File system in Linux:

1. ls

The ls command is used in Linux/Unix systems to list directory contents. It's a fundamental command and has various options to display information in different formats. Here are some of the common options used with the ls command:

Basic Usage:

ls: Simply lists the names of files and directories in the current working directory.

Options:

Long Format:

- **-l:** Displays detailed information about files and directories in long format, showing permissions, owner, group, size, modification time, and name.
- **-a:** Shows all files, including hidden files (those starting with a dot .).
- **-h:** Prints file sizes in a human-readable format (e.g., KB, MB, GB).

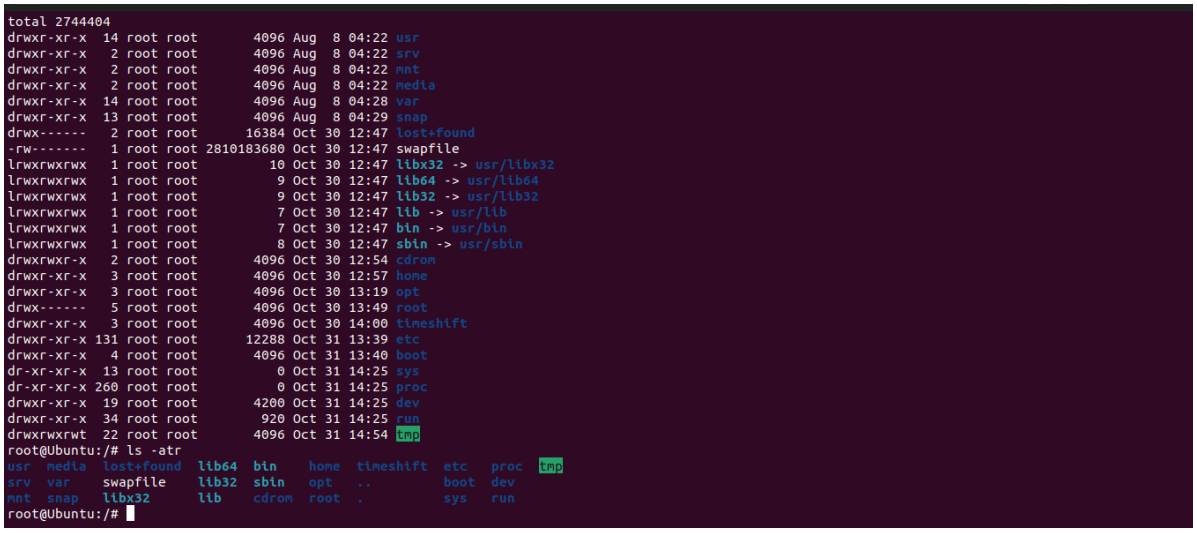
Sorting:

- **-t:** Sorts by modification time, with the newest files first.
- **-r:** Reverses the sorting order.
- **-S:** Sorts by file size.
- **-X:** Sorts by extension.

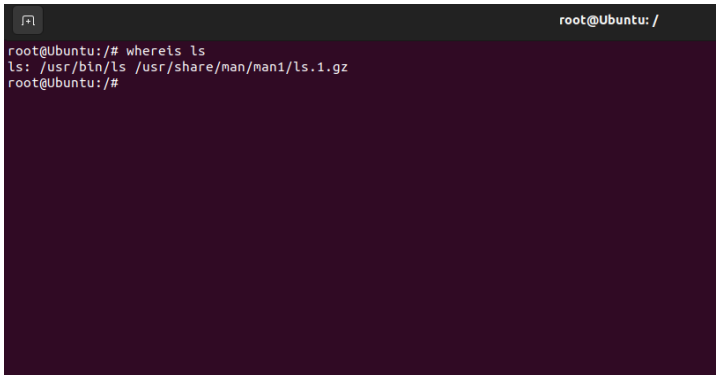
Directories:

-d: Lists directories themselves, not their contents.

-R: Lists subdirectories recursively.

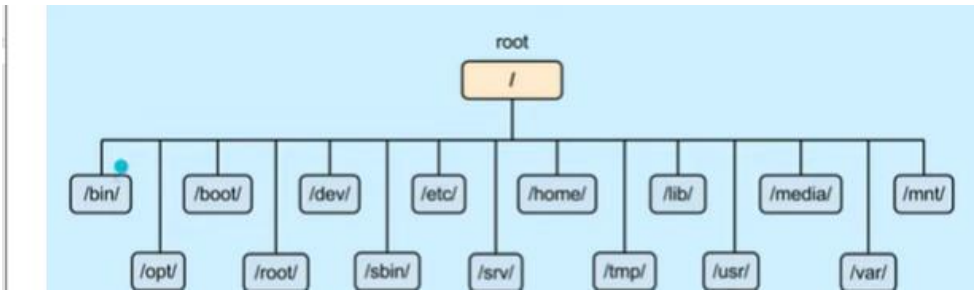


- In Linux even when we use ls commands this is also a file here



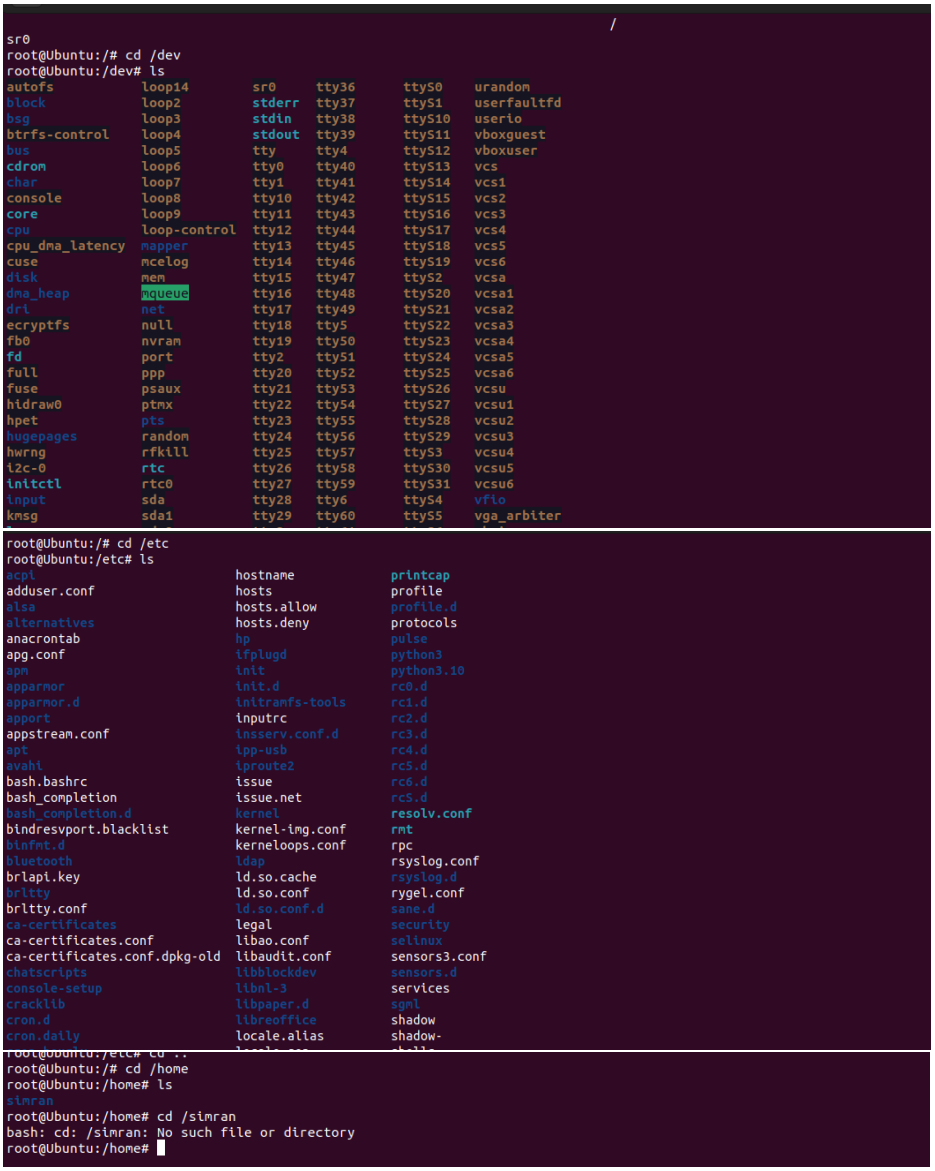
Types of File system in Linux:

- Ext3
- Ext4
- XFS
- FAT

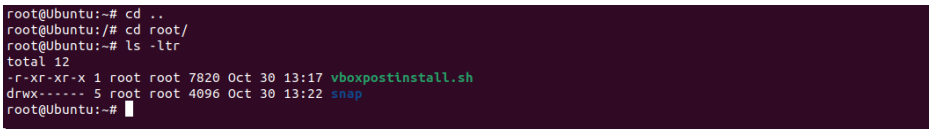


File System Structure:

- Root Directory (/): The top-level directory in the Linux file system.
- /bin: Contains essential system binaries (commands) accessible to all users.
- /etc: Configuration files for the system and installed applications.
- /home: Home directories for regular users.
- /var: Variable files like logs, temporary files, spool, mail, etc.
- /usr: User-related programs and files.
- /tmp: Temporary files.



- Now check Root Folder



Commands:

$: represents Normal user

#: represents root user

- To check the host name: hostname
- To check the user logged in right now: whoami
- To check IP : ip addr
- To know the present working directory: pwd

```
simran@Ubuntu: ~
simran@Ubuntu:~$ hostname
Ubuntu
simran@Ubuntu:~$ whoami
simran
simran@Ubuntu:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:3a:48:c2 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 83196sec preferred_lft 83196sec
    inet6 fe80::b248:5ac0:fb9c:1231/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
simran@Ubuntu:~$ pwd
/home/simran
```

- To list the file and folders/directories: ls
- To create a folder/directory: mkdir folder1
- Switch inside the directory which you have created : cd folder1
- To see what's inside the folder: ls

```
simran@Ubuntu:~$ ls
Desktop    Downloads  Pictures  snap       Videos
Documents  Music      Public    Templates
simran@Ubuntu:~$ mkdir folder1
simran@Ubuntu:~$ ls
Desktop    Downloads  Music     Public  Templates
Documents  folder1    Pictures  snap    Videos
simran@Ubuntu:~$ cd folder1
simran@Ubuntu:~/folder1$ pwd
/home/simran/folder1
simran@Ubuntu:~/folder1$
```

- Now let's create another folder inside this directory.
  Mkdir folder2

```
simran@Ubuntu:~/folder1$ mkdir folder2
simran@Ubuntu:~/folder1$ ls
folder2
simran@Ubuntu:~/folder1$ pwd
/home/simran/folder1
simran@Ubuntu:~/folder1$
```

- Now lets get outside of this folder: cd ..
- If directly wants to go in folder2: cd folder1/folder2
- Want to come outside directly to home location: cd

```
simran@Ubuntu:~/folder1$ cd ..
simran@Ubuntu:~$ pwd
/home/simran
simran@Ubuntu:~$ cd folder1/folder2
simran@Ubuntu:~/folder1/folder2$ pwd
/home/simran/folder1/folder2
simran@Ubuntu:~/folder1/folder2$ cd
simran@Ubuntu:~$
```

- To find a folder if multiple folders are there: find folder1/ -name folder2
- Now if there are directories with different names present in folder1

```
/home/simran/folder1/folder2
simran@Ubuntu:~/folder1/folder2$ cd
simran@Ubuntu:~$ find folder1/ -name folder2
folder1/folder2
simran@Ubuntu:~$ cd folder1
simran@Ubuntu:~/folder1$ ls
folder2
simran@Ubuntu:~/folder1$ mkdir media
simran@Ubuntu:~/folder1$ mkdir folder3
simran@Ubuntu:~/folder1$ mkdir folder4
simran@Ubuntu:~/folder1$ ls
folder2  folder3  folder4  media
simran@Ubuntu:~/folder1$ ls folder*
folder2:

folder3:

folder4:
simran@Ubuntu:~/folder1$
```

- Creating a file: touch file1
- Now wants to find this file then,

```
simran@Ubuntu:~/folder1$ cd folder2
simran@Ubuntu:~/folder1/folder2$ touch file1
simran@Ubuntu:~/folder1/folder2$ cd ..
simran@Ubuntu:~/folder1$ cd..
cd..: command not found
simran@Ubuntu:~/folder1$ cd ..
simran@Ubuntu:~$ find folder1/ -name file1
folder1/folder2/file1
simran@Ubuntu:~$
```

- How to delete a folder/directory: rmdir folder2
- Delete a directory with all of its files: rm -r media

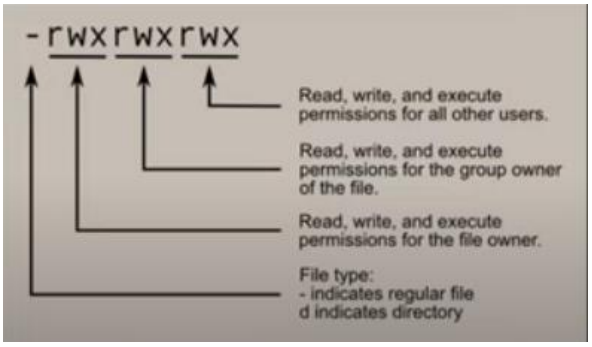
```
simran@Ubuntu:~$ cd folder1
simran@Ubuntu:~/folder1$ ls
folder2  folder3  folder4  media
simran@Ubuntu:~/folder1$ rmdir folder2
rmdir: failed to remove 'folder2': Directory not empty
simran@Ubuntu:~/folder1$ rm -r folder2
simran@Ubuntu:~/folder1$ ls
folder3  folder4  media
simran@Ubuntu:~/folder1$ rmdir media
simran@Ubuntu:~/folder1$ ls
folder3  folder4
simran@Ubuntu:~/folder1$
```

- To list file in order of modification dates: ls -lt
- To see reverse order of modification date: ls -ltr
- To edit the file use VIM editior
- Command: vi file2
    - Insert mode: press ‘I’
    - Exit Mode: esc+ :wq
- See whats inside the file file2: cat file2

```
simran@Ubuntu:~/folder1$ vi file2
simran@Ubuntu:~/folder1$ cat file2
Hello there
simran@Ubuntu:~/folder1$
```

- To see full details: ls-ltr

```
simran@Ubuntu:~/folder1$ ls -ltr
total 12
drwxrwxr-x 2 simran simran 4096 Oct 31 15:30 folder3
drwxrwxr-x 2 simran simran 4096 Oct 31 15:30 folder4
-rw-rw-r-- 1 simran simran   12 Oct 31 15:41 file2
simran@Ubuntu:~/folder1$
```

- 

Practice commands

Switch to root user and where as an Other user we created the file there create the another file but now as an root user

```
root@Ubuntu:/home/simran/folder18# ls
audio  folder1  folder2  folder3  media
root@Ubuntu:/home/simran/folder18# cd folder1
root@Ubuntu:/home/simran/folder18/folder1# ls
file18
root@Ubuntu:/home/simran/folder18/folder1# touch file2
root@Ubuntu:/home/simran/folder18/folder1# ls -l
total 0
-rw-rw-r-- 1 simran simran 0 Oct 31 17:51 file18
-rw-r--r-- 1 root   root   0 Nov  1 14:20 file2
root@Ubuntu:/home/simran/folder18/folder1# cd
root@Ubuntu:~# exit
exit
simran@Ubuntu:~$
```

Now exit from root user and as an other user try to delete that file2 created by Root user.It will show permission denied.

So file created by root user, we can't do any change in that file

```
simran@Ubuntu:~/folder18/folder1$ ls -l
total 0
-rw-rw-r-- 1 simran simran 0 Oct 31 17:51 file18
-rw-r--r-- 1 root   root   0 Nov  1 14:20 file2
simran@Ubuntu:~/folder18/folder1$ rm file2
rm: remove write-protected regular empty file 'file2'?
simran@Ubuntu:~/folder18/folder1$ vi file2
simran@Ubuntu:~/folder18/folder1$
```

To see the file content we can use cat command or less command

Cat file18 or less file18

```
simran@Ubuntu:~/folder18/folder1$ cat file18
hello There!
simran@Ubuntu:~/folder18/folder1$ less file18
simran@Ubuntu:~/folder18/folder1$
simran@Ubuntu:~/folder18/folder1$ less file18
simran@Ubuntu:~/folder18/folder1$
simran@Ubuntu:~/folder18/folder1$
```

Use of word count commands: wc file18

1 is for 1 line is present

2 is for number of words

13 is for number of total characters

```
simran@Ubuntu:~/folder18/folder1$ wc file18
 1  2 13 file18
simran@Ubuntu:~/folder18/folder1$
```

To compare the file we use diff command

Diff file18 file181

```
simran@Ubuntu:~/folder18/folder1$ touch file181
simran@Ubuntu:~/folder18/folder1$ vi file181
simran@Ubuntu:~/folder18/folder1$ diff file18 file181
1c1
< hello There!
---
> Hello There What a beautiful day it is!
simran@Ubuntu:~/folder18/folder1$
```

Now how to compress and decompress the file(like zip or unzip)

Tar cvf files.tar file18 file181

Ls

```
simran@Ubuntu:~/folder18/folder1$ tar cvf files.tar file18 file181
file18
file181
simran@Ubuntu:~/folder18/folder1$ ls
file18  file181  file2  files.tar
simran@Ubuntu:~/folder18/folder1$
```

The command tar cvf specifically means:

- c: Create a new archive. This flag tells tar to create an archive file.
- v: Verbose mode. This flag provides detailed information about the files being processed, displaying the names of the files as they are added to the archive.
- f: File. This flag specifies the filename of the archive. The f flag should be followed by the name of the archive file.

Now compress this file

Gzip files.tar

```
simran@Ubuntu:~/folder18/folder1$ ls
file18  file181  file2  files.tar
simran@Ubuntu:~/folder18/folder1$ gzip files.tar
simran@Ubuntu:~/folder18/folder1$ ls
file18  file181  file2  files.tar.gz
simran@Ubuntu:~/folder18/folder1$ ls -ltr
total 12
-rw-r--r-- 1 root   root     0 Nov  1 14:20 file2
-rw-rw-r-- 1 simran simran  13 Nov  1 14:27 file18
-rw-rw-r-- 1 simran simran  40 Nov  1 14:32 file181
-rw-rw-r-- 1 simran simran 198 Nov  1 14:34 files.tar.g
simran@Ubuntu:~/folder18/folder1$
```

Now if we want to move this zip file from folder1 to folder5 which is present inside folder1

Cp files.tar.gz folder5/ (destination_folder)

```
simran@Ubuntu:~/folder18/folder1$ ls
file18  file181  file2  files.tar.gz  folder18
simran@Ubuntu:~/folder18/folder1$ mkdir folder5
simran@Ubuntu:~/folder18/folder1$ cp files.tar.gz folder5/
simran@Ubuntu:~/folder18/folder1$ cd folder5
simran@Ubuntu:~/folder18/folder1/folder5$ ls
files.tar.gz
simran@Ubuntu:~/folder18/folder1/folder5$
```

Unzip this file now

Gunzip files.tar.gz

Untar the file

Tar xvf files.tar

```
simran@Ubuntu:~/folder18/folder1/folder5$ gunzip files.tar.gz
simran@Ubuntu:~/folder18/folder1/folder5$ ls
files.tar
simran@Ubuntu:~/folder18/folder1/folder5$ tar xvf files.tar
file18
file181
simran@Ubuntu:~/folder18/folder1/folder5$ ls -ltr
total 20
-rw-rw-r-- 1 simran simran    13 Nov  1 14:27 file18
-rw-rw-r-- 1 simran simran    40 Nov  1 14:32 file181
-rw-rw-r-- 1 simran simran 10240 Nov  1 14:40 files.tar
simran@Ubuntu:~/folder18/folder1/folder5$
```

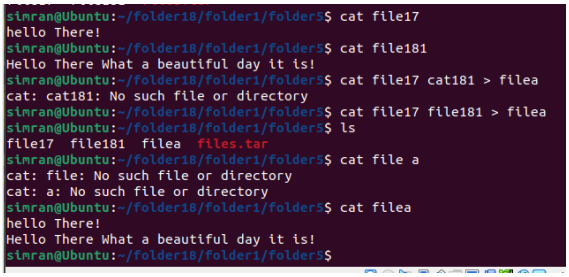
To rename the files

mv file18 file17 (move the file18 to file17 and delete the file18 now)

```
simran@Ubuntu:~/folder18/folder1/folder5$ mv file18 file17
simran@Ubuntu:~/folder18/folder1/folder5$ ls
file17  file181  files.tar
simran@Ubuntu:~/folder18/folder1/folder5$
```

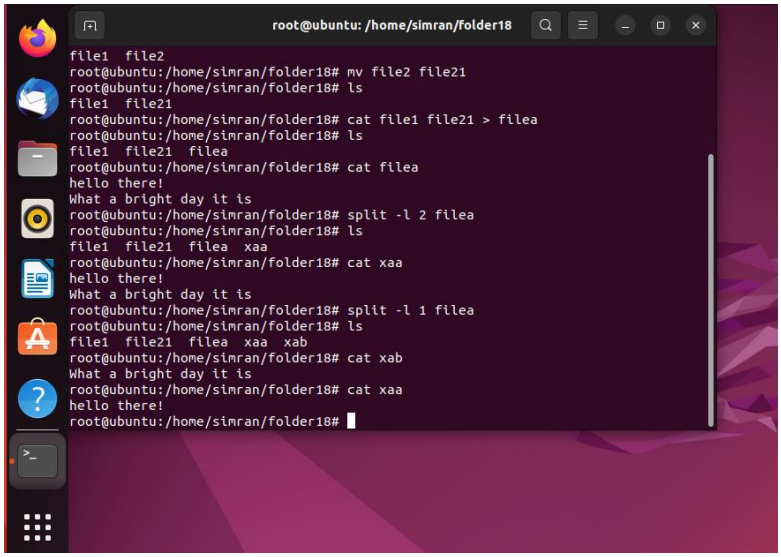
To add the content of 2 filess into third file

Cat file17 file181 > filea



To split the file into two parts

Split -l 1 filea (1 representd number of lines want to split)



Usage of RM command:

The `rm` command in Linux is used to remove files and directories. It's a powerful command, so be very careful when using it as deleted files are usually not recoverable. Here are several ways to use the `rm` command:

1. Remove a Single File

To delete a single file:

```

**rm filename**

```

For example, to delete a file named `example.txt`:

```
rm example.txt
```

2. Remove Multiple Files

You can remove multiple files at once:

```
rm file1 file2 file3
```

For example, to delete files named `file1.txt`, `file2.txt`, and `file3.txt`:

```
rm file1.txt file2.txt file3.txt
```

3. Remove Files Using Wildcards

Wildcards can be used to match multiple files based on a pattern:

```
rm *.txt
```

This command will remove all files with a ".txt" extension in the current directory.

4. Remove a Directory and Its Contents

To remove a directory and all its contents (including files and subdirectories):

```

**rm -r directory_name**

```

For example, to delete a directory named `folder` and all its contents:

```

**rm -r folder**

```

5. Prompt Before Removal

To be prompted before each file is removed, use the `-i` flag:

```

**rm -i filename**

```

For example:

```

**rm -i example.txt**

```

This will prompt for confirmation before deleting `example.txt`.

6. Force Removal (No Confirmation)

To force deletion without confirmation, use the `-f` flag:

```

**rm -f filename**

```

For example:

```
rm -f file.txt
```

This will delete `file.txt` without confirmation.

7. Verbose Output

To show a more detailed output while deleting files, use the `-v` flag:

```
rm -v filename
```

For example:

```
rm -v example.txt
```

This will display the process of removing `example.txt`.

#CAT Command:

The cat command (short for concatenate) is used to display the contents of one or multiple files, creating, or combining files.

Usage:

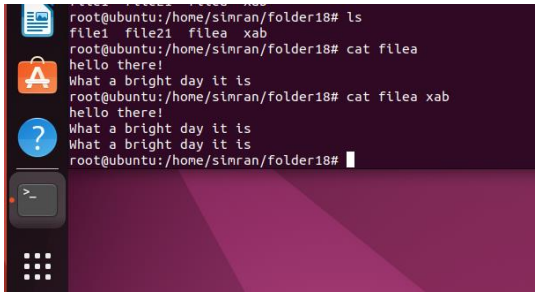
cat filea

This command displays the contents of the file named filename in the terminal.

bash

cat filea xab

This command displays the contents of both file1 and file2 in the order listed.



2. less Command

The less command is a **terminal pager** that allows you to view and navigate through the contents of a file.

Usage:

less filename

This command opens filename in a paginated view, allowing you to scroll up and down through the content.

3. more Command

Similar to less, the more command is **a terminal pager** used to display the contents of a file one screen at a time.

Usage:

more filename

This command displays the content of the file filename one page at a time. Press the spacebar to view the next page and press q to exit.

4. head Command

The head command is used to display the beginning lines of a file.

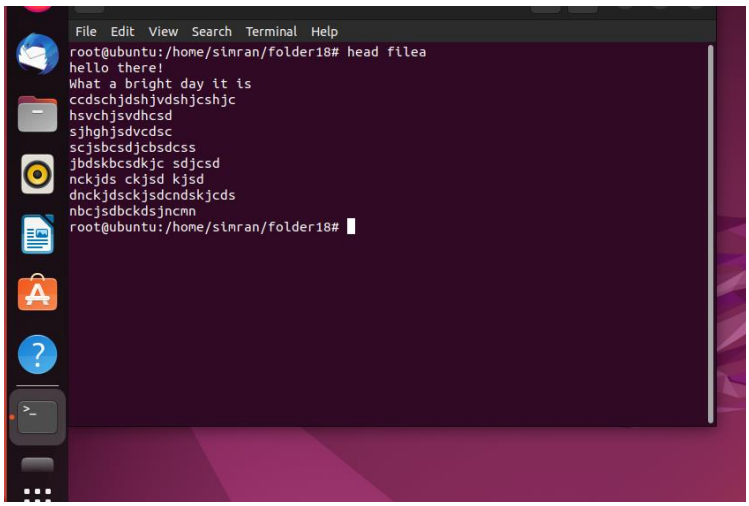
Usage:

head filename

This command shows the first ten lines of the file filename by default. You can also specify the number of lines to display using the -n option. For example:

head -n 20 filename

This will display the first 20 lines of the file.



5. tail Command

The tail command is used to display the ending lines of a file.

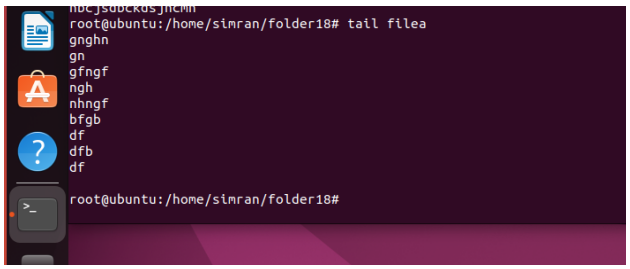
Usage:

tail filename

This command displays the last ten lines of the file filename by default. Similar to head, you can specify the number of lines to display using the -n option. For example:

tail -n 15 filename

This will display the last 15 lines of the file.



Remember, these commands are used to view or display the content of files in the terminal. They don't modify the files themselves. They are useful for reading or quickly viewing the content of files directly within the terminal.

Permissions in FileSystem:

Ubuntu (and other Linux-based systems), the chmod command is used to change the permissions of files and directories. It allows users to modify the read, write, and execute permissions for the owner, group, and others. Here's a detailed breakdown of how to use chmod:

Basic Permissions

Each file and directory in Linux has permissions represented by three categories: owner, group, and others. These permissions can be set as **read (r), write (w), and execute (x)**. These permissions are represented numerically as well: **read (4), write (2), and execute (1).**

chmod Syntax

The chmod command follows this syntax:

chmod who=permissions file

who: Represents who the permission change will apply to and can be:

u (user/owner)

g (group)

o (others)

a (all/ugo - user, group, others)

permissions: Represents the permissions to be set, using combinations of:

r (read) 4

w (write) 2

x (execute) 1

file: Specifies the file or directory whose permissions are to be changed.

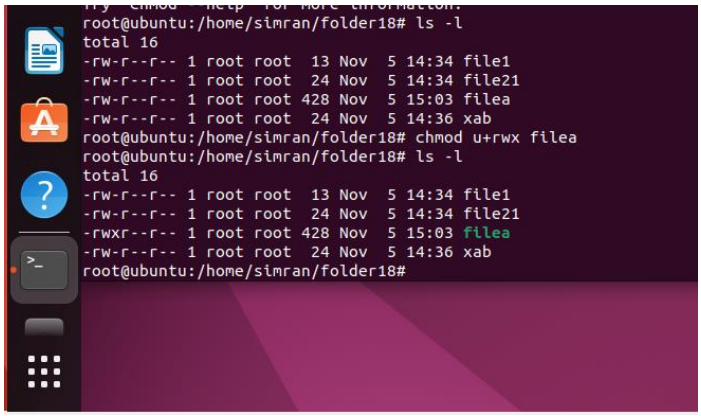
Usage Examples:

Changing File Permissions

Symbolic Mode:

To give the owner of the file read, write and execute permissions:

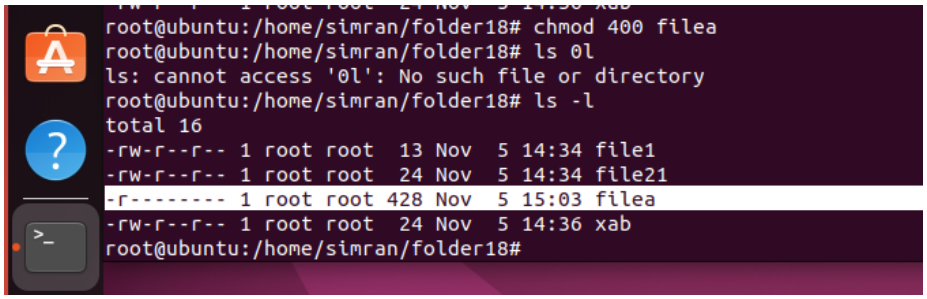
Chmod u+rwx filea



Numeric Mode:

To give the owner only read permission:

Chmod 400 filea

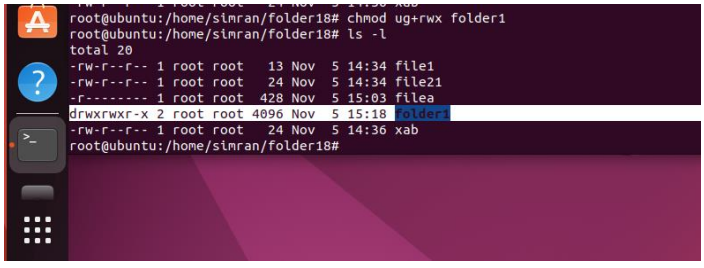


Changing Directory Permissions

Symbolic Mode:

To give the owner and group read, write, and execute permissions:

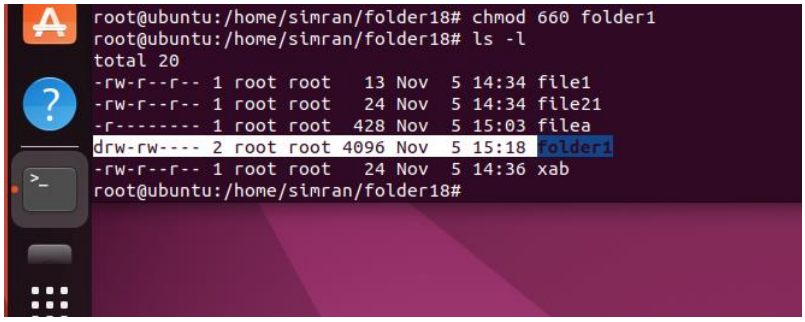
Chmod ug+rwx folder1



Numeric Mode:

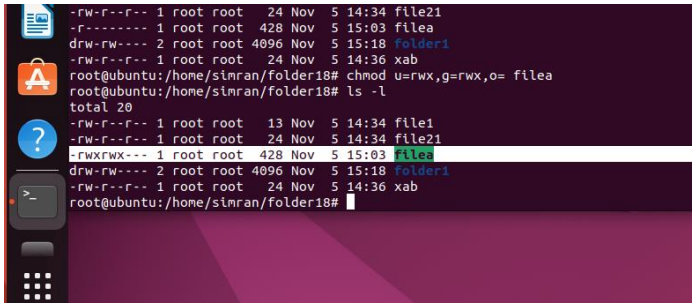
To give the owner and group read and write permissions:

Chmod 660 folder1



Permission Combination Examples

Symbolic Representation:

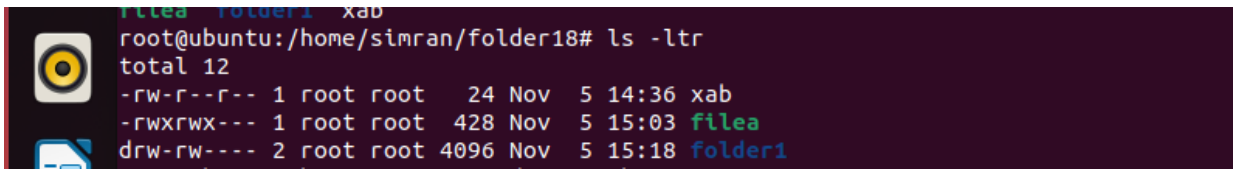


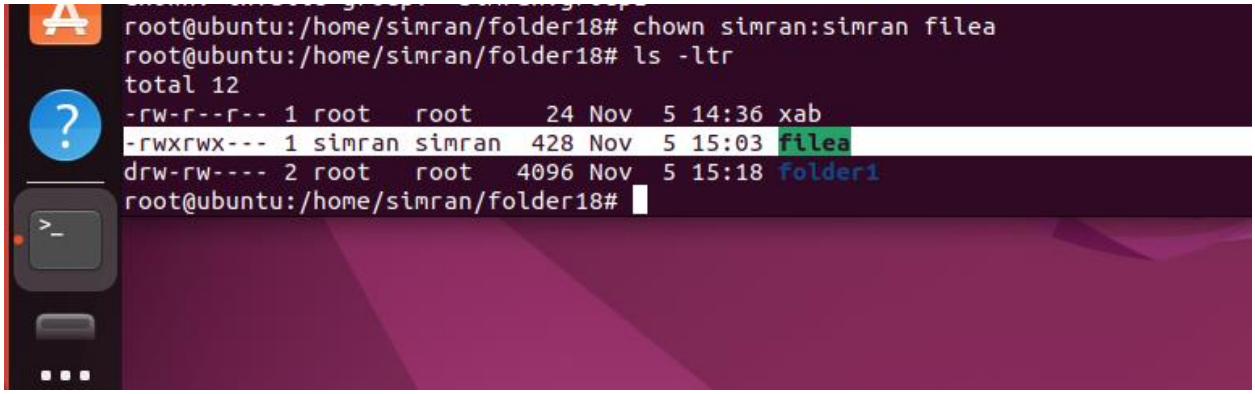
The **chmod** command often requires administrative permissions (use sudo or be logged in as the owner or with relevant permissions)

CHOWN Command: (Change ownership)

Chown owner:group filename

Chown simran:simran filea





Linux File Types:

File Symbol	File Type
-	Regular file
d	Directory
l	Link
c	Device File
s	Socket
p	FIFO or Named Pipe
b	Block device

In Linux, files are categorized into several types based on their nature and purpose. Here are some common file types:

1. Regular Files (Regular Text Files):

- These are the most common type of files.

- They contain data, such as text files, binary files, and program executables.

```

$ ls -l
```

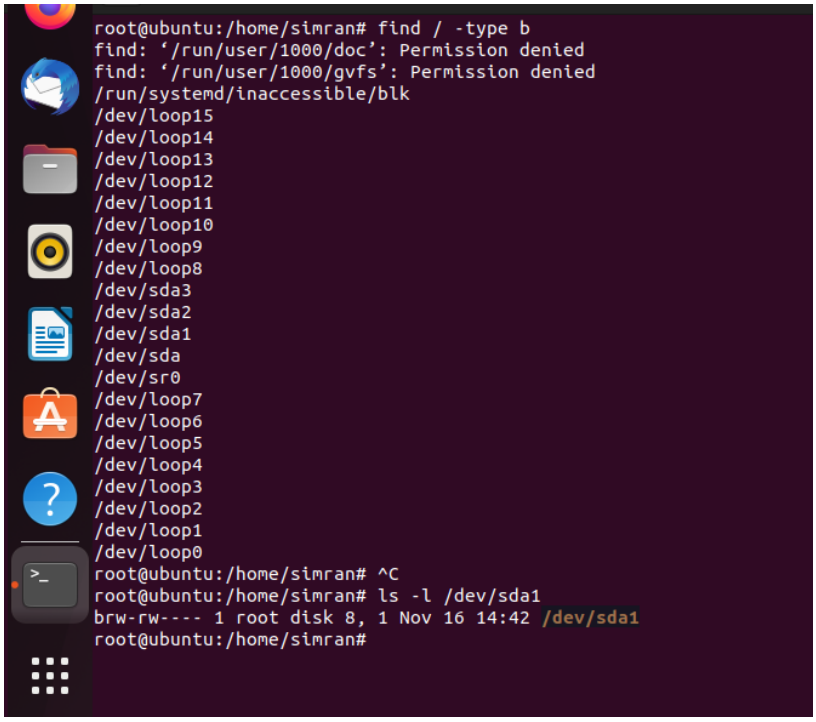
-rw-r--r-- 1 user user 12345 Nov 16 10:00 example.txt

```

**2. Directories:**

- Directories contain lists of file names and their corresponding inode numbers.

- They are analogous to folders in other operating systems.

```
$ ls -l

drwxr-xr-x 2 user user 4096 Nov 16 09:59 my_directory/
```

**3. Symbolic Links (Soft Links):**

- Symbolic links are shortcuts or pointers to another file or directory.

- They are similar to Windows shortcuts.

```
$ ls -l

lrwxrwxrwx  1 user user     8 Nov 16 10:02 link_to_file -> myfile
```

4. Block Special Files:

- These files provide buffered input/output for device drivers.

- They are often used for accessing devices like hard drives.

```
$ ls -l

brw-rw----  1 root disk  8,   1 Nov 16 10:04 sda1
```
```



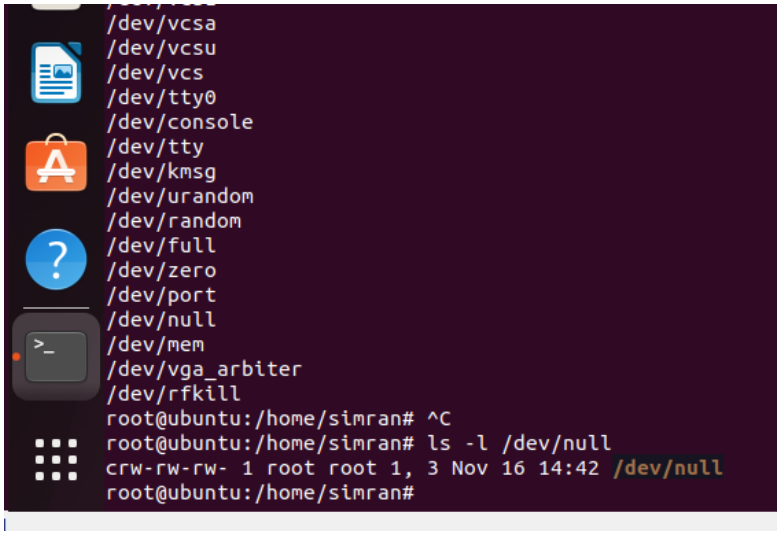
5. Character Special Files:

- Similar to block special files, but provide unbuffered access.

- Often used for devices like terminals.

- Files that read/write data character by character

```
$ ls -l
crw-rw-rw-  1 root tty   5,   1 Nov 16 10:06 tty1
```



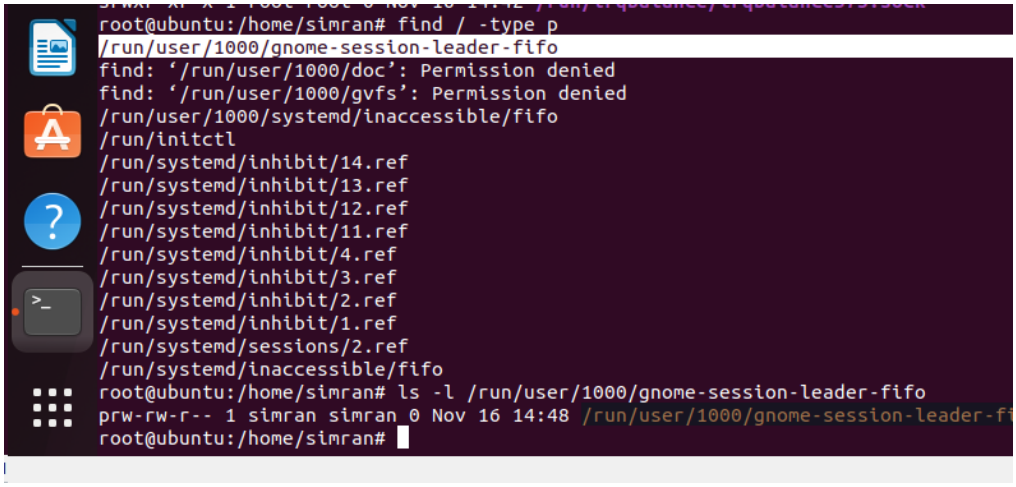
6. FIFO (Named Pipes):

- FIFOs are special files used for communication between processes.

- They are named pipes.

- data getting transferred between processes in FIFO manner

```
$ ls -l

prw-r--r--  1 user user     0 Nov 16 10:08 my_pipe
```

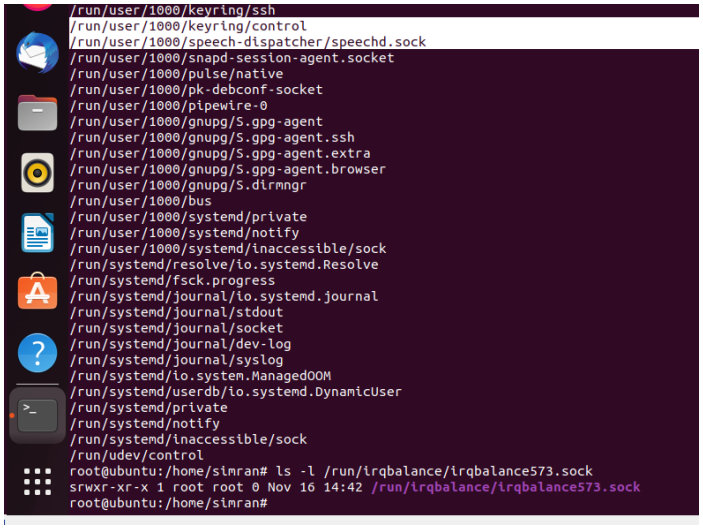


7. Sockets:

- Sockets are special files used for inter-process communication.

- They enable communication between processes on the same or different machines.

```
$ ls -l

srwxr-xr-x  1 user user     0 Nov 16 10:10 my_socket
```



8. Device Files:

- Device files represent physical and virtual devices on the system.

- They are found in the `/dev` directory.

```
$ ls -l /dev/sda

brw-rw----  1 root disk  8,   0 Nov 16 10:12 /dev/sda
```

These are the main file types in Linux, and they serve various purposes in the system. The output of the `ls -l` command provides information about file permissions, ownership, size, and modification time.

#Link Files in Linux:

In Linux, a "link" typically refers to two types of links: symbolic links (soft links) and hard links.

1. Symbolic Links (Soft Links):

- A symbolic link is a separate file that acts as a pointer or reference to another file or directory.

- It is similar to a Windows shortcut.

- Symbolic links can span different file systems and can link to files or directories that may not exist.

Creating a symbolic link:

```bash
ln -s target_file link_name
```

Example:

```bash
ln -s /path/to/target/file /path/to/link
```

Example Output:

```bash
$ ls -l
lrwxrwxrwx  1 user user     8 Nov 16 10:02 link_to_file -> myfile
```

2. Hard Links:

- A hard link is a reference to the same inode (data and disk block location) as the original file.

- Changes to the data in one file are reflected in all hard links.

- Hard links cannot span different file systems, and they must be on the same file system as the original file.

Creating a hard link:

```bash
ln target_file link_name
```

Example:

```bash
ln /path/to/target/file /path/to/link
```

```

**Example Output:**

```bash

$ ls -l

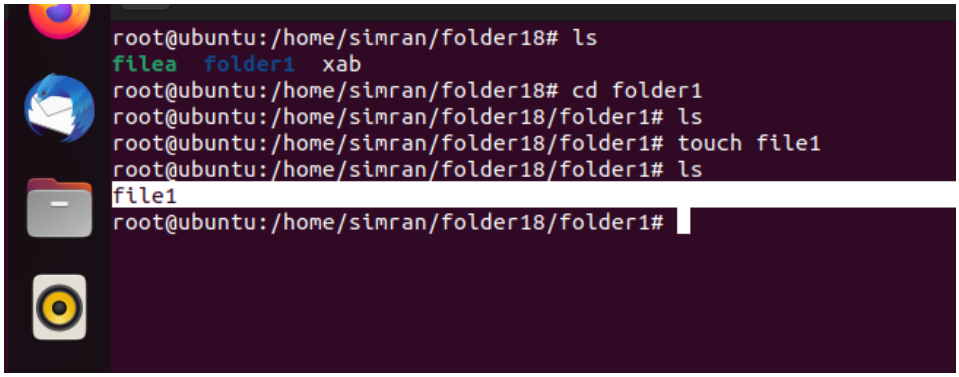
-rw-r--r--  2 user user  12345 Nov 16 10:00 myfile

```

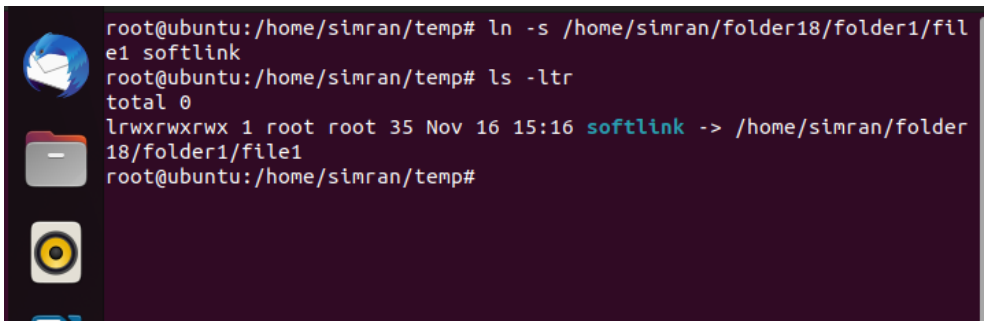
It's important to note that when the last hard link to a file is deleted, the data is not immediately deleted from the disk. It is only freed when there are no more references to the inode, which happens when all links to the file are removed.

Practical:

Let’s say we have a file1 inside a folder1

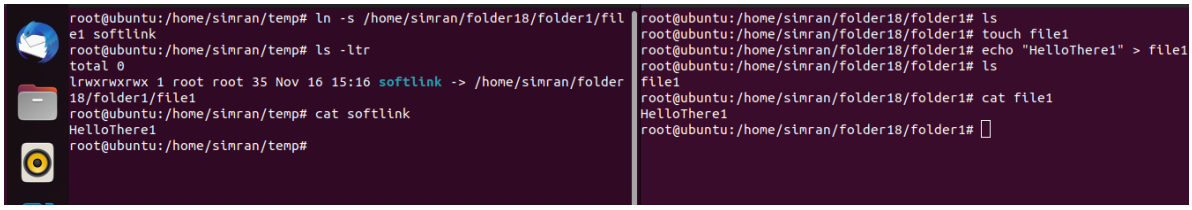


- Now we want to create a soft link (shortcut) for this file1
- Inside a temp folder we will create a soft link

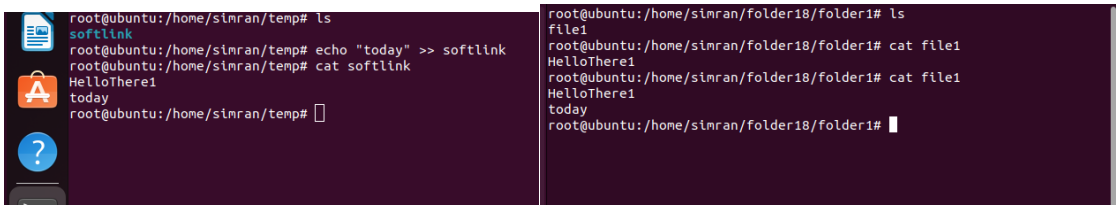


Now ill go to the folder1 where our file1 exists originally and make some changes in the file at that location.

You will see the change you are making by staying in the original location those changes are getting reflected in the softlink as well
```



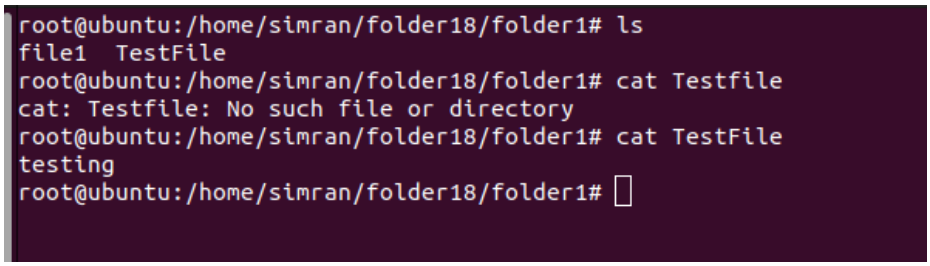
Also the changes you make in shortcuts are visible in the file1 present in original location.



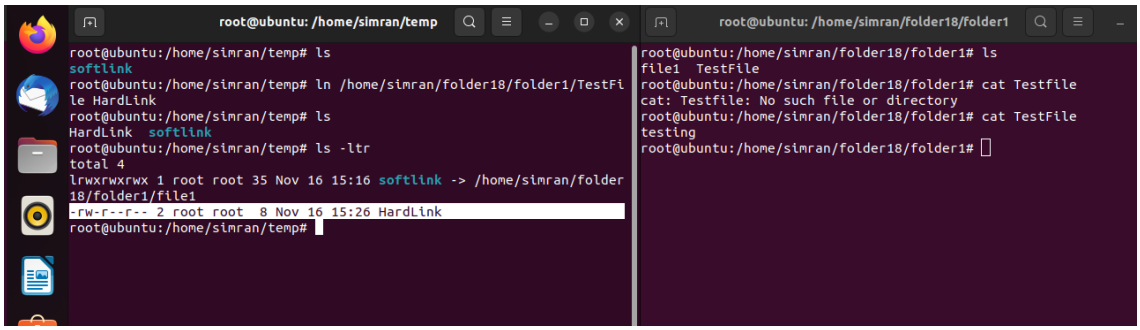
If you delete the original file the softlink will also be deleted

HardLink:

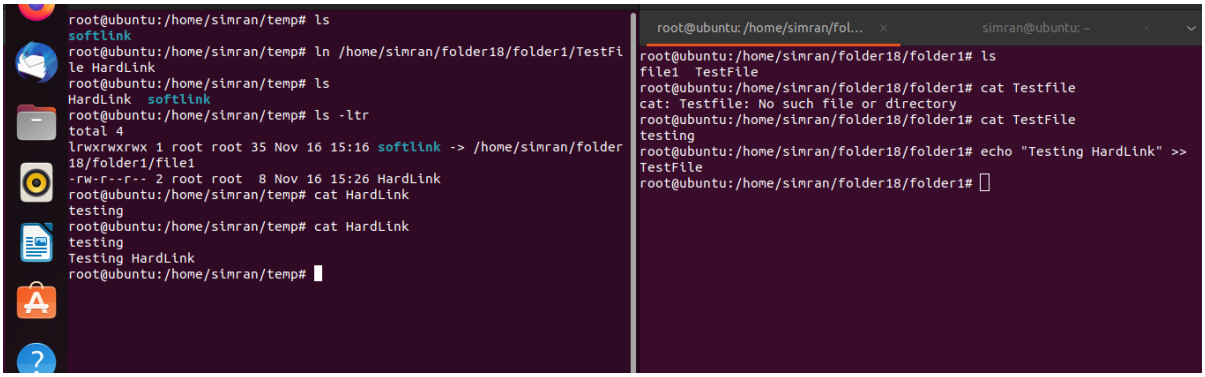
First create a test File inside Folder1 TestFile



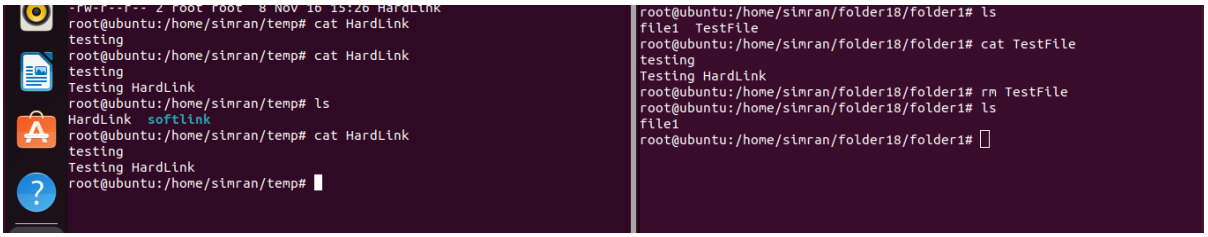
Now wil create a hardlink



- Hardlink will appear to you as a normal file only
- Adding some content in the Test File present in original location
- You will see whatever you are updating in Test file present in the original location those same changes are getting reflected in the Hardlink file as well this works same as your soft file.



- Now let’s try to delete the original file



- Content is still there in the Hardlink Fill even when we have deleted the original File