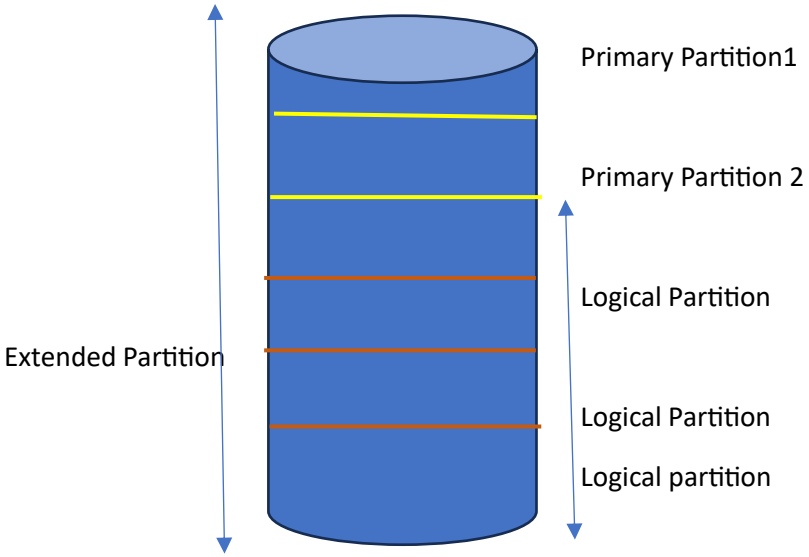
Partitioning

In Linux, partitioning refers to the process of dividing a physical hard drive into multiple logical storage units known as partitions. Each partition acts as a separate entity with its own file system, allowing users to organize and manage data more efficiently. Here's a detailed overview of partitioning in Linux:

Types of Partitions:

1. *Primary Partition:* These are the basic partitions on a hard disk and are used to boot an operating system. A maximum of four primary partitions can exist on a disk.
2. *Extended Partition:* If you need more than four partitions on a disk, you can create one extended partition, which serves as a container for additional logical partitions.
3. *Logical Partition:* These partitions exist within the extended partition and are used to further divide the extended partition into more than four partitions.

    - Primary Partition : which we use to install OS and users data
    - We can create only 4 partition in MBR
    - But if we want to create more partition then we go for Logical partitions



- We can’t install OS in Logical Partitions
- We actually create partition to effectively use the disk space
- Command to see Partitions: lsblk

File System Basics:

A file system is a method used by an operating system to store, retrieve, and manage files on a storage device.

It organizes data into files and directories, manages file access, and keeps track of where data is stored on the storage medium.

Common Linux File Systems:

1. **Ext2** (Second Extended File System): One of the earliest file systems used in Linux. It lacks journaling, making it more susceptible to data corruption in case of unexpected shutdowns.
2. **Ext3:** An improvement over Ext2, providing journaling functionality for better reliability. It's backward-compatible with Ext2.
3. **Ext4:** Successor to Ext3, offering improvements in performance, scalability, and reliability. It supports larger file systems and files, better resistance to fragmentation, and faster file system checks.
4. **Btrfs** (B-tree File System): A modern file system with advanced features like snapshots, integrated RAID, and better data integrity through checksums.
5. **XFS:** Designed for high-performance environments, XFS supports large files and volumes, high scalability, and efficient handling of concurrent read/write operations.
6. **FAT** (File Allocation Table): A simple file system commonly used on removable storage devices for compatibility with various operating systems.
7. **NTFS** (New Technology File System): Developed by Microsoft, NTFS is supported in Linux through third-party drivers. It provides features like journaling, file compression, and access control.

Features and Characteristics:

- **Journaling**: File systems like Ext3, Ext4, and XFS use journaling to track changes before they're actually committed, reducing the risk of data corruption in case of power failures or system crashes.
- **Permissions and Ownership:** File systems support access control mechanisms to determine who can read, write, or execute files, and which users or groups have ownership.
- **File Metadata:** File systems store metadata information such as file names, creation/modification dates, file size, and permissions.
- **Compression:** Some file systems, like Btrfs and NTFS, support file compression, reducing disk space usage.
- **Checksums and Data Integrity:** Modern file systems like Btrfs use checksums to ensure data integrity. If data becomes corrupted, these file systems can often repair it using redundant copies or checksums.

Choosing the Right File System:

- The choice of file system depends on various factors such as the specific use case, performance requirements, features needed, and compatibility.
- For the root file system, consider reliability, journaling, and performance. Ext4 is a commonly used and reliable option.

- For specific use cases like large-scale storage or where advanced features like snapshots and integrity checking are needed, Btrfs or XFS might be preferable.
- Compatibility with other operating systems might necessitate the use of FAT or NTFS for external drives.

Commands: fdisk -l or lsblk to see partitions in your machine

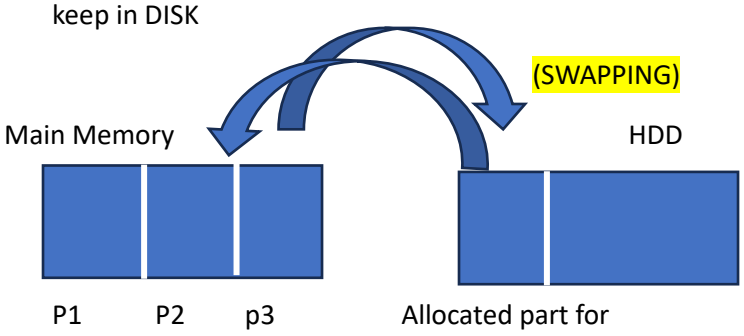
In the **lsblk** command's output in Ubuntu or other Linux systems, the "loop" type refers to loop devices or loopback devices. These devices are pseudo devices that allow a file to be accessed as a block device.

Loop devices are commonly used for mounting disk images, virtual filesystems, or any file that contains a filesystem within it. For instance, if you have an ISO file of an operating system or a disk image, a loop device can be used to access the content within the file as if it were a physical disk.

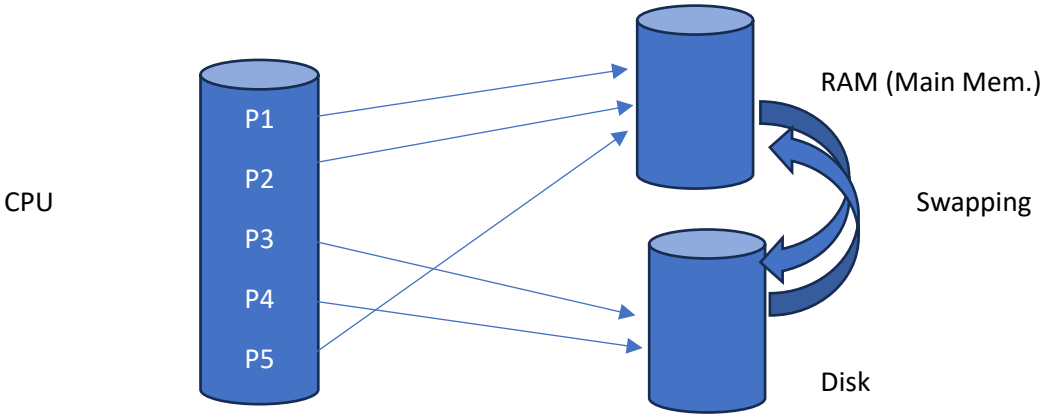
The loop devices allow the operating system to access and interact with the content of these files as if they were regular block devices or disks. They are versatile tools for working with disk images or virtual filesystems in Linux.

Virtual Memory:

- Virtual memory in Linux is a memory management technique that expands the available memory by allowing the operating system to use a combination of RAM (physical memory) and disk storage.
- When use access a program that needs more memory or extends the size of main memory then we need to use the concept of Virtual memory
- Transferring the data between Disk and Main Memory is called Swapping
- Here if we have a program of 16 GB and Main memory size is just 4 GB, then we keep the active portion of program in main memory and rest portion kept in Disk only
- Main idea of Virtual memory to keep active part of program in Main memory and rest to keep in DISK



- In Windows to implement virtual memory we have a file pagefile.sys
- In Linux to implement it we use Separate Partition



- CPU accessing some Here some processes which are present in RAM
- But if CPU access the Process P3 and P4 that are present in DISK then by swapping those process gets swapped in to Main Memory to get processed and any inactive processes present in RAM can be placed in DISK for some time
- This process we have in Virtual Memory

Partitioning:

Mountpoint: In Linux, a "mount point" refers to the location within the overall filesystem where a separate filesystem is attached for access. When you create partitions on a disk in Linux, each partition needs to be connected or "mounted" to a specific location in the overall file system hierarchy to be accessible and utilized.

To check the filesystem table: cat /etc/fstab

About /dev/loop:

When you use the **fdisk -l** command in Linux to list the available disks and their partitions, you might encounter entries with /dev/loop in the output. /dev/loop devices are special pseudo-devices that allow a file to be accessed as a block device. These devices are associated with loopback mounts, which enable you to mount filesystem images or files as if they were actual block devices or partitions.

The loop device enables you to mount disk images, ISO files, or other file-based storage as if they were physical partitions. This is often used for testing, virtualization, or running systems from disk images. When you see entries like /dev/loopX, where X is a number, it represents a specific loopback device.

m-Help

n- to create new partition

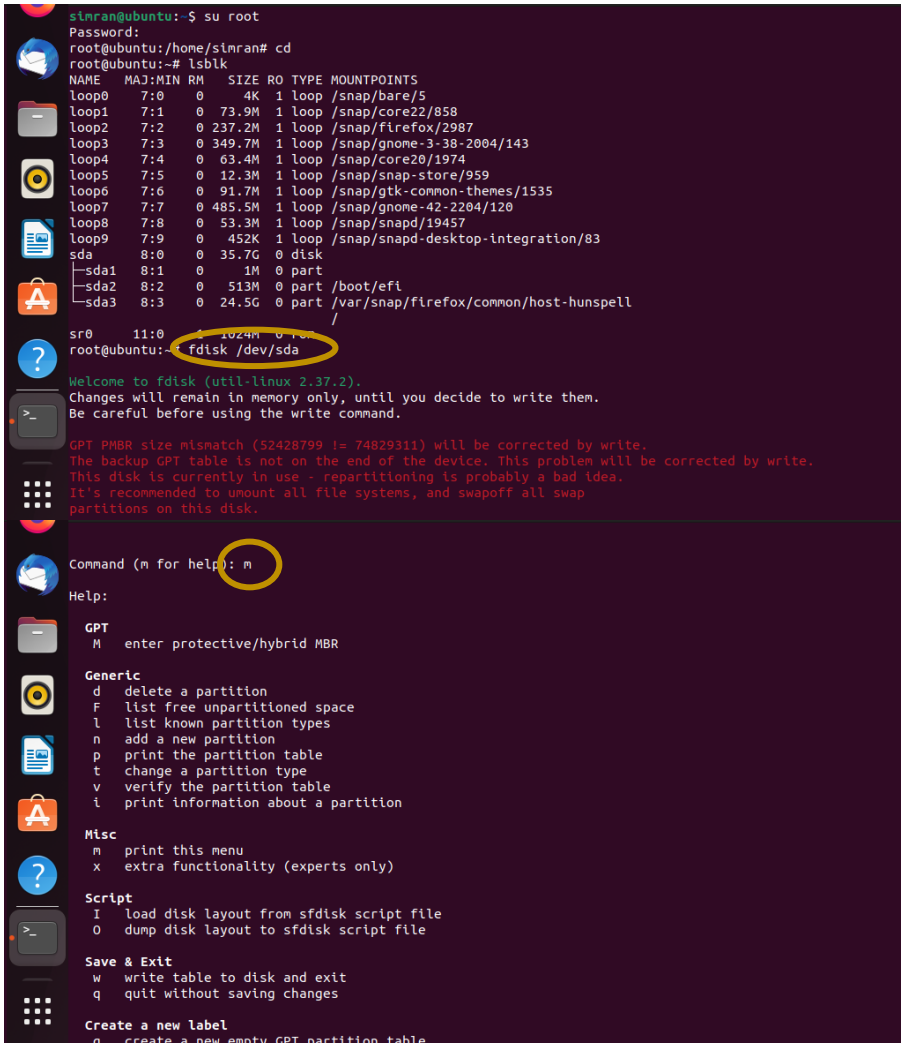
p- to print the partition which we have created

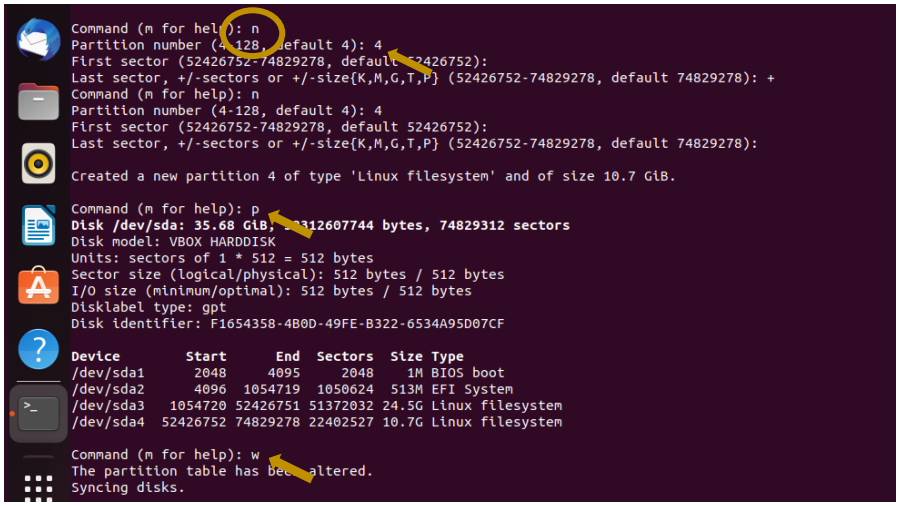
if sure about the partition then use,

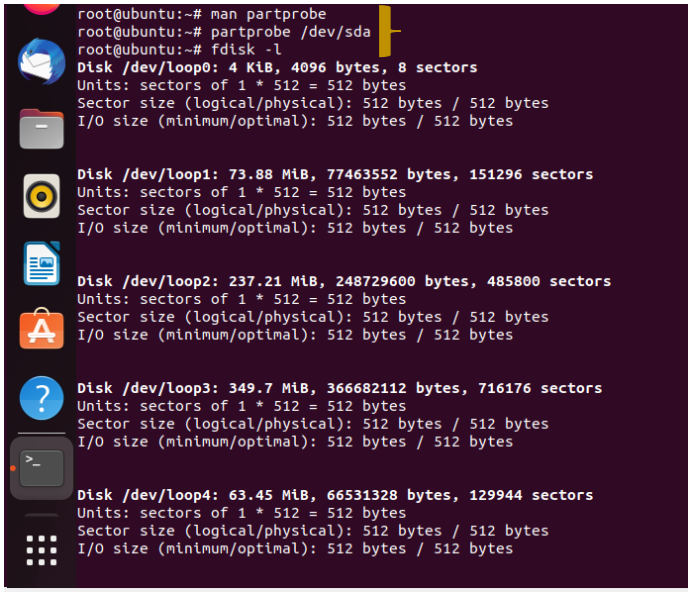
w- to write the new partition changes into OS

Otherwise, use

Q – to quit

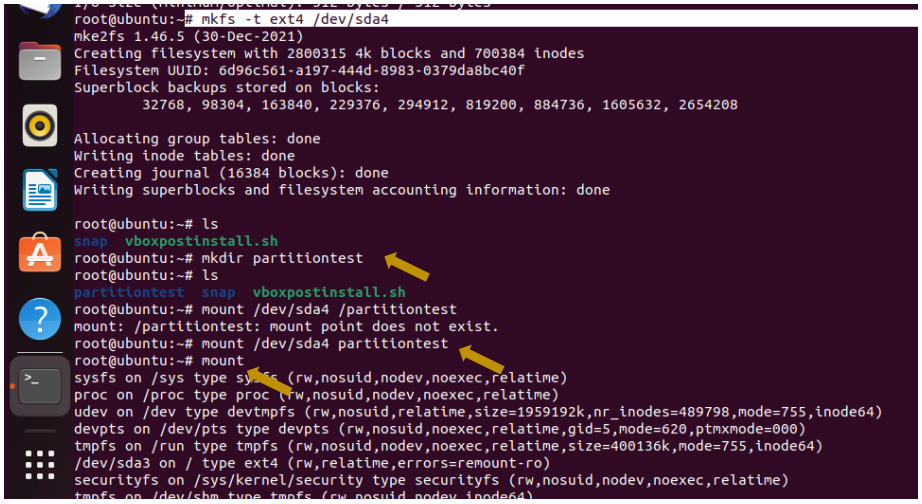






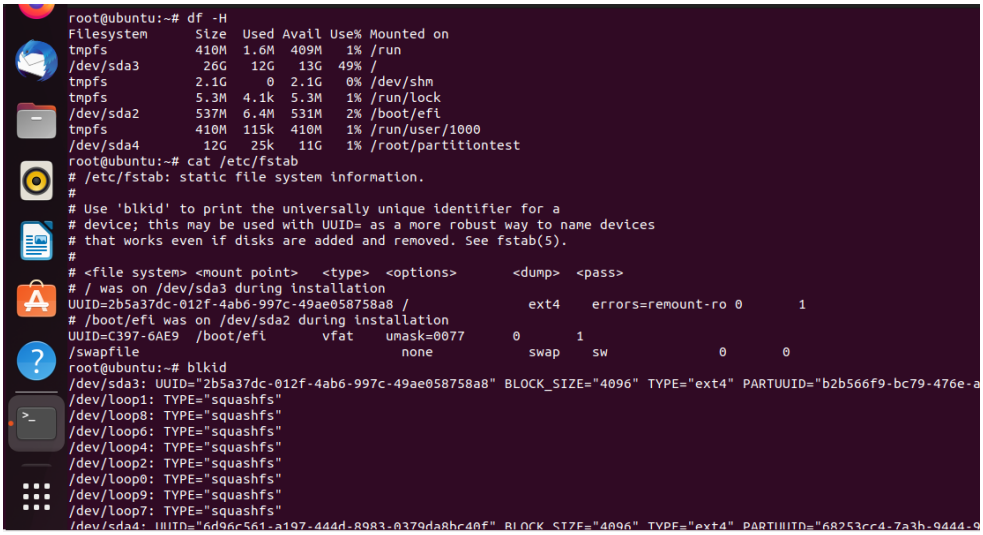
**** till now Kernel is still following the previous partitioned table so to inform the OS**

- Use command- man partprobe or partprobe /dev/sda
- Now use fdisk -l to see the partition we have created
- mkfs -t ext4 /dev/sda4 = command to put the file system in this new partition

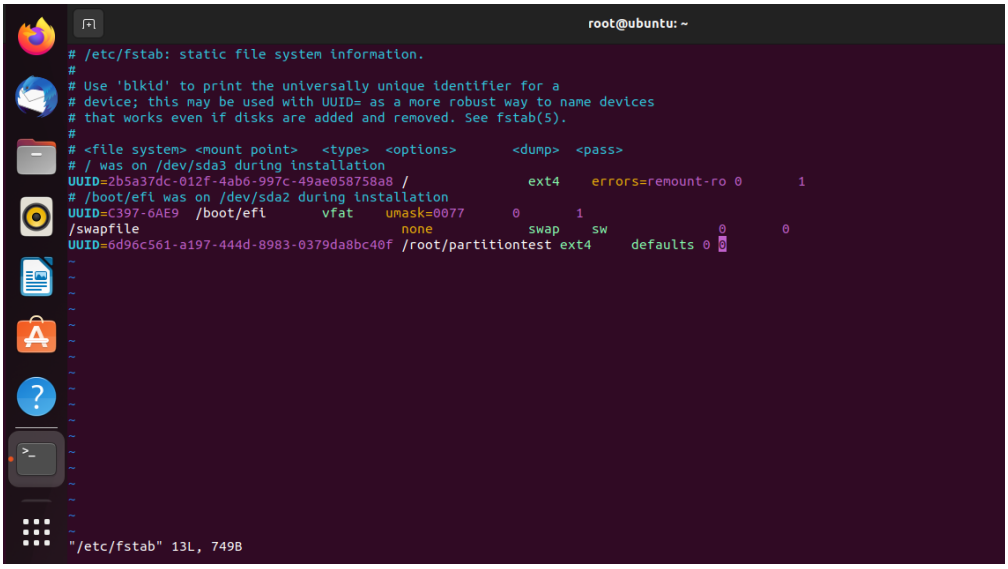


- In order to use this partition now we need to mount it
- Create a new mount point
    - Mkdir partitiontest
    - Mount /dev/sda4 partitiontest
    - Mount

- But where we have mounted the partition this is something temporary but if we need to write these changes permanently then need to write changes in a configurate file
- Cat /etc/fstab
-



Now to make these changes permanent open this fstab file with VIM editor there you will see UUID of other partitions these just add the UUID of the partition you have created and SAVE (Do this carefully)



-
- Run Command mount -a (if there is any mistake done in fstab file after running this command it will show error)
-
- Now if you will try to delete this directory where we created partition it will show currently busy can't delete it (because its not a normal directory but a mount point for the partition) rm -rf /root/partitiontest

```
root@ubuntu:~# vim /etc/fstab
root@ubuntu:~# mount -a
root@ubuntu:~# rm -rf /root/partitiontest
rm: cannot remove '/root/partitiontest': Device or resource busy
```

How to delete the partition then:

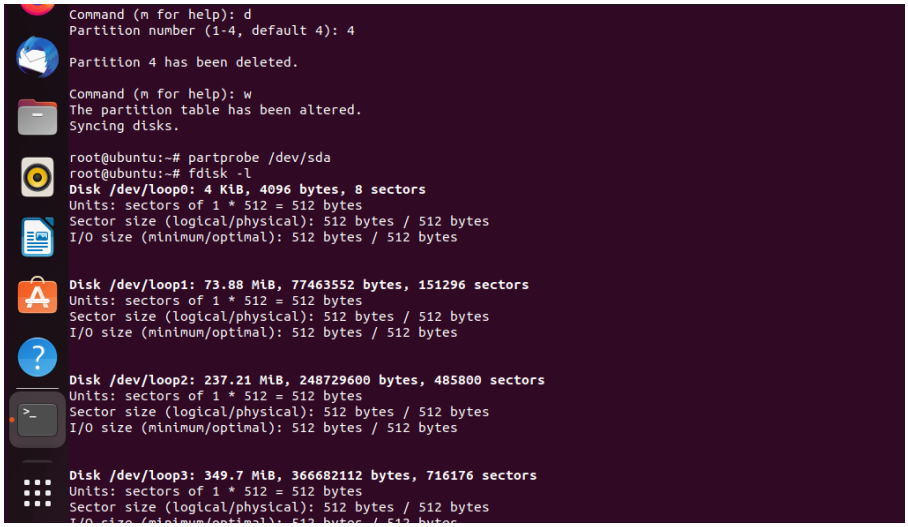
Reverse the whole process

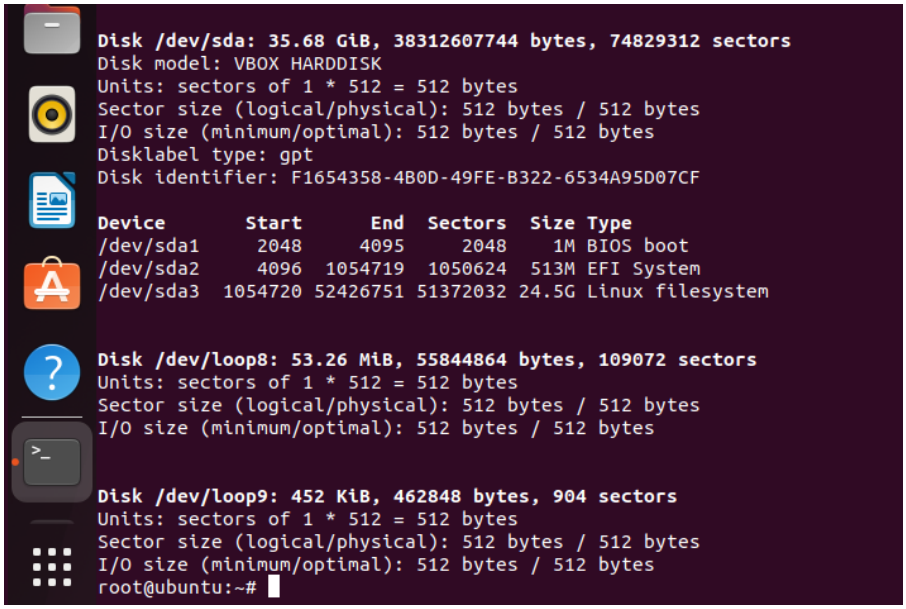
- First of all the change we did in **fstab file** remove those changes

```
root@ubuntu:~# vim /etc/fstab
root@ubuntu:~# cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda3 during installation
UUID=2b5a37dc-012f-4ab6-997c-49ae058758a8 /               ext4    errors=remount-ro 0       1
# /boot/efi was on /dev/sda2 during installation
UUID=C397-6AE9  /boot/efi       vfat    umask=0077      0       1
/swapfile                                 none            swap    sw              0       0

root@ubuntu:~#
```

- Unmount from the directory: umount /root/partitiontest
- Remove the directory: rm -rf /root/partitiontest
- Use commands: fdisk /dev/sda
    - P to see the partitions
    - D to delete the partition
        - Mention the partition number we have created like 4
    - W to write the changes
    - Tell these changes to OS: partprobe /dev/sda
    - See the total partitions: fdisk -l

```
root@ubuntu:~# umount /root/partitiontest
root@ubuntu:~# rm -rf /root/partitiontest
root@ubuntu:~# fdisk /dev/sda

Welcome to fdisk (util-linux 2.37.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

This disk is currently in use - repartitioning is probably a bad idea.
It's recommended to umount all file systems, and swapoff all swap
partitions on this disk.


Command (m for help): p

Disk /dev/sda: 35.68 GiB, 38312607744 bytes, 74829312 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: F1654358-4B0D-49FE-B322-6534A95D07CF

Device         Start      End  Sectors  Size Type
/dev/sda1       2048     4095     2048    1M BIOS boot
/dev/sda2       4096  1054719  1050624  513M EFI System
/dev/sda3    1054720 52426751 51372032 24.5G Linux filesystem
/dev/sda4   52426752 74829278 22402527 10.7G Linux filesystem
```





In fdisk which is MBR partitioning scheme, it supports 4 primary partition and we can have max 15 logical partition

In GPT where we use gdisk command and it supports 128 partitions

-