

This is a pseudo filesystem contains information about running process. For example:

/proc/{pid} directory contains information about the process with that particular pid.

This is a virtual filesystem with text information about system resources. For example:

/proc/uptime

7. /var – Variable Files

var stands for variable files.

Content of the files that are expected to grow can be found under this directory.

This includes — system log files (/var/log); packages and database files (/var/lib); emails (/var/mail); print queues (/var/spool); lock files (/var/lock); temp files needed across reboots (/var/tmp);

8. /tmp – Temporary Files

Directory that contains temporary files created by system and users.

Files under this directory are deleted when system is rebooted.

9. /usr – User Programs

Contains binaries, libraries, documentation, and source-code for second level programs.

/usr/bin contains binary files for user programs. If you can't find a user binary under

/bin, look under /usr/bin. For example: at, awk, cc, less, scp

/usr/sbin contains binary files for system administrators. If you can't find a system

binary under /sbin, look under /usr/sbin. For example: atd, cron, sshd, useradd, userdel

/usr/lib contains libraries for /usr/bin and /usr/sbin

/usr/local contains users programs that you install from source. For example, when you install apache from source, it goes under /usr/local/apache2

10. /home – Home Directories

Home directories for all users to store their personal files.

For example: /home/john, /home/nikita

11. /boot – Boot Loader Files

Contains boot loader related files.

Kernel initrd, vmlinuz, grub files are located under /boot

For example: initrd.img-2.6.32-24-generic, vmlinuz-2.6.32-24-generic

12. /lib – System Libraries

Contains library files that supports the binaries located under /bin and /sbin

Library filenames are either ld* or lib*.so.*

For example: ld-2.11.1.so, libncurses.so.5.7

13. /opt – Optional add-on Applications

opt stands for optional.

Contains add-on applications from individual vendors.

add-on applications should be installed under either /opt/ or /opt/sub-directory.

14. /mnt – Mount Directory

Temporary mount directory where sysadmins can mount filesystems.

15. /media – Removable Media Devices

Temporary mount directory for removable devices.

➤ Files in Linux

The Linux philosophy is “everything is a file” .

The term “file” not only means a file of data (the typical meaning), it can also refer to an input device (such as a scanner), an output device (such as the monitor), a hardware component (such as the hard drive), or a process (such as the shell that you work with) .

Linux divides files into 7 different types:

1. regular file: a file of data. Data can be text (human readable) or binary (machine readable)

2. directory: a file that can “contain” other files (equivalent to folders in Windows)

3. character special file: IO device that processes one character at a time, such as a printer

4. block special file: IO device that processes a block of characters at a time, such as a hard disk

5. symbolic link: a file that holds the address of another file

➤ Getting Started with Linux Command Lines :

A command is a program that tells the Linux system to do something. It has the form:

\$ command [options] [arguments]

where an argument indicates on what the command is to perform its action, usually a file or series of files. An option modifies the command, changing the way it performs.

Commands are case sensitive. command and Command are not the same.

Options are generally preceded by a hyphen (-), and for most commands, more than one option can be strung together, in the form:

\$ command -[option][option][option]

e.g.:

\$ ls -lart

For most commands you can separate the options, preceding each with a hyphen, e.g.:

command -option1 -option2 -option3

➤ **Basic Linux Commands :**

pwd (Present Working Directory) :

'pwd' command is used to show which directory the terminal is currently working in

\$ pwd

Prints the path to the current working directory. Tells you the folder the terminal is currently working in .

who :

\$who -> display login user name terminal and time

whoami :

\$whoami -> display login user name only

date :

\$ date – display time and date

Ex \$date +%H:%M%S -> 11:44:54
%S -> seconds

%M	→ minutes
%H	→ hours
%T	→ hh:mm:ss
%D	→ mm/dd/yy
%d	→ day value
%m	→ month value
%y	→ yy year value
%Y	→ yyyy year value

cal :

cal command is a calendar command in Linux which is used to see the calendar of a specific month or a whole year.

\$ cal

exit :

exit command in linux is used to exit the shell where it is currently running.

\$ exit

➤ **Displaying Files and Directories :**

ls (Listing Files & Directories) :

The ls command takes the location of a directory and prints all the files and directories within the location. It can also print additional file information like file permissions, file ownership, file size, etc.

Command structure / Syntax:

This is the command structure that all ls commands must follow.

\$ ls [-options] [directory]

If no directory is specified, then ls performs its action on the current directory

The ls command without any options lists files and directories in a plain format without displaying much information like file types, permissions, modified date and time etc.

```
$ ls
```

Lets see options used with ls command

a) ls -t : Open last file edited

It lists the files in order of the time when they were last modified (newest first) rather than in alphabetical order.

```
$ ls -t
```

b) ls -a : Display the hidden files

The ls command will not show hidden files. In Linux, a hidden file is any file that begins with a dot (.).

To display all files including the hidden files use the -a option

```
$ ls -a
```

c) ls -d */ : Display only directories

If you want to display on directories present in our current location use command syntax as

```
$ ls -d */
```

d) ls -R : Displays directories and subdirectories under them

The -R option tells the ls command to display the contents of the directories and subdirectories recursively

```
$ ls -R
```

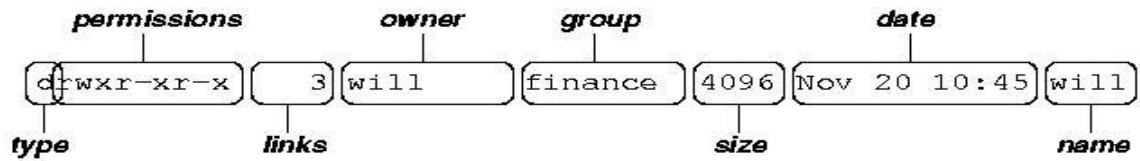
e) ls -p | grep -v / : Display only files

If we want to display only files present in our current location use syntax as

```
$ ls -p | grep -v /
```

f) ls -l : Display all information about files/Directories

The "ls -l" option displays the contents of the current directory in a long listing format, one per line. The line begin with the file or directory permission, owner and group name, file size, created/modified date and time, file/folder name as some of the attributes. Output will be as follows



- File type- '-' normal file
'd' directory
's' socket file
'l' link file
- File permissions
- Number of links
- Owner
- Group
- Size
- Last modified date & time
- File name

g) ls -i : Display the inode(index node) number :

Inode is a Index Node and it is a unique identification number for each file and directories being created in linux. The 'ls -i' option will list the index (called inode) number of each file and directory.

```
$ ls -i
```

h) ls -h : Human-readable format:

It is used to display the file or directory sizes in human readable format and it can used in combination with -s , -t options.

```
$ ls -h
```

```
$ ls -sh
```

```
$ ls -lh
```

i) ls -s : Display sizes of all files and directories :

It is used to display the sizes of each files and directories

```
$ ls -s
```

j) ls -1 : Display each files and directories in single line :

It will display all files and directories in single line format
\$ ls -1

k) ls -r : Reverse sort order :

It is used to list files in reverse order while sorting. It most useful when associated with the -t option to have the least recently modified file displayed first

\$ ls -r

\$ ls -rt

- Options combinely used with ls command
 - ls -lh : Display file size in human readable format i.e. MB or KB
 - ls -ld : Display directory information
 - ls -lt : Order files based on last modified time
 - ls -ltr: Order files based on last modified time in reverse order
 - ls -lS : Sort files by size
- Important options used with ls command on daily basics

ls -a , ls -t , ls -s , ls -i , ls -l , ls -r , ls -h , ls -1 , ls -lart , ls -ltr

➤ Working with Directories :

cd (Change Directory) :

'cd' command is used to navigate between directories in Linux. It changes the current working directory to the desired directory that you wish to navigate to.

The syntax for cd command is following:

\$ cd [option] [directory]

a. Switch to root directory:

It is the parent directory to all other directories present in the file system. It is denoted by /. You can navigate to the root directory from any other directory by using the following command.

```
$ cd /
```

b. Switch to directory path :

Changes your current working directory to mentioned path. The path can be an absolute or a relative .

```
$ cd /home/akshay/downloads/work
```

OR

```
$ cd downloads/work
```

The above both commands meaning are same but path representation are different its an Absolute and Relative path

1) Absolute Path :

Absolute path always starts from root(/) directory

Ex. /home/akshay/Downloads/Work

(The path start with / denotes the root location)

Consider a user name 'akshay' created a directory named 'work' in his 'home' location , then the absolute path is

```
/home/akshay/work
```

2) Relative Path :

Relative path starts from current directory. When we change the directory , relative path also changes

Ex.

If you are in the 'home' directory & you want to access 'work' directory then relative path will be

```
/akshay/work
```

c. To go up to the directory one level above :

To navigate to the directory which is one level above the directory you can use the following command.

```
$ cd ..
```

d. Switch back to the home directory :

~(tilde operator) is used to navigate back to the home directory from any other directory.

```
$ cd ~
```

mkdir (make directory):

mkdir command is used to create one or more directories.

SYNTAX:

mkdir [options] directories

OPTIONS:

-m Set the access mode for the new directories.
-p Create child directories under parent directory (Directory hierarchy)

EXAMPLE:

1. Create single directory:
\$ mkdir mydir

The above command is used to create the directory 'mydir'.

2. Create multiple directories :
\$ mkdir dir dir1 dir2 dir3

3. Create directory and set permissions:
\$ mkdir -m 666 mydir

The above command is used to create the directory 'mydir' and set the read and write permission.

\$ mkdir -m a=rwx mydir

4. Create directory in a specific location
\$ mkdir /home/username/work/dirname

\$ mkdir ~/dirname

rm (Remove Directory):

Used for removes/deletes directory only if is empty.

Example:

```
$rm -r mydir
```

A directory needs to be empty before you can remove it. If it's not, you need to remove the files first. Also, you can't remove a directory if it is your present working directory; you must first change out of it.

rm - remove files and directories :

(for directories -r option is required)

Syntax :

```
$ rm -options File/Directory
```

Options :

- r -> recursively remove all files and directories from specified directory
- R -> same as -r option
- i -> interactive deletion ,ask permission for deletion of file/directories (yes/no)
- f -> forcefully delete
- v -> verbose information displayed
- d -> delete empty directory

Example :

1. Removes files

```
$ rm filename
```

```
$ rm file1 file2 file3
```

```
$ rm *.txt (removes all .txt files from current location)
```

2. Delete files interactively

```
$ rm -i filename
```

3. Remove directories (delete all files & sub-directories under them)

```
$ rm -r dirname
```

4. Removes/Delete write protected files/directory

```
$ rm -f filename
```

```
$ rm -rf dirname
```

5. Remove empty directory (same as 'rmdir')

```
$ rm -d dirname
```

```
$ rm -d dir dir1 dir2
```

6. Display verbose information

```
$ rm -v filename
```

7. Deleting a directory recursively & interactively

```
$ rm -ri dirname
```

➤ **Working with Files :**

Cat (Concatenation) command :

The cat (short for “concatenate”) command is one of the most frequently used commands in Linux. cat command allows us to create single or multiple files, view content of a file, concatenate files and redirect output in terminal or files ,append files , copy file contents etc.

Syntax

```
cat [options] [file]
```

Options :

-n display each line with a line number

Example :

1 . Display content of a file (single or multiple files)

```
$ cat filename
```

```
$ cat file1 file2 file3
```

Or

```
$ cat < filename
```


2. Display file contents with each line number

```
$ cat -n filename
```

3. Create a new file

```
$ cat > file1.txt
```

This command creates a new file file1.txt. After typing into the file press control+d (^d) simultaneously to end the file.

4. To Append data into the file

```
$ cat file.txt >> file1.txt
```

To append data into the file use append operator >> (double re-direction output operator) to write into the file, else the file will be overwritten (i.e., all of its contents will be erased).

```
$ cat >> file1.txt
```

It will append data into same file, previous content will be deleted.

5. To concatenate several files and transfer the output to another file.

```
$ cat file1.txt file2.txt > file3.txt
```

In the above example the output is redirected to new file file3.txt. The cat command will create new file file3.txt and store the concatenated output into file3.txt.

```
$ cat file1.txt > file2.txt
```

Touch command :

The touch command is a standard command used in UNIX/Linux operating system which is used to create empty files, change and modify timestamps of a file.

Syntax :

```
$ touch filename
```

Example :

1. To create empty file/s
\$ touch filename
\$ touch file1 file2 file3
2. To create a hidden file
\$ touch . filename

3. Change File's Access Time

We can change the access time of a file using -a option. By default it will take the current system time and update the atime field. Before change atime of a file check the status of a file by using 'stat' command

```
$ touch -a filename
```

4. To change the file's Modification Time

You can change the modification time of a file using -m option.
\$ touch -m filename

5. To change the timestamp of a file

You can change the timestamp of a file by using -t option
\$ touch -t YYYYMMDDhhmm file.txt
\$ touch -t 202310120909 file.txt

cp (Copying files and directories) command:

cp command is used to copy one or more files or directories from source to destination.. If the destination is an existing file, then the file is overwritten; if the destination is an existing directory, the file is copied into the directory (the directory is not overwritten).

Syntax :

```
cp [option] source destination
```

Options :

-b This option will make the backup of all the contents of the file

- f This option is used to forcefully copied all the contents of the file from source to destination (write protected)
- i This option will confirm from you before copying the files by displaying a message
- r This option will recursively copy all directory with child files & dir to destination

Example :

1. Create a backup before copying

```
$ cp -b file1.txt file2.txt
```

If I do a copy without the -b option in the above case, the 'file2.txt' file would've been overwritten by 'file1.txt' and I would lose the contents of the second file. The file with the tilde (~) symbol as a suffix is the backup file that contains the original contents of the destination file.

2. Display Prompt message before overwritten

```
$ cp -i file1 file2
```

3. Copy directory into directory (Recrusively copy)

```
$ cp -r dir1 dir2
```

When copying directories, the cp command will display an error if the -r option is not specified. Why does it do that? Because the cp command doesn't know what to do with the files and folders inside a parent directory when you simply specify the parent directory name.

With the recursive option, you can specifically ask the cp command to copy the entire directory structure along with the parent.

4. Copy files into directory

```
$ cp file.txt mydir
```

```
$ cp file1 file2 file3 file4 mydir
```

```
$ cp file1.txt /home/linux/documents/mydir
```

(to copy the files to the specific location)

5. Display verbose information while copying

```
$ cp -v file.txt dir1
```

mv (move or rename) command:

mv command which is short for move. It is used to move/rename file from one directory to another. mv command is different from cp command as it completely removes the file from the source and moves to the directory specified, where cp command just copies the content from one file to another.

Syntax:

```
mv [-options] oldname newname  
mv [-options] [source file] [destination file]
```

Example :

1. Rename files

```
$ mv oldfile newfile  
$ mv file1.txt newfile.txt
```

2. Rename directories

```
$ mv olddir newdir  
$ mv mydir newdir
```

3. Move files to directory

```
$ mv file.txt mydir  
$ mv file1 file2 file3 mydir
```

4. Move files interactively (prompt before move)

```
$ mv -i file1 mydir  
$ mv -i file1 file2 file3 mydir
```

5. Creating file backups before moving

```
$ mv -b file1 newfile
```

wc (wordcount) command :

The wc command in Linux is a command line utility for printing newline, word and byte counts for files. It can return the number of lines in a file, the number of characters in a file and the number of words in a file. It can also be combine with pipes for general counting operations.

Syntax :

Wc [option] [filename]

Options:

- l --> Display only no of lines
- w --> Display only no of words
- c --> Display only no of character
- L --> Display longest line

Example:

1. Display no of lines , no of words and no of character
\$ wc filename.txt
2. Display only no lines
\$ wc -l file.txt
3. Display only no of words
\$ wc -w file.txt
4. Display only no of character
\$ wc -c file.txt

nl command :

The nl command in Linux is an abbreviation for number lines. If you've ever had a large text document and needed to add line numbers to it, the nl command is used. Rather than going through the painstaking process of numbering each line manually, or importing your text document into a GUI text editor, you can simply use nl from the command line and be done in a matter of seconds.

```
$ nl filename.txt
```

➤ **File & Directory Permission Management in Linux:**

Linux is a multiuser environment, so many different people are often working with a single set of files and directories at the same time. Some of these files are intended to be shared, so that many people can view and edit the file.

If any user could access & modify all files belonging to other user this could be a security risk. This is why Linux has a built in security. This ensures that a file or directory can be accessed, modified or executed by only desired users.

Which file would be accessed by which user is decided by two factors in Linux.

- File Ownership
- File Permission

- File Ownership :

Every Linux system has three types of owner:

User: A user is the one who created the file. By default, whosoever, creates the file becomes the owner of the file. A user can create, delete, or modify the file.

Group: A group can contain multiple users. All the users belonging to a group have same access permission for a file.

Other: Any one who has access to the file other than user and group comes in the category of other. Other has neither created the file nor is a group member.

Users and groups can be locally managed in **/etc/passwd** or **/etc/group**.

- File Permissions

Every file and directory in Linux system has following 3 permissions defined for all the 3 owners.

Read (r):

This permission give you the authority to open and read a file. Read permission on a directory gives you the ability to lists its content.

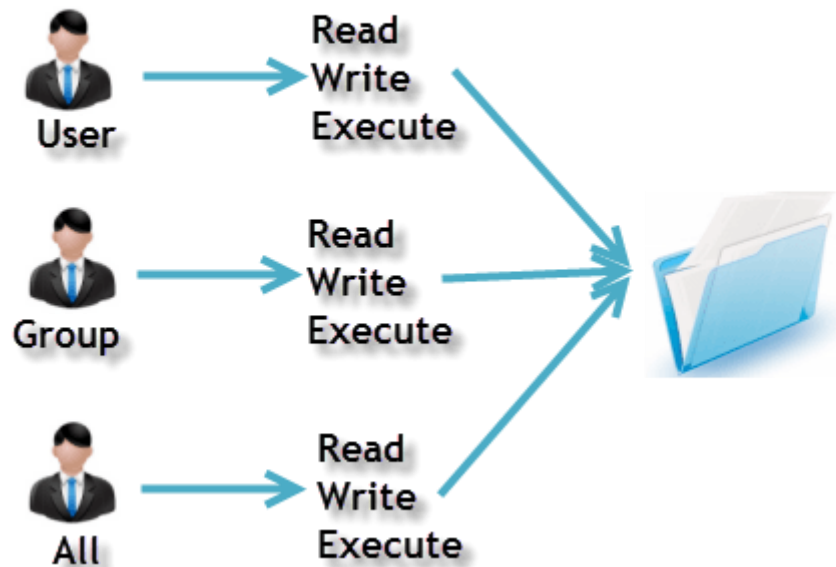
Write (w) :

The write permission gives you the authority to modify the contents of a file. The write permission on a directory gives you the authority to add, remove and rename files stored in the directory.

Execute (e):

In Linux, you cannot run a program unless the execute permission is set. If the execute permission is not set, you might still be able to see/modify the program code(provided read & write permissions are set), but not run it.

Owners assigned Permission On Every File and Directory



View File/Directory Permissions :

To view the permissions for all files or a directory, use the ls command with the -l option otherwise we can use 'stat' command to view file permissions.

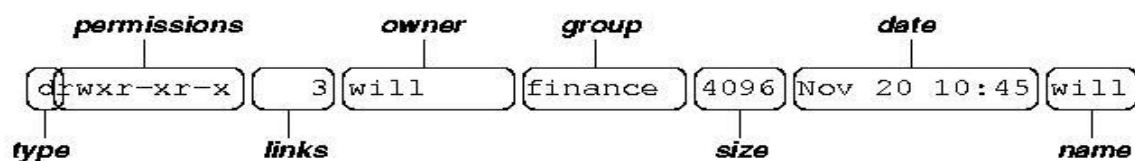
```
$ ls -l filename/dirname
```

```
$ stat filename
```

The output will be like –


```
-rw-r--r-- 1 user1 group1 62 Jan 15 16:10 myfile.txt
drwxr-xr-x 2 user1 group1 2048 Jan 15 17:10 Example
```

In the output example above, the first character in each line indicates whether the listed object is a file or a directory. Directories are indicated by a (d); the absence of a d at the beginning of the first line indicates that myfile.txt is a regular file.



The output contains :

where:

- o type is a single character which is either 'd' (directory), '-' (ordinary file), 'l' (symbolic link), 'b' (block-oriented device) or 'c' (character-oriented device).
- o permissions is a set of characters describing access rights. There are 9 permission characters, describing 3 access types given to 3 user categories. The three access types are read ('r'), write ('w') and execute ('x'), and the three users categories are the user who owns the file, users in the group that the file belongs to and other users (the general public). An 'r', 'w' or 'x' character means the corresponding permission is present; a '-' means it is absent.
- o links refers to the number of filesystem links pointing to the file/directory.
- o owner is usually the user who created the file or directory.
- o group denotes a collection of users who are allowed to access the file according to the group access rights specified in the permissions field.

- o size is the length of a file, or the number of bytes used by the operating system to store the list of files in a directory.
- o date is the date when the file or directory was last modified (written to). The -u option display the time when the file was last accessed (read).
- o name is the name of the file or directory.
The group that file belongs to, and any user in that group will have the permissions given in the third field over that file.

Manage File & Directory Permissions :

The chmod command is commonly used to change Linux file permissions. Any user with sudo privileges, the root, and the file owner are capable of changing the file permissions.

Chmod (Change Mode) :

This command is used to change the permissions of files and directories in Linux.

There are two ways to use the chmod command

1. Absolute Mode
2. Symbolic Mode

1. Absolute Mode (Numeric Mode) :

In Absolute Mode, permissions are represented in numeric form i.e.(Octal Form 0-7). Octal values are base 8 numbers, which means they use the numbers 0 to 7 to represent different combinations of file permissions. Each number corresponds to the read, write, and execute

permissions for the owner, group, and others, respectively. The numbers are calculated by adding the values of the permissions you want to set:

- 4 for read permission
- 2 for write permission
- 1 for execute permission

To calculate the octal value for a set of file permissions, you add the values of the permissions that are granted. For example:

rw- (read, write, and execute) has an octal value of 7 ($4 + 2 + 1 = 7$)

rw- (read and write and no perm) has an octal value of 6 ($4 + 2 + 0 = 6$)


r-- has an octal value of 4 ($4 + 0 + 0 = 4$)

--- (no read, write, or execute permission) has an octal value of 0 ($0 + 0 + 0 = 0$)

Permissions	Binary	Octal	Description
---	000	0	No permissions
--x	001	1	Execute-only permission
-w-	010	2	Write-only permission
-wx	011	3	Write and execute permissions
r--	100	4	Read-only permission
r-x	101	5	Read and execute permissions
rw-	110	6	Read and write permissions
rwX	111	7	Read, write, and execute permissions

d =directory	7	rwX	111
r = read	6	rw-	110
w = Write	5	r-X	101
	4	r--	100
	3	-wX	011
	2	-w-	010
	1	--X	001
	0	---	000

chmod 777



rwx|rwx|rwx

Owner|Group|Others

To use

octal values to set file permissions, you use the chmod command with a three-digit octal value. The first digit represents the permissions for the owner, the second digit represents the permissions for the group, and the third digit represents the permissions for others.

Examples:

Read by owner	400
Write by owner	200
Execute by owner	100
Read by group	040
Write by group	020
Execute by group	010
Read by others	004
Write by others	002
Execute by others	001

Anyone can do anything (read, write, or execute)	777
User can do anything; others can only read and execute	755
User can do anything; others can only execute	711
User can read and write; others can only read	644

2. Symbolic Mode :

In symbolic mode, you use symbols to add or remove permissions. The symbols used in symbolic mode are as follows:

"+" adds the specified permissions

"-" removes the specified permissions

"=" sets the specified permissions and removes all others

"r" for read permission

"w" for write permission

"x" for execute permission

For example, to add execute permission for the owner and group and remove all permissions for others, you would use the following command:

```
$ chmod u+x,g+x,o-rwx file.txt
```

In this command, "u" represents the owner, "g" represents the group, and "o" represents others. The "+x" adds execute permission for the owner and group, and "-rwx" removes all permissions for others.

if you want to take all permissions away from everyone, cmd is:

```
$ chmod ugo-rwx xyz.txt
```

The above cmd removes all the read(r), write(w) and execute(x) permission from all user(u), group(g) and others(o) for the file xyz.txt

```
$ chmod ug+rw,o-x file.txt
```

The above cmd adds read(r) and write(w) permission to both user(u) and group(g) and remove execute(x) permission from others(o) for the file file.txt.

```
$ chmod ug=rx,o+r file.txt
```

The above cmd assigns read(r) and execute(x) permission to both user(u) and group(g) and add read permission to others for the file file.txt

Umask (User Mask) :

Every file that gets created comes with a default set of permissions. If you ever wanted to change that default set of permissions, you can do so with the umask command. This command takes the 3 bit permission set we see in numerical permissions. The umask command provides the facility to manage the privileges for the directories and files. It seems to be complicated, but actually, It is just like the chmod command. The umask command is applied to change the default privileges on the folders and files that are newly created, while chmod is applied to the files and folders that already exist.

To check the umask value, just type the umask in the terminal and hit enter button:

```
$ umask  
→ 0022
```

By default umask value is 0022, the first 0 represents the sticky bit (special permission). The remaining three bits are the representation of the octal values. These three octal values (022) represent permissions for the owner, group members, and everyone else. Users can also change the default umask value by simply typing the umask and then the value in the terminal:

```
$ umask 444
```

Every time you create a file or a directory, your system will apply the mask, consisting of applying consecutive bitwise operations to your initial set of permissions.

The only thing you have to remember is that files are created with a 666 permission, or a "r w - r w - r w -" permission.

Similarly, directories are created with a 777 permission, or a "r w x r w x r w x" permission.

The default creation permissions for files and directories are 666 and 777, respectively. Subtract the umask value from the default value to get the permission bits for the new files.

Use the following formula to determine how umask 022 will effect newly created files and directories:

Files: $666 - 022 = 644$. The files can be accessed and modified by the owner. The files can only be read by the group and others.

Directories: $777 - 022 = 755$. The owner can list, read, modify, create, and remove files in the directory using cd. cd into the directory and list and read the files for the group and others.

Understanding the Linux mask



Mask = 022



$$\begin{array}{r} 666 \\ - 022 \\ \hline 644 \end{array}$$

or

r w - r - - r - -

$$\begin{array}{r} 777 \\ - 022 \\ \hline 755 \end{array}$$

or

r w x r w - r w -

Note : By default Umask value is 0022
By default Directory permission is 755
By default File permission is 644

➤ **Input/Output Redirection in Linux :**

- **Standard files:**

1 . Standard input file (stdin):

In Linux, when a user executes a command that requires input, the shell interprets the command and assigns the keyboard as the default source of input. The keyboard is referred to as the standard input file.

Let us take an example of the cat command. When followed by a file name, all the lines in the file are displayed. Without the file name, however, the cat command takes its input from the standard input File as shown below:

```
$ cat > filename
```

The cat command waits for input from the keyboard. As you enter characters from the keyboard and press <Enter>, the characters are displayed on the screen.

Not all commands expect input. For example, the cd command does not use the standard input file.

In Linux, all open files, including the standard files, are assigned a number called the file descriptor. The file descriptor, 0, is assigned to the standard input file.

2. Standard output file (stdout):

In Linux, the shell assigns the monitor as the default destination for the output of any command that it executes. The monitor is referred to as the standard output file.

For example, when you issue the command,

```
$ ls
```

The shell executes the command and sends its output – the directory listing – to the standard output file.

Not all commands generate output, For example, the mkdir command does not use the standard output file.

The file descriptor, 1, is assigned to the standard output file.

3. Standard Error file (stderr):

Command line utilities display error messages on the monitor. There could be many reasons for error messages, such as typing an invalid command, or a command for which the user does not have permission. The monitor is thus also the standard error file. For example, the cat command followed by a file name that does not exist will generate an error message on the monitor. Another example could be the command, cp file1 dir1, which displays an error message on the standard error file if you do not have write permission on dir1. or read permission on file1. Although the command is correct, this will generate an error since you may not have permissions on the file.

The file descriptor, 2, is assigned to the standard error file.

• **Redirection:**

Redirection changes the assignments for the standard input, standard output, and standard error. Using redirection, the input to a command can be taken from a file other than the keyboard. Similarly, the output of a command or the error can be written to a disk file or printed, instead of the monitor. The three types of redirection are:

- Input redirection
- Output redirection
- Error redirection

Input Redirection:

The following example illustrates the usage of input redirection:

```
$ cat < test 1
```

Here, the less than symbol, <, implies input redirection from the file, test1. The cat command will take each line of the file, test1, as input and display it on the monitor. Thus, it is possible to specify that the standard input, instead of coming from the standard input file, comes from a disk file.

Output Redirection:

The following example illustrates the usage of output redirection:

```
$ cat test1 > test2
```

Here, the greater than symbol, >, implies redirection of output to the file, test2. The output of the cat command is written to a disk file, test2.

In output redirection, the file to which the output is redirected is first created on the disk as an empty file and then the output is sent to this file. However, if the file already exists, its contents are deleted before the output is written to it.

If you want to append the output to the file, test2, the command is:

```
$ cat test1 >> test2
```

Error Redirection:

The following example illustrates the usage of error redirection:

```
$ cat test > errormsg
```

Assume that the file, test, does not exist in the current directory. So, when a user tries to execute this command, Linux will generate an error message since the execution is unsuccessful. This message, which would otherwise be displayed on the monitor (the standard error file), will now be written to the file, error-mesg. As in the case of output redirection, error redirection also first creates the file to which the error messages are redirected and then writes the error output to the file.

➤ **Archiving & Compression in Linux:**

Tar (Tape Archive) Command:

Archiving is the process of combining multiple files into a single package called an archive. This archive can then be easily distributed to another machine, backed up in a repository somewhere, or simply kept on your own machine as a way to organize and cleanup your file system. Archives are also an essential component of the Linux ecosystem, because all the software you install via your distribution's package manager will initially be downloaded as a compressed archive from a remote repository. Therefore, working with archives is an important aspect of using a Linux based operating system effectively.

The tar command on Linux is often used to create .tar archive files, also called "tape Archive." This command has a large number of options, but you just need to remember a few letters to quickly create archives with tar. It can create a .tar archive and then compress it with gzip or bzip compression in a single command. That's why the resulting file is a .tar.gz file or .tar.bz file.

Syntax:

```
tar [-options] [archive file name].tar [list of files]
```

Options :

- c -> create archive file
- t -> display archive file content
- x -> extract archive file content

- v -> display verbose information
- z -> compress by using gzip compression technique
- j -> compress by using bzip compression technique
- f -> creates archive with given filename

Examples:

1. Creating an uncompressed archive/ backup of files

```
$ tar -cvf filename.tar file1 file2 file3 file4
```

In the above command

c – create a new archive

v – verbosely list files which are processed.

f – following is the archive file name

2. To listout the content of a uncompressed archive file

```
$ tar -tvf filename.tar
```

3. To extract uncompressed archive files

```
$ tar -xvf filename.tar
```

4. To compress the archived / backup files using gzip compression technique

```
$ tar -czvf filename.tar.gz file1 file2 file3
```

5. To view the content of compressed archive files

```
$ tar -tzvf filename.tar.gz
```

6. To extract the compressed archive files

```
$ tar -xzvf filename.tar.gz
```

7. To compress the archived files using bzip compression technique

```
$ tar -cjvf filename.tar.bz f1 f2 f3
```

8. To view content of compressed archive

```
$ tar -tjvf filename.tar.bz
```

9. To extract the compressed archive file contents

```
$ tar -xjvf filename.tar.bz
```

➤ **File Filter Commands in Linux :**

head command:

The Linux head command prints the first lines of one or more files to standard output. By default, it shows the first 10 lines.

Syntax :

```
head [option] file_name
```

Examples :

1. To print print 1st 10 lines by-default

```
$ head filename.txt
```

This command displays default 10 lines of the file file1.txt

2. To print starting 15 lines of a file

```
$ head -n 15 file1.txt
```

This command displays 15 lines of the file file1.txt

3. To print the starting 3 lines of a file

```
$ head -n 3 file1.txt
```

This command displays 3 lines of the file file1.txt

tail command :

The Linux tail command prints the last lines of one or more files to standard output. By default, it shows the last 10 lines.

Syntax:

```
Tail [options] [filename]
```

Example :

1. To print print last 10 lines by-default

```
$ tail filename.txt
```

This command displays default last 10 lines of the file file1.txt

2. To print last 15 lines of a file

```
$ tail -n 15 file1.txt
```

This command displays last 15 lines of the file file1.txt

3. To print the last 3 lines of a file


```
$ tail -n 3 file1.txt
```

This command displays last 3 lines of the file file1.txt

4. To monitor the logs using tail command (Important)

```
$ tail -f log_file_name
```

The tail -f command prints the last 10 lines of a text or log file, and then waits for new additions to the file to print it in real time.

(Que .: How to monitor the real time logs

Que.: How to track a file for changes)

Using the tail and head together :

Combining head and tail commands allows you to output a in between lines from a file. First, use the head command to extract the first lines from a file. Then pipe the data as input to the tail command, which displays the last lines from that particular section.

if we want to get some part in the middle of a file then head and tail command are used together.

1. To print the lines from 51 to 55

```
$ head -n 55 file.txt | tail -n 5
```

This will print line no from 52,53,54 &55 from file

2. To print the lines from 10 to 15

```
$ head -n 15 file.txt | tail -n 6
```

3. To print the lines from 4 to 10

```
$ head -n 10 file.txt | tail -n 7
```

more command :

In the Linux operating system, the more command is used to view or read the larger file size on the command prompt. The more command is used to display large file contents and to check the file contents slowly use more command. It will display file contents in forward direction , page by page and line by line.

In Linux, the “cat” command is also useful to see or read the file content but when the size of the files is too high then initial or starting file content will not visible in the “cat” command. It will be visible in more command.

Syntax :

```
$ more filename
```

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 / scroll page by page

q key : To exit/quit from file.

Less command :

Less command is used to display large file contents in forward and backward direction scroll by page by page .Less is similar to more command, but less allows both forward and backward movements.

Syntax :

`$ less filename`

Following keys are used in 'less' command to scroll the page in forward and backward direction:

- f key : To scroll down page by page in forward direction.
- b key : To scroll back page by page in backward direction
- q key : To exit/quit from file.

Sort command :

The sort command is used to sort lines by alphabetical order and sorting of a file is done from 1st character of a line. By default sort command without option displays lines in alphabetical order.

Syntax :

```
sort [options] <filename>
```

Example :

1. Sort file in alphabetical order

```
$ sort file.txt
```

The output display the default content and is sorted in ascending order(alphabetically / numerically) basis on the first character.

2. Sorting in reverse order

```
$ sort -r file.txt
```

The -r option is used to sort the input in reverse order. This command sorts the lines of text in the file.txt file in reverse order and displays the result on the screen.

3. Numerical sorting

```
$ sort -n number.txt
```

This command sorts the lines of text in the number.txt file numerically and displays the result on the screen.

4. Sort numerical values in reverse order

```
$ sort -nr number.txt
```

5. Sorting by field or column

```
$ sort -k 2 file.txt
```

This command sorts the lines of text in the file.txt file based on the second field (column)

```
$ sort -k1 file.txt
```

```
$ ls -l | sort -k2
```

6. sorting numerical value by field or column number

```
$ sort -nk5 file.txt
```

```
$ ls -l | sort -nk5
```

```
$ ls -l | sort -nrk5
```

7. Removing duplicate lines

```
$ sort -u file.txt
```

This command sort the lines of text in the file.txt file and remove any duplicate lines from the output.

Uniq command :

Uniq command is helpful to remove or detect duplicate entries in a file. It is used for filtering duplicate text. It can be used by itself but it is commonly used in along with other commands. This command is also used to display unique content in the sorted file but first we have to sort file and use uniq with pipeline.

Syntax :

```
Uniq [-options] [filename]
```

Example :

1. Remove duplicate lines

```
$ uniq file.txt
```

This command will display only unique lines and removes duplicates lines on command line

2. To display no of occurrences / count of each repeated lines

```
$ uniq -c file.txt
```

3. To display only duplicate lines /repeated lines

```
$ uniq -d file.txt
```

4. Show only unique lines (repeated lines will not displayed)

```
$ uniq -u file.txt
```

5. Remove duplicate lines from file and display in alphabetical order (Display uniq content in alphabetical order)

```
$ sort file.txt | uniq
```

```
$ sort -u file.txt
```

6. To display the no of occurrences of each repeated lines in alphabetical order

```
$ sort file.txt | uniq -c
```

7. To display only duplicate lines in alphabetical order

```
$ sort file.txt | uniq -d
```

8. To display the unique lines in alphabetical order

```
$ sort file.txt | uniq -u
```

Grep Command :

Grep is used to search for a specified pattern from a specified file and display all lines containing the pattern. grep is an acronym for 'globally search a regular expression and print it'. The command searches the specified input fully (globally) for a match with the supplied pattern and displays it. The pattern that is searched in the file is referred to as the regular expression.

Syntax :

```
$ grep [options] [pattern] filename
```

Options:

- | | |
|----------|--|
| -i | ignore case |
| -o | display only pattern , not complete line |
| -w | display exact pattern matched lines |
| -c | display no of occurrences of matched string |
| -v | invert the search, displaying only lines that do not match |
| -n | display the line number before result |
| -l | list filenames, but not lines, in which matches were found |
| -e | search by multiple pattern |
| | |
| -H | filename and matching line are displayed |
| -b | byte offset or byte position display |
| -f | file contain pattern for searching string |
| -A num | display matched line and number of lines after that |
| -B num - | display matched line and number of lines before that |

-C num -	display matched line and number of lines after and before that
--include	search only included files
--exclude	search only in non included files
-E	normal grep work as egrep

Useful options : -e , -i , -c , -n , -l , -v , -w , -o

Examples :

1. Search for the given string in a single file.

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

```
$ grep "string" filename
```

2. Checking for the given string in multiple files.

```
$ grep "string" file1 file2
```

This cmd will find the given string in multiple file

3. Case insensitive search using grep -i

```
$ grep -i "string" filename
```

This searches for the given string/pattern case insensitively. So it matches all the words such as "linux", "LINUX" and "Linux" case insensitively

4. Invert match using grep -v

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

```
$ grep -v "linux" filename
```


5. Counting the number of matches using grep -c

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

```
$ grep -c "pattern" filename
```

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

```
$ grep -vc "pattern" filename
```

6. Show only the matched string using -o option

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

```
$ grep -o "pattern" filename
```

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

To show the line number of file with the line matched.

```
$ grep -n "linux" filename
```

8. Display the exact pattern matched lines

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

When you search for "is", without any option it will show out "bin", "sbin", and everything which has the substring "bin", to avoid this or to print exact searched pattern use -w option

```
$ grep -w "bin" filename
```

9. To print the filename which file contains the searching pattern

```
$ grep -l "pattern" filename
```

10. To search by multiple pattern

```
$ grep -e "linux" -e "unix" filename
```

Regular Expressions in Grep command :

Regular Expression: Regular expressions in the grep command are a set of rules used to match patterns in text. They provide a powerful way to search for and manipulate text data. grep stands for "global regular expression print," and it is a command-line tool used to search for specific text patterns in files or streams of data.

Regular expressions in grep are written using a special syntax that allows you to specify complex patterns of text to search for. For example, you could use a regular expression to search for all occurrences of a word that begins with a certain letter or that contains a specific sequence of characters. Regular expressions can also be used to search for patterns that follow certain rules, such as email addresses or phone numbers.

Metacharacter	Function	Example	What It Matches
^	Beginning of line anchor	'^love'	Matches all lines beginning with <i>love</i> .
\$	End of line anchor	'love\$'	Matches all lines ending with <i>love</i> .
.	Matches one character	'l.e'	Matches lines containing an <i>l</i> , followed by two characters, followed by an <i>e</i> .
*	Matches zero or more characters	'*love'	Matches lines with zero or more spaces, of the preceding characters followed by the pattern <i>love</i> .
[]	Matches one character in the set	'[L]ove'	Matches lines containing <i>love</i> or <i>Love</i> .
[^]	Matches one character not in the set	'[^A-K]ove'	Matches lines not containing <i>A</i> through <i>K</i> followed by <i>ove</i> .
<	Beginning of word anchor	'<love'	Matches lines containing a word that begins with <i>love</i> .
>	End of word anchor	'love>'	Matches lines containing a word that ends with <i>love</i> .
(.)	Tags matched characters	'(love)ing'	Tags marked portion in a register to be remembered later as number 1. To reference later, use <i>\1</i> to repeat the pattern. May use up to nine tags, starting with the first tag at the leftmost part of the pattern. For example, the pattern <i>love</i> is saved in register 1 to be referenced later as <i>\1</i> .

EXAMPLE 3.5

grep NW

datafile

Examples with Regular Expression :

1) `$ grep '^n' datafile`

Prints all lines beginning with an n. The caret (^) is the beginning of line anchor.

2) `$ grep '4$' datafile`

Prints all lines ending with a 4. The dollar sign (\$) is the end of line anchor.

3) `$ grep '5\.'` datafile

Prints a line containing the number 5, followed by a literal period and any single character. The “dot” metacharacter represents a single character, unless it is escaped with a backslash. When escaped, the character is no longer a special metacharacter, but represents itself, a literal period.

4) `$ grep '\.5' datafile`

Prints any line containing the expression .5.

5) `$ grep '[we]' datafile`

Prints lines beginning with either a w or an e. The caret (^) is the beginning of line anchor, and either one of the characters in the brackets will be matched.

6) `$ grep '[^0-9]' datafile`

Prints all lines containing one non-digit. Because all lines have at least one non-digit, all lines are printed. (See the -v option.)

7) `$ grep '[A-Z][A-Z] [A-Z]' datafile`

Prints all lines containing two capital letters followed by a space and a capital letter.

8) `$ grep 'ss*' datafile`

Prints all lines containing an s followed by zero or more consecutive s's and a space.

9) `$ grep '[a-z]\{9\}' datafile`

Prints all lines where there are at least nine consecutive lowercase letters.

10) `$ grep '\(3\) \.[0-9].*\1 *\1' datafile`

Prints the line if it contains a 3 followed by a period and another number, followed by any number of characters (`.*`), another 3 (originally tagged), any number of tabs, and another 3. Since the 3 was enclosed in parentheses, `\(3\)`, it can be later referenced with `\1`. `\1` means that this was the first expression to be tagged with the `\(\)` pair.

11) `$ grep '\<north' datafile`

Prints all lines containing a word starting with north. The `\<` is the beginning of word anchor.

12) `$ grep '\<north\>' datafile`

Prints the line if it contains the word north. The `\<` is the beginning of word anchor, and the `\>` is the end of word anchor.

13) `$ grep '\<[a-z].*n\>' datafile`

Prints all lines containing a word starting with a lowercase letter, followed by any number of characters, and a word ending in n. Watch the `.*` symbol. It means any character, including white space.

14) `$ grep '^$' datafile`

Prints all blanked or empty lines from file

Some Review Examples of Grep Command :

Grep Command	What It Does
<code>grep '\<Tom\>' file</code>	Prints lines containing the word
<code>Tom.grep 'Tom Savage' file</code>	Prints lines containing <i>Tom Savage</i> .
<code>grep '^Tommy' file</code>	Prints lines if <i>Tommy</i> is at the beginning of the line.
<code>grep '\.bak\$' file</code>	Prints lines ending in <i>.bak</i> . Single quotes protect the dollarsign (\$) from interpretation.
<code>grep '[Pp]yramid' *</code>	Prints lines from all files containing <i>pyramid</i> or <i>Pyramid</i> in the current working directory.
<code>grep '[A-Z]' file</code>	Prints lines containing at least one capital
<code>letter.grep '[0-9]' file</code>	Prints lines containing at least one number.
<code>grep '[A-Z]...[0-9]' file</code>	Prints lines containing five-character patterns starting with a capital letter and ending with a
<code>number.grep -w '[tT]est' files</code>	Prints lines with the word <i>Test</i> and/or <i>test</i> .
<code>grep -s "Mark Todd" file</code>	Finds lines containing <i>Mark Todd</i> , but does not print the line. Can be used when checking <i>grep</i> 's exit status.
