## Disks and File Systems

Steps:

- Install a physical drive
- Partition it
- Create a file system on the partition
- Mount the filesystem in the root partition
- Use FS
- Umount FS

---

## What is a partition?

A logical division of a hard disk. The OS treats a partition as if it were a distinct physical device.

---

## Why partition?

- Different OS's on each partition
- Separate system data from user space data (i.e. home drives)
- Protection (in case of failure, or corruption)
- Able to grow or resize partitions
- Better performance

---

## More reasons for partitioning (from tldp.org)

- Encapsulate your data. Since file system corruption is local to a partition, you stand to lose only some of your data if an accident occurs.

- Increase disk space efficiency. You can format partitions with varying block sizes, depending on your usage. If your data is in a large number of small files (less than 1k) and your partition uses 4k sized blocks, you are wasting 3k for every file. In general, you waste on average one half of a block for every file, so matching block size to the average size of your files is important if you have many files.

---

## More reasons for partitioning (from tldp.org)

- Limit data growth. Runaway processes or maniacal users can consume so much disk space that the operating system no longer has room on the hard drive for its bookkeeping operations. This will lead to disaster. By segregating space, you ensure that things other than the operating system die when allocated disk space is exhausted.

---

## Linux Partitions

- In Linux, at least 1 partition is required for the `/`.
- `Mounting` is the action of connecting a filesystem to a particular point in the `/` root filesystem. I.e. When a usb stick is inserted, it is assigned a particular mount point and is available to the filesytem tree.

---

## Partition Details

- PC style machines used to use DOS partitions
- Still seen today but largely has been replaced by GPT
- 4 primary partitions
  - One of these 4 can be used as an extended partition
  - Any number of logical may be created in the extended partition

---

## Primary vs Logical

One primary partition of a hard drive may be subpartitioned. These are logical partitions. This effectively allows us to skirt the historical four partition limitation.

---

## More about devices

- Device naming conventions
  - Dependent on hardware used
  - Partitions are numbered
    - /dev/sda1 (refers to first partition of first SATA or SCSI hd)
    - /dev/sdc3 (refers to third partition of third SATA or SCSI hd)

---

## Partition tools

- fdisk
- cfdisk
- parted (also has a gparted GUI)
- sfdisk

---

## Filesystem Types

Once we have partitioned the disk, then we choose the way it should be formatted. What type of data organization do we want on the disk?

Types to choose from:

- ext3: old linux
- ext4: linux default
- ntfs: windows default
- vfat: flash drives, cameras, minimal
- fat: old dos (windows)
- etc ...

---

## The EXT filesystem

- Data is organized for the OS
- Superblocks

---

## The EXT filesystem tools

- mkfs -t ext4 /dev/sdc1
  - Format the disk with type ext4
- mount
  - makes device part of / tree
- umount
  - removed device from fs tree
- tune2fs
  - Tune various parameters (beyond our scope)

---

## What is an inode?

- The filesystem is made up of inodes.
- Inodes contains metadata about a file.
  - size
  - dev id
  - UID, GID
  - OTher stuff
  - DOES NOT STORE FILENAME.

---

## Viewing inodes

- `df -i` will show us how many inodes we have and how many available to us.
- Can we have more files than inodes?
- `ls` doesn’t access inode metadata, but `ls -l` does.
    - `ls -i` will give us information about inode
- `stat f1.txt` will give us metadeta information.

---

## More testing with inodes

- Examine inodes when:
    - we create a symbolic link?
    - we create a hard link?
- So, files can have multiple names.

---

## Other inode implications

- An inode may have no links
- A files inode number stays the same when moved to another directory on the same device.
- root generally has an inode of 2? (`ls -lai /`)

---

## More Inode stuff

Where is the NAME of the file. Or the Path? It’s NOT in the inode. It’s NOT in the data blocks. It’s *in* the directory. That’s right. A “file” is really in three (or more) places on the disk.

You see, the directory is just a table that contains the filenames in the directory, and the matching inode. Think of it as a table, and the first two entries are always “.” and “..” The first points to the inode of the current directory, and the second points to the inode of the parent directory.

---

## One more piece of information

When you create a hard link, it just created a new name in the table, along with the inode, without moving the file. When you move a file (or rename it), you don’t copy the data. That would be Slow. You just create the (name,inode) entry in a new directory, and delete the old entry in the table inside the old directory entry. In other words, moving a gigabyte file takes very little time. In the same way, you can move/rename directories very easily. That’s why “mv /usr /Old_usr” is so fast, even though “/usr” may contain (for example) 57981 files.

---

## Inode Examples

- Create a dir and cd into it
    - What does output of `ls -id .` show?
    - Cd back to parent directory and `ls -id testdir`, should show same inode number
- Now do an `ls -la` on that directory. What is the field after the permissions? This is count of hard links to the file.
    - We should see that there are (2) hard links, one for filename and one for `.`.

---

## Inode Examples

- Look at this output:

```
drwxrwxr-x  6 joe  joe        4096 Oct  6 10:15 it1100
drwxrwxr-x 41 joe  joe        4096 Oct 22 11:22 lab6
```

- What does the 41 mean on line 2?

- Create a dir (look at that column), create subdir (look), create sub,subdir(look)

---

## Finding with inodes.

- `find / -inum 147649`

---

## Steps for partitioning More Detailed

1. Use cfdisk or another program to partition free space
2. Run the `mkfs` command to set up the filesystem(i.e. mkfs.ext2 /dev/sdb5)
3. Create a mount point (i.e. `mkdir testmount`)
4. Run the mount command (i.e. `mount /dev/sdb5 testmount`)

---

## Making mounts persist

When issuing a mount command at the command line, the mount will only persist until the machine is rebooted. If we always want that particular mount to persist we can put an entry in `/etc/fstab`.

- See `/etc/fstab`
- Here is an example entry
    - `/dev/sda6 /home/joe/op auto defaults 0 0`

---

## Fstab explained

Here is a short explanation:

- `/dev/sda6` - the device we are going to mount
- `/home/joe/op` - the mount point
- `auto` - automatically detect what type fs this is (ext3 or ext4 for example)
- `defaults` - options associated with mount
- `0 0` - advanced, usually these are always 0.

---

## One more note about fstab

Ubuntu now uses UUID to identify partitions. To find out what the UUID is for a particular device we can use:

- `sudo blkid`

Why use a UUID instead of /dev/sda1 to identify the device? Say that that you plugged another disk into your computer and booted up. It probably won't happen, but it is possible that the new disk might be identified as /dev/sda, causing the system to look for the contents of /boot on the first partition of that disk.

---

## What is the MBR

The information about how a hard disk has been partitioned is stored in its first sector (that is, the first sector of the first track on the first disk surface). The first sector is the master boot record (MBR) of the disk; this is the sector that the BIOS reads in and starts when the machine is first booted. The master boot record contains a small program that reads the partition table, checks which partition is active (that is, marked bootable), and reads the first sector of that partition, the partition's boot sector (the MBR is also a boot sector, but it has a special status and therefore a special name). This boot sector contains another small program that reads the first part of the operating system stored on that partition (assuming it is bootable), and then starts it.

---

## Side Note about MBR Partitioning

We have been examining how partitioning has been done using the MBR. To see if we are in-fact using the MBR partitioning scheme we can do a `sudo parted -l`. If we see the `ms-dos` partition table this is what we are using. MBR only supports addressing 2TB

There are new and better alternatives to using the MBR partition table such as the GUID partition table (GPT).

---

## UEFI

Unified Extensible Firmware Interface

- Replaces the basic input/output system (BIOS)
- Developed by Intel (ca. 1998)

---

## UEFI Advantages

- Can boot from large disks (over 2TB)
- Now uses GPT instead of MBR
- CPU independent
- Flexible pre-OS environment (including networking)

---

## UEFI Booting

Unlike BIOS, UEFI does not rely on a boot sector, defining instead a boot manager as part of the UEFI specification. When a computer is powered on, the boot manager checks the boot configuration and, based on its settings, loads and executes the specified operating system loader or operating system kernel. The boot configuration is a set of global-scope variables stored in NVRAM, including the boot variables that indicate the paths to operating system loaders or kernels, which as a component class of UEFI applications are stored as files on the firmware-accessible EFI system partition (Wikipedia)

---

## UEFI Can do things BIOS cannot

- Can read a partition table
- Can access files in some specific filesystems
- Can execute code in a particular format

---

## GPT

- Supports 128 partitions
- Supports Zettabyte hard disks(one billion terabytes)
- Usually used in conjunction with UEFI
- Can be used on BIOS