Linux File System Structure :-

1. **/bin** :- bin - stands for binary.s

.
Binary is a file which contains the compiled source code.

We can also call it as executable, because it can be executed on the computer.

/bin is a sub-directory of the root directory in Unix/Linux OS.

this directory contains basic commands which is enough for the minimal system function
ex :- ls, cat, cp

2. **/sbin** : system binaries or super user binaries. This folder contains commands which are required for changing system properties.
Ex:- adduser,reboot,shutdown

3. **/boot** : The contents are mostly Linux kernel files and bootloader files(files needed to start up the operating system)

4. **/dev** : This contains device files
This file represents your speaker device,keyboard

5. **/etc** :- it contains all system related configuration files in here or in its sub-directories
A "configuration file" is defined as a local file used to control the operation of a program;it cannot be an executable binary.
Ex:- adduser.conf, theme config

6. **/cdrom** :- directory is a standard practice to mount cd, but not necessary. We use media and mnt to mount anything these days

/home :- The home directory can be said as a personal working space for all the users except root. There is a separate directory for every user. For example, two users ‘jitendra' and ‘jack' will have directories like "/home/jitendra“ and "/home/jack“

/lib :-
directory contains those shared library files needed to boot the system and run the commands in the filesystem,
ie. by binaries in /bin and /sbin

Only the shared libraries required to run binaries in /bin and /sbin will be here
Difference between lib, lib32, lib64, libx32

lib :- architecture independent files.
lib32 :- for 32 bit architecture libraries
lib64 :- for 64 bit architecture libraries
libx32 :- for 64 bit architecture libraries but the pointer size is 32 bit,
Normally software using the x86-64 instruction set uses 64-bit pointer size.

/media :- When you connect a removable media such as USB disk, SD card or DVD, a directory is automatically created under the /media directory for them. You can access the content of the removable media from this directory.

/mnt – Mount directory
This is similar to the /media directory but instead of automatically mounting the removable media, mnt is used by system administrators to manually mount a filesystem

/opt – Optional software
Traditionally, the /opt directory is used for installing/storing the files of third-party applications that are not available from the distribution's repository.

In the old days, "/opt" was used by UNIX vendors like AT&T, Sun, DEC and 3rd-party vendors to hold "Option" packages; i.e. packages that you might have paid extra money for.

/proc :- It contains useful information about the processes that are currently running.
It could be used for obtaining information about a system, we can also edit the config files related to kernel here.

/root :- it works as the home directory of the root user. So instead of /home/root, the home of root is located at /root. root directory (/) is different from root user directory.

/tmp :- this directory holds temporary files. Many applications use this directory to store temporary files. Even you can use directory to store temporary files.

the contains of the /tmp directories are deleted when your system restarts. Some Linux system also delete files old files automatically so don' store anything important here.

/var :- stores system-generated variable data files. This includes spool directories and files, administrative and logging data, cache, transient and temporary files.
Ex :- /var/spool contains data which is awaiting some kind of later processing

/run :- runtime variable data. The purposes of this directory were once served by /var/run, system may use both

/srv :- This directory gives users the location of data files for a particular service.
For example, if you run a HTTP or FTP server, it's a good practice to store the website data in the /srv

directory.

/usr :- User System Resources. in '/usr' go all the executable files, libraries, source of most of the system programs. For this reason, most of the files contained therein is read only (for the normal user).

why /usr/bin , /usr/sbin ?

/sys :- allows you to get information about the system and its components (mostly attached and installed hardware) in a structured way.
Ex:- device, kernel, firmware

/snap :- The /snap directory is, by default, where the files and folders from installed snap packages appear on your system.

Types of Files in Linux:-

In Linux everything is treated as File.

All files are divided into 3 types

1) Normal or Ordinary files:

These files contain data. It can be either text files (like abc.txt) OR binary files (like images, videos etc).

2) Directory Files:

- These files represent directories.
- In windows, we can use folder terminology where as in linux we can use directory terminology.
- Directory can contains files and sub directories.

3) Device Files:

In Linux, every device is represented as a file. By using this file we can communicate with that device.

Note: short-cut commands to open and close terminal

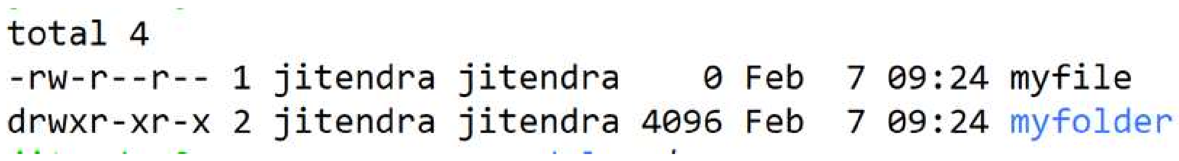
ctrl+alt+t ➔ To open terminal

ctrl+d ➔ To close terminal

How to check File Type:

In Ubuntu, blue color files represents directories and all remaining are considered as normal files. This color conventions are varied from flavour to flavour. Hence it is not standard way to check file type.

We have to use 'ls -l' command



```
total 4
-rw-r--r-- 1 jitendra jitendra    0 Feb  7 09:24 myfile
drwxr-xr-x 2 jitendra jitendra 4096 Feb  7 09:24 myfolder
```

The first character represents the type of file.

d = Directory File

-   = Normal File

l = Link File

c = Character Special File

b = Block Special File

s = Socket File

Note: c, b, s are representing system files and mostly used by super user (also known as root user or admin user)

1. ls :-

We can use ls command to listout all files and directories present in the given directory.

We can get manual documentation for any command by using man.

man ls

It provides complete information about ls command.

Various options of ls Command:

1) ls

It will display all files and directories according to alphabetical order of names.

2) ls -r

It will display all files and directories in reverse of aplhabetical order.

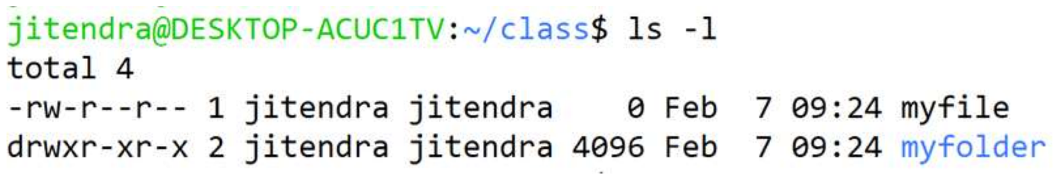
3) ls | more

To display content line by line

(To come out we have to use q)

4)ls -l

To display long listing of files



```
jitendra@DESKTOP-ACUC1TV:~/class$ ls -l
total 4
-rw-r--r-- 1 jitendra jitendra    0 Feb  7 09:24 myfile
drwxr-xr-x 2 jitendra jitendra 4096 Feb  7 09:24 myfolder
```

5) ls -t

To display all files based on last modified date and time. Most recent is at top and old are at bottom.

6) ls -rt

To display all files based on reverse of last modified date and time. Old files are at top and recent files are at bottom.

7) ls -a

a means all

To display all files including hidden files. Here . and .. also will be displayed.

8) ls -A

A means almost all

To display all files including hidden files except . and ..

9)) ls -h

display in human readable format

10) ls -R

🕐 R means Recursive.

🕐 It will list all files and directories including sub directory contents also. By default ls will display only direct contents but not sub directory contents.

Eg: All the following commands are equal

```
$ ls -l -t -r
$ ls -t -r -l
$ ls -l -r -t
$ ls -ltr
$ ls -trl
```

Which Command will make a Long listing of all the Files in our System including Hidden Files,
sorted by Modification Date (Oldest First)?
ls -lat

5. date :-

We can use date command to display date and time of system.

Various Format Options:
1) date +%D
To display only date in the form: mm/dd/yy

2) date +%T
To display only time in the form: hh:mm:ss

3) date +%d
To display only day value

4) date +%m
To display only month value

5) date +%y
To display only year value in yy form

6) date +%Y
To display only year value in yyyy form.

7) date +%H
To display only Hours value (in 24 hours scale format)

8) date +%M
To display only Minutes value

9) date +%S
To display only Seconds value

10) date +%A
To display full day name ex:- Sunday

11) date +%a

abbreviated weekday name ex:- sun

12) date +%B
to full month name ex:- January

13) date +%b
abbreviated month name ex :- jan

14) date +%w

To display day of the week (0..6), 0 is Sunday

15) date +%W

To display week number of year

16) date +%q

To display quarter of year

17) date +%j

To display day of year (1----> 366)

18) date +%I

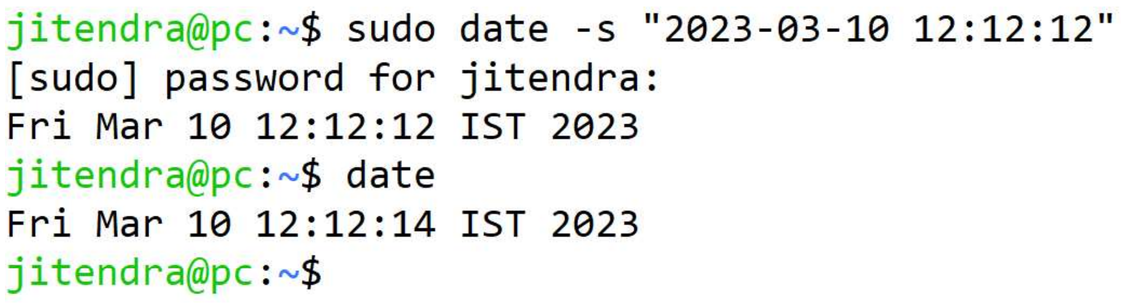
to display hour in 12 hour format (01..12)

19) date +%r

12-hour clock format   ex :-   11:11:04 PM

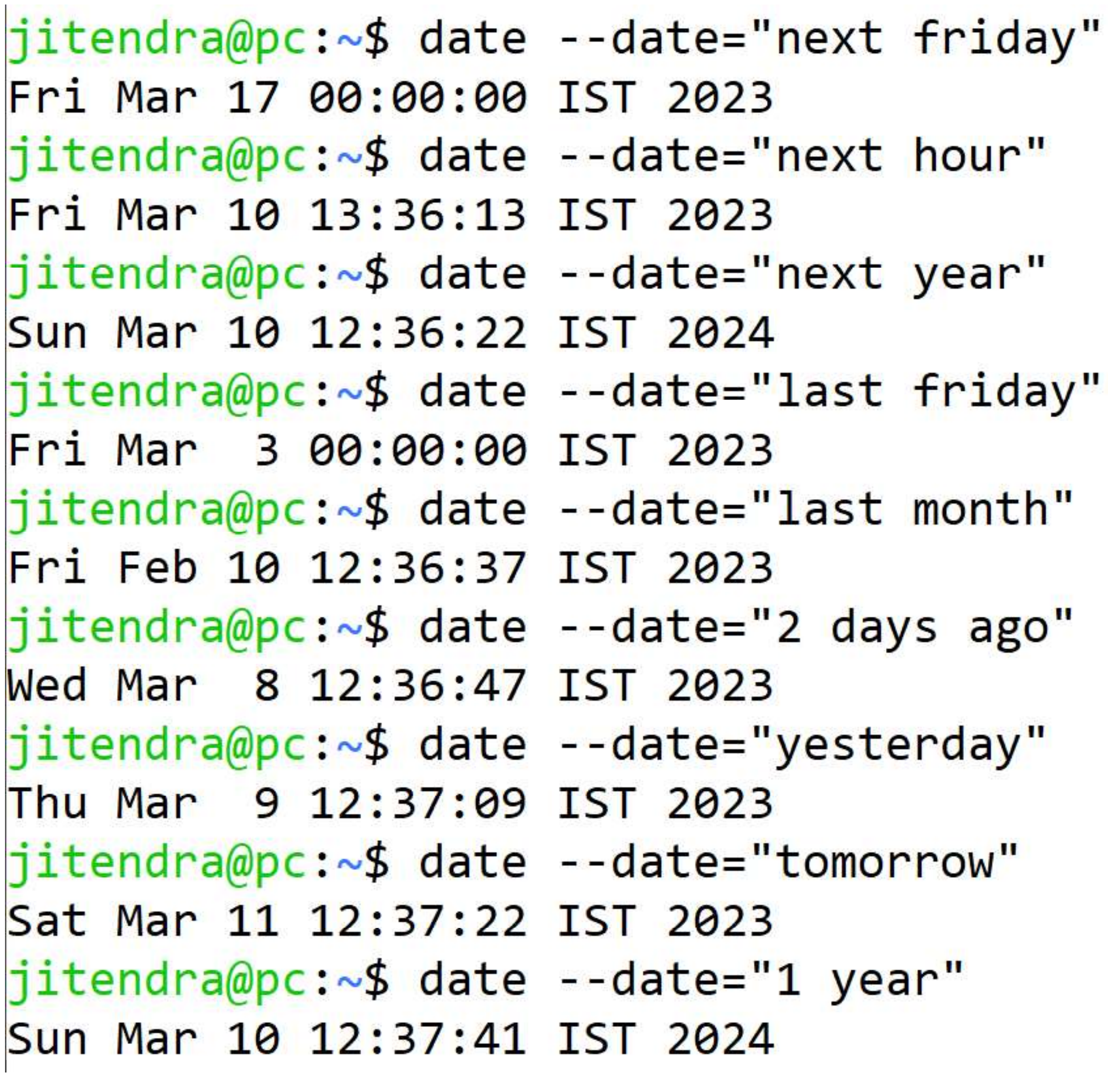
Date Flags :-

1) date -u ----> to display UTC time (universal standard)
2) date -s "string" ----> to set date as given string



```
jitendra@pc:~$ sudo date -s "2023-03-10 12:12:12"
[sudo] password for jitendra:
Fri Mar 10 12:12:12 IST 2023
jitendra@pc:~$ date
Fri Mar 10 12:12:14 IST 2023
jitendra@pc:~$
```

3) date --date="string" ----> to display past and future date time



```
jitendra@pc:~$ date --date="next friday"
Fri Mar 17 00:00:00 IST 2023
jitendra@pc:~$ date --date="next hour"
Fri Mar 10 13:36:13 IST 2023
jitendra@pc:~$ date --date="next year"
Sun Mar 10 12:36:22 IST 2024
jitendra@pc:~$ date --date="last friday"
Fri Mar  3 00:00:00 IST 2023
jitendra@pc:~$ date --date="last month"
Fri Feb 10 12:36:37 IST 2023
jitendra@pc:~$ date --date="2 days ago"
Wed Mar  8 12:36:47 IST 2023
jitendra@pc:~$ date --date="yesterday"
Thu Mar  9 12:37:09 IST 2023
jitendra@pc:~$ date --date="tomorrow"
Sat Mar 11 12:37:22 IST 2023
jitendra@pc:~$ date --date="1 year"
Sun Mar 10 12:37:41 IST 2024
```

3. cal :-

$ cal To display current month calendar.

$ cal 2020 To display total year calendar.

$ cal 1 To display 1st year calendar.

$ cal 9999 To display 9999th year calendar.

$ cal 10000 cal: year '10000' not in range 1..9999

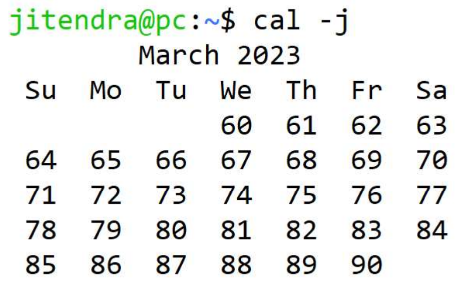
$ cal 08 2019 To display august 2019th calendar

cal -j ----> show julian calender

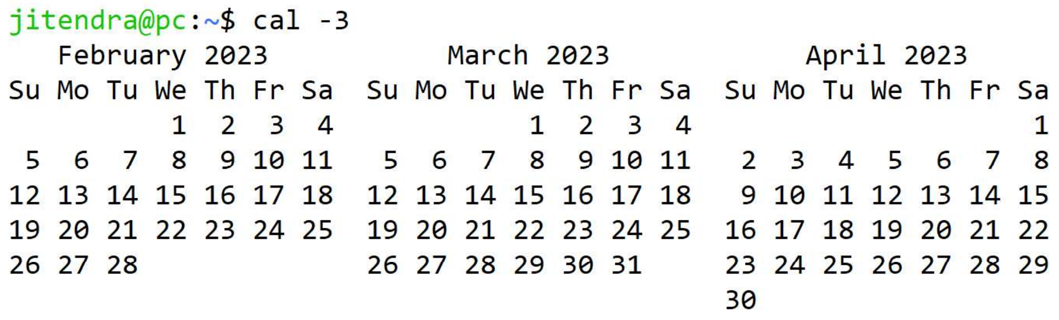
cal -3 ----> past current next month calender cal -m 5 ---> 5th month calender

cal -y 2024 ---> year 2024 calender

cal -1 ---> current month calender (this is default)

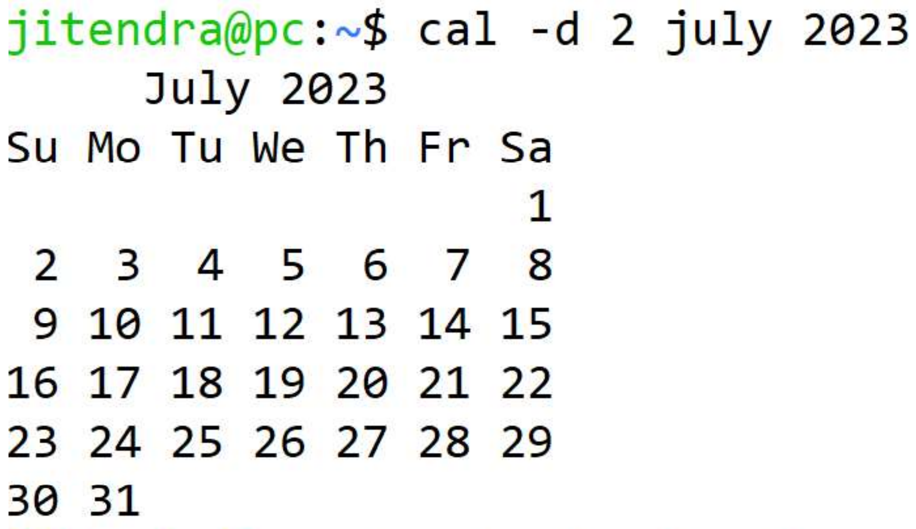


```
jitendra@pc:~$ cal -j
       March 2023
 Su  Mo  Tu  We  Th  Fr  Sa
             60  61  62  63
 64  65  66  67  68  69  70
 71  72  73  74  75  76  77
 78  79  80  81  82  83  84
 85  86  87  88  89  90
```

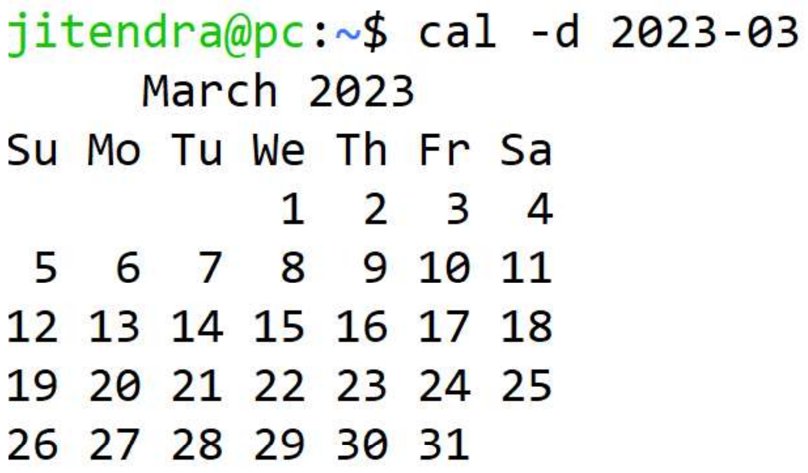


```
jitendra@pc:~$ cal -3
   February 2023          March 2023            April 2023
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
          1  2  3  4            1  2  3  4                     1
 5  6  7  8  9 10 11   5  6  7  8  9 10 11   2  3  4  5  6  7  8
12 13 14 15 16 17 18  12 13 14 15 16 17 18   9 10 11 12 13 14 15
19 20 21 22 23 24 25  19 20 21 22 23 24 25  16 17 18 19 20 21 22
26 27 28              26 27 28 29 30 31     23 24 25 26 27 28 29
                                            30
```

cal -d string ------> string will be a date which support many formats including below



```
jitendra@pc:~$ cal -d 2 july 2023
     July 2023
Su Mo Tu We Th Fr Sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```



```
jitendra@pc:~$ cal -d 2023-03
     March 2023
Su Mo Tu We Th Fr Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

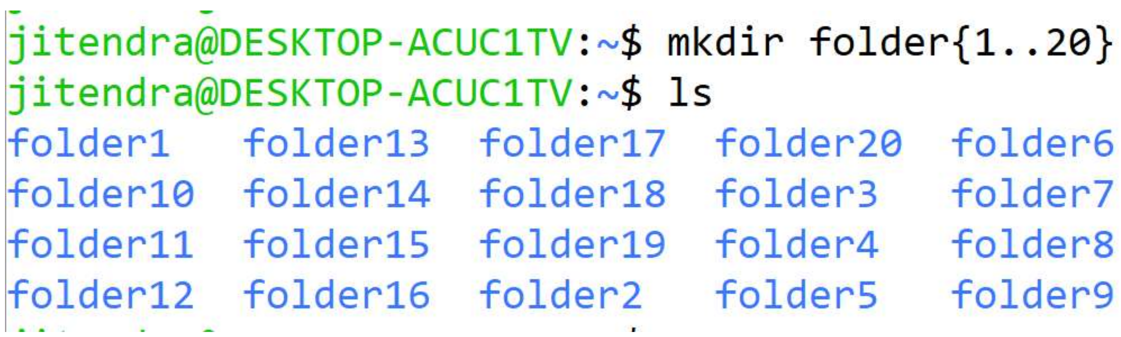
6. mkdir:-

We can create directories by using mkdir command.
1) mkdir dir1
To create a directory

2) mkdir dir1 dir2 dir3
To create multiple directories

3) mkdir dir1/dir2/dir3
To create dir3. But make sure dir1 and in that dir2 should be available already.

4) mkdir -p dir1/dir2/dir3
 -p means path of directories.
 All directories in the specified path will be created.
 First dir1 will be created and in that dir2 will be created and within that dir3 will be created



```
jitendra@DESKTOP-ACUC1TV:~$ mkdir folder{1..20}
jitendra@DESKTOP-ACUC1TV:~$ ls
folder1   folder13  folder17  folder20  folder6
folder10  folder14  folder18  folder3   folder7
folder11  folder15  folder19  folder4   folder8
folder12  folder16  folder2   folder5   folder9
```

7. rmdir :-

We can remove directories by using rmdir command.
1) $ rmdir dir1
To remove empty directory dir1
2) 2. $ rmdir dir1 dir2 dir3
To remove multiple empty directories
Note: rmdir command will work only for empty directories. If the directory is not empty then we will get error. We cannot use rmdir for files. Hence the most useless (waste) command in linux is rmdir.

8. rm :-

to delete files
rm file1 file2
To delete non empty folders

```
$ rm -r mydir
$ rm -R folder
```

Note: In Linux operating system, there is no way to perform undo operation. Once we delete a file or directory, it is impossible to retrieve that. Hence while using rm command we have to take special care.
The following command is the most dangerous command in linux, because it removes total file system.
Various options with rm Command:
1) interactive Option(-i)
While removing files and directories, if we want confirmation then we have to use -i

2)verbose Option(-v):
If we want to know the sequence of removals on the screen we should go for -v option.

9. Copy (cp)

1) To Copy from File1 to File2 (File to File)
 $ cp source_file destination_file
 $ cp file1 file2
 Total content fo file1 will be copied to file2.
 If file2 is not already available, then this command will create that file.
 If file2 is already available and contains some data, then this data will be over write with file1 content.

2) To Copy File to Directory:
 $ cp file1 file2 output
 file1 and file2 will be copied to output directory.
 Here we can specify any number of files, but last argument should be directory.
 output directory should be available already.

3) To Copy all Files of One Directory to another Directory:
 $ cp dir1/* dir2
 All files of dir1 will be copied to dir2
 But dir2 should be available already.

4) To Copy Total Directory to another Directory:
 $ cp dir1 dir2
 cp: -r not specified; omitting directory 'dir1'
Whenever we are copying one directory to another directory,
compulsory we should use -r option. 🕐 $ cp -r dir1 dir2 total dir1 will be copied to dir2

5) To Copy Multiple Directories into a Directories:
 $ cp -r dir1 dir2 dir3 dir4 dir5
 dir1,dir2,dir3 and dir4 will be copied to dir5

10. mv :-

Moving and Renaming Directories:
Both moving and renaming activities can be performed by using single command: mv

1) Renaming of files:
 $ mv oldname newname
 Eg: $ file1.txt file2.txt
 file1.txt will be renamed to file2.txt

2) Renaming of Directories:
 $ mv dir1 dir2
 dir1 will be renamed to dir2

3) Moving files to directory:
 $ mv a.txt b.txt c.txt output
 a.txt,b.txt and c.txt will be moved to output directory.

4) Moving of all files from one directory to another directory:
 $ mv dir1/* dir2
 All files of dir1 will be moved to dir2. After executing this command dir1 will become empty.

5) Moving total directory to another directory:
 $ mv dir1 dir2

Note: If dir2 is already available then dir1 will be moved to dir2.
 If dir1 is not already available then dir1 will be renamed to dir2.

--------Creation of files -----------

Creation of Files:
In Linux, we can create files in the following ways:
1) By using touch command (to create empty file)
2) By using cat command
3) By using editors like gedit, vi, nano etc

9. cat :-

cat > file1.txt

Eg:
```
$ cat > file1.txt
Hello Friends
Listen Carefully
```

Otherwise Linux will give Left and Right
ctrl+d To save and exit

If file1.txt is not already available, then file1.txt will be created
with our provided data.
If file1.txt is already available with some content,
then old data will be over written with
our provided new data.
Instead of overwriting, if we want append operation
then we should use >> with cat
command.
cat >> file1.txt
extra content
ctrl+d

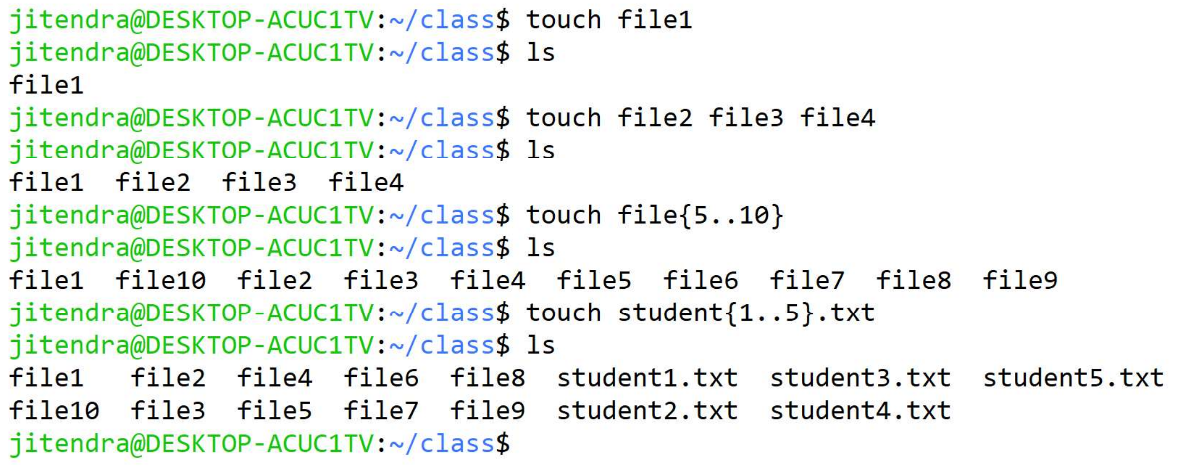
>> for appending

) If we are using Touch Comamnd, but the File is already available then
what will happend?
The content of the file won't be changed. But last modified date and time
(i.e., timestamp) will be updated

10. Touch :-

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

creating files using touch



```
jitendra@DESKTOP-ACUC1TV:~/class$ touch file1
jitendra@DESKTOP-ACUC1TV:~/class$ ls
file1
jitendra@DESKTOP-ACUC1TV:~/class$ touch file2 file3 file4
jitendra@DESKTOP-ACUC1TV:~/class$ ls
file1  file2  file3  file4
jitendra@DESKTOP-ACUC1TV:~/class$ touch file{5..10}
jitendra@DESKTOP-ACUC1TV:~/class$ ls
file1  file10  file2  file3  file4  file5  file6  file7  file8  file9
jitendra@DESKTOP-ACUC1TV:~/class$ touch student{1..5}.txt
jitendra@DESKTOP-ACUC1TV:~/class$ ls
file1   file2  file4  file6  file8  student1.txt  student3.txt  student5.txt
file10  file3  file5  file7  file9  student2.txt  student4.txt
jitendra@DESKTOP-ACUC1TV:~/class$
```

touch file1   ------> change timestamp (both access and modify time)

touch -a file1   -----> change access file of file1

touch -m file1 ------> change modify time of file1

touch -r file1 file2 -----> use file1's timestamp as reference and change timestamp of file2

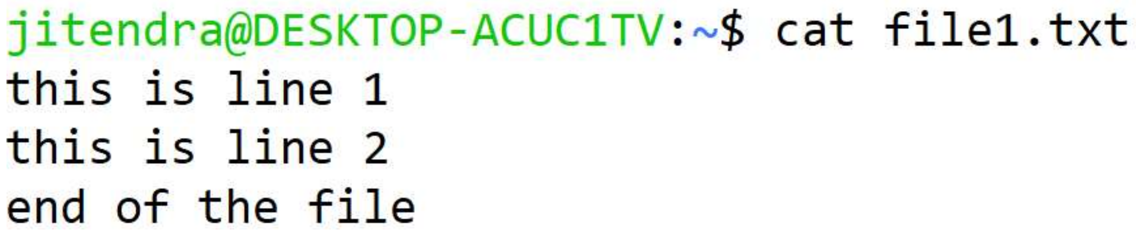
(now file2 timestamp will change and become same as file1) try :- touch -r file2 -a file1   and observe using stat command what happens

---------View Content of the Files -------------

We can view content of the file by using the following commands
1) cat
2) tac
3) rev
4) head
5) tail
6) less
7) more

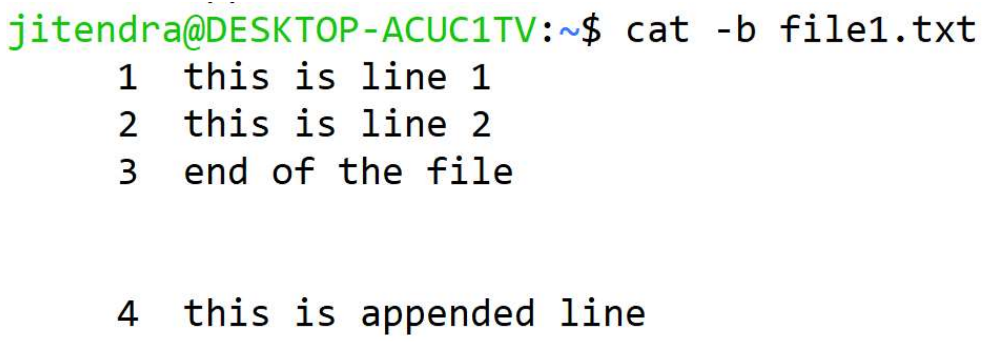
11. cat :-

$ cat < file1.txt
OR
$ cat file1.txt < is optional



```
jitendra@DESKTOP-ACUC1TV:~$ cat file1.txt
this is line 1
this is line 2
end of the file
```

-n option to give line numbering to file content
-b to give numbering to all lines apart from blank lines



```
jitendra@DESKTOP-ACUC1TV:~$ cat -b file1.txt
     1  this is line 1
     2  this is line 2
     3  end of the file


     4  this is appended line
```

We can view multiple files content at a time by using cat command.
$ cat file1.txt file2.txt file3.txt

Various utilities of cat Command:
1) To create new file with some content
```
$ cat > filename
 data
 ctrl+d
```

2) To append some extra data to existing file
```
$ cat >> filename
 extra data
 ctrl+d
```

3) To view content of file
```
 $ cat < filename or $ cat filename
```

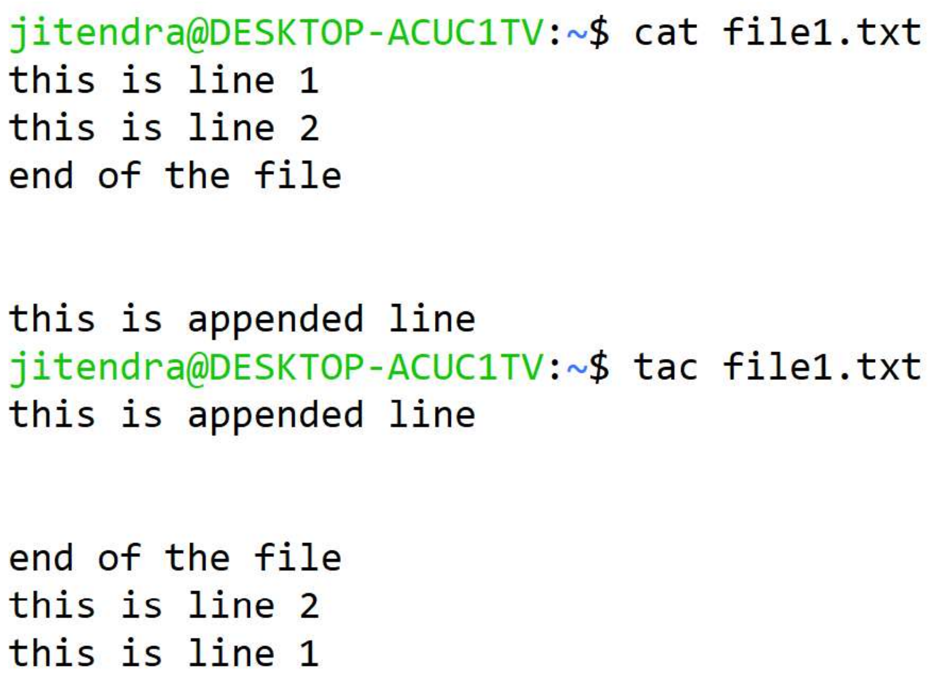
4) Copy content of one file to another file
```
 $ cat input.txt > output.txt
```

5) To copy content of multiple files to a single file
```
 $ cat file1.txt file2.txt file3.txt > file4.txt
```

6) Merging/appending of one file content to another file
```
 $ cat file1.txt >> file2.txt
```

12. tac :-

It is the reverse of cat.
It will display file content in reverse order of lines. i.e first line will become last line and last line will become first line.
This is vertical reversal.



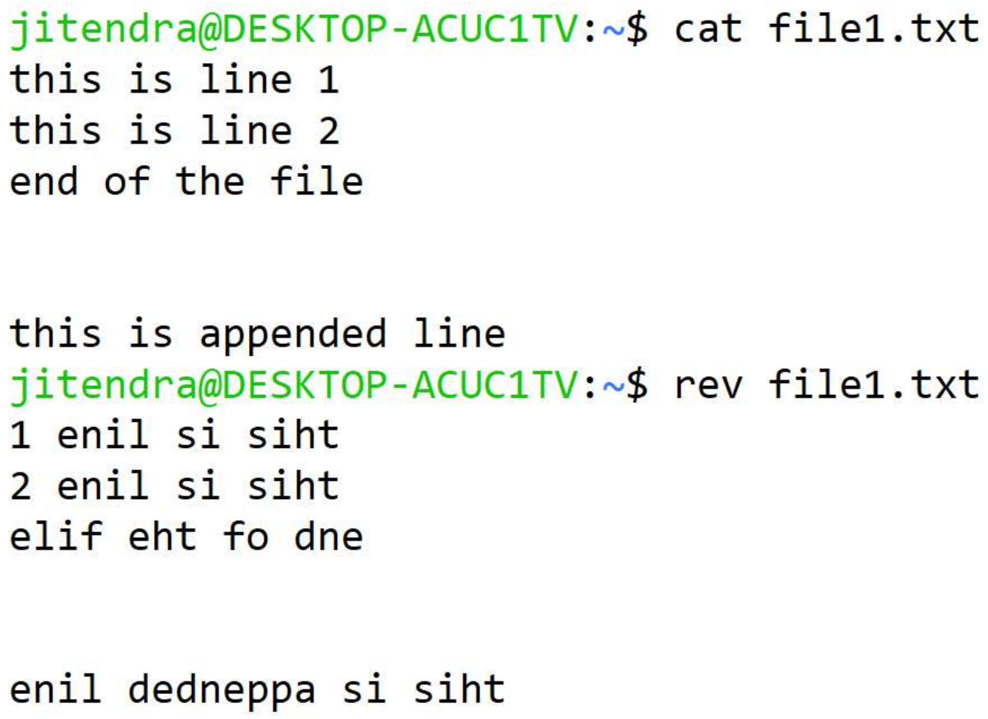
```
jitendra@DESKTOP-ACUC1TV:~$ cat file1.txt
this is line 1
this is line 2
end of the file


this is appended line
jitendra@DESKTOP-ACUC1TV:~$ tac file1.txt
this is appended line


end of the file
this is line 2
this is line 1
```

13. rev :

rev means reverse. Here each line content will be reversed. It is horizontal reversal.



```
jitendra@DESKTOP-ACUC1TV:~$ cat file1.txt
this is line 1
this is line 2
end of the file


this is appended line
jitendra@DESKTOP-ACUC1TV:~$ rev file1.txt
1 enil si siht
2 enil si siht
elif eht fo dne


enil dedneppa si siht
```

cat command will display total file content at a time. It is best suitable for small files. If the file contains huge lines then it is not recommended to use cat command. We should go for head, tail, less and more commands.

14. head :

We can use head command to view top few lines of content.

✻ head file1.txt

 It will display top 10 lines of file1.txt.

 10 is the default value of number of lines.

✻ head -n 30 file1.txt OR head -30 file1.txt

 To display top 30 lines of the file.

 Instead of 30 we can specify any number.

✻ head -n -20 file1.txt

To display all lines of file1.txt except last 20 lines.

✻ head -c 100 file1.txt

To display first 100 bytes of file content

15. tail :

 We can use tail command to view few lines from bottom of the file.

 It is opposite to head command.

✻ tail file1.txt

Last 10 lines will be displayed.

✻ tail -n 30 file1.txt OR tail -30 file1.txt OR tail -n -30 file1.txt

It will display last 30 lines.

✻ tail -n +4 file1.txt

It will display from 4th line to last line

✻ tail -c 200 file1.txt

It will display 200 bytes of content from bottom of the file.

16. more :

We can use more command to view file content page by page.

✻ more file1.txt

 It will display first page.

 Enter To view next line

 Space Bar To view next page

 q To quit/exit

✻ more -d file1.txt

-d option meant for providing details like
--More--(5%)[Press space to continue, 'q' to quit.]

17. less :

 By using more command, we can view file content page by page only in forward direction.
 If we want to move either in forward direction or in backward direction then we should go for less command.

d To go to next page.(d means down) **b** To go to previous page. (b means backward)

Creation of Hidden Files and Directories:-

 If any file starts with '.' , such type of file is called hidden file.

 If we don't want to display the files then we have to go for hidden files.

 Hidden files meant for hiding data. All system files which are internally required by kernal are hidden files.

 We can create hidden files just like normal files, only difference is file name should starts with dot.

```
touch .securefile1.txt
cat > .securefile1.txt
```

Even by using editors also we can create hidden files.

We can create hidden directories also just like normal directories.
```
mkdir .db_info
```

Note: By using hidden files and directories we may not get full security. To make more secure we have to use proper permissions. For this we should use 'chmod' command.

Interconversion of Normal Files and Hidden Files:

Based on our requirement, we can convert normal file as hidden file and viceversa.
```
mv a.txt .a.txt
```
We are converting normal file a.txt as hidden file.
```
mv .a.txt a.txt
```
Similarly directories also
```
mv dir1 .dir1
mv .dir1 dir1
```

------------------Comparing Files---------------