420-N23-LA Operating Systems & Scripting Using Linux

PHYSICAL STORAGE

Topics

- Partitions
- How to create partitions
- How to format a hard drive (build file system)
- Mounting a partition
- Permanently mounting a partition or drive
- Boot Sector
- Disk Utilities

Logical Partitions

DISKS

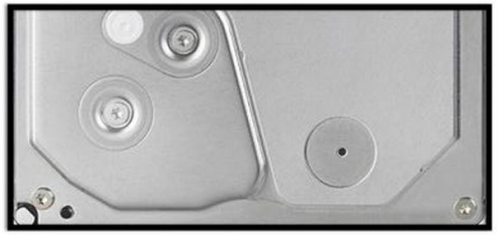
What is a partition

A partition can be thought of as a division or "part" of a real hard disk drive.

A partition is really only a logical separation from the whole drive, but it appears as though the division creates multiple physical drives. Example: C: D: E: (on the same drive).

Partitions are also sometimes called **disk partitions** and when someone uses the word **drive**, they usually mean a partition with a drive letter assigned.

"A partition is a contiguous space of storage on a physical or logical disk that functions as if it were a physically separate disk." Microsoft



Why partition?

One Partition: Define a place to install the operating system. This is the default.

Multiple Partitions: Install a different operating system per partition (Linux, Windows, etc.)

System Partitions: To keep data invisible – like a recovery partition, or to hide information.

Partitioning + Security Pros and Cons

Security Considerations

PROS

- Partitions look like different drives to the computer.
- If one partition gets corrupted, the other partition may not be affected.
- A partition can be invisible or inaccessible, makes it harder for a user or virus to modify it.

CONS

- If the whole drive dies for any reason, all partitions are lost.

The Boot Record

Boot Process

The computer boot process is orchestrated by the BIOS.

The BIOS finds all partitions using a MASTER BOOT RECORD, and then runs the code from that partition.

Usually, you can choose which partition to boot from, in the BIOS yourself.

Bios reads the PARTITION TABLE.

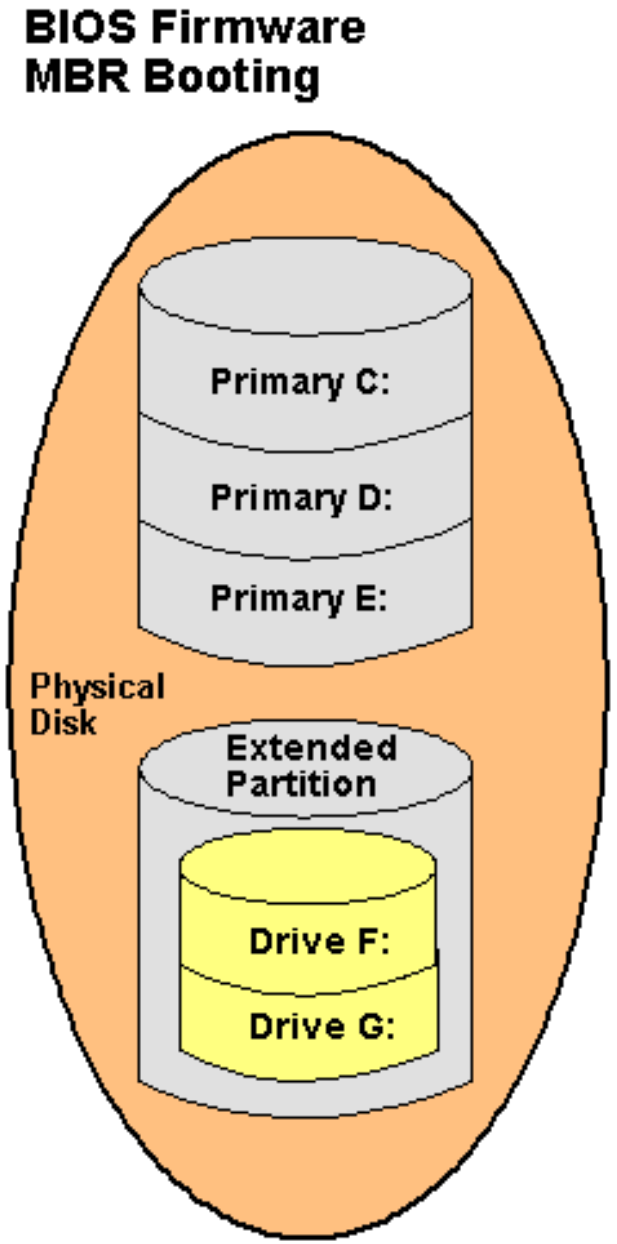
[Boot Process Video]

A Master Boot Record

In the context of the Master Boot Record (MBR) disk partitioning scheme, a primary partition is one of the four possible partitions directly accessible by the system BIOS or the disk's MBR. These are typically used to store operating systems and their bootloaders.

On the other hand, an extended partition is a special type of primary partition that does not directly contain data. Instead, it functions as a container that can hold unlimited logical partitions. Logical partitions are where data and other operating systems can be stored when more than four partitions are needed on a disk.

Primary partitions hold boot information and the operating system, while an extended partition is a workaround for MBR's four-partition limit, allowing you to create additional logical partitions within it.

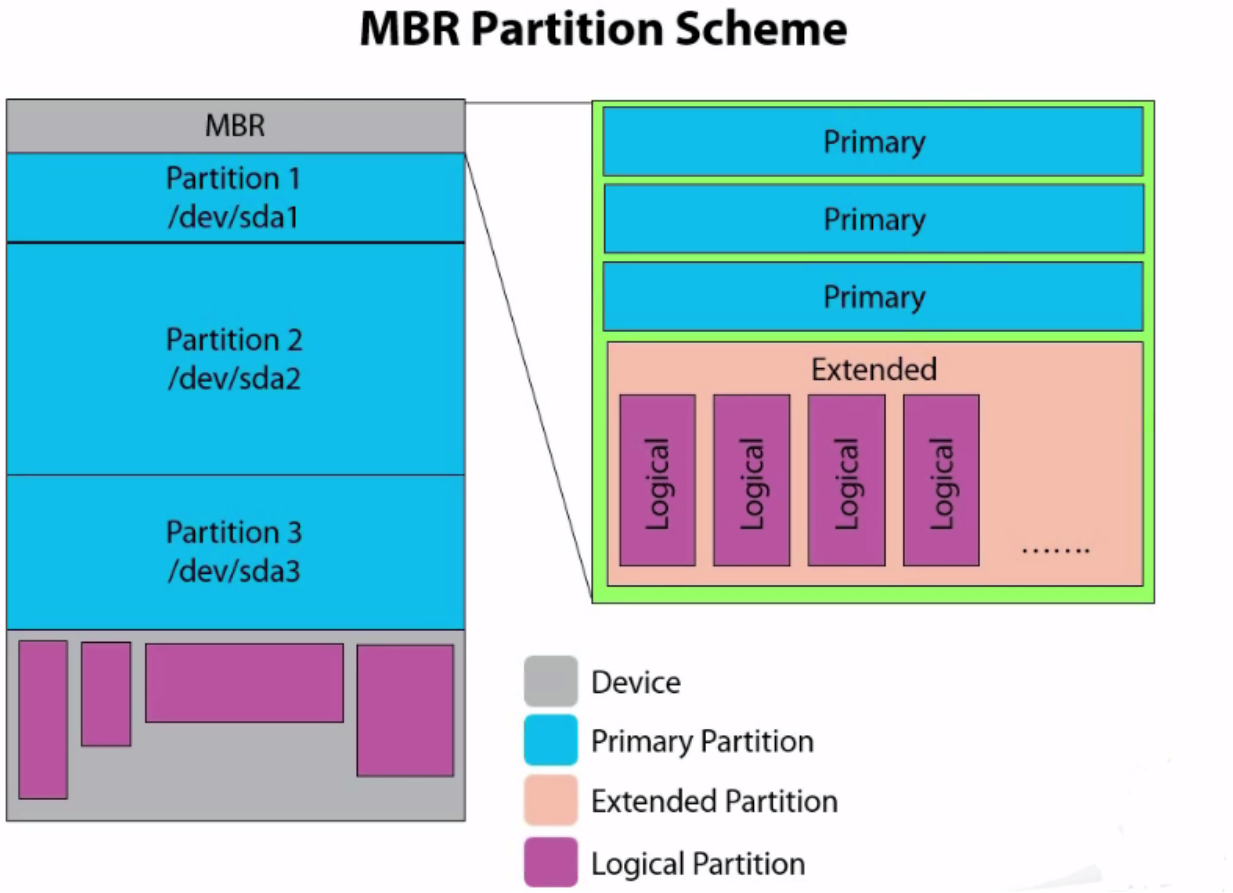


MBR Partitions

4 Primary partitions

1 can be an extended partition

Extended can host 1-n partitions in it.



Traditional boot records (MBR) vs. GPT boot recs.

MBR – MASTER BOOT RECORD

Older **legacy** partition style = Compatibility

Boot data stored at the BEGINNING

Allows up to **4 primary partitions**

Supports drives (volumes) **up to 2TB capacity.**

Supports partitions up to 2TB capacity.

Sensitive to corruption – Only one place for boot data – no redundancy

GLOBALLY UNIQUE IDENTIFIER - GUID

GUID PARTITION TABLE – GPT

Modern disk partition style

Boot data is scattered @ different locations on the disk

Allows up to **128 primary partitions**

Supports disks **more than 2TB** (9ZB*)

Resistant to corruption due to redundancy of boot data

Note: GUID –> Global Unique Identifier

GPT Boot record

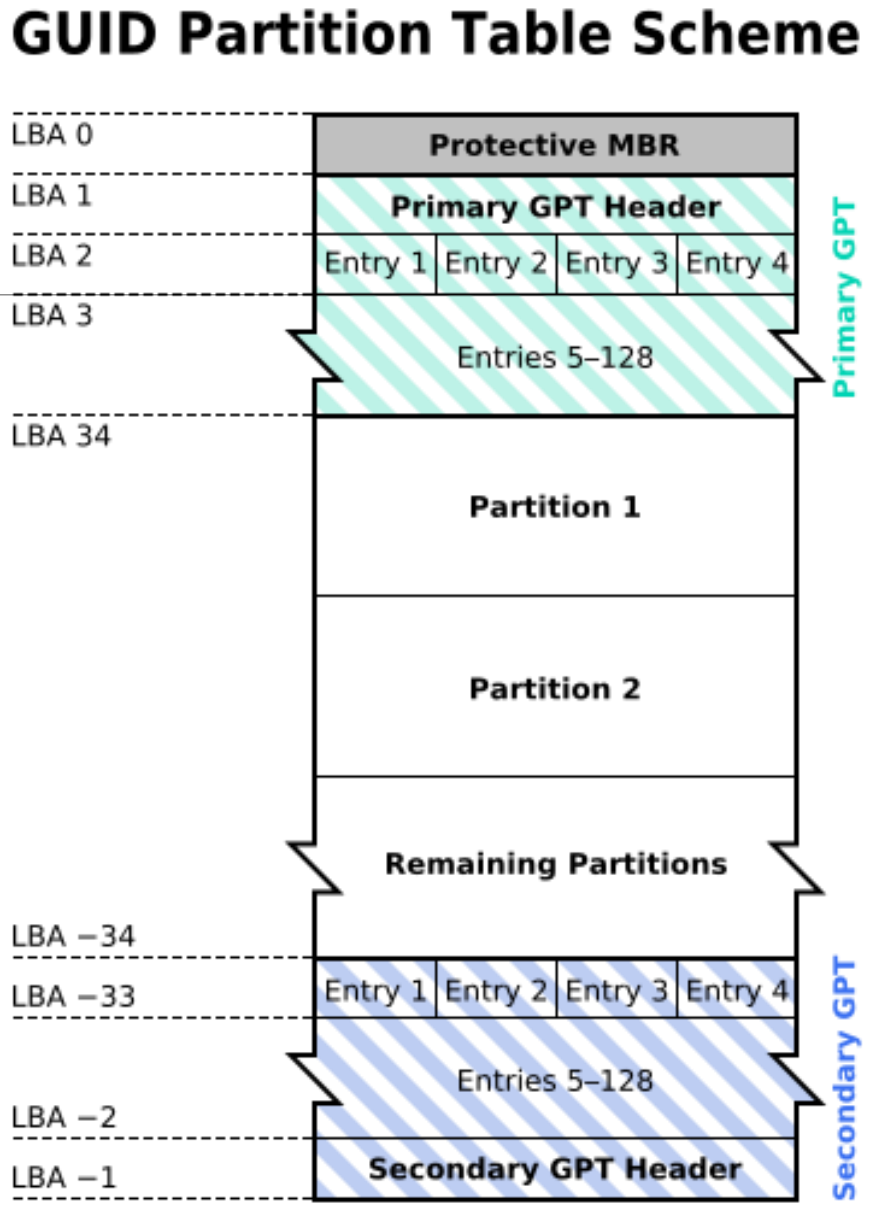
Contains a “MBR” also so that it can be seen by older disk utilities.

Contains unlimited partitions. (Microsoft limited to 128 for Windows). (Linux can use more)

Every partition on your drive has a “globally unique identifier,” or GUID

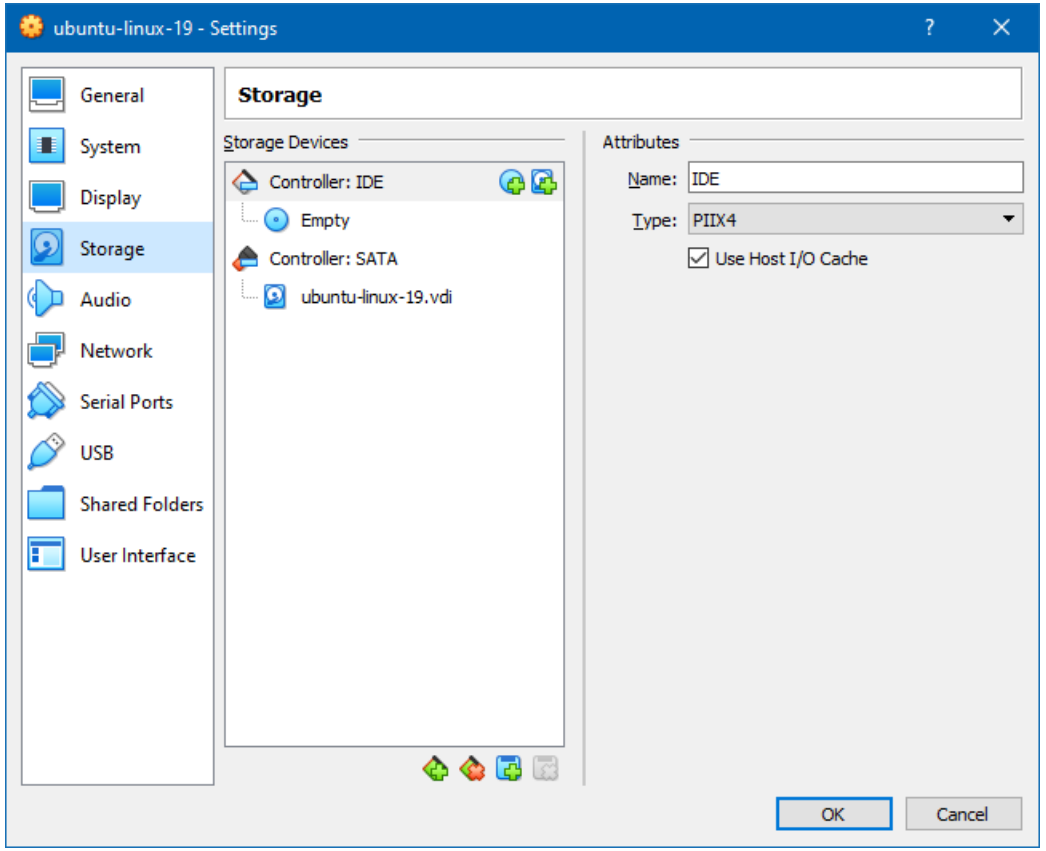
GPT also stores cyclic redundancy check (CRC) values to check that its data is intact. MBR does not.

An entry in a GUID Partition Table (GPT) partition refers to a record within the GPT's partition table. The GPT header defines the number and size of partition entries. Each entry corresponds to an individual partition on the disk. These entries contain the unique identifiers (GUIDs) and attributes for the partitions, such as starting and ending sectors, name, and type, which collectively define the layout and properties of the partitions on a GPT-formatted disk.



Hard Drive Setup – Using a Virtual Machine

1. Go to the "settings" of the virtual machine.
2. Click the Storage tab



Add a New Drive

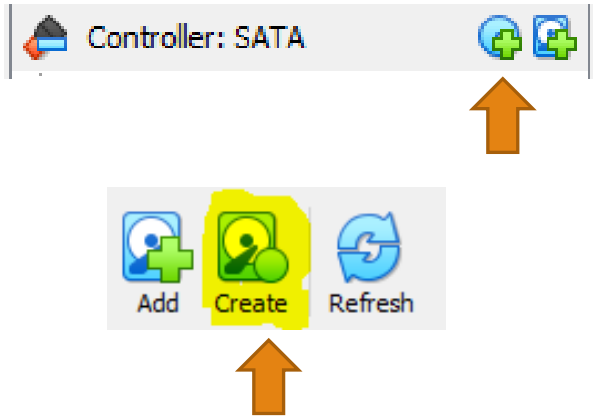
Click on the new hard drive icon.

Click on "Create" to create a logical drive.

Choose "VDI" image

Choose dynamic or fixed.

At the end, select "Choose" to add it.



In VirtualBox, a VDI file is a Virtual Disk Image. It is a storage file format that represents a virtual hard disk drive (HDD). A VDI file can be used as the storage media for a virtual machine's operating system, applications, and data. It acts like a physical hard disk but is stored as a file on your host computer's file system. VirtualBox uses VDI files as one of several supported disk image formats for virtual machines.

VirtualBox 7 How to add an additional hard drive (youtube.com)

What drive did we just Add?

In the "dev" directory, we can see a clue about what drives are available to the system.

```
$ ls /dev/sd* -l
brw-rw---- 1 root disk 8,  0 Mar 20 21:33 /dev/sda
brw-rw---- 1 root disk 8,  1 Mar 20 21:33 /dev/sda1
brw-rw---- 1 root disk 8, 16 Mar 20 21:33 /dev/sdb
```

Note:

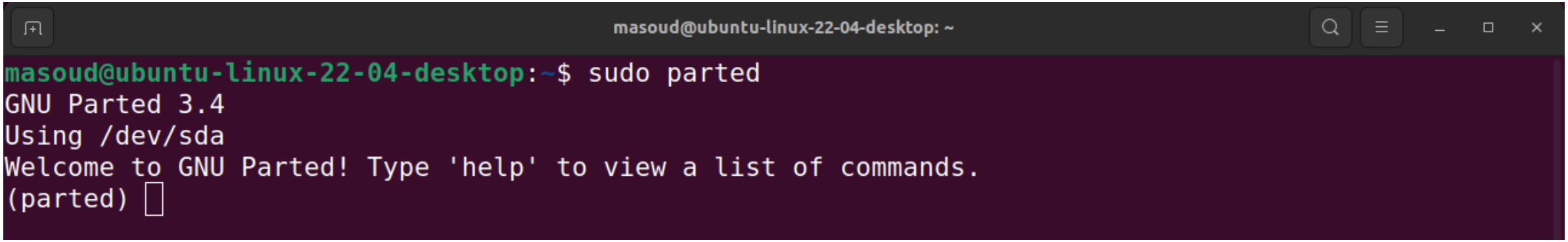
- sda = The hard drive #1
- sda1 = Partition #1 on hard drive #1.
- sdb = Hard drive #2 (with no partitions on it).

sda: SCSI drive a

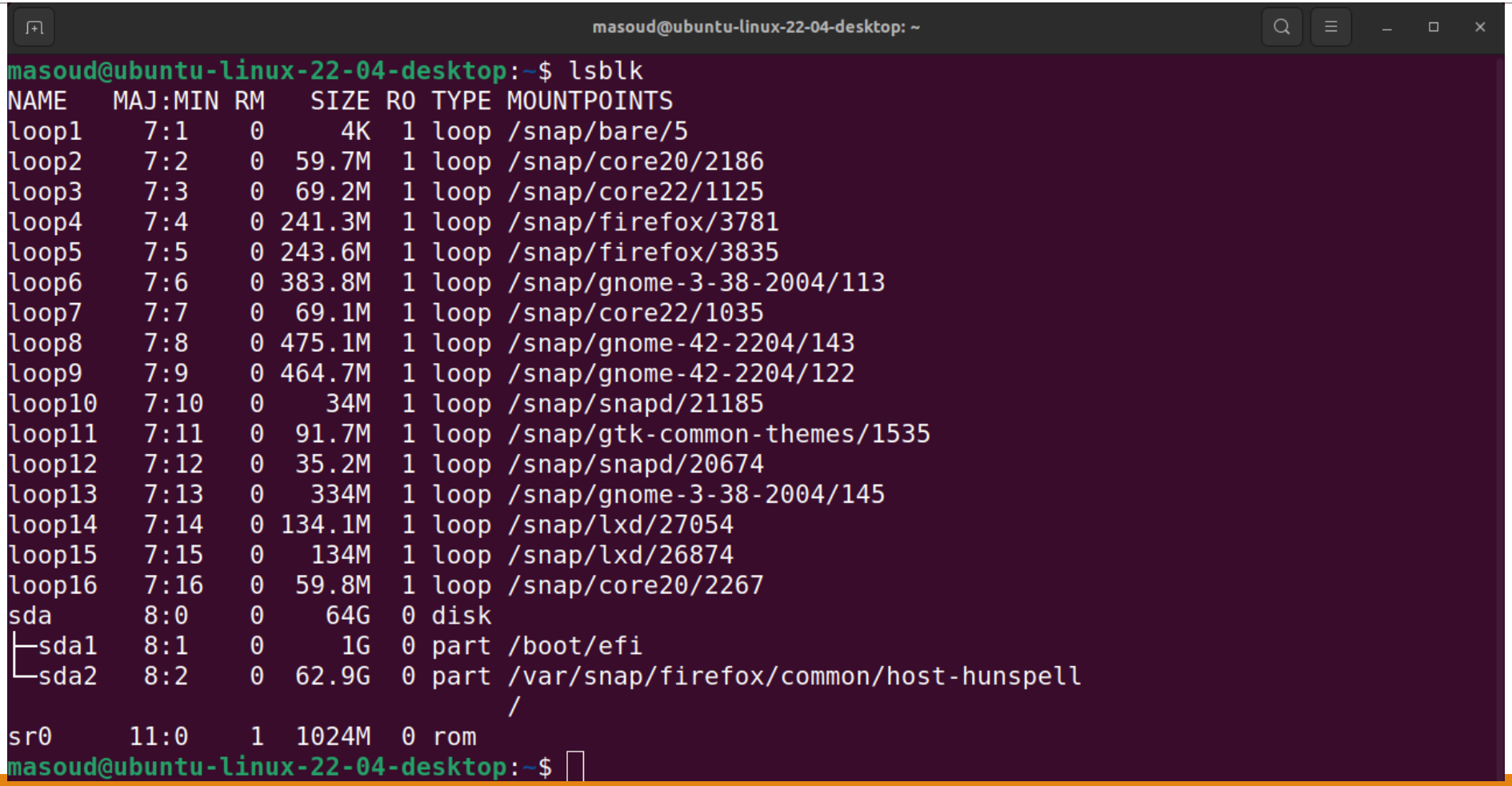
SCSI: Small Computer System Interface

parted

In Linux, `parted` is a powerful command-line tool used for managing disk partitions. It allows you to create, resize, delete, and modify partitions on a disk without data loss. `parted` supports a variety of disk labels and is capable of handling multiple partition table formats, including GPT (GUID Partition Table) and MBR (Master Boot Record).



list block devices



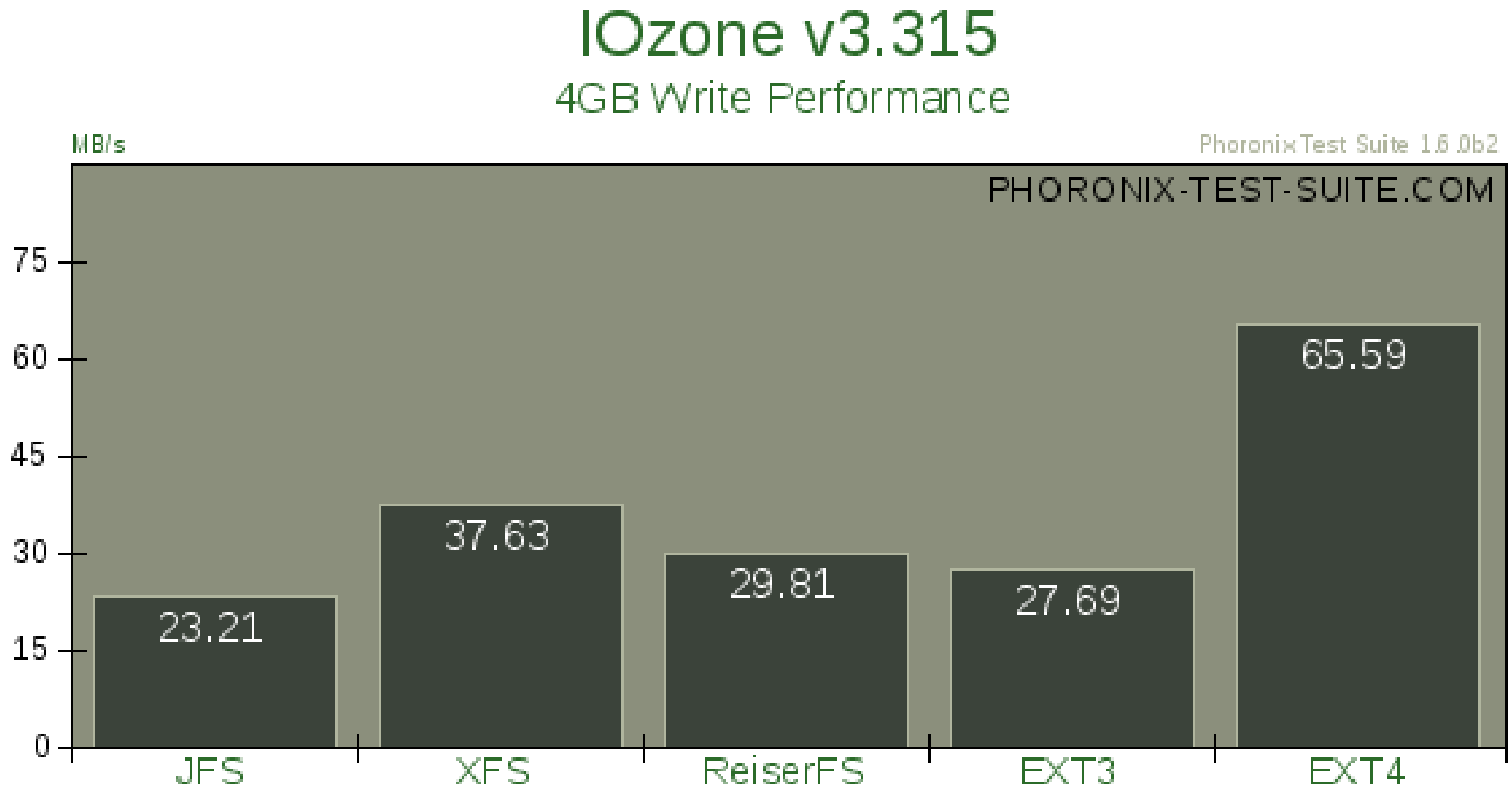
Linux File system types

Ext2 is not a journaling file system. When introduced, it was the first file system to support extended file attributes and 2-terabyte drives. Ext2's lack of a journal means it writes to disk less, which makes it useful for flash memory like USB drives. However, file systems like exFAT and FAT32 also don't use journaling and are more compatible with different operating systems, so we recommend you avoid Ext2 unless you know you need it for some reason.

Ext3 is basically just Ext2 with journaling. Ext3 was designed to be backward compatible with Ext2, allowing partitions to be converted between Ext2 and Ext3 without any formatting required. It's been around longer than Ext4, but Ext4 has been around since 2008 and is widely tested. At this point, you're better off using Ext4.

Ext4 was also designed to be backward compatible. You can mount an Ext4 file system as Ext3, or mount an Ext2 or Ext3 file system as Ext4. It includes newer features that reduce file fragmentation, allows for larger volumes and files, and uses delayed allocation to improve flash memory life. This is the most modern version of the Ext file system and is the default on most Linux distributions.

Write performance for EXT3 / EXT4



Useful commands for disk management

df

- Checks the disk space usage
- Gives device name, free space, used space, total space.

du

- Shows usage by directory.

fdisk

- Manage partitions for MBR

gdisk

- Manage partitions for GPT

parted

- Manage partitions for both GPT and MBR

Useful commands for disk management

lsblk

- Lists out all the storage blocks, which includes disk partitions and optical drives. Details include the total size of the partition/block and the mount point if any.
- Does not report the used/free disk space on the partitions.

blkid

- Prints the block device (partitions and storage media) attributes like uuid and file system type. Does not report the space on the partitions.

gdisk

- Manage partitions for GPT

hwinfo

- The hwinfo is a general purpose hardware information tool and can be used to print out the disk and partition list.

Important utility: mkfs
Make file system.

mkfs is the utility that writes the file system itself on the drive. A partition does nothing without a file system in place.

In Windows, this is the equivalent of "FORMAT".

mkfs *[options] [-t type fs-options] device [size]*

-t specify ext2, ext3, ext4

Can also specify types for windows like msdos or ntfs

Can be rewritten as “mkfs.type /dev/partition”

Example:

- mkfs.ext4 /dev/sdb1

FSTAB

FSTAB Means "File System Table"

This table automatically mounts drives at boot up time.

To ensure that the drives are mounted you must do two things:

1. Create a directory as a mountpoint
2. Put an entry in fstab to map the physical drive to the mountpoint.

/etc/fstab Contents

```
sudo vi /etc/fstab

/dev/sdb1 /mnt/drive1 ext4 defaults 0 0

/dev/sdb2 /mnt/drive2 ext4 defaults 0 0
```

The entry structure for each mapping is:

- The drive or partition : /dev/sdb1
- The mountpoint on Linux FS: /mnt/drive1 (must exist, create it!)
- The filesystem expected: ext4 (Linux usually is ext4)
- Defaults (leave as is)
- 0 0 : 0 = Backup, 0 = Skip Check

parted Create New Partitions

Create 2 primary ext4 partitions and use MBR, taking exactly half the drive each.

```
sudo parted –align optimal  /dev/sdb
mklabel msdos
mkpart primary ext4 0% 50%
mkpart primary ext4 50% 100%
```

Example Partitioning Scenario

Start PARTED

- sudo parted /dev/sdb

Tell it to use a GPT type table

- (parted) mklabel gpt
- (parted) quit

Make a partition

- Start parted with the "-a optimal" to choose optimal alignment automatically.
  - $ parted --align optimal /dev/sdb
  - (parted) **mkpart primary 0% 100%**
  - (parted) quit
  - Information: You may need to update /etc/fstab.

Example - mount

Once the drive has been partitioned and formatted, you can "mount" the drive on a mountpoint to save files to it.

The –o option, means "options" – and here we just accept the default options.

The first mount here means, "Mount sdb1 into the mountpoint called /mnt/data_drive_1".

Note: The mount disappears once you reboot, to make it permanent you need to create an "fstab" entry for it (See next slides)

```
mkdir /mnt/data_drive_1
mount -o defaults /dev/sdb1 /mnt/data_drive_1
mkdir /mnt/data_drive_2
mount -o defaults /dev/sdb2 /mnt/data_drive_2
```

Example mkfs

Make the file systems on the drives you created.

sudo mkfs.ext4 /dev/sdb1

Example Scenario

We need the following:

Using an MBR boot record:

- 3 Primary partitions of 2GB each (ext4)
- 1 Extended from 6GB to end
- 2 Logical of 2GB each – but make the last one take ALL remaining space.

End

LINUX STORAGE