Partition in Linux

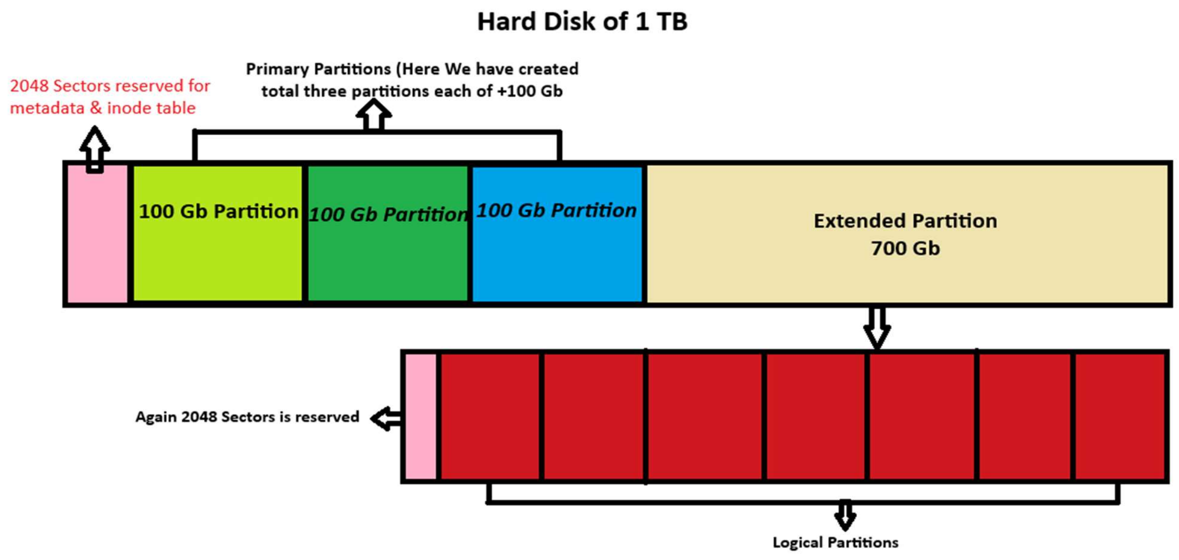
Define Partition:

Disk partitioning is the process of dividing a disk into one or more logical sections called partitions. Partitioning also allows us to divide a hard drive into isolated sections, where each section behaves as it's own hard drive. Partitioning is particularly useful to run multiple operating systems.

If a partition is created, the disk will store the information about the location & size of partitions in the partition table.

Sectors:

Each platter of hard disk is divided into tracks & each track is divided into sectors. Sectors are fixed at 512bytes in size on most file systems. Creation of partition is based on sectors & while creating partitions, 2048 is a common starting sector because it ensures that the partition is aligned with the 4kb sector boundary.


Hard Disk of 1 TB
2048 Sectors reserved for metadata & inode table
Primary Partitions (Here We have created total three partitions each of +100 Gb
100 Gb Partition
100 Gb Partition
100 Gb Partition
Extended Partition 700 Gb
Again 2048 Sectors is reserved
Logical Partitions

Imagine a hard disk of 1TB in size, Now as we cannot use the disk device directly for writing and reading the data, we will create the partitions because partitions are the only way to use our hard disk for storing data.

There are 3 types of partitions

- Primary Partitions -> We can create only 4 Primary Partitions in single drive
- Extended Partitions -> We can Create only 1 Extended Partition in single drive
- Logical Partitions -> We can create upto 128 Logical Partitions in extended partition.

which allows for a maximum of 4 primary partitions and 128 logical partitions in the extended partition.

Primary Partition:

Primary partitions are the buildings blocks of a hard drive. These are the partitions where the operating system & essential files are typically installed.

A hard drive can have a maximum of **4 primary partitions**. This limitation arises from the partition table format, Particularly the Master Boot Record (MBR). The reason behind is the partition table stores the metadata of each partition and total size it can store is 64 bytes and the size of each metadata is 16 bytes.

Each primary partition operates as an independent entity with it's file system. For instance, we might have 1 primary partition for operating system & another for important data.

In diagram we can see that we created only 3 primary partitions of 100Gb to make more partitions. We configured 4th partition as a new hard disk that is known as Extended Partition.

Extended Partition:

To overcome the limitation of only 4 primary partitions, the extended partition comes into play. It serves as a container, allowing us to create additional logical partitions within it.

There can be only one extended partition on a disk. The extended partition doesn't hold data itself but acts as a boundary that houses logical partitions. It extends the capability to have more than 4 partitions on a disk.

Our 4th partition is extended partition of size 700Gb and being an extended partition table reserves the initial space for partition. We can't store data directly in it so we have to create additional partitions inside it which is known as logical partitions.

Logical Partitions:

Logical Partitions exist within the extended partition and are essentially sub-divisions of it. They provide a way to overcome the 4 primary partition limit.

The number of logical partitions that can be created in an **extended partition** is still limited to **128**. This is due to the constraints of the MBR partitioning system itself.

Disk Identification:

Disk identification is critical for managing storage, partitioning, formatting, and mounting storage devices. It helps the system and users differentiate between physical and virtual disks.

1. IDE (Integrated Drive Electronics) drive:

- Older type of disk commonly found in desktop computers.
- The first IDE drive is /dev/hda, the second is /dev/hdb, and soon.

2. SCSI (Small Computer System Interface) drive:

- Common in servers and high-performance systems.
- The first SCSI drive is /dev/sda, the second is /dev/sdb, and soon.

3. Virtual Drive (e.g., for virtual machines or cloud environments):

- Virtual disks (like those in KVM, VMware, or cloud environments like AWS and Azure) are often identified as /dev/vda, /dev/vdb, /dev/vdc, etc.
- The first virtual disk is typically /dev/vda, and subsequent disks are /dev/vdb, /dev/vdc, etc.

4. SATA (Serial ATA) drives:

- In newer systems, SATA drives are often recognized similarly to IDE drives in terms of device names, but usually, modern systems use /dev/sda, /dev/sdb, etc. for both SCSI and SATA drives.

5. NVMe (Non-Volatile Memory Express) drive:

- High-speed SSDs using the NVMe protocol are identified differently.
- These drives are typically named /dev/nvme0n1, /dev/nvme1n1, and so on, where "nvme" indicates the type of storage, and the number after it refers to the drive number.

File Systems:

The Linux file system is a method of organizing and managing files in Linux. It defines how to store, access, and retrieve data on the disk.

Linux supports various file system types such as ext4, XFS, Btrfs, and ZFS, each offering different features like journaling, scalability, and snapshot capabilities.

The most commonly used file system on Linux is **ext4**, known for its balance between performance and reliability.

Linux has evolved from the limited Minix system to more advanced options like ext, ext2, and ext3. Each version improved file management and storage capabilities, with ext3 adding journaling for better data integrity.

Currently ext4 is the default Linux file system, offering faster performance, larger file support, and improved reliability for modern usage.

File System	Maximum Volume Size	Maximum File Size
ext4	1 EiB	16 TiB
XFS	8 EiB	8 EiB
Btrfs	16 EiB	16 TiB
ZFS	256 EiB	16 TiB
exFAT	128 EiB	16 TiB
NTFS	256 EiB	16 TiB

Conversions
1 EiB = 1048576 TiB
1 TiB = 0.0000009536743 EiB

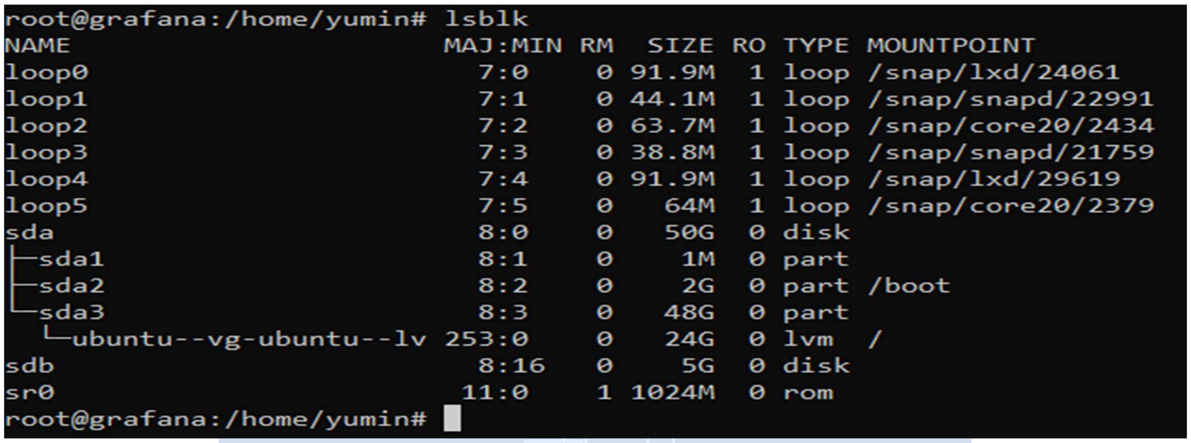
File System	Use Case	Features	Main Pros/Cons
ext4	Default on most Linux distributions. Suitable for general-purpose systems, including desktops and servers.	Journaling for crash recovery.	Mature and stable with extensive community support.
		Support for volumes up to 1 EiB and files up to 16 TiB.	Lacks advanced features like snapshots.
		Fast mount times and delayed allocation for better performance.	
XFS	Ideal for high-performance environments that handle large files, such as media servers or databases.	High throughput with large files and scalable to massive storage.	Poor performance with many small files.
		Good support for parallel I/O operations.	Complex to tune and manage effectively.
		Online defragmentation and resizing.	
Btrfs	Used for advanced use cases that require snapshots, sub volumes, or integrated volume management; often found in modern servers and storage appliances.	Snapshotting, RAID support, self-healing via checksums.	Less mature than ext4.
		Support for compression and deduplication.	Stability issues in certain use cases.
		Online resizing and defragmentation.	
ZFS	Suitable for environments that require extreme data integrity, such as enterprise servers, backups, and NAS devices.	Combines file system and volume management.	Requires substantial memory for optimal performance (8 GB+).
		RAID-Z, data deduplication, and end-to-end data integrity verification.	Licensing restrictions (CDDL vs. GPL).
		Supports pools with immense storage capacities.	
exFAT & NTFS (via ntfs-3g)	Typical choice for dual-boot setups or external drives shared between Linux and Windows.	Read/write support for NTFS and exFAT partitions on Linux.	ntfs-3g performance can be slower compared to native NTFS on Windows. exFAT does not support journaling, which limits data recovery options.
		exFAT is optimized for flash drives and SD cards.	
		NTFS supports advanced Windows-specific features.	

Creating Partitions:

To see how many disk and partitions we currently have in our system, we will use this command.

lsblk

The lsblk command in Linux lists all block devices (like hard drives, partitions, and storage devices) in a tree structure, showing their names, sizes, and mount points. It helps in identifying device relationships and partition details without displaying filesystem data by default.



```
root@grafana:/home/yumin# lsblk
NAME                      MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
loop0                       7:0    0  91.9M  1 loop /snap/lxd/24061
loop1                       7:1    0  44.1M  1 loop /snap/snapd/22991
loop2                       7:2    0  63.7M  1 loop /snap/core20/2434
loop3                       7:3    0  38.8M  1 loop /snap/snapd/21759
loop4                       7:4    0  91.9M  1 loop /snap/lxd/29619
loop5                       7:5    0    64M  1 loop /snap/core20/2379
sda                         8:0    0    50G  0 disk
├─sda1                      8:1    0     1M  0 part
├─sda2                      8:2    0     2G  0 part /boot
└─sda3                      8:3    0    48G  0 part
  └─ubuntu--vg-ubuntu--lv 253:0    0    24G  0 lvm  /
sdb                         8:16   0     5G  0 disk
sr0                        11:0    1  1024M  0 rom
root@grafana:/home/yumin#
```

We can see we have two disk one is primary disk that is attached for installing all system softwares. And second we have our disk drive /dev/sdb.

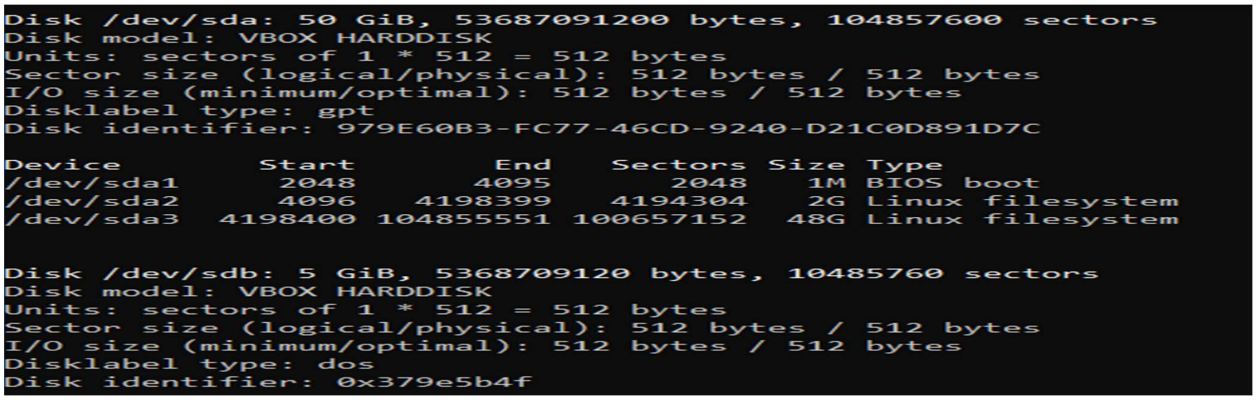
fdisk

The fdisk command in Linux is used to manipulate disk partitions, allowing users to create, delete, and modify partitions on a storage device. It provides an interactive interface for managing partition tables on MBR (Master Boot Record) disks.

To manipulate partition inside disks, we have fdisk command.

To see the details of drives and paritions we use the command **fdisk -l**

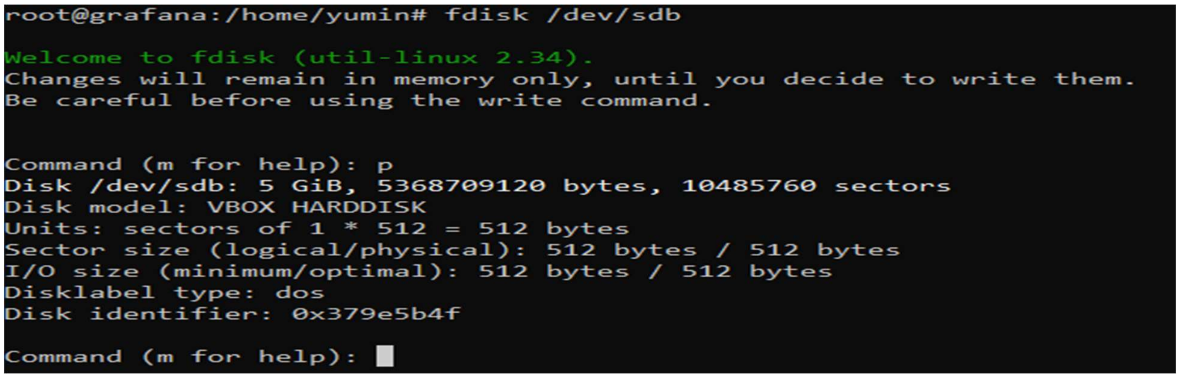
The **fdisk -l** command in Linux lists all disk partitions and their details, including device names, sizes, types, and partition tables for all available disks.



```
Disk /dev/sda: 50 GiB, 53687091200 bytes, 104857600 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: 979E60B3-FC77-46CD-9240-D21C0D891D7C

Device          Start       End   Sectors Size Type
/dev/sda1        2048      4095      2048   1M BIOS boot
/dev/sda2        4096   4198399   4194304   2G Linux filesystem
/dev/sda3     4198400 104855551 100657152  48G Linux filesystem


Disk /dev/sdb: 5 GiB, 5368709120 bytes, 10485760 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: dos
Disk identifier: 0x379e5b4f
```

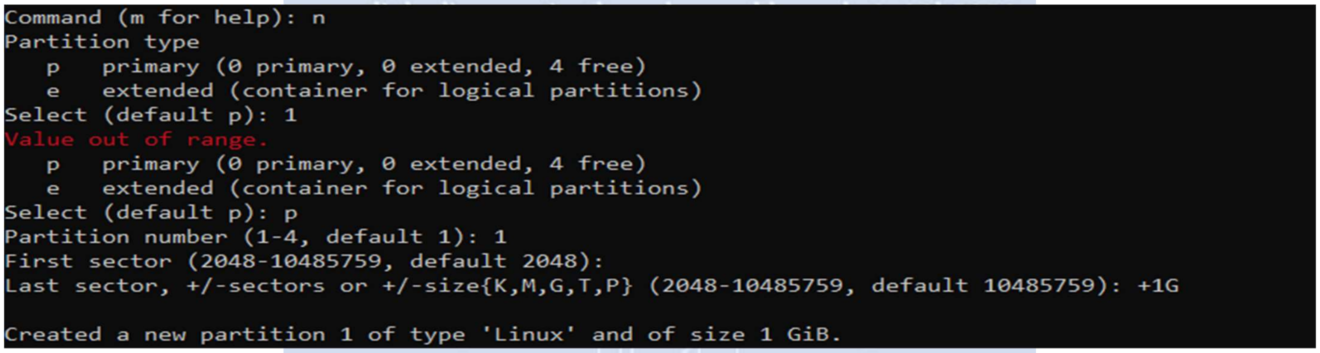
fdisk /dev/sdb

To enter into the disk /dev/sdb for creating partition, we will use fdisk <diskname> command and for having info of that disk we have “p” command.

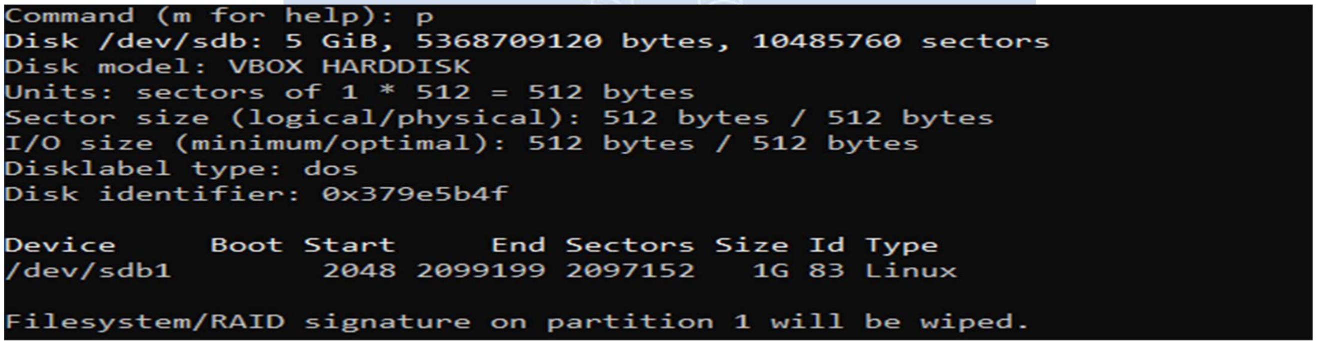


```
root@grafana:/home/yumin# fdisk /dev/sdb

Welcome to fdisk (util-linux 2.34).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): p
Disk /dev/sdb: 5 GiB, 5368709120 bytes, 10485760 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: dos
Disk identifier: 0x379e5b4f

Command (m for help):
```

To create new partition in that disk, we have “n” command.



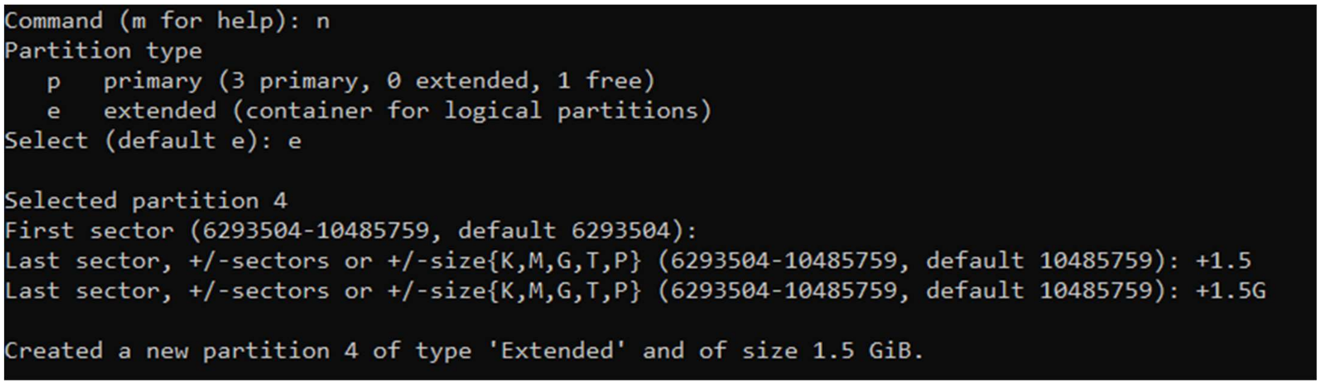
```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): 1
Value out of range.
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-10485759, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-10485759, default 10485759): +1G

Created a new partition 1 of type 'Linux' and of size 1 GiB.
```



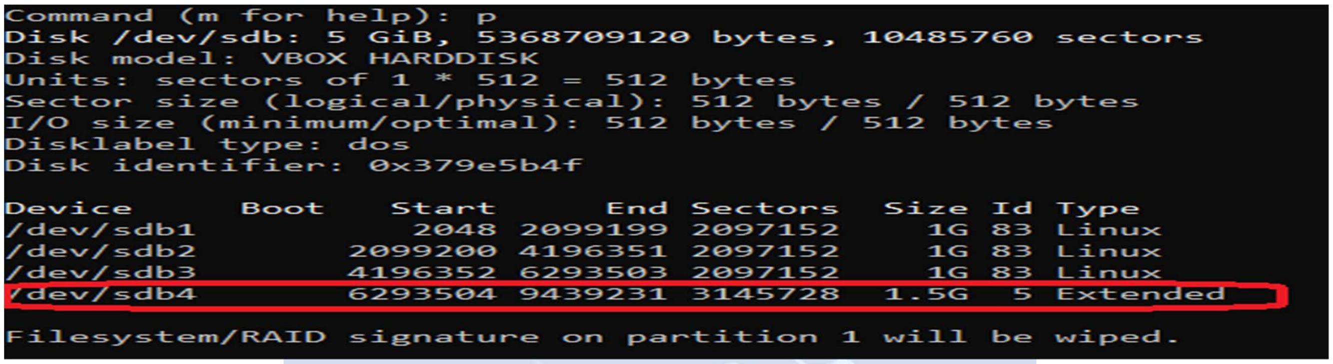
```
Command (m for help): p
Disk /dev/sdb: 5 GiB, 5368709120 bytes, 10485760 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x379e5b4f

Device     Boot Start     End Sectors Size Id Type
/dev/sdb1        2048 2099199 2097152   1G 83 Linux

Filesystem/RAID signature on partition 1 will be wiped.
```

We can see that the start of partition is from 2048 means that it had reserved some space for partition table.

Similarly we will create the 2 more partitions but after that you will reach your limit.

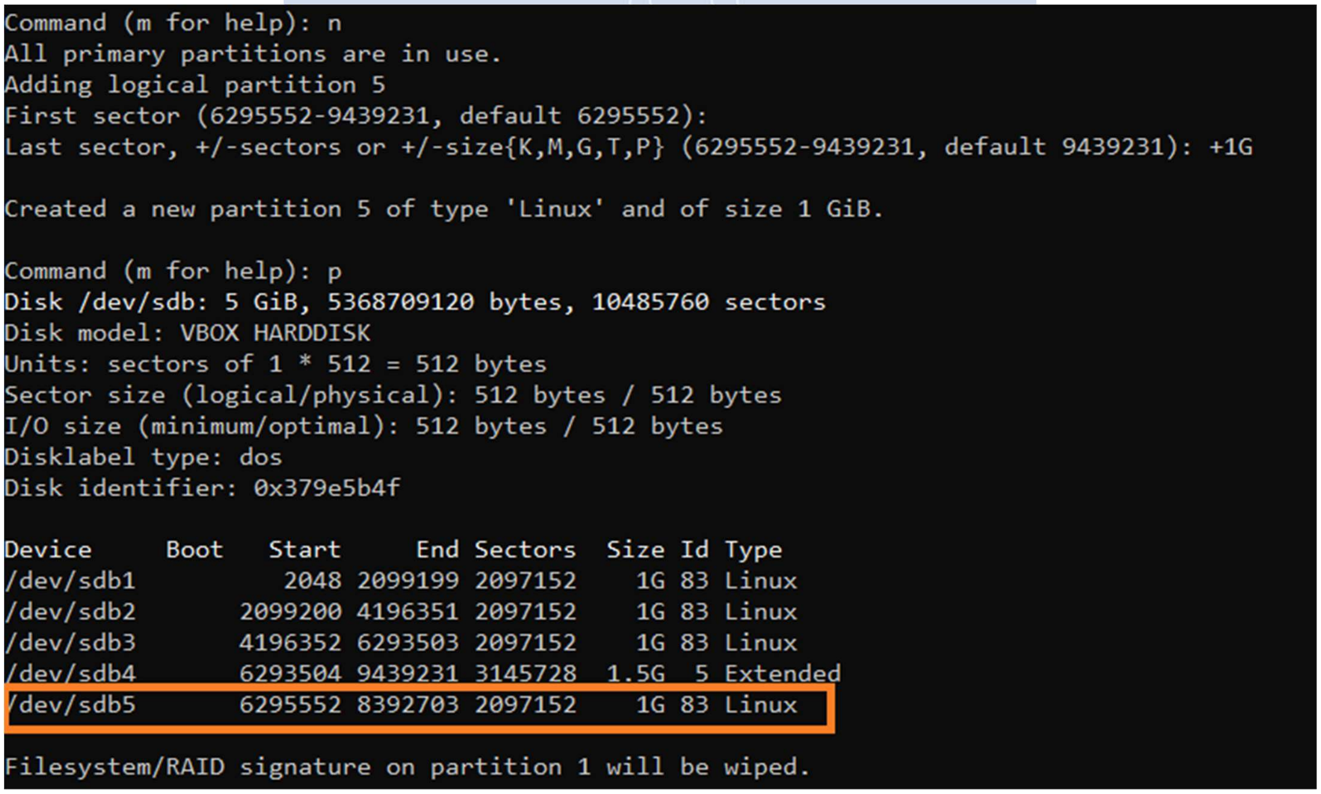


```
Command (m for help): n
Partition type
   p   primary (3 primary, 0 extended, 1 free)
   e   extended (container for logical partitions)
Select (default e): e

Selected partition 4
First sector (6293504-10485759, default 6293504):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (6293504-10485759, default 10485759): +1.5
Last sector, +/-sectors or +/-size{K,M,G,T,P} (6293504-10485759, default 10485759): +1.5G

Created a new partition 4 of type 'Extended' and of size 1.5 GiB.
```



```
Command (m for help): p
Disk /dev/sdb: 5 GiB, 5368709120 bytes, 10485760 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: dos
Disk identifier: 0x379e5b4f

Device     Boot   Start     End Sectors  Size Id Type
/dev/sdb1          2048 2099199 2097152    1G 83 Linux
/dev/sdb2       2099200 4196351 2097152    1G 83 Linux
/dev/sdb3       4196352 6293503 2097152    1G 83 Linux
/dev/sdb4       6293504 9439231 3145728  1.5G  5 Extended

Filesystem/RAID signature on partition 1 will be wiped.
```

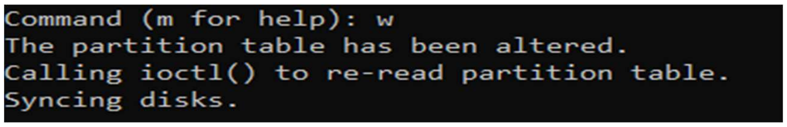
Now we can easily create more partition that is known as logical partitions and you will notice that those partitions are within the limit of these extended ones.



```
Command (m for help): n
All primary partitions are in use.
Adding logical partition 5
First sector (6295552-9439231, default 6295552):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (6295552-9439231, default 9439231): +1G

Created a new partition 5 of type 'Linux' and of size 1 GiB.

Command (m for help): p
Disk /dev/sdb: 5 GiB, 5368709120 bytes, 10485760 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: dos
Disk identifier: 0x379e5b4f

Device     Boot   Start     End Sectors  Size Id Type
/dev/sdb1          2048 2099199 2097152    1G 83 Linux
/dev/sdb2       2099200 4196351 2097152    1G 83 Linux
/dev/sdb3       4196352 6293503 2097152    1G 83 Linux
/dev/sdb4       6293504 9439231 3145728  1.5G  5 Extended
/dev/sdb5       6295552 8392703 2097152    1G 83 Linux

Filesystem/RAID signature on partition 1 will be wiped.
```

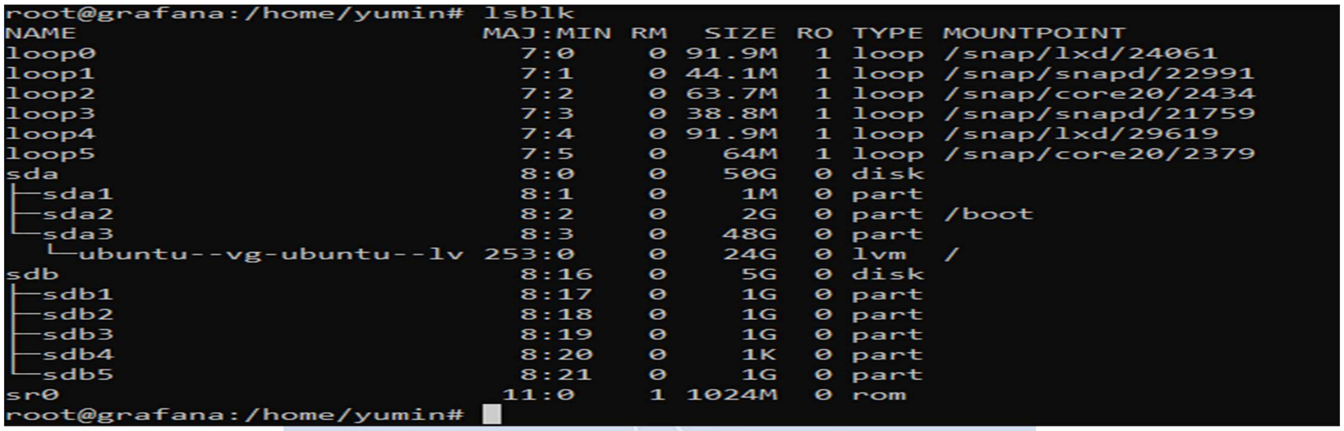
And also we can notice one more thing here that the starting sector of the logical partition is 6295552 which is 2048 sectors ahead of the start of extended sector 62935504. Hence we can conclude that the extended partition also reserved some space that 2048 sectors for storing partition table and some more metadata in it.

Don’t forget to press “w” before exiting from the disk, it will save all the changes we made.



```
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

So now our disk is ready with the partitions:

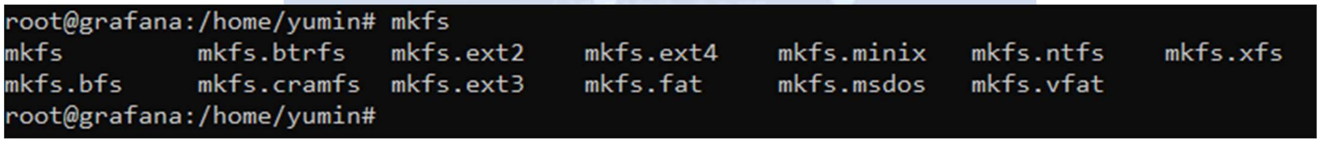


```
root@grafana:/home/yumin# lsblk
NAME                        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
loop0                         7:0    0  91.9M  1 loop /snap/lxd/24061
loop1                         7:1    0  44.1M  1 loop /snap/snapd/22991
loop2                         7:2    0  63.7M  1 loop /snap/core20/2434
loop3                         7:3    0  38.8M  1 loop /snap/snapd/21759
loop4                         7:4    0  91.9M  1 loop /snap/lxd/29619
loop5                         7:5    0    64M  1 loop /snap/core20/2379
sda                           8:0    0    50G  0 disk
├─sda1                        8:1    0     1M  0 part
├─sda2                        8:2    0     2G  0 part /boot
└─sda3                        8:3    0    48G  0 part
  └─ubuntu--vg-ubuntu--lv   253:0    0    24G  0 lvm  /
sdb                           8:16   0     5G  0 disk
├─sdb1                        8:17   0     1G  0 part
├─sdb2                        8:18   0     1G  0 part
├─sdb3                        8:19   0     1G  0 part
├─sdb4                        8:20   0     1K  0 part
└─sdb5                        8:21   0     1G  0 part
sr0                          11:0    1  1024M  0 rom
root@grafana:/home/yumin#
```

Create a file system on the partition:

Formatting a partition is the process of creating or changing the file system on a disk. A file system is a way of organizing and storing data on a disk, such as EXT3, EXT4, & XFS etc. It is a compulsory or mandatory step to be done after creating partitions.

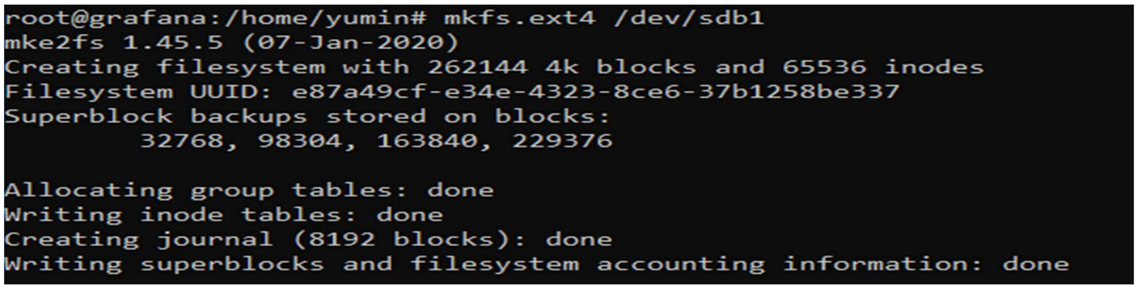
The mkfs command is a command line tool that formats a disk or partition into a specific file system. The letters in mkfs stand for "make file system”. Inside mkfswe have many types of file system.



```
root@grafana:/home/yumin# mkfs
mkfs         mkfs.btrfs   mkfs.ext2    mkfs.ext4    mkfs.minix   mkfs.ntfs    mkfs.xfs
mkfs.bfs     mkfs.cramfs  mkfs.ext3    mkfs.fat     mkfs.msdos   mkfs.vfat
root@grafana:/home/yumin#
```

mkfs.ext4 <partition-name>

mkfs.ext4 /dev/sdb1



```
root@grafana:/home/yumin# mkfs.ext4 /dev/sdb1
mke2fs 1.45.5 (07-Jan-2020)
Creating filesystem with 262144 4k blocks and 65536 inodes
Filesystem UUID: e87a49cf-e34e-4323-8ce6-37b1258be337
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

Similarly, we will format the other partitions also to make it usable.

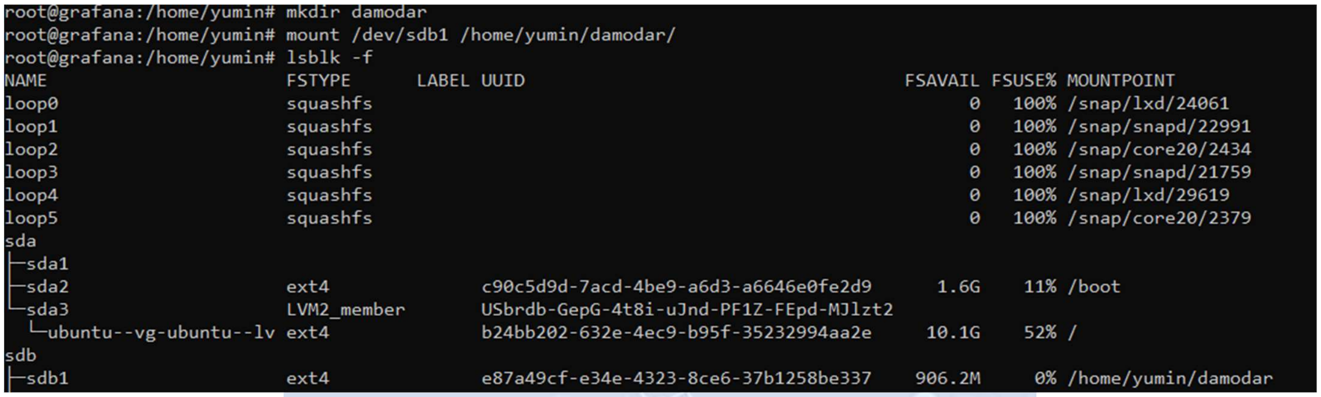
Mount the filesystem:

The very next step after formatting the partition is to mount it with some folder as there is no way out there for connecting to partition directly and start storing data on them directly. In any OS for storing data, we have to create Folder and inside folder we have files and inside files we have our final data.

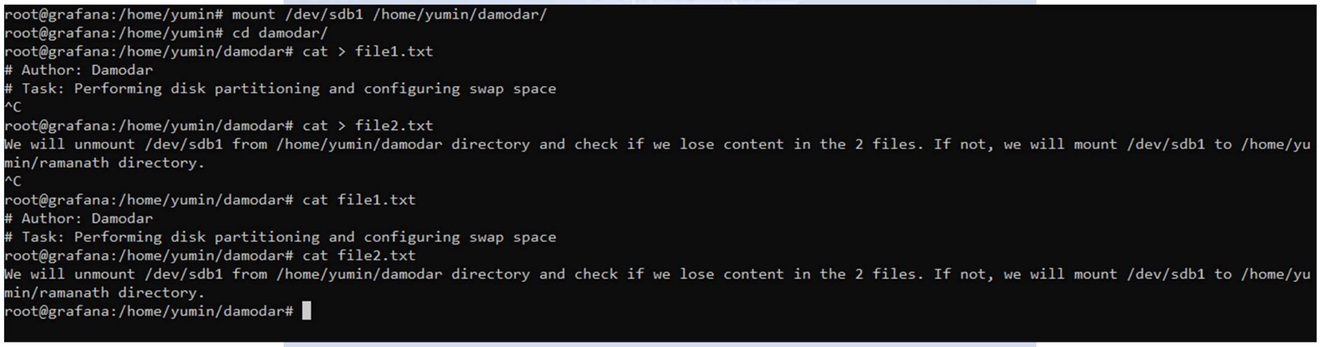
Hence we will mount it with some directory but the data is actually storing inside partitions.

mkdir -p /home/yumin/damodar

mount /dev/sdb1 /home/yumin/damodar



```
root@grafana:/home/yumin# mkdir damodar
root@grafana:/home/yumin# mount /dev/sdb1 /home/yumin/damodar/
root@grafana:/home/yumin# lsblk -f
NAME                      FSTYPE      LABEL UUID                                   FSAVAIL FSUSE% MOUNTPOINT
loop0                     squashfs                                                       0   100% /snap/lxd/24061
loop1                     squashfs                                                       0   100% /snap/snapd/22991
loop2                     squashfs                                                       0   100% /snap/core20/2434
loop3                     squashfs                                                       0   100% /snap/snapd/21759
loop4                     squashfs                                                       0   100% /snap/lxd/29619
loop5                     squashfs                                                       0   100% /snap/core20/2379
sda
├─sda1
├─sda2                    ext4              c90c5d9d-7acd-4be9-a6d3-a6646e0fe2d9      1.6G    11% /boot
└─sda3                    LVM2_member       USbrdb-GepG-4t8i-uJnd-PF1Z-FEpd-MJlzt2
  └─ubuntu--vg-ubuntu--lv ext4              b24bb202-632e-4ec9-b95f-35232994aa2e     10.1G    52% /
sdb
├─sdb1                    ext4              e87a49cf-e34e-4323-8ce6-37b1258be337    906.2M     0% /home/yumin/damodar
```



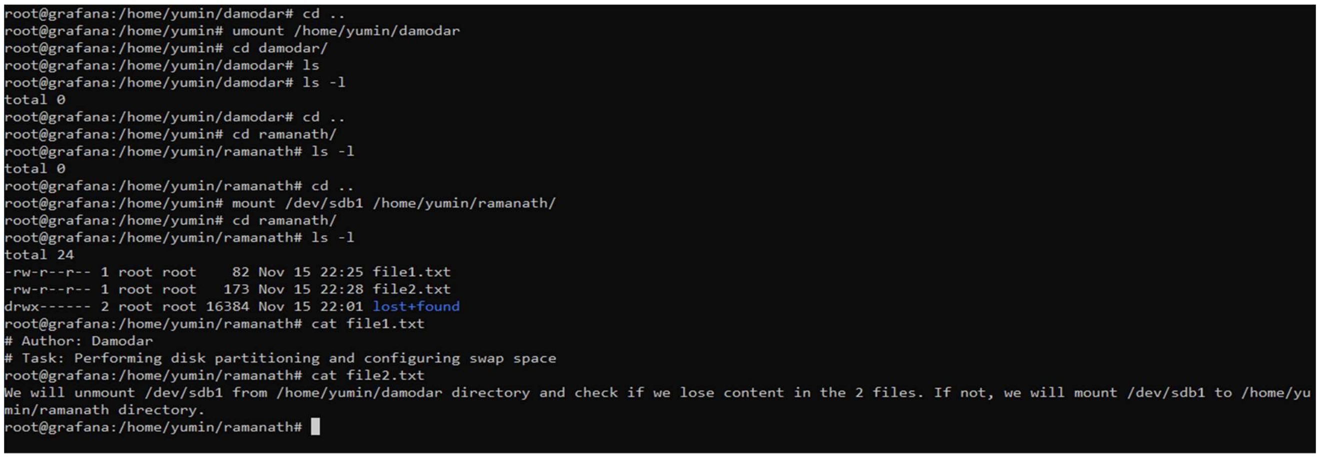
```
root@grafana:/home/yumin# mount /dev/sdb1 /home/yumin/damodar/
root@grafana:/home/yumin# cd damodar/
root@grafana:/home/yumin/damodar# cat > file1.txt
# Author: Damodar
# Task: Performing disk partitioning and configuring swap space
^C
root@grafana:/home/yumin/damodar# cat > file2.txt
We will unmount /dev/sdb1 from /home/yumin/damodar directory and check if we lose content in the 2 files. If not, we will mount /dev/sdb1 to /home/yu
min/ramanath directory.
^C
root@grafana:/home/yumin/damodar# cat file1.txt
# Author: Damodar
# Task: Performing disk partitioning and configuring swap space
root@grafana:/home/yumin/damodar# cat file2.txt
We will unmount /dev/sdb1 from /home/yumin/damodar directory and check if we lose content in the 2 files. If not, we will mount /dev/sdb1 to /home/yu
min/ramanath directory.
root@grafana:/home/yumin/damodar#
```

Now lets unmount the partitions and then mount it to some other directory and see what will happen.

umount /dev/sdb1 /home/yumin/Damodar

mkdir -p /home/yumin/ramanath

mount /dev/sdb1 /home/yumin/ramanath



```
root@grafana:/home/yumin/damodar# cd ..
root@grafana:/home/yumin# umount /home/yumin/damodar
root@grafana:/home/yumin# cd damodar/
root@grafana:/home/yumin/damodar# ls
root@grafana:/home/yumin/damodar# ls -l
total 0
root@grafana:/home/yumin/damodar# cd ..
root@grafana:/home/yumin# cd ramanath/
root@grafana:/home/yumin/ramanath# ls -l
total 0
root@grafana:/home/yumin/ramanath# cd ..
root@grafana:/home/yumin# mount /dev/sdb1 /home/yumin/ramanath/
root@grafana:/home/yumin# cd ramanath/
root@grafana:/home/yumin/ramanath# ls -l
total 24
-rw-r--r-- 1 root root    82 Nov 15 22:25 file1.txt
-rw-r--r-- 1 root root   173 Nov 15 22:28 file2.txt
drwx------ 2 root root 16384 Nov 15 22:01 lost+found
root@grafana:/home/yumin/ramanath# cat file1.txt
# Author: Damodar
# Task: Performing disk partitioning and configuring swap space
root@grafana:/home/yumin/ramanath# cat file2.txt
We will unmount /dev/sdb1 from /home/yumin/damodar directory and check if we lose content in the 2 files. If not, we will mount /dev/sdb1 to /home/yu
min/ramanath directory.
root@grafana:/home/yumin/ramanath#
```

We can see that the data is gone from that folder after unmount it from partition. Let's again mount the partition but this time to some other directory.

After mounting the partition to some other directory the entire data that we have stored in the previous directory is now present inside this directory.

Hence this practical shows that the data is actually stored in partition but the medium is directory.

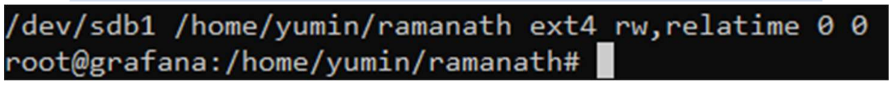
Please caution while executing the umount command.

Make it Persistent Across Reboots:

Still, the /dev/sdb1 is only temporarily mounted. After rebooting the server, it gets unmounted.

cat /etc/mtab

The /etc/mtab file contains a list of currently mounted filesystems, but it reflects **temporary mounts** (those made during the current session) rather than persistent ones. Permanent mounts are stored in /etc/fstab.



```
/dev/sdb1 /home/yumin/ramanath ext4 rw,relatime 0 0
root@grafana:/home/yumin/ramanath#
```

To make the mounting of /dev/sdb1 persistent across reboots.

nano /etc/fstab

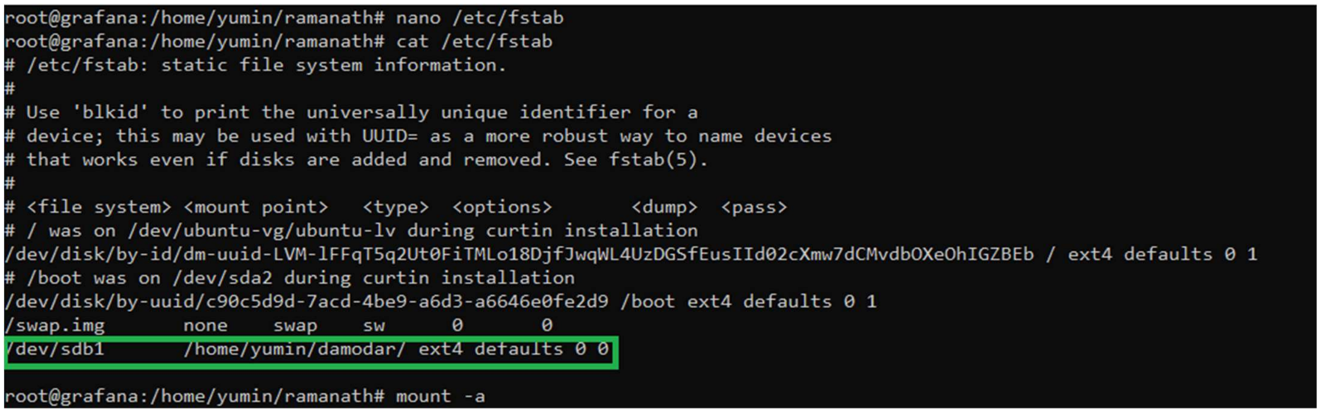
```
/dev/sdb1   /home/yumin/ramanath   ext4   defaults   0   0
```

Save & Exit

- /dev/sdb1: The partition you want to mount.
- /home/yumin/ramanath: The directory where it will be mounted.
- ext4: The filesystem type. We will change this if it's not ext4 (e.g., xfs, ntfs, etc.).
- defaults: Mount options (you can adjust these if needed).
- 0 0: Dump and fsck options (commonly set to 0 for non-root filesystems).

mount -a

Mounts all filesystems listed in the /etc/fstab file that are not currently mounted, ensuring they are active.



```
root@grafana:/home/yumin/ramanath# nano /etc/fstab
root@grafana:/home/yumin/ramanath# 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/ubuntu-vg/ubuntu-lv during curtin installation
/dev/disk/by-id/dm-uuid-LVM-lFFqT5q2Ut0FiTMLo18DjfJwqWL4UzDGSfEusIId02cXmw7dCMvdbOXeOhIGZBEb / ext4 defaults 0 1
# /boot was on /dev/sda2 during curtin installation
/dev/disk/by-uuid/c90c5d9d-7acd-4be9-a6d3-a6646e0fe2d9 /boot ext4 defaults 0 1
/swap.img       none    swap    sw      0       0
/dev/sdb1       /home/yumin/damodar/ ext4 defaults 0 0

root@grafana:/home/yumin/ramanath# mount -a
```

Interview Questions:

1. **What is disk partitioning, and why is it important in Linux?**

Disk partitioning is the process of dividing a physical disk into multiple logical sections, known as partitions. Each partition can then be formatted with a filesystem and used to store data independently. Partitioning is essential in Linux for:

- Organizing data: Different partitions can be used for different purposes (e.g., /home, /var, /boot, swap), which improves system organization and security.
- Efficient space management: Partitions allow better space allocation, preventing one filesystem from consuming all available space.
- Data isolation: Separating critical system files from user data, or isolating swap space, minimizes risks of data corruption or accidental deletion.
- System recovery: In case of a failure, non-critical partitions can be mounted and recovered without affecting system partitions.

2. **Can you explain the difference between primary, extended, and logical partitions in Linux?**

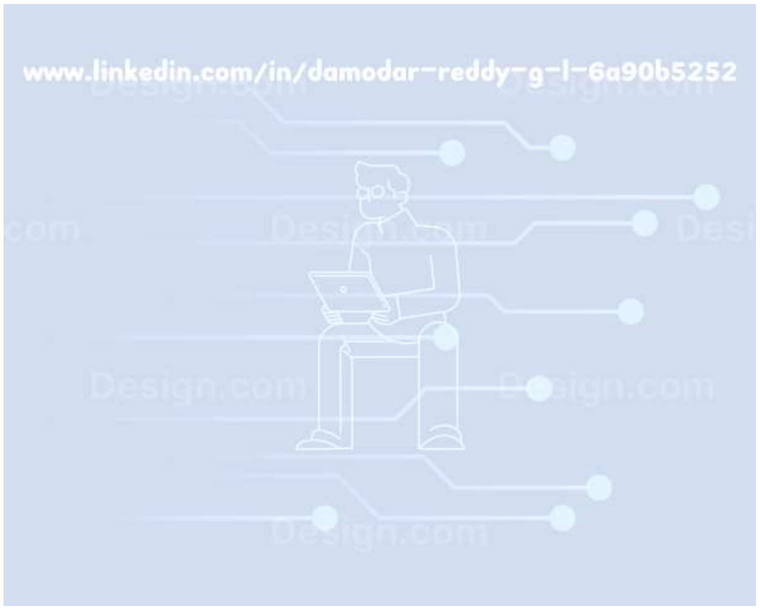
In Linux (and generally with MBR partitioning), there are three types of partitions:

- **Primary partition**: A primary partition is a basic partition that can be directly used by the system. You can have up to four primary partitions on a disk. They are used to store system files or user data.
- **Extended partition**: A special type of partition that acts as a container for logical partitions. A disk can have only one extended partition, but it can contain multiple logical partitions.
- **Logical partition**: These are partitions inside an extended partition. Unlike primary partitions, you can have more than four logical partitions. Logical partitions are typically used when you need to create more than four partitions on a single disk.

Example:

- A disk with 4 partitions might have 3 primary partitions and 1 extended partition containing multiple logical partitions.

3. **How do you create and mount a partition in Linux? Can you explain the process of adding a new partition to /etc/fstab to make it persistent across reboots?**


www.linkedin.com/in/damodar-reddy-g-l-6a90b5252