

**S.C.O.D.E.E.N GLOBAL**

# Linux /UNIX Fundamentals



**S.C.O.D.E.E.N GLOBAL**  
School Of Code Engineering

By: D.B.Biradar

@2021

URL: <https://www.scodeen.in/>

Email: [hr@Scodeen.in](mailto:hr@Scodeen.in)

Address:

SCODEEN GLOBAL, GIRINAGAR HOSINGSOCIETY,  
FLATNO25, BALAJINAGAR, DHANKAWADI, KATRAJ,  
PUNE-43 MAHARASHTAR,INDIA

## Introduction to UNIX

### What is UNIX?

UNIX is an open source operating system. It is widely used for networking purpose. This can be used by its copyrighters or license holder.

### What is Linux?

Linux is an open source operating system which is freely available to everyone.

### Difference between Linux and UNIX:

Linux	Unix
1. It is an open-source operating system which is freely available to everyone.	1. It is an operating system which can be only used by its copyrighters.
2. It has different flavors like Ubuntu, Redhat, Fedora, etc	2. Vendors of Unix are IBM AIX, HP-UX and Sun Solaris.
3. Nowadays, Linux is in great demand. Anyone can use Linux whether a home user, developer or a student.	3. It was developed mainly for servers, workstations and mainframes.
4. Linux is used everywhere from servers, PC, smart phones, tablets to mainframes and supercomputers.	4. It is used in servers, workstations and PCs.
5. Linux is freely distributed, downloaded, and distributed through magazines And priced flavors of Linux are also cheaper than Windows.	5. Unix copyright vendors decide different costs for their respective Unix Operating systems.
6. As it is open source, it is developed by sharing and collaboration of codes by world-wide developers.	6. Unix was developed by AT&T Labs, various commercial vendors and non-profit organizations.
7. Linux is command based but some Flavors provide GUI based Linux.	7. Initially it was command based OS, but later Common Desktop Environment was created.
8. The default interface is BASH (Bourne Again Shell). But some flavors have developed their own interfaces.	8. It originally used Bourne shell. But is also compatible with other GUIs.
9. Linux supports more file system than Unix.	9. It also supports file system but lesser than Linux.
10. Linux is a Unix clone, behaves like Unix but doesn't contain its code.	10. Unix contains a completely different coding developed by AT&T Labs.

### Flavors:

Linux - Ubuntu, Redhat, Fedora etc

UNIX - IBM AIX, HP-Ux, Sun Solaris etc

## Basic Commands:

### 1. Date:

It will display date and time.

**Syntax:** \$date

**O/p** - Thu Jun 3 19:08:04 DST 2021

### 2. Cal:

It will display current month calendar

**Syntax:** \$ cal,

\$cal 2017,

\$cal march 2021,

\$cal 5 2022

**O/p**- By default it will display present month.

### 3. Who

Display current logged users in system.

**Syntax:** \$who

### 4. Who I am

It will display current user information.

**Syntax:** \$who i am

**O/p** - sagar 2021-11-09 5:45

### 5. Whoami

It will display only user name

**Syntax:** \$whoami

**O/p**- maven

### 6. Hostname

Display system name or host name

**Syntax:** \$Hostname, hostname -i

**O/p**-

### 7. tty

It will display name of computer.

**Syntax:** \$tty

**O/p:** /dev/tty1

## 8. Whatis

It will display command description.

Syntax: \$whatis

Ex: \$whatis mkdir or any command which we want knows the operation.

## 9. wc

It will display no of lines, words and character.

**Syntax:** \$wc option: -l,-w,-c

**O/p:** 2 4 45 t15

## 10. Man

It will provide the complete manual related to particular command with syntax and options which to be used in Linux.

**Syntax:** \$man mkdir

## 11. alias

Linux 'alias' command replaces one string from the shell with another string. It is a shell built-in command. It converts a complicated command into a simpler command or in other words, it creates a shortcut by replacing it with the simpler one.

**Syntax:**

1. alias <newName>=<command> (To create alias for commands)
2. alias <newName>=<'command arg1 arg2.'>(To create alias for more than one argument)
3. alias <newName>=<'/home/sssit/path/... '> (To create alias by a path)

## How to run batch by using putty terminal?

Batch: \$ ./ A\_load

ls -ltr

A\_load.log

cat A\_log

grep error A\_log

## File related commands

**Q: Tell me any 5 basic commands you're across?**

**Q: which commands you're using with your project?**

**Q: Tell me your project related commands?**

### 1. Touch:

It is use to create empty .txt file

**Q: how to create empty or simple .txt file?**

**Syntax:** \$ touch filename

### 2. Cat:

**Q: what is the purpose of cat?**

**Q: how we can overwrite a file?**

**Q: How to transfer or how to copy first 5 data into second file without overwriting and with overwriting?**

This is command is used for many purposes

1. Create a file. (\$cat > file)
2. To display the content of file. (\$cat file or \$cat < file)
3. To append (add) data at the end of file. (\$cat >> file)
4. Concatenate more than one file. (\$cat file file1)
5. Copying many files into a single file. (\$cat file file1 > file2)
6. To overwrite existing file (\$cat > file)

**Syntax:** \$ cat

**Q: How to display file with line number?**

```
root@DESKTOP-JOBS5P6:~/scodeen# cat -s -n awk1
 1  awk command is ude to display rows and columns
 2  this is awknowlegment command
 3  we have to follow syntax rule to display olumns
 4  and rows
```

### 3. More

To show content in a file. But we can't move up once we have displayed the bottom content in a file.

**Syntax:** \$more

**Ex:** \$more +5 filename

\$more -9 filename

### 4. Less

To show content in a file with 'END' keyword. It is providing the flexibility to move up and down into the file by using up and down arrow.

These commands are used to open large data/content file in real time scenarios. To come out of the more command press q.

**Syntax:** \$less

## 5. mv

Rename a file or move the file.

**Syntax:** \$mv f1 f2 - moving the data from f1 to f2

\$mv d1 d2 - While using the purpose of rename but condition is that the file should exist in directory then only we can rename the file.

**Note:** f1 data copies f2 i.e. f1 will be permanently deleted by copying the data into new file i.e. f2.

## 6. CP

It will erase or overwrite data into file.

**Syntax:** \$cp f1 f2

**Note:** it will erase or overwrite data into f2 and copy f1 data into f2.

## 7. Head

It will display top 10 records by default from a file.

**Syntax:** \$head filename, head option filename.

Ex: head p1, head -9 p1, \$head -22 p1

## 8. Tail

It will display bottom 10 records by default from file.

**Syntax:** \$tail filename or tail option filename

Ex: tail -2 p1, tail p1, tail -22 p1

**Q: How to display 77th record from file?**

Ans: \$head -77 filename | tail -1

## 9. rm

It is used to remove file from directory. We can delete more than one file at one time by using same syntax.

**Syntax:** \$rm filename, rm option filename (-i) -> it will display a pop up msg while removing a file and mention whether we want to delete the file or not.

Ex: rm p18, rm, p1,

## 10. comm (Compare)

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,

Third column indicates matching items of both the files.

Both the files have to be in sorted order for 'comm' command to be executed.

**Syntax:** comm <file1> <file2>

```
root@DESKTOP-JOBS5P6:~# cat file1
sachin
praveen
naveen
harsha
soham
root@DESKTOP-JOBS5P6:~# cat file2
sachin
praveen
naveen
kiran
vaman
root@DESKTOP-JOBS5P6:~# comm file1 file2
      sachin
      praveen
      naveen
harsha
      kiran
soham
      vaman
```

## 11. tac

The tac command is the reverse of cat command, as its name specified. It displays the file content in reverse order (from the last line).

**Syntax:** tac <file name>

## 12. tr

The tr command is used to translate the file content like from lower case to upper case.

**Syntax:** command | tr 'old' 'new'

**Ex:** cat case | tr 'name' 'NAME'

```
root@DESKTOP-JOBS5P6:~/scodeen# cat > case
goa
pune
nasik
sangali
root@DESKTOP-JOBS5P6:~/scodeen# cat case | tr 'gpki' 'GPKI'
Goa
Pune
nasIK
sanGali
```

## Filter related commands:

### 1. Grep (global regular expression print)

It is used to search particular pattern or regular expression

**Syntax:** \$grep [option] filename

**Ex :** grep error pattern(filename),(It will highlight the pattern in file )

grep -c error pattern,(It will count how many time the pattern is present in a file)

grep -n error pattern,(It will display the line no in which pattern is present)

grep -v error pattern,(It will display only those lines where the pattern is not present)

grep -i error pattern (It will ignore the upper case and lower case in pattern)

### 2. Sort:

To display content from a file in ascending and descending order.

**Syntax:** \$ sort filename (by default ascending)

sort -r filename (descending order)

### 3. gzip

The gzip command is used to truncate the file size. It is a compressing tool. It replaces the original file by the compressed file having '.gz' extension.

**Syntax:** gzip <file1> <file2> <file3>...

```
root@DESKTOP-JOB55P6:~/scodeen# ls
f1 f2
root@DESKTOP-JOB55P6:~/scodeen# gzip f1 f2
root@DESKTOP-JOB55P6:~/scodeen# ls
f1.gz f2.gz
```

### 4. gunzip

The gunzip command is used to decompress a file. It is a reverse operation of gzip command.

**Syntax:** gunzip <file1> <file2> <file3>.

```
root@DESKTOP-JOB55P6:~/scodeen# ls
f1.gz f2.gz
root@DESKTOP-JOB55P6:~/scodeen# gunzip f1 f2
root@DESKTOP-JOB55P6:~/scodeen# ls
f1 f2
```



## Directory Related Commands

### 1. pwd

The pwd command is used to display the location of the current working directory.

**Syntax:** pwd

**O/P:**

```
root@DESKTOP-JOB55P6:~/scodeen# pwd
/root/scodeen
```

### 2. mkdir

The mkdir command is used to create a new directory under any directory.

**Syntax:** mkdir <directory name>

**Ex:** mkdir scodeen

### 3. rmdir

The rmdir command is used to delete a directory.

By using this command we can delete only empty directory.

**Syntax:** rmdir <directory name>

**Ex:** rmdir scodeen

### ***Q: How to delete a non-empty directory from terminal?***

Syntax : rm -rf <directory\_name> or rm -r <directory\_name>

### 4. ls

The ls command is used to display a list of content of a directory.

**Syntax:** ls

**O/P-**

```
root@DESKTOP-JOB55P6:~/scodeen# ls
f1  f2
```

**Ex :** ls -l

```
root@DESKTOP-JOB55P6:~# ls -l
total 0
drwxrwxrwx 1 root root 4096 Jun  6 17:05 08
-rw-rw-rw- 1 root root    0 Jun  3 20:33 T15
-rw-r--r-- 1 root root    0 Jun 15 14:25 file
-rw-r--r-- 1 root root   35 Jun 15 14:25 file1
-rw-r--r-- 1 root root   34 Jun 15 14:25 file2
drwxr-xr-x 1 root root 4096 Jun 14 23:49 jobhunt
-rw-rw-rw- 1 root root   32 Jun  1 19:20 mat
drwxr-xr-x 1 root root 4096 Jun 15 15:00 scodeen
drwxrwxrwx 1 root root 4096 Jun  6 17:47 t13
-rw-rw-rw- 1 root root    0 Jun  3 20:33 t15
-rw-rw-rw- 1 root root    0 Jun  3 20:26 ty
```

Columns above indicate specific things:

Column 1 - indicates information regarding file permission.

Column 2 - indicates the number of links to the file.

Column 3 & 4 - indicates the owner and group information.

Column 5 - indicates size of the file in bytes.

Column 6 - shows the date and time on which the file was recently modified.

Column 7 - shows the file or directory name.

### With ls command we can use below multiple combination

ls option	Description
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. Size of the file is very difficult to read when displayed in terms of byte. The (ls -lh) command will give you the data in terms of Mb, Gb, Tb, etc.
ls -lhS	If you want to display your files in descending order (highest at the top) according to their size, then you can use (ls -lhS) command.
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.
ls -d */	It is used to display only subdirectories.
ls -g or ls -lG	With this you can exclude column of group information and owner.
ls -n	It is used to print group ID and owner ID instead of their names.
ls --color=[VALUE]	This command is used to print list as colored or discolored.
ls -li	This command prints the index number if file is in the first column.
ls -p	It is used to identify the directory easily by marking the directories with a slash (/) line sign.
ls -r	It is used to print the list in reverse order.
ls -R	It will display the content of the sub-directories also.
ls -lX	It will group the files with same extensions together in the list.
ls -lt	It will sort the list by displaying recently modified files at top.
ls ~	It gives the contents of home directory.
ls ../	It gives the contents of parent directory.

## 5. cd

The cd command is used to change the current directory.

**Syntax:** cd <directory name>

To go up to one directory back

**Syntax:** cd ..

O/P-

```

root@DESKTOP-JOBS5P6:~# ls
DE T15 file file1 file2 jobhunt mat scodeen t15 ty
root@DESKTOP-JOBS5P6:~# cd jobhunt
root@DESKTOP-JOBS5P6:~/jobhunt# ls
cap cap1 error harsha line new p1 p14 p18 pattern permit.gz scodeen

```

**Q: How to reverse multiple directory back?****Q: What is the difference between `cd` and `cd ..`?**

## Terminal Commands:

### 1. Clear:

It will clear the terminal screen completely.

**Syntax:** `$Clear`

### 2. Man

It will display manual pages with all the details of selected commands.

**Syntax:** `$man command-name`Ex: `$man grep`

## Permission Command:

### 1.Chmod

This command is used to set the permission for file or directory.

Basically there are two approaches we can use with `chmod` command to change the permission

1. By symbolic method
2. By using CHMOD-777 table

#### 1. Symbolic method

To change file and directory permissions, use the command `chmod` (change mode). The owner of a file can change the permissions for user (u), group (g), or others (o) by adding (+) or subtracting (-) the read, write, and execute permissions.

The first way is the relative (or symbolic) method, which lets you specify permissions with single letter abbreviations. A `chmod` command using this method consists of at least three parts from the following lists:

Access class	Operator	Access Type
u (user)	+ (add access)	r (read)
g (group)	- (remove access)	w (write)
o (other)	.= (set exact access)	x (execute)
a (all: u, g, and o)		

For example, to add permission for everyone to read a file in the current directory named **myfile**, at the UNIX prompt, enter:

**EX: chmod a+r myfile**

The 'a' stands for "all", the '+' for "add", and the r for "read".

**chmod u=x,g+w,o-r f1**

```

root@DESKTOP-JOBS5P6: ~/scodeen
root@DESKTOP-JOBS5P6:~/scodeen# ls -l
total 0
-rw-r--r-- 1 root root 387 Jun 15 17:55 editor
-rw-r--r-- 1 root root 102 Jun 15 14:59 f1
-rw-r--r-- 1 root root 71 Jun 15 14:59 f2
-rw-r--r-- 1 root root 29 Jun 16 18:19 file
-rw-r--r-- 1 root root 238 Jun 16 18:59 newsedcom
-rw-r--r-- 1 root root 21 Jun 16 19:16 num
-rw-r--r-- 1 root root 404 Jun 15 19:50 python
-rw-r-x--x 1 root root 119 Jun 16 19:13 sedcom
-rw-r--r-- 1 root root 160 Jun 16 19:13 sedcom,
-rw-r--r-- 1 root root 29 Jun 16 18:20 sedcommand
-rw-r--r-- 1 root root 35 Jun 16 19:32 sedop
root@DESKTOP-JOBS5P6:~/scodeen# chmod u=x,g+w,o-r f1
root@DESKTOP-JOBS5P6:~/scodeen# ls -l
total 0
-rw-r--r-- 1 root root 387 Jun 15 17:55 editor
---xrw--- 1 root root 102 Jun 15 14:59 f1
-rw-r--r-- 1 root root 71 Jun 15 14:59 f2
-rw-r--r-- 1 root root 29 Jun 16 18:19 file
-rw-r--r-- 1 root root 238 Jun 16 18:59 newsedcom
-rw-r--r-- 1 root root 21 Jun 16 19:16 num
-rw-r--r-- 1 root root 404 Jun 15 19:50 python
-rw-r-x--x 1 root root 119 Jun 16 19:13 sedcom
-rw-r--r-- 1 root root 160 Jun 16 19:13 sedcom,
-rw-r--r-- 1 root root 29 Jun 16 18:20 sedcommand
-rw-r--r-- 1 root root 35 Jun 16 19:32 sedop


```

## 2. CHMOD -777

The second way to use the chmod command is the CHMOD-777 table form, in which you specify a set of three numbers that together determine all the access classes and types. Rather than being able to change only particular attributes, you must specify the entire state of the file's permissions.

The three numbers are specified in the order: user (or owner), group, and other. Each number is the sum of values that specify read, write, and execute access:


Permission	Number
Read (r)	4
Write (w)	2
Execute (x)	1

CHMOD-777 Table  1 root root 16 Jun 12 19:01 cap  
-rwxrwxrwx 1 root root 16 Jun 12 19:01 cap

owner/user(u)				group(g)				others(o)			
Read(r/4)	write(w/2)	execute(x/1)	Permission	Read(r/4)	write(w/2)	execute(x/1)	Permission	Read(r/4)	write(w/2)	execute(x/1)	Permission
0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	0	0	1	1
0	1	0	2	0	1	0	2	0	1	0	2
0	1	1	3	0	1	1	3	0	1	1	3
1	0	0	4	1	0	0	4	1	0	0	4
1	0	1	5	1	0	1	5	1	0	1	5
1	1	0	6	1	1	0	6	1	1	0	6
1	1	1	7	1	1	1	7	1	1	1	7

**Note:** By default permission of file is 664 (i.e RW-RW-R--)  
By default permission of directory is 755 (i.e drwxr-xr-x)

Ex: chmod 777 filename

 root@DESKTOP-JOB55P6: ~/scodeen

```
root@DESKTOP-JOB55P6:~/scodeen# ls -l
total 0
drwxr-xr-x 1 root root 4096 Jun 18 17:34 cd
-rw-r--r-- 1 root root 387 Jun 15 17:55 editor
---xrw--- 1 root root 102 Jun 15 14:59 f1
-rw-r--r-- 1 root root 71 Jun 15 14:59 f2
-rw-r--r-- 1 root root 29 Jun 16 18:19 file
-rw-r--r-- 1 root root 238 Jun 16 18:59 newsedcom
-rw-r--r-- 1 root root 21 Jun 16 19:16 num
-rw-r--r-- 1 root root 404 Jun 15 19:50 python
-rw-r-x--x 1 root root 119 Jun 16 19:13 sedcom
-rw-r--r-- 1 root root 160 Jun 16 19:13 sedcom,
-rw-r--r-- 1 root root 29 Jun 16 18:20 sedcommand
-rw-r--r-- 1 root root 35 Jun 16 19:32 sedop
root@DESKTOP-JOB55P6:~/scodeen# chmod 777 f1
root@DESKTOP-JOB55P6:~/scodeen# ls -l
total 0
drwxr-xr-x 1 root root 4096 Jun 18 17:34 cd
-rw-r--r-- 1 root root 387 Jun 15 17:55 editor
-rwxrwxrwx 1 root root 102 Jun 15 14:59 f1
-rw-r--r-- 1 root root 71 Jun 15 14:59 f2
-rw-r--r-- 1 root root 29 Jun 16 18:19 file
-rw-r--r-- 1 root root 238 Jun 16 18:59 newsedcom
-rw-r--r-- 1 root root 21 Jun 16 19:16 num
-rw-r--r-- 1 root root 404 Jun 15 19:50 python
-rw-r-x--x 1 root root 119 Jun 16 19:13 sedcom
-rw-r--r-- 1 root root 160 Jun 16 19:13 sedcom,
-rw-r--r-- 1 root root 29 Jun 16 18:20 sedcommand
-rw-r--r-- 1 root root 35 Jun 16 19:32 sedop
```

## 2. Chown

Different users in the operating system have ownership and permission to ensure that the files are secure and put restrictions on who can modify the contents of the files. In Linux there are different users who use the system:

- ❖ Each user has some properties associated with them, such as a user ID and a home directory. We can add users into a group to make the process of managing users easier.
- ❖ A group can have zero or more users. A specified user can be associated with a “default group”. It can also be a member of other groups on the system as well.

**Ownership and Permissions:** To protect and secure files and directory in Linux we use permissions to control what a user can do with a file or directory. Linux uses three types of permissions:

- ❖ **Read:** This permission allows the user to read files and in directories, it lets the user read directories and subdirectories stores in it.
- ❖ **Write:** This permission allows a user to modify and delete a file. Also it allows a user to modify its contents (create, delete and rename files in it) for the directories. Unless the execute permission is not given to directories changes does do affect them.
- ❖ **Execute:** The write permission on a file allows it to get executed. For example, if we have a file named *php.sh* so unless we don't give it execute permission it won't run.

### Types of file Permissions:

- ❖ **User:** This type of file permission affects the owner of the file.
- ❖ **Group:** This type of file permission affects the group which owns the file. Instead of the group permissions, the user permissions will apply if the owner user is in this group.
- ❖ **Other:** This type of file permission affects all other users on the system.

**Note:** To view the permissions we use: `ls -l`

**chown** command is used to change the file Owner or group. Whenever you want to change ownership you can use chown command.

### Syntax:

`chown [OPTION]... [OWNER][:[GROUP]] FILE...`

`chown [OPTION]... --reference=RFILE FILE...`

### Ex:

To change owner of the file:

`chown owner_name file_name`

In our case we have files as follows:

```
root@kali:~# ls -l
total 80
drwxr-xr-x 2 root root 4096 Feb 2 05:29 Desktop
drwxr-xr-x 2 root root 4096 Feb 2 05:05 Documents
drwxr-xr-x 2 root root 4096 Feb 3 06:41 Downloads
-rw-r--r-- 1 root root 202 Feb 4 12:08 example.a
-rw-r--r-- 1 master group1 12 Feb 4 12:04 file1.txt
-rw-r--r-- 1 master group1 61 Feb 4 12:06 file2.txt
-rw-r--r-- 1 root group1 7 Feb 4 12:09 greek1
-rw-r--r-- 1 master group1 6 Feb 4 12:10 greek2
-rw-r--r-- 1 master group1 6 Feb 4 12:21 greek3
-rw-r--r-- 1 root root 208 Feb 4 12:23 greeks.a
drwxr-xr-x 2 root root 4096 Feb 2 05:05 Music
drwxr-xr-x 2 root root 4096 Feb 2 05:05 Pictures
drwxr-xr-x 2 root root 4096 Feb 2 05:05 Public
-rw-r--r-- 1 root root 6 Feb 5 14:04 star1.txt
-rw-r--r-- 1 root root 7 Feb 5 14:01 star2.txt
-rw-r--r-- 1 root root 6 Feb 5 14:01 star3.txt
-rw-r--r-- 1 root root 8 Feb 5 12:33 star.a
-rw-r--r-- 1 root root 208 Feb 5 13:12 super.a
drwxr-xr-x 2 root root 4096 Feb 2 05:05 Templates
drwxr-xr-x 2 root root 4096 Feb 2 05:05 Videos
root@kali:~#
```

first three column indicates user group and other respectively

Now if I use *file1.txt* in my case, to change ownership I will use the following

**Syntax:** # chown master file1.txt

```
root@kali:~# ls -l file1.txt
-rw-r--r-- 1 root root 12 Feb 4 12:04 file1.txt
root@kali:~# chown master file1.txt
root@kali:~# ls -l file1.txt
-rw-r--r-- 1 master root 12 Feb 4 12:04 file1.txt
root@kali:~#
```

where the *master* is another user in the system. Assume that if you are user named user1 and you want to change ownership to root (where your current directory is user1). use “sudo” before

**Syntax:** sudo chown root file1.txt

### Options:

**-c:** Reports when a file change is made.

**Ex:** chown -c master file1.txt

```
root@kali:~# chown -c master file1.txt
changed ownership of 'file1.txt' from root to master
root@kali:~# ls -l file1.txt
-rw-r--r-- 1 master root 12 Feb 4 12:04 file1.txt
root@kali:~#
```

**-v:** It is used to show the verbose information for every file processed.

Ex: chown -v master file1.txt

```
root@kali:~# chown -v master file1.txt
changed ownership of 'file1.txt' from root to master
root@kali:~# ls -l file1.txt
-rw-r--r-- 1 master root 12 Feb  4 12:04 file1.txt
root@kali:~# █
```

**-f:** It suppresses most of the error messages. When you are not permitted to change group permissions and shows error, this option forcefully/silently changes the ownership.

**Ex:** To Change group ownership In our case I am using group1 as a group in the system.

To change ownership we will use

**Syntax:** chown : group1 file1.txt

```
root@kali:~# ls -l file1.txt
-rw-r--r-- 1 master root 12 Feb  4 12:04 file1.txt
root@kali:~# chown -v :group1 file1.txt
changed ownership of 'file1.txt' from master:root to :group1
root@kali:~# █
```

You can see that the group permissions changed to group1 from root, if you use -v option it will report that. We just need to add a ":" to change group.

**1. To change the owner as well as group:** Again taking master as user and group1 as a group in the system

chown master:group1 greek1

Here, greek1 is a file.

```
root@kali:~# ls -l greek1
-rw-r--r-- 1 root root 7 Feb  4 12:09 greek1
root@kali:~# chown -v master:group1 greek1
changed ownership of 'greek1' from root:root to master:group1
root@kali:~# █
```

**2. To change the owner from particular ownership only:** Suppose we want to change ownership from master to root where current owner must be master only.

chown --from=master root greek1



```
root@kali:~# ls -l greek1
-rw-r--r-- 1 master group1 7 Feb  4 12:09 greek1
root@kali:~# chown -v --from=master root greek1
changed ownership of 'greek1' from master to root
root@kali:~#
```

### 3. To change group from a particular group:

`chown --from=:group1 root greek1`

```
root@kali:~# chown -v --reference=greek1 greek2
changed ownership of 'greek2' from root:root to root:group1
root@kali:~# ls -l greek2
-rw-r--r-- 1 root group1 6 Feb  4 12:10 greek2
root@kali:~#
```

Here, the group of greek1 is changed to root.

### 4. To copy ownership of one file to another:

`chown --reference=greek1 greek2`

```
root@kali:~# chown -v --reference=greek1 greek2
changed ownership of 'greek2' from root:root to root:group1
root@kali:~# ls -l greek2
-rw-r--r-- 1 root group1 6 Feb  4 12:10 greek2
root@kali:~#
```

### 5. To change ownership of multiple files:

`chown master:group greek2 greek3`

```
root@kali:~# chown -c master:group1 greek2 greek3
changed ownership of 'greek2' from root:root to master:group1
changed ownership of 'greek3' from root:root to master:group1
root@kali:~#
```

## Memory Related Commands:

### 1. du

Display the amount of disk usage

**Syntax:** \$du

```
root@DESKTOP-JOB55P6:~/jobhunt# ls -l
total 0
----- 1 root root 16 Jun 12 19:01 cap
-rw-r--r-- 1 root root 16 Jun 12 19:02 cap1
-rw-r--r-- 1 root root 129 Jun 12 19:09 error
-rw-r--r-- 1 root root 3 Jun 13 08:44 harsha
-rw-r--r-- 1 root root 3 Jun 13 08:43 line
drwxr-xr-x 1 root root 4096 Jun 14 19:44 new
-r--w---x 1 root root 18 Jun 13 09:12 p1
-rw-r--r-- 1 root root 23 Jun 10 19:14 p14
-rw-r--r-- 1 root root 96 Jun 10 19:50 p18
-rw-r--r-- 1 root root 150 Jun 12 18:43 pattern
-rw-r--r-- 1 root root 27 Jun 14 19:42 permit.gz
drwxr--r-- 1 root root 4096 Jun 12 19:28 scodeen
-rw-r--r-- 1 root root 41 Jun 12 18:59 sort.gz
drwxrwxrwx 1 root root 4096 Jun 14 23:24 unix
drwxr-xr-x 1 root root 4096 Jun 14 23:26 unixnew
-rw-r--r-- 1 root root 24 Jun 13 08:53 word
-rw-r--r-- 1 root root 13 Jun 13 08:35 wordcount
root@DESKTOP-JOB55P6:~/jobhunt# du
0      ./new
0      ./scodeen
0      ./unix
0      ./unixnew
0      .
```

## 2. df

Display the amount of free disk.

```

root@DESKTOP-JOBS5P6: ~/jobhunt
----- 1 root root 16 Jun 12 19:01 cap
-rw-r--r-- 1 root root 16 Jun 12 19:02 cap1
-rw-r--r-- 1 root root 129 Jun 12 19:09 error
-rw-r--r-- 1 root root 3 Jun 13 08:44 harsha
-rw-r--r-- 1 root root 3 Jun 13 08:43 line
drwxr-xr-x 1 root root 4096 Jun 14 19:44 new
-r---w---x 1 root root 18 Jun 13 09:12 p1
-rw-r--r-- 1 root root 23 Jun 10 19:14 p14
-rw-r--r-- 1 root root 96 Jun 10 19:50 p18
-rw-r--r-- 1 root root 150 Jun 12 18:43 pattern
-rw-r--r-- 1 root root 27 Jun 14 19:42 permit.gz
drwxr--r-- 1 root root 4096 Jun 12 19:28 scodeen
-rw-r--r-- 1 root root 41 Jun 12 18:59 sort.gz
drwxrwxrwx 1 root root 4096 Jun 14 23:24 unix
drwxr-xr-x 1 root root 4096 Jun 14 23:26 unixnew
-rw-r--r-- 1 root root 24 Jun 13 08:53 word
-rw-r--r-- 1 root root 13 Jun 13 08:35 wordcount
root@DESKTOP-JOBS5P6:~/jobhunt# df
Filesystem      1K-blocks      Used Available Use% Mounted on
rootfs          101887996 71620940 30267056 71% /
none            101887996 71620940 30267056 71% /dev
none            101887996 71620940 30267056 71% /run
none            101887996 71620940 30267056 71% /run/lock
none            101887996 71620940 30267056 71% /run/shm
none            101887996 71620940 30267056 71% /run/user
tmpfs           101887996 71620940 30267056 71% /sys/fs/cgroup
C:\             101887996 71620940 30267056 71% /mnt/c
D:\             180222972 59814256 120408716 34% /mnt/d
E:\             205760508 165381648 40378860 81% /mnt/e

```

## Process related command:

### 1. PS

Display the Active process or display the running process.

Syntax: \$ps

```

root@DESKTOP-JOBS5P6:~/jobhunt# ps
  PID TTY          TIME CMD
    8 tty1      00:00:00 init
    9 tty1      00:00:00 bash
   36 tty1      00:00:00 su
   37 tty1      00:00:00 bash
   47 tty1      00:00:00 sudo
   48 tty1      00:00:00 su
   49 tty1      00:00:00 bash
   59 tty1      00:00:00 sudo
   60 tty1      00:00:00 su
   61 tty1      00:00:00 bash
  120 tty1      00:00:00 ps

```

## 2. Kill

This command is used to terminate the process or to stop the process.

**Syntax:** \$kill

```
root@DESKTOP-JOBS5P6:~/jobhunt# ps
  PID TTY          TIME CMD
    8 tty1      00:00:00 init
    9 tty1      00:00:00 bash
   36 tty1      00:00:00 su
   37 tty1      00:00:00 bash
   47 tty1      00:00:00 sudo
   48 tty1      00:00:00 su
   49 tty1      00:00:00 bash
   59 tty1      00:00:00 sudo
   60 tty1      00:00:00 su
   61 tty1      00:00:00 bash
  120 tty1      00:00:00 ps
root@DESKTOP-JOBS5P6:~/jobhunt# kill 61
```

## Networking Related Commands:

### 1. Telnet:

The telnet command is used to create a remote connection with a system over a TCP/IP network. This command is used to connect remote computer/Server.

**Syntax:** \$telnet hostname/IP address

Ex: \$telnet

telnet > 127.0.1.1


But this command will not work on standalone (Ad-Hoc) system, to operate this command we need to use on workstation or server

```
root@DESKTOP-JOBS5P6:~/scodeen# telnet
telnet>
```

```
root@DESKTOP-JOBS5P6: ~/scodeen
root@DESKTOP-JOBS5P6:~/scodeen# telnet
telnet> 127.0.8.2
?Invalid command
```

It shows invalid because this root is connected to stand alone system.

The shell facilitates with various commands to create a remote connection. We can list all the commands by executing the help command, execute the help command by typing **h**. Consider the below output:

 root@DESKTOP-JOB55P6: ~/scodeen

```
root@DESKTOP-JOB55P6:~/scodeen# telnet
telnet> h
Commands may be abbreviated.  Commands are:

close          close current connection
logout         forcibly logout remote user and close the connection
display        display operating parameters
mode           try to enter line or character mode ('mode ?' for more)
open           connect to a site
quit           exit telnet
send           transmit special characters ('send ?' for more)
set            set operating parameters ('set ?' for more)
unset          unset operating parameters ('unset ?' for more)
status         print status information
toggle         toggle operating parameters ('toggle ?' for more)
slc            set treatment of special characters

z             suspend telnet
environ        change environment variables ('environ ?' for more)
telnet>
```

## 2. FTP (File Transfer Protocol):

This command is used to transfer files but you must have at least read permissions on the source file and write permission on the target system.

When transferring large files it is recommended to run the ftp command inside a screen the directory from where you run the ftp command is the local working directory.

**Syntax:** \$ftp

### Establishing an FTP Connection

To open an ftp connection to a remote system, invoke the ftp command followed by the remote server IP address or domain name.

**Ex:** To connect to an FTP server at “192.168.46.88”

You would type: ftp 192.168.46.88 Copy

Name (192.168.46.88:localuser): linuxizeCopy

You may see a different confirmation message depending on the FTP service running on the remote server.

Copy

1. Once you enter the username you will be prompted to type your password:

Password:Copy

2. If the password is correct, the remote server will display a confirmation message and the ftp> prompt.

### Common FTP Commands

Most of the FTP commands are similar or identical to the commands you would type in the Linux shell prompt.

Below are some of the most common FTP commands

- ❖ **help or ?** : list all available FTP commands.
- ❖ **cd** : change directory on the remote machine.
- ❖ **lcd** : change directory on the local machine.
- ❖ **ls** : list the names of the files and directories in the current remote directory.
- ❖ **mkdir** : create a new directory within the current remote directory.
- ❖ **pwd** : print the current working directory on the remote machine.
- ❖ **delete** : remove a file in the current remote directory.
- ❖ **rmdir** : remove a directory in the current remote directory.
- ❖ **get** : copy one file from the remote to the local machine.
- ❖ **mget** : copy multiple files from the remote to the local machine.
- ❖ **put** : copy one file from the local to the remote machine.
- ❖ **mput** : copy multiple files from the local to the remote machine.

### 3. SSH command

In Linux, ssh is a protocol, which stands for Secure Shell or Secure Socket Shell. The secure shell is useful for security while connecting to a remote server.

The ssh command uses a ssh protocol, which is a secure protocol, as the data transfer between the client and the host takes place in encrypted form.

It executes through TCP/IP port 22. The encrypted connection is also used to run the commands on a Linux server.

**Syntax:** ssh user\_name@host(IP/Domain\_name)

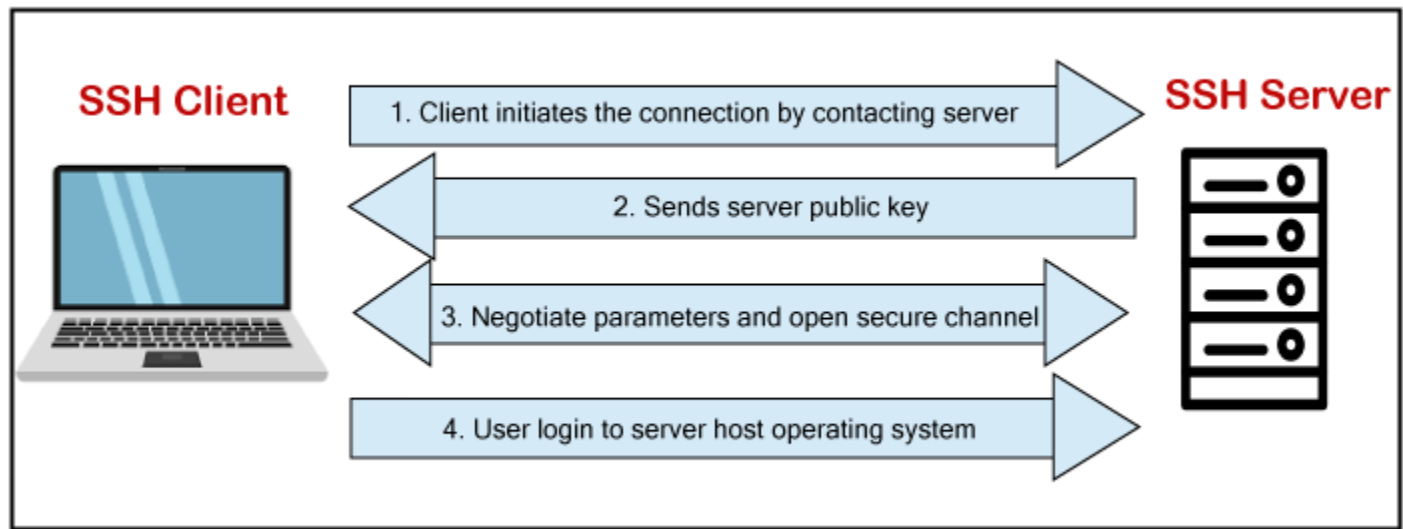
The ssh command consists of three different types of components:

- ssh command: It instructs the machine to create a secure encrypted connection with the host system.
- User name: User name is the name of the Linux user, which is being accessed by the host machine.
- Host: A host is a machine that is being accessed by the user, such as a computer or a router. A domain name or an IP address also refers as Host.

### How SSH works?

To establish an SSH connection, we need two primary components; a client and a host, which can be a server, domain name, IP address, and more. Also, we require a ssh client to connect with another computer or server. The client uses the specified host information to establish the connection; if the provided credential verified, it will establish an encrypted connection.

Difference between JDK, JRE, and JVM



The server (Host) contains an SSH process that is ready to take a request for the client connection through a TCP/IP port.

To install the OpenSSH client, execute the below command:

**1. sudo apt update** : The above sudo command will update the package of the Linux system.

**2. sudo apt install OpenSSH-client** : After updating the Linux system, execute the below command to install the OpenSSH client:

```
root@DESKTOP-JOBS5P6:~/scodeen# sudo apt install openssh -client
W: Unable to read client - open (2: No such file or directory)
root@DESKTOP-JOBS5P6:~/scodeen# sudo apt install openssh-client
Reading package lists... Done
Building dependency tree
Reading state information... Done
openssh-client is already the newest version (1:7.6p1-4ubuntu0.3).
openssh-client set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

### 3.ssh localhost :

To make an SSH connection, we need to have the server-side part of the SSH software in our machine. To check the installation status of the server, open the terminal and execute the above command. If our machine does not have the server tool kit of the OpenSSH client, then it will display the output as follows:

```
root@DESKTOP-JOBS5P6: ~/scodeen
```

```
root@DESKTOP-JOBS5P6:~/scodeen# ssh localhost
ssh: connect to host localhost port 22: Connection refused
```

In the above case, we have to install the OpenSSH server. To install the SSH server, execute the below command:

#### 4. sudo apt-get install openssh-server

We have to install the OpenSSH server. execute the above command, Consider the below output:

```
root@DESKTOP-J0BS5P6:~/scodeen# sudo apt-get install openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
openssh-server is already the newest version (1:7.6p1-4ubuntu0.3).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

#### Techniques of SSH Protocol

The SSH protocol is more secure as compared to other protocols such as telnet, and the encryption techniques are quite good than other protocols.

There are three major encryption techniques which are used by the SSH. They are as following:

1. **Hashing:** Hashing is an authentication technique that is used to ensure whether the received data is coming from a genuine sender and is unaltered or not.
2. **Symmetrical encryption:** This technique generates a single key for encryption as well as decryption. The generated key is distributed among the hosts and clients and creates a secure connection.
3. **Asymmetric encryption:** The asymmetric encryption technique is considered as more secure than other technologies, as it uses the ssh keys (Public and private keys) for encryption.

#### SSH Commands

The client ssh has many functions for the ssh command, such as creating a key, configuring a key, opening an SSH server, holding a key for single sign-on, file transfer client, and more. Some most useful ssh commands are as follows:

- ❖ **ssh-keygen:** It is used to create a key pair for establishing a connection and public key authentication.
- ❖ **ssh-copy-id:** It is used to configure a public key as a valid user on a server.
- ❖ **ssh-agent:** It is used to create an agent to hold private key for single sign-on.
- ❖ **ssh-add:** It is a tool to add a key to the agent.
- ❖ **scp:** It is a file transfer client that provides an RCP-like command-line interface.
- ❖ **sftp:** It is a file transfer command that provides an FTP-like command-line interface.
- ❖ **sshd:** It is an OpenSSH server for the Linux system.

#### Options:

There are many command-line options are available to specify the different specification of SSH output. Some useful options are as following:

- ❖ **-c:** It is used to specify query class for non-IN data.
- ❖ **-C:** It is used to compare SOA records on authoritative nameservers.
- ❖ **-d:** This option is considered as equivalent to -v.
- ❖ **-i:** It is used for IP6.INT reverse lookups.
- ❖ **-l:** It is used to list all hosts in a domain using AXFR.
- ❖ **-m:** This option sets the memory debugging flag, such as trace|record|usage.
- ❖ **-N:** It is used to change the number of dots allowed before root lookup is done.
- ❖ **-r:** It is used to disable recursive processing.
- ❖ **-R:** It specifies the number of retries for UDP packets.
- ❖ **-s:** It is used for a SERVFAIL response should stop query.



- ❖ -t: It is used to specify the query type.
- ❖ -T: It is used to enable the TCP/IP model.
- ❖ -v: It is used for verbose output.
- ❖ -V: It is used to print the version number and exit.
- ❖ -w: It is used to specify the wait forever for a reply.
- ❖ -W: It is used to specify how long to wait for a reply.
- ❖ -4: It is used only for IPv4 query transport.
- ❖ -6: It is used only for IPv6 query transport.

### How to connect via SSH

As we have installed the SSH client and server, we can establish a secure connection with other machines. For a secure connection between two machines, they both have ssh client and server installed.

To establish a connection, execute the below command:

1. **ssh your\_username@host\_ip\_address**

If the user name is verified by the machine that you want to connect, execute the below command:

2. **ssh host\_ip\_address**

The above command will ask for the password, type the password, and press ENTER key.

If we are making a connection for the first time, it will ask for the continue connecting; type yes and press Enter. It will add an ECDSA (Elliptical curve Digital Signature Algorithm) key and connect you to a remote server.

You are now eligible to control and manage a remote machine by your terminal. If you face any difficulty in establishing a connection, consider the following points:

- ❖ If the provided IP address of the remote machine is valid.
- ❖ The port SSH daemon is listening to is not blocked by a firewall or forwarded incorrectly.
- ❖ The username and password that you entered are correct.
- ❖ SSH software is installed properly.

### Acknowledgement commands:

#### 1. awk:

To Process Rows and Columns in a file.

#### Syntax: -- To display columns

```
awk '{print $ column no}' filename
```

--- to display 2<sup>nd</sup> row.

```
awk 'NR1 && NR3' file name
```

**Note :** You can display multiple columns and rows by using awk command

Ex: to display 3<sup>rd</sup> column from a file

```
root@DESKTOP-JOBS5P6: ~/scodeen
root@DESKTOP-JOBS5P6:~/scodeen# cat awk2
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
root@DESKTOP-JOBS5P6:~/scodeen# awk '{print $3}' awk2
3
8
13
18
```

To display 3<sup>rd</sup> row from file

```
root@DESKTOP-JOBS5P6: ~/scodeen
root@DESKTOP-JOBS5P6:~/scodeen# cat awk2
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
root@DESKTOP-JOBS5P6:~/scodeen# awk 'NR>2 && NR<4' awk2
11 12 13 14 15
```

Q: How to print '11' at the end of every line in a file

Q: How to print '11' at the start of every line in a file

```
root@DESKTOP-JOBS5P6: ~/scodeen
root@DESKTOP-JOBS5P6:~/scodeen# cat awk2
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
root@DESKTOP-JOBS5P6:~/scodeen# awk '{print $0 "11"}' awk2
1 2 3 4 511
6 7 8 9 1011
11 12 13 14 1511
16 17 18 19 2011
root@DESKTOP-JOBS5P6:~/scodeen# awk '{print "11" $0}' awk2
111 2 3 4 5
116 7 8 9 10
1111 12 13 14 15
1116 17 18 19 20
```

**Q: To print the squares of first numbers from 1 to n say 6:**

```
$ awk 'BEGIN { for(i=1;i<=6;i++) print "square of", i, "is",i*i; }'
```

```
root@DESKTOP-JOB55P6: ~/scodeen
root@DESKTOP-JOB55P6:~/scodeen# awk 'BEGIN{for(i=1;i<=6;i++) print "square of ", i, "is",i*i;}'
square of  1 is 1
square of  2 is 4
square of  3 is 9
square of  4 is 16
square of  5 is 25
square of  6 is 36
```

## 2. cut :

The cut command in UNIX is a command for cutting out the sections from each line of files and writing the result to standard output. It can be used to cut parts of a line by **byte position, character and field**.

**Syntax:** cut OPTION... [FILE]...

**EX:** cut -b 3 file, cut -c 4 file etc

**Options and their Description with examples:**

**1. -b (byte):** To extract the specific bytes, you need to follow -b option with the list of byte numbers separated by comma. Range of bytes can also be specified using the **hyphen (-)**. It is necessary to specify list of byte numbers otherwise it gives error. Tabs and backspaces are treated like as a character of 1 byte.

```
root@DESKTOP-JOB55P6: ~/scodeen
root@DESKTOP-JOB55P6:~/scodeen# cat cut1
abcde
efghij
klmno
pqrst
uvwxy
root@DESKTOP-JOB55P6:~/scodeen# cut -b 3 cut1
c
g
m
r
w
root@DESKTOP-JOB55P6:~/scodeen# cut -b 3-5 cut1
cde
ghi
mno
rst
wx
root@DESKTOP-JOB55P6:~/scodeen# cut -b 1,3,5 cut1
ace
egi
kmo
prt
uw
```

**2. -c (column):** To cut by character use the -c option. This selects the characters given to the -c option. This can be a list of numbers separated comma or a range of numbers separated by hyphen (-). **Tabs and backspaces** are treated as a character.

It is necessary to specify list of character numbers otherwise it gives error with this option.

```
root@DESKTOP-JOBS5P6: ~/scodeen

root@DESKTOP-JOBS5P6:~/scodeen# cat awk2
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
root@DESKTOP-JOBS5P6:~/scodeen# cut -c 3 awk2
2
7

root@DESKTOP-JOBS5P6:~/scodeen# cut -c 3-5 awk2
2 3
7 8
12
17

root@DESKTOP-JOBS5P6:~/scodeen# cut -c 1,3,5 awk2
123
678
1 2
1 7
```

**Q: How to cut 2<sup>nd</sup> word of each row?**

`cut -d ' ' -f2 filename`

```
root@DESKTOP-JOBS5P6: ~/scodeen

root@DESKTOP-JOBS5P6:~/scodeen# cat awk1
awk command is ude to display rows and columns
this is awknowlegment command
we have to follow syntax rule to display olumns
and rows
root@DESKTOP-JOBS5P6:~/scodeen# cut -d ' ' -f2 awk1
command
is
have
rows
```

### 3. find :

The Linux **Find Command** is one of the most important and frequently used command command-line utility in Unix-like operating systems. Find command is used to search and locate the list of files and directories based on conditions you specify for files that match the arguments.

Find can be used in a variety of conditions like you can find files by **permissions, users, groups, file type, date, size**, and other possible criteria.

## Basic Find Commands for Finding Files with Names

### 1. Find Files Using Name in Current Directory

Find all the files whose name is **f2.txt** in a current working directory.

```
root@DESKTOP-JOB55P6: ~/scodeen
root@DESKTOP-JOB55P6:~/scodeen# ls
cd editor f1 f2 file newscdcom num python sedcom sedcom, sedcommand sedop
root@DESKTOP-JOB55P6:~/scodeen# find -name f2
./f2
```

### 2. Find Files under Home Directory

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

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

```
/home/tecmint.txt
```

### 3. Find Files Using Name and Ignoring Case

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

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

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

### 4. Find Directories Using Name

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

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

```
/Tecmint
```

### 5. Find PHP Files Using Name

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

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

#### 6. Find all PHP Files in Directory

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

```
# find . -type f -name "*.php"  
  
./tecmint.php  
./login.php  
./index.php
```

### Find Files Based on their Permissions

#### 7. Find Files with 777 Permissions

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

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

#### 8. Find Files without 777 Permissions

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

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

#### 9. Find SGID Files with 644 Permissions

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

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

#### 10. Find Sticky Bit Files with 551 Permissions

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

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

#### 11. Find SUID Files

Find all **SUID** set files.

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

**12. Find SGID Files**

Find all **SGID** set files.

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

**13. Find Read Only Files**

Find all **Read Only** files.

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

**14. Find Executable Files**

Find all **Executable** files.

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

**15. Find Files with 777 Permissions and Chmod to 644**

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

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

**16. Find Directories with 777 Permissions and Chmod to 755**

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

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

**17. Find and remove single File**

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

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

**18. Find and remove Multiple File**

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

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

OR

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

**19. Find all Empty Files**

To find all empty files under a certain path.

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

## 20. Find all Empty Directories

To find all empty directories under a certain path.

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

## 21. Find all Hidden Files

To find all hidden files, use the below command.

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

## Search Files Based On Owners and Groups

### 22. Find Single File Based on User

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

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

### 23. Find all Files Based on User

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

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

### 24. Find all Files Based on Group

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

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

### 25. Find Particular Files of User

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

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

## Find Files and Directories Based on Date and Time

### 26. Find Last 50 Days Modified Files

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

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

### 27. Find Last 50 Days Accessed Files

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

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



**28. Find Last 50-100 Days Modified Files**

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

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

**29. Find Changed Files in Last 1 Hour**

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

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

**30. Find Modified Files in Last 1 Hour**

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

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

**31. Find Accessed Files in Last 1 Hour**

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

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

**Find Files and Directories Based on Size****32. Find 50MB Files**

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

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

**33. Find Size between 50MB – 100MB**

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

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

**34. Find and Delete 100MB Files**

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

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

**35. Find Specific Files and Delete**

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

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

## 4. VI (Visual Instruments) Editor

The vi editor is elaborated as visual editor. It is installed in every UNIX system.

In other words, it is available in all Linux flavors.

It is user-friendly and works same on different flavors and platforms.

It is a very powerful application. An improved version of vi editor is vim.

**The vi editor has three modes:**

### 1. Command Mode:

- ❖ In command mode, actions are taken on the file.
- ❖ The vi editor starts in command mode.
- ❖ In vi editor typed words will act as commands in vi editor.
- ❖ To pass a command, you need to be in command mode.

### 2. Insert Mode:

- ❖ In insert mode, entered text will be inserted into the file.
- ❖ The Esc key will take you to the command mode from insert mode.
- ❖ By default, the vi editor starts in command mode.
- ❖ To enter text, you have to be in insert mode, just type 'i' and you'll be in insert mode.
- ❖ After typing 'i' nothing will appear on the screen but you'll be in insert mode.

### 3. Escape mode

- ❖ To exit from insert mode press **Esc** key, you'll be directed to command mode.
- ❖ If you are not sure which mode you are in, press Esc key twice and you'll be in command mode

VI editor tool is an interactive tool as it displays changes made in the file on the screen while you edit the file. In vi editor you can insert, edit or remove a word as cursor moves throughout the file. Commands are specified for each function like to delete its x or dd.

The vi editor is case-sensitive.

For example, p allows you to paste after the current line while P allows you to paste before the current line.

**Syntax:** \$ vi <fileName>

In the terminal when you'll type vi command with a file name, the terminal will get clear and content of the file will be displayed. If there is no such file, then a new file will be created and once completed file will be saved with the mentioned file name.

To start vi open your terminal and type vi command followed by file name. If your file is in some other directory, you can specify the file path. And if in case, your file doesn't exist, it will create a new file with the specified name at the given location.

Once you have done with your typing, press esc key to return to the command mode.

To save and quit

You can save and quit vi editor from command mode. Before writing save or quit command you have to press colon (:). Colon allows you to give instructions to vi.

#### Exit vi table:

Commands	Action
<b>:wq</b>	Save and quit
<b>:w</b>	Save
<b>:q</b>	Quit
<b>:w fname</b>	Save as fname
<b>ZZ</b>	Save and quit
<b>:q!</b>	Quit discarding changes made
<b>:w!</b>	Save (and write to non-writable file)

To exit from vi, first ensure that you are in command mode. Now, type: wq and press enter. It will save and quit vi.

Type: wq to save and exit the file.

If you want to quit without saving the file, **use: q**. this command will only work when you have not made any changes in the file.

#### To switch from command to insert mode:

Command	Action
<b>I</b>	Start typing before the current character
<b>l</b>	Start typing at the start of current line
<b>A</b>	Start typing after the current character
<b>a</b>	Start typing at the end of current line
<b>O</b>	Start typing on a new line after the current line
<b>o</b>	Start typing on a new line before the current line

#### To move around a file:

Commands	Action
<b>J</b>	To move down
<b>K</b>	To move up
<b>H</b>	To move left
<b>L</b>	To move right

#### To jump lines:

Commands	Action
<b>G</b>	Will direct you at the last line of the file
<b>``</b>	Will direct you to your last position in the file

#### To delete:

Commands	Action
<b>x</b>	Delete the current character
<b>X</b>	Delete the character before the cursor
<b>r</b>	Replace the current character
<b>xp</b>	Switch two characters
<b>dd</b>	Delete the current line
<b>D</b>	Delete the current line from current character to the end of the line
<b>dG</b>	delete from the current line to the end of the file

**To repeat and undo:**

Commands	Action
<b>u</b>	Undo the last command
<b>.</b>	Repeat the last command

**Command to cut, copy and paste:**

Commands	Action
<b>dd</b>	Delete a line
<b>yy</b>	(yank yank) copy a line
<b>p</b>	Paste after the current line
<b>P</b>	Paste before the current line

**Command to cut, copy and paste in blocks:**

Commands	Action
<b>&lt;n&gt;dd</b>	Delete the specified n number of lines
<b>&lt;n&gt;yy</b>	Copy the specified n number of lines

**Start and end of line:**

Commands	Action
<b>θ</b>	Bring at the start of the current line
<b>^</b>	Bring at the start of the current line
<b>\$</b>	Bring at the end of the current line
<b>dθ</b>	Delete till start of a line
<b>d\$</b>	Delete till end of a line

**Joining lines:**

Commands	Action
<b>J</b>	Join two lines
<b>yyp</b>	Repeat the current line
<b>ddp</b>	Swap two lines

**Move forward or backward:**

Commands	Action
<b>w</b>	Move one word forward
<b>b</b>	Move one word backward
<b>&lt;n&gt;w</b>	Move specified number of words forward
<b>dw</b>	Delete one word
<b>yw</b>	Copy one word
<b>&lt;n&gt;dw</b>	Delete specified number of words

**5. sed (Stream Editor) Command**

- ❖ It is used to edit streams (files) using regular expressions. But this editing is not permanent.
- ❖ it is used to replace string in a file.
- ❖ it is used to delete or remove specific lines.
- ❖ It remains only in display, but in actual, file content remains the same.

The sed command allows us to edit files without opening them. Regular expression support makes it a more powerful text manipulation tool.

**Syntax:** sed [OPTION]... {Script-only-if-no-other-script} [Input-file]...

**Options:**

The following are some command line options of the sed command:

**-n --quiet, --silent:** It forcefully allows us to print of pattern space.

**-e script, --expression=script:** It is used to add the script to the commands to be executed.

**-f script-file, --file=script-file:** It is used to add the contents of script-file to the commands to be executed.

**EX:** sed -i, '1d' filename, sed -i '\$d' filename, sed 's/hi/hello/g' filename.

**Examples of sed Command**

Let's see the following examples:

**1. Applying to the STDIN directory:**

The sed command is not just limited to manipulate files also; we can apply it to the STDIN directory.

```
echo class7 | sed 's/class/jtp/'
echo class7 | sed 's/7/10/'
cat msg.txt | sed 's/learn/study/'
```

*From the above output, first, we have performed 'sed' command on a string 'class7' where 'class' is changed into 'jtp' and 7 into 10.*

*Then we have performed 'sed' command on a stream 'msg.txt' where 'learn' is converted into 'study.'*

**2. Global Replacement:**

In the earlier example, all 'learn' words were not edited into 'study'. To edit every word, we have to use a global replacement 'g'. It will edit all the specified words in a file or string.

**Syntax:** sed 's/<oldWord>/<newWord>/g'

Consider the below examples:

```
echo class7 class9 | sed 's/class/jtp/g'
cat msg.txt | sed 's/learn/study/g'
```

The above commands will replace the entire specified text pattern.  
From the above output, by executing the command

"echo class7 class9 | sed 's/class/jtp/g'" all the 'class' is converted into 'jtp'

And with command

"cat msg.txt | sed 's/learn/study/g'" all the 'learn' was converted into 'study'.

### 3. 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.

But, note that all the lines having that same word will be deleted. It will be executed as:

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

Consider the below command:

```
cat msg.txt | sed '/jtp/d'
```

The above command will delete the lines having the word 'jtp'.  
by executing the command "cat msg.txt | sed '/jtp/d'" all lines containing the word 'jtp' are deleted.

### 4. Using the Multiple sed Command:

The '-e' option allows us to execute the multiple sed commands at once. We can perform more than one sed operation by executing the command as:

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

Consider the below command:

```
sed -e 's/red/blue/; s/yellow/black/' exm.txt
```

The above command will apply all the specified operations in file 'exm.txt'.

all the **red** words are replaced with **blue**, and all the **yellow** words are replaced with **black**.

We can also separate commands like this:

```
sed -e '
>s/red/blue/;
>s/yellow/black/' exm.txt
```

The result will be the same as the above command.

## 5. Reading Commands From a File:

We can save the sed commands in a file and apply them at once in any file. It can be done by specifying the '-f' option as follows:

**Syntax:** sed -f <sed file> <file name>

From the above command, the '<sed file>' is a file that has a sed command list.

Consider the below command:  
sed -f SedCommands exm.txt

The above command will apply all the specified commands in the 'SedCommand' file on 'exm.txt'.

## 6. Replacing Characters:

We can use the exclamation mark (!) as a string delimiter. For example, we want to replace bash shell and replace it with csh shell in the "/etc/passwd". To do so, execute the below command:

```
sed 's/\bin\bash/\bin\csh/' /etc/passwd
```

We can achieve the same result by executing the below command:

```
sed 's!/bin/bash!/bin/csh!' /etc/passwd
```

The basic use of the sed command process the entire file. But, we can limit the sed command and specify any line.

**There are two ways to limit the sed command:**

1. A range of lines.
2. A pattern that matches a specific line.

We can provide a number to specify a line as follows:

```
sed '3s/Red/Blue/' exm.txt
```

The above command will apply the specified operation on the third line.

From the above output, only the line three is modified.

We can also specify a range of lines. To specify a range of lines, execute the command as follows:

```
sed '1,3s/Red/Blue/' exm.txt
```

The above command will update the specified text in lines 1 and 3.

## 7. Inserting and Appending Text:

- ❖ The 'i' and 'a' flag is used to insert and append the text on a file.
- ❖ The 'i' flag will add the text before the string, and
- ❖ the 'a' flag is used to add text after the string.

**Ex:** `echo "Another Demo" | sed 'i\First Demo'`

The above command will insert the text before the text "Another Demo".

To append text, execute the command as follows:

`echo "Another Demo" | sed 'a\First Demo'`

The above command will append the text.

## 8. Modifying Lines:

The 'c' flag is used to modify a specific line.

To modify a line, execute the command as follows:

`sed '3c\This is a modified line.' exm.txt`

The above command will update the line three.

We can also use a regular expression to update more than one lines having the same pattern.

`sed '/Apple is /c Line updated.' exm.txt`

The above command will update all the lines having string 'Apple is'.

## 9. Transformation of Characters:

The 'y' flag is used to transform the characters.

The transformation of characters cannot be limited to specific occurrences.

To transform characters, execute the command as follows:

`sed 'y/abc/def/' exm.txt`

The above command will transform the characters 'a', 'b', 'c' into 'd', 'e', 'f'.

## 10. Printing the Line Numbers

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

To print the line number, execute the command as follows:

`sed '=' exm.txt`

The above command will display the line number of file content.

The equal sign with the '-n' option specifies the line number that contains a matching script.

`sed -n '/mango/= ' exm.txt`

The above command will display the line number that contains the word 'mango'.

From the above output, we can see the line number 2 has the 'mango' word



**Interview Questions:**

- ❖ Is there a way to erase all files in the current directory, including all its sub-directories, using only one command?
- ❖ What is a directory?
- ❖ Differentiate relative path from an absolute path.
- ❖ Enumerate some of the most commonly used network commands in UNIX
- ❖ Differentiate cmp command from diff command.
- ❖ What is the use of -l when listing a directory?
- ❖ What is piping?
- ❖ How do you determine and set the path in UNIX?
- ❖ Is it possible to see information about a process while it is being executed?
- ❖ Differentiate cat command from more command.
- ❖ What is pid?
- ❖ How does the system know where one command ends and another begins?
- ❖ What is the output of this command? \$who | sort -logfile > newfile
- ❖ What would be the effect of changing the value of PATH to: ./usr/della/bin: /bin: /usr/bin
- ❖ Write a command that will display files in the current directory, in a colored, long format.
- ❖ What does this command do? cat food 1 > kitty
- ❖ What is wrong with this interactive shell script?
- ❖ Describe the usage and functionality of the command "rm -r \*" in UNIX?
- ❖ What is the UNIX command to list files/folders in alphabetical order?
- ❖ What is the behavioral difference between "cmp" and "diff" commands?
- ❖ What are the duties of the following commands: chmod, chown, chgrp?
- ❖ What is the command to find today's date?
- ❖ Describe the zip/unzip command using gzip?
- ❖ Explain the method of changing file access permission?
- ❖ How to display the last line of a file?
- ❖ How to Kill a process in UNIX?
- ❖ Explain the advantage of executing processes in the background?
- ❖ What is the command to find maximum memory taking process on the server?
- ❖ What is the command to find hidden files in the current directory?
- ❖ What is the command to find the currently running process in Unix Server?
- ❖ What is the command to find remaining disk space in the UNIX server?
- ❖ What is the UNIX command to make a new directory?
- ❖ What is the method to see command line history?
- ❖ What is the UNIX command to find how many days the server is up?
- ❖ What is the purpose of the "echo" command?
- ❖ What is the method to edit a large file without opening it in UNIX?
- ❖ What is the process to count the number of characters and lines in a file?
- ❖ Write a command to erase all files in the current directory including all its subdirectories.
- ❖ What do you understand by command substitution?
- ❖ Enlist some commonly used network commands.
- ❖ How is cmp command different from diff command?
- ❖ Explain the types of pathnames that can be used in UNIX.
- ❖ Absolute Pathname: It defines a complete path specifying the location of a file/ directory from the beginning of the actual file system i.e.
  - ❖ from the root directory (/)
- ❖ Absolute pathname addresses system configuration files that do not change location.
- ❖ Relative Pathname: It defines the path from the current working directory where the user is i.e. the present working directory (pwd)
- ❖ Enlist some filename manipulation commands in UNIX.
- ❖ Explain the alias mechanism.

- ❖ What do you know about wildcard interpretation?
- ❖ Explain pid.
- ❖ What are the possible return values of kill
- ❖ What do you know about tee command and its usage?
- ❖ Explain mount and unmount command.
- ❖ What is the “chmod” command?

-----Thank You-----