Linux Core Concepts

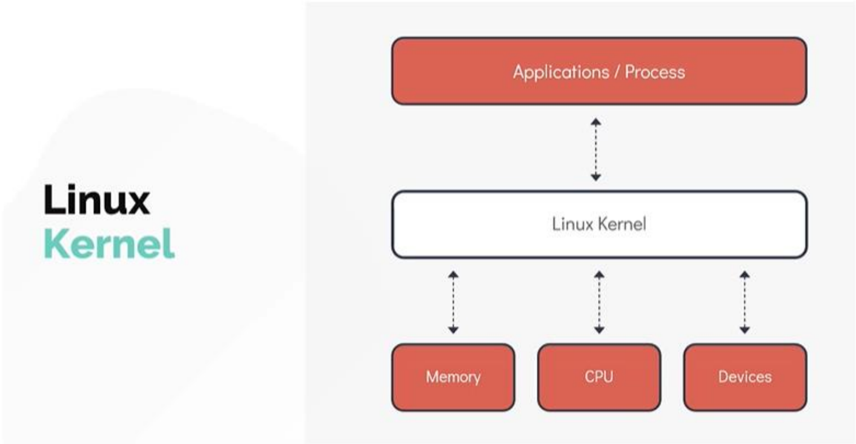
In this section, we will take a look at the core concepts of a linux operating system.

- We will start with introduction to the linux kernel.
- We will then learn about the kernel space and user space.

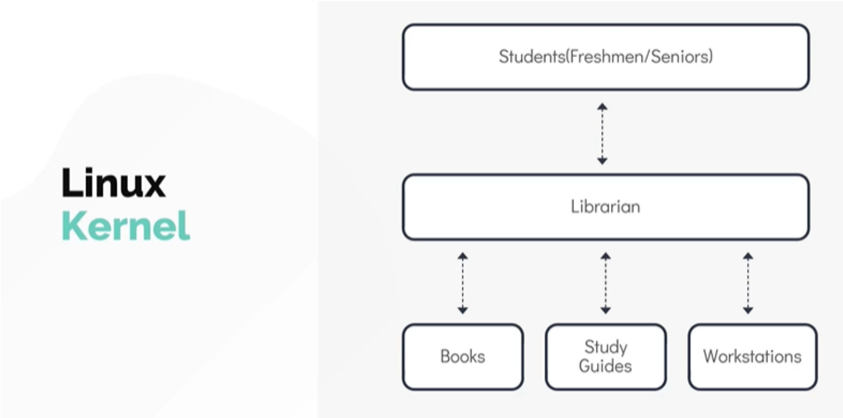
Linux Kernel

If you have worked with any operating system, you have run into the term kernel.

- The Linux kernel is monolithic, this means that the kernel carrries out CPU scheduling, memory management and several operations by itselfs.
- The Linux Kernel is also modular, which means it can extends its capabilities through the use of dynamically loaded kernel modules



To understand a kernel in simple terms, let us use an analogy of a `College Library`. Here the librarian is equal to Linux Kernel.



The Kernel is responsible for 4 major tasks

1. Memory Management
2. Process Management
3. Device Drivers
4. System calls and Security

Linux Kernel Versions

let us know identify the ways to identify linux kernel versions

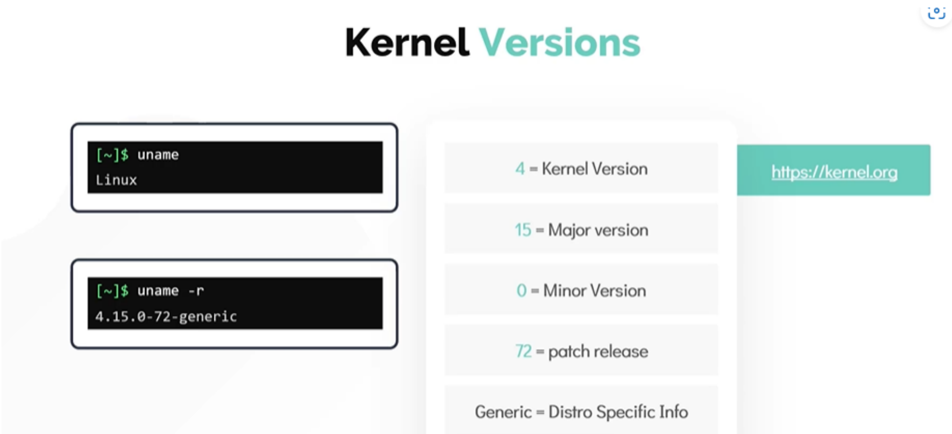
Use `uname` command to get the information about the kernel (by itself it doesn't provide much information except that the system uses the `Linux` Kernel.

```
$ uname
```

Use the `uname -r` or `uname` comamnd and option to print the kernel version

$ uname -r

$ uname -a

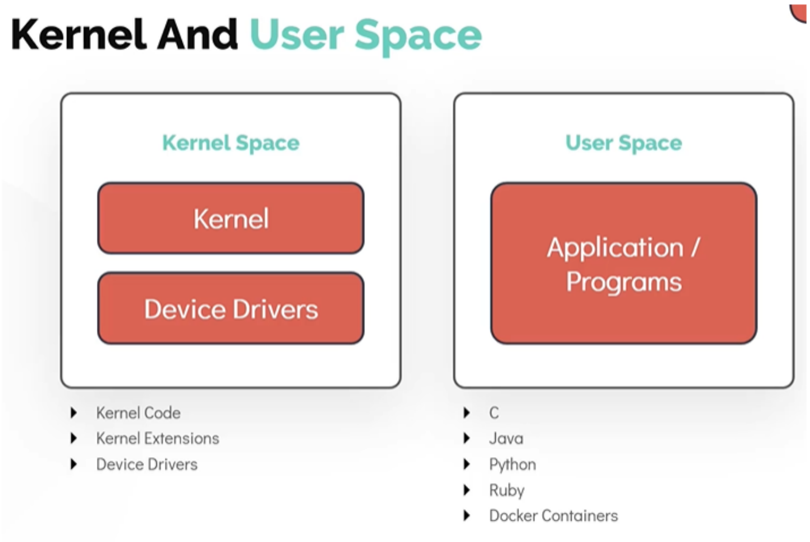


Kernel and User Space

One of the important functions of the linux kernel is the `Memory Management` . We will now see how memory is seperated within the linux kernel

Memory is divded into two areas.

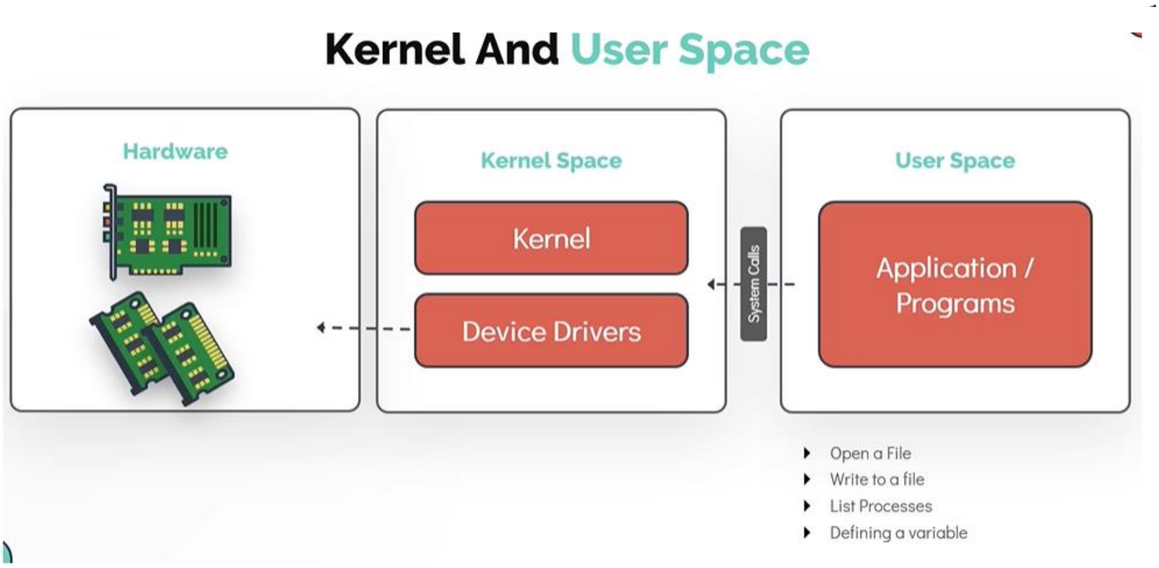
1. Kernel Space
    i. Kernel Code
    ii. kernel Extensions
    iii. Device Drivers
2. User Space
    i. C
    ii. Java
    iii. Python
    iv. Ruby e.t.c
    v. Docker Containers



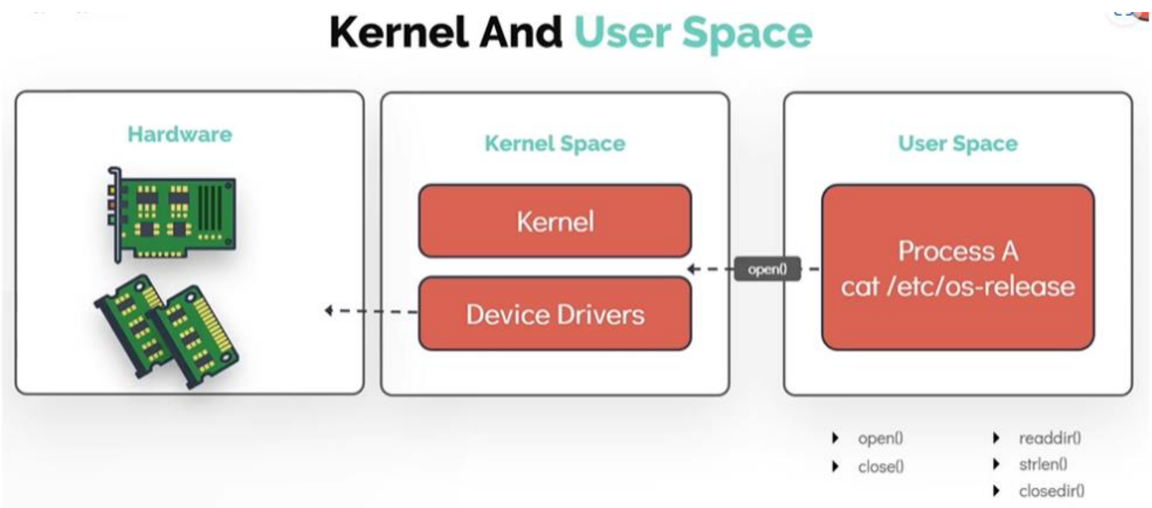
Let us know see how programs running in the `User Space` work

All user programs function by manipulating data that is stored in memory and on disk. User programs get access to data by making special request to the kernel called `System Calls`

- Examples include, allocating memory by using variables or opening a file.



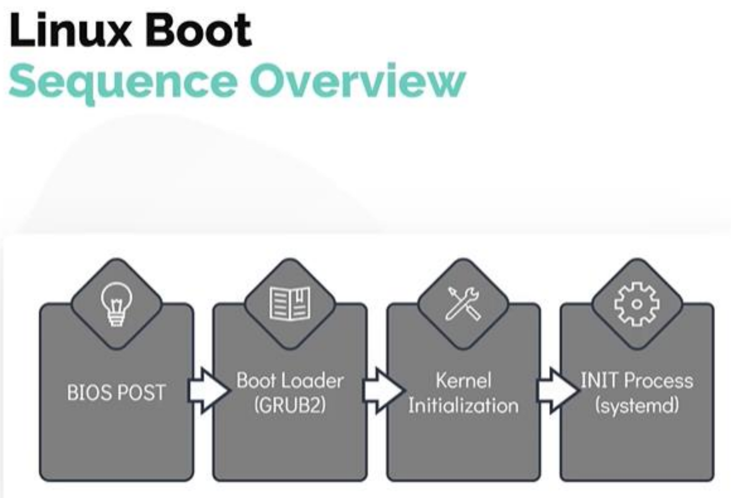
- For example, opening a file such as the `/etc/os-release` to see the operating system installed, results in a `system call`



Linux Boot Sequence

In this section, we look at the boot process in a simplied manner by dividing it into four broader steps.

- The boot process can be broken down into four stages
  1. BIOS POST
  2. Boot Loader (GRUB2)
  3. Kernel Initialization
  4. INIT Process



How to initiate a linux boot process?

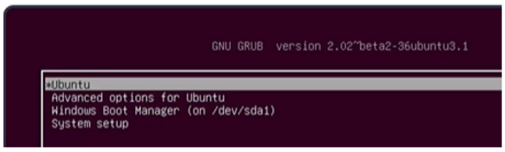
- This can be achieved in one of the two ways.
  - The first method is to start a linux device which is in a halted or stopped state
  - Second method is to reboot or reset a running system

BIOS POST

- The first stage, called `BIOS POST` has very little to do with linux itself.
- `POST` Stands for `Power On Self Test`.
- In this stage, BIOS runs a POST test, to ensure the hardware components that are attached to the device are functioning correctly, if POST fails the computer may not be operable and the system will not be proceed to next stage of the boot process

Boot Loader

- The next stage after BIOS POST is `Boot Loader` after successful of POST test.
- BIOS loads and executes the boot code from the boot device, which is located in the first sector of the harddisk. In Linux this is located in the `/boot` file system.
- The boot loader will provide the user with the boot screen, often with multiple options to boot into. Such as Microsoft windows O.S or Ubuntu 18.04 O.S in an example of a dual boot system.
- Once the selection is made at the boot screen, boot laoder loads the kernel into the memory supplies it with some parameters and handsover the control to kernel
- A popular example of the boot loader is `GRUB2` (GRand Unified Bootloader Version 2). Its a primary boot loader now for most of the operating system.



Kernel Initialization

- After the selected kerenl is selected and loads into the memory, it usually decompress and then loads kernel into the memory.
- At this stage, kernel carries out tasks such as initializing hardware and memory management tasks among other things.
- Once it is completely operational , kernel looks for `INIT Process` to run. Which sets up the `User Space` and the process is needed for the environment.

INIT Process

- In most of the current day linux distribution, the `INIT` function then calls the `systemd` daemon.
- The `systemd` is responsible for bringing the linux host to usable state.
- `systemd` is responsible for mounting the file systems, starting and managing system services.
- `systemd` is the universal standard these days, but not too long ago another initialization process called `system V (five) init` was used. It is also called ** `Sys5`
  - For example these were used in `RHEL 6` or `CentOS 6` distribution
- Once of the key advantages of using `systemd` over `system V(five) init` is that it reduces the system startup time by parallelizing the startup of services.

To check the `init` system used run `ls -l /sbin/init`, if it is systemd then you will see a pointer to `/lib/systemd/systemd`

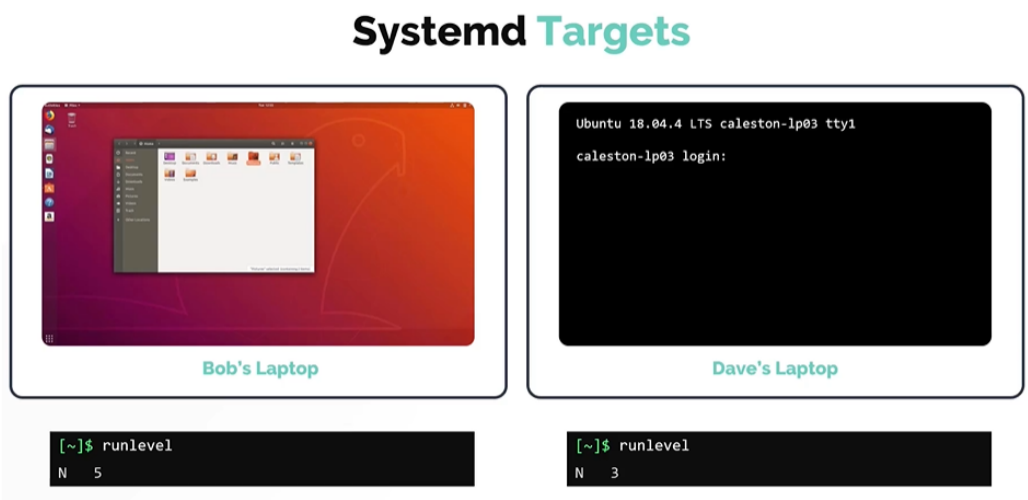
```
$ ls -l /sbin/init
```

Run Levels

We can setup the server to boot either into graphical mode or non-graphical mode. Linux can run in multiple modes and these modes are set by something called `runlevel`

- The operation mode which provide a graphical interface is called `runlevel 5`

- The operation mode which provide a non-graphical mode is called `runlevel 3`

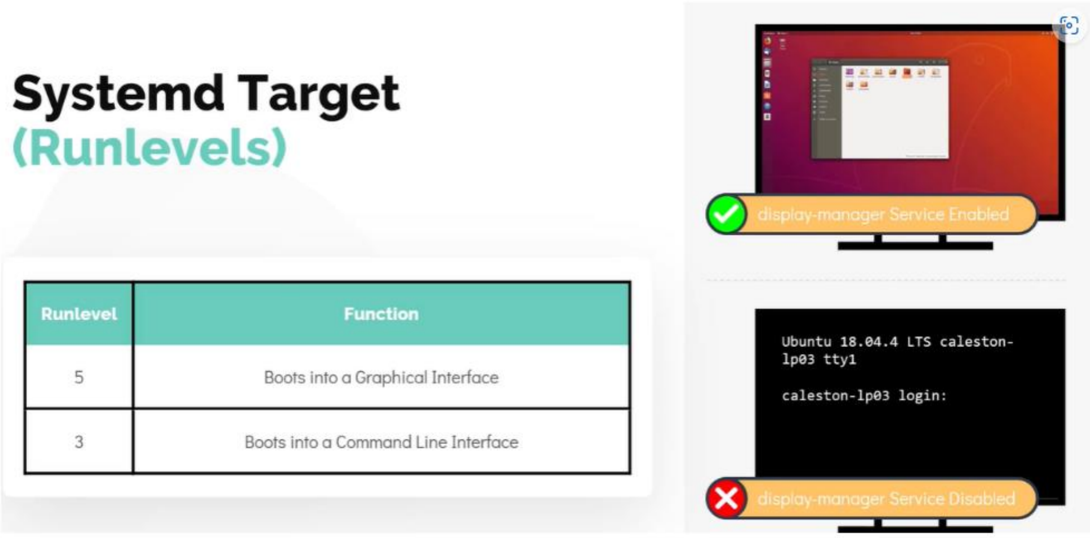


To see the operation mode run in the system. Run the command `runlevel` from the terminal

```
$ runlevel
```

During boot, the `init` process checks the `runlevel`, it make sure that all programs need to get the system operation in that mode are started.

- For example: The `Graphical User` mode requires a `display manager` service to run for the GUI to work, however this service is not required for the `non-graphical mode`



In the boot process section, we saw that the `systemd` is used as the `init` process in most new linux distributions suchs as `Ubuntu 18.04`.

- In `systemd`, runlevels are called as `targets`.
  - The RunLevel 5 is called as the `graphical target`
  - The Runlevel 3 is called as the `multiuser target`

Now that we are familiar with runlevels in systemd target unit. Lets now take a look at how we change these values from a shell.

To see the default target, run the command `systemctl get-default`. This command looks at the file located at `/etc/systemd/system/default.target`

```
$ systemctl get-default
```

To change the default target, we can make use of `systemctl set-target <desired target name goes here as an argument>`

```
$ systemctl set-default multi-user.target
```

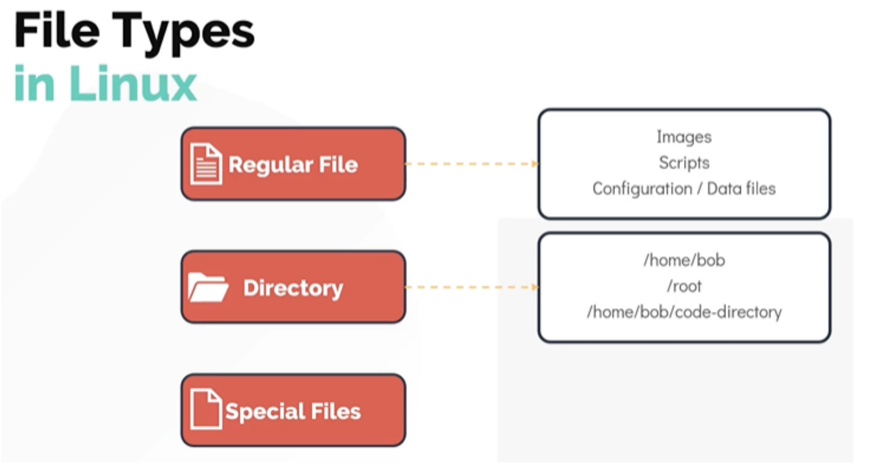
File Types in Linux

In this section, we will take a look at different types of files in linux.

- Everything is a file in Linux.
  - Every object in linux can be considered to be a type of file, even a directory for example is a special type of file.

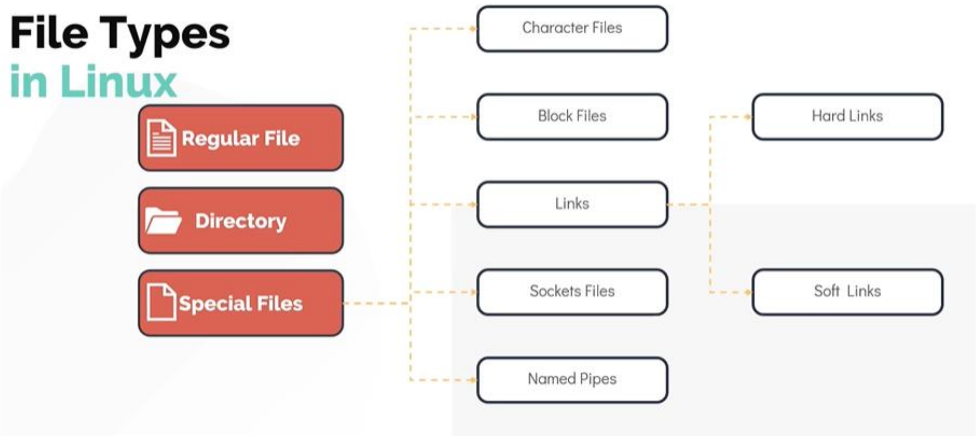
There are three types of files.

1. Regular File
2. Directory
3. Special Files



`Special files` are again catagorized into five other file types.

1. Character Files
   - These files represent devices under the `/dev` file system.
   - Examples include the devices such as the `keyboard` and `mouse`.
2. Block Files
   - These files represent block devices also located under `/dev/` file system.
   - Examples include the `harddisks` and `RAM`
3. Links
   - Links in linux is a way to associate two or more file names to the same set of file data.
   - There are two types of links
     - The Hard Link
     - The Soft Link
4. Sockets
   - A sockets is a special file that enables the communication between two processes.
5. Named Pipes
   - The Named Pipes is a special type of file that allows connecting one process as an input to another



Let us now see how to identify different file types in Linux.

One way to identify a file type is by making use of the `file` command.

```
$ file /home/michael
$ file bash-script.sh
$ file insync1000.sock
$ file /home/michael/bash-script
```

Another way to identify a file type is by making use of the `ls -ld` command

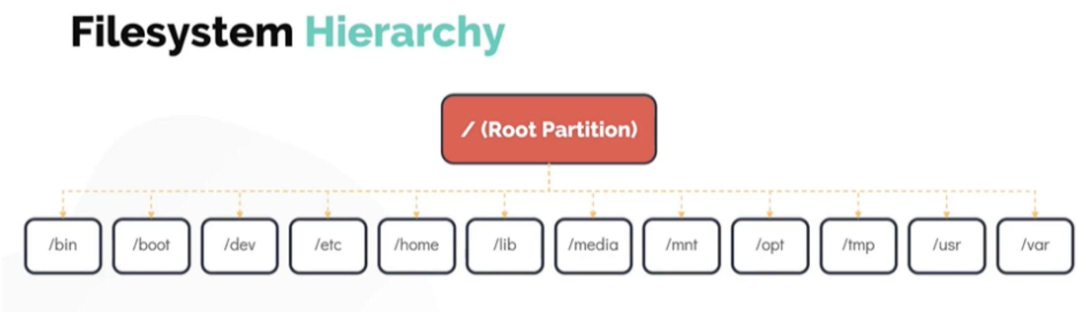
```
ls -ld /home/michael
ls -l basg-script.sh
```

File Type	Identifier
DIRECTORY	d
REGULAR FILE	-
CHARACTER DEVICE	c
LINK	l
SOCKET FILE	s
PIPE	p
BLOCK DEVICE	b

Filesystem Hierarchy

In this section, lets take a look at the `filesystem hierarchy`

- Linux uses single rooted, inverted tree like file system
    - `/home` : It is the location that contains the home directories for all users, except the `root` user (root user home directory is located at `/root`)
    - `/opt` : If you want to install any third party programs put them in the `/opt` filesystem.
    - `/mnt` : It is the default mount point for any partition and it is empty by default. It is used to mount filesystems temporarly in the system
    - `/tmp` : It is used to store temporary data
    - `/media` : All external media is mounted on `/media`
    - `/dev` : Contains the special block and character device files
    - `/bin` : The basic programs such as binaries `cp` , `mv` , `mkdir` are located in the `/bin` directory
    - `/etc` : It stores most of the configuration files in Linux.
    - `/lib` : The directory `/lib` and `/lib64` is the place to look for shared libraries to be imported into your program
    - `/usr` : In older systems, `/usr` directory is used for `User Home Directories` , however in the modern linux operating systems it is the location where all user land applciations in their data reside
    - `/var` : It contains variable data like mails, log files



To print all the mounted filesystems, run `df` (disk filesystem) command

```
$ df -hP
```