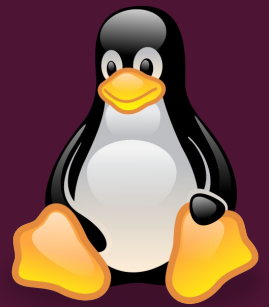
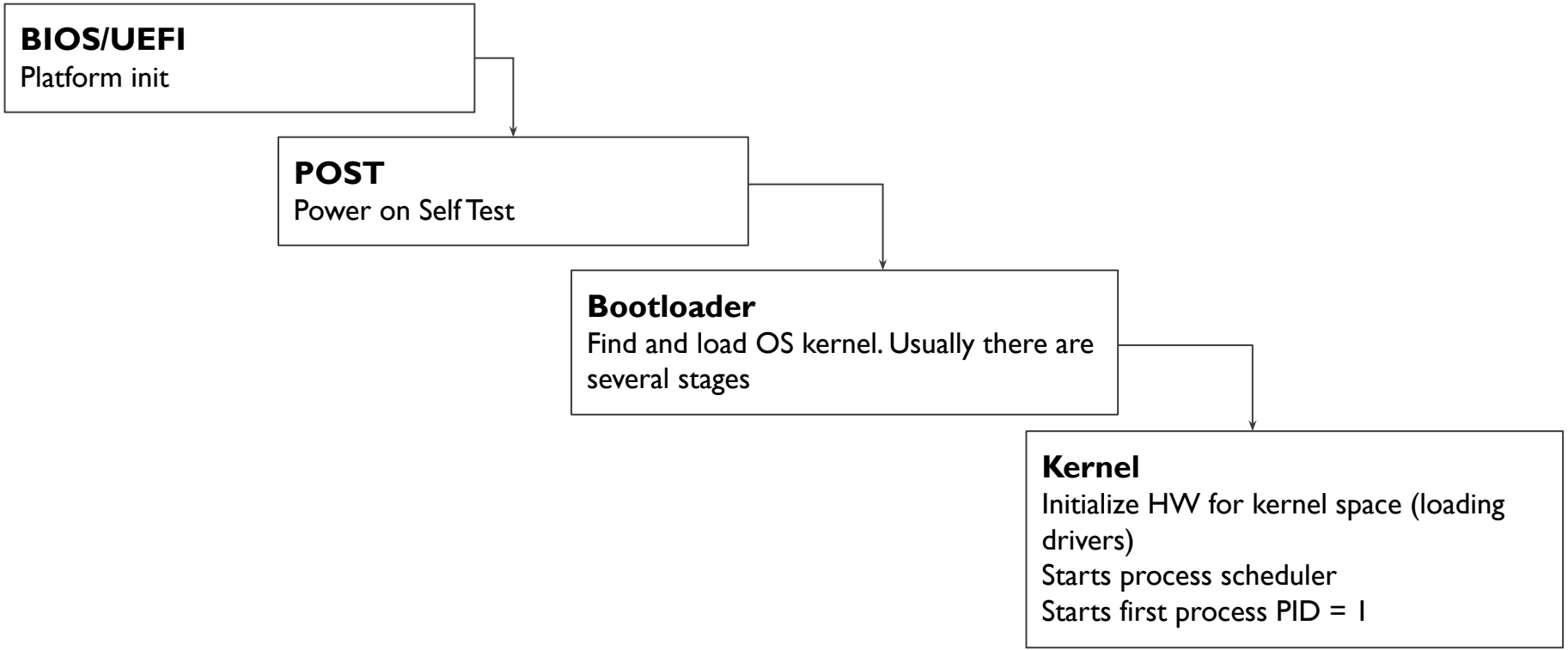
System and Network Engineering - Lecture 2

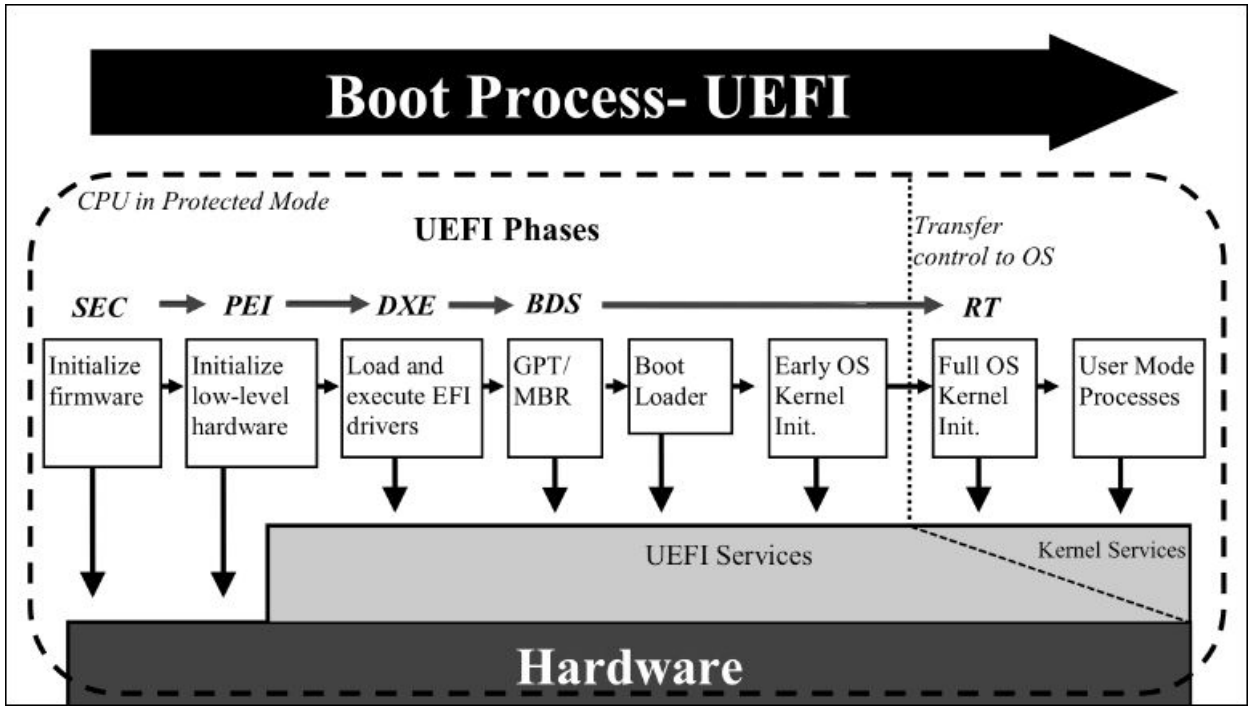
$ Linux Main Components



OS Boot Process



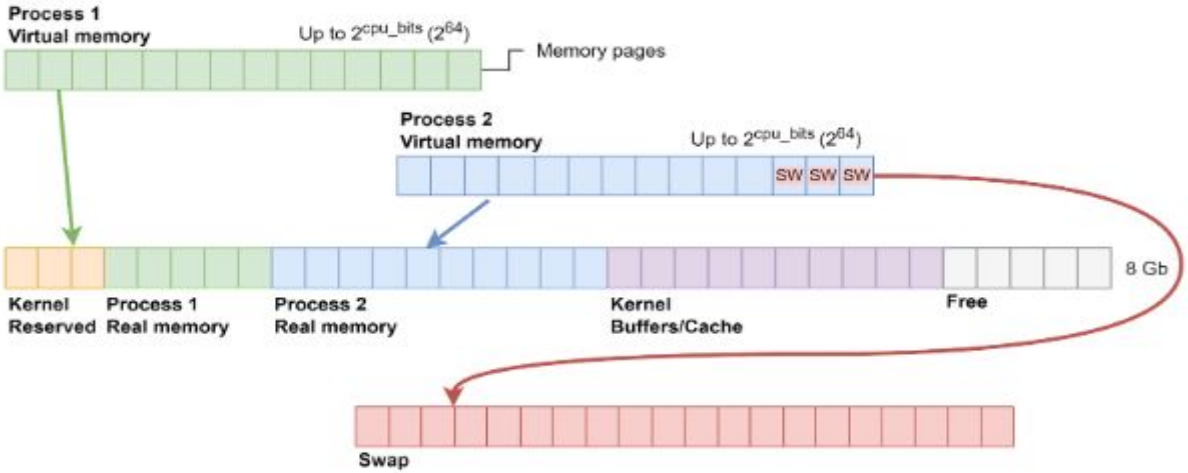
OS Boot Process: UEFI - Ubuntu OS loader - Kernel services



src: UEFI secure boot

Memory Management

Virtual Memory - OS shows the process that it owns all memory on the computer
Resident Memory - part of the real, physical memory that consumed by the process
Kernel I/O buffers and cache - memory used for I/O data storage/caching
Swap - on-disk additional memory emulation (optional)



Process - Kernel

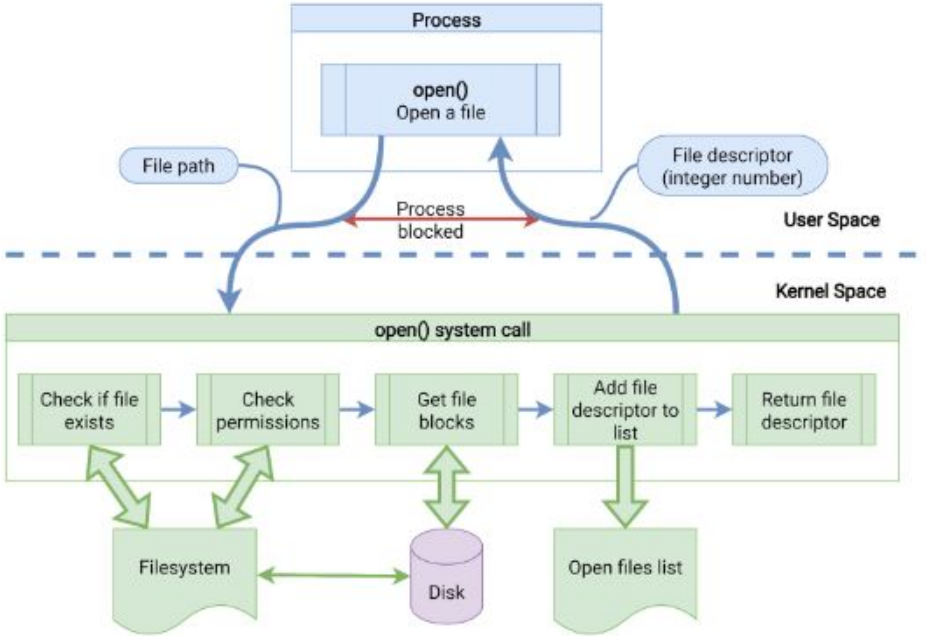
Process interacts with hardware only through the kernel

System calls:

- malloc ()
- open(), read(), write(), close()
- Lots of them

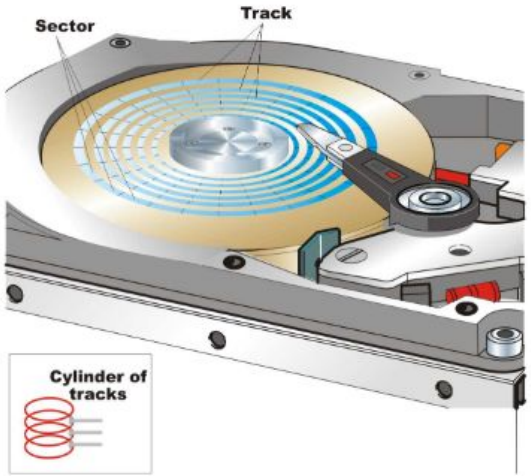
System calls are interrupting process execution via Software interrupts

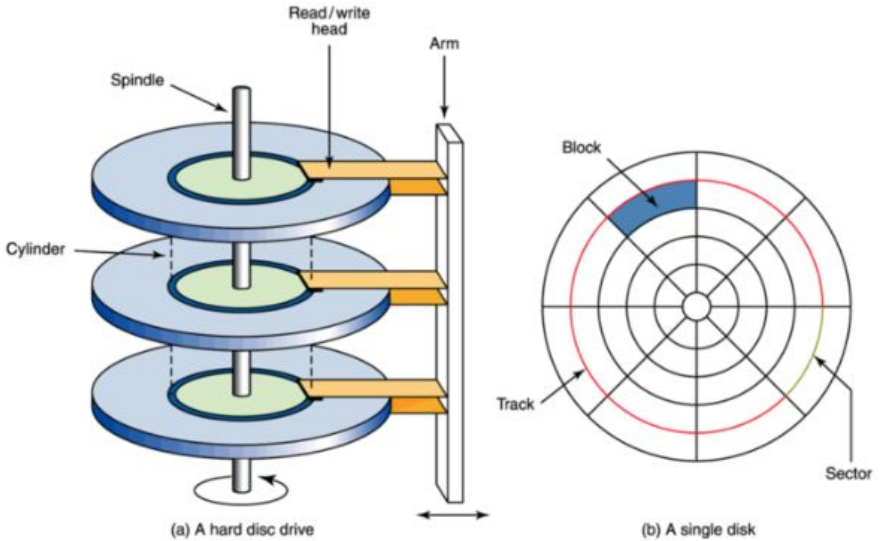
Software interrupt is triggered by process and considered one of the ways to communicate with kernel or to trigger system calls, especially during error or exception handling.



Storage - Block devices

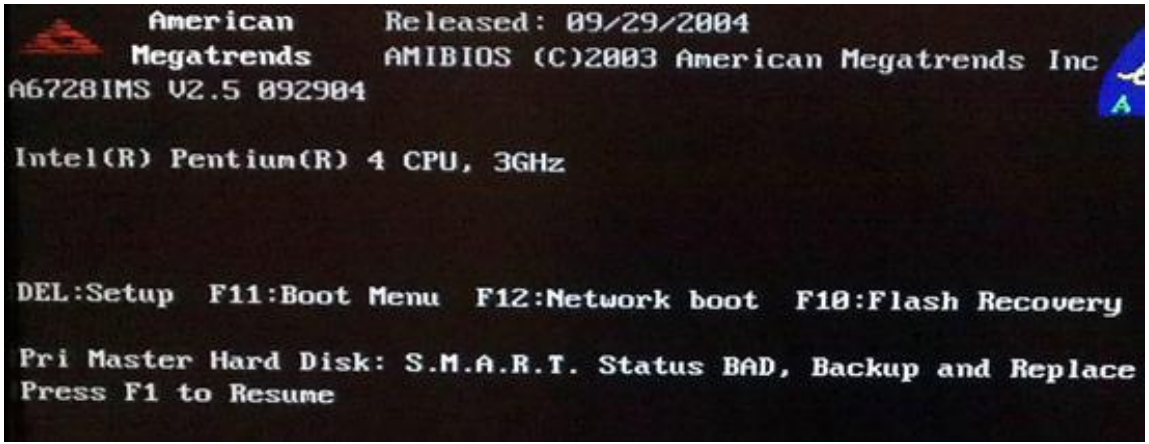
Most **storage devices** (HDDs, SSDs, flash drives, CD/DVD, etc) write and read data in blocks





Storage - Block devices - S.M.A.R.T.

S.M.A.R.T. (Self-Monitoring, Analysis and Reporting Technology) is an internal monitoring and reporting technology built into most modern hard drives. Its main purpose is to detect anomalies and predict failure.



Storage - Block devices - S.M.A.R.T.

- Some attributes could be used for an analysis:
  - **#Power-On Hours:** this could help establish for how long a hard drive was used
  - **#Power Cycle Count:** this could help establish how many times the drive was power cycled
- Acquiring attributes could be done by *smartmoontools*
- Each time disk will be connected and run, S.M.A.R.T. attributes will be changed even if write protection is enabled

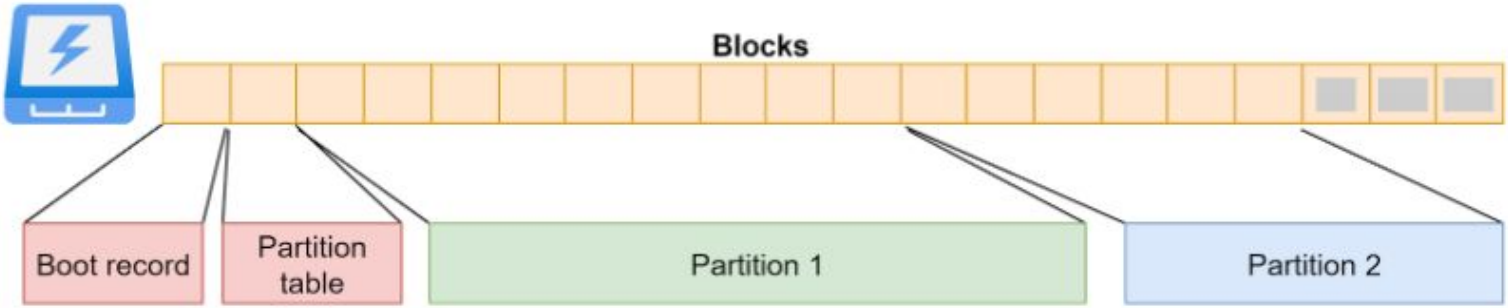
ID	Current	Worst	Threshold	Data	Status
(01) Raw Read Error Rate	114	99	6	81696809	ok
(03) Spin Up Time	97	97	0	0	ok
(04) Start/Stop Count	100	100	20	116	ok
(05) Reallocated Sector Count	100	100	36	0	ok
(07) Seek Error Rate	85	60	30	377203607	ok
(09) Power On Hours Count	65	65	0	31269	ok
(0A) Spin Retry Count	100	100	97	0	ok
(0C) Power Cycle Count	100	100	20	58	ok
(B7) SATA Downshift Count	100	100	0	0	ok
(B8) End To End Error Detection	100	100	99	0	ok
(BB) Uncorrectable Error Count	100	100	0	0	ok
(BC) Command Timeout	100	100	0	0	ok
(BD) Unknown Attribute	100	100	0	0	ok
(BE) Airflow Temperature	72	61	45	572194844	ok
(C2) Temperature	28	40	0	81604378652	ok
(C3) Hardware ECC Recovered	48	39	0	81696809	ok

Storage - Partitions and data layout

Partitions - a way to divide a storage device into several virtual devices

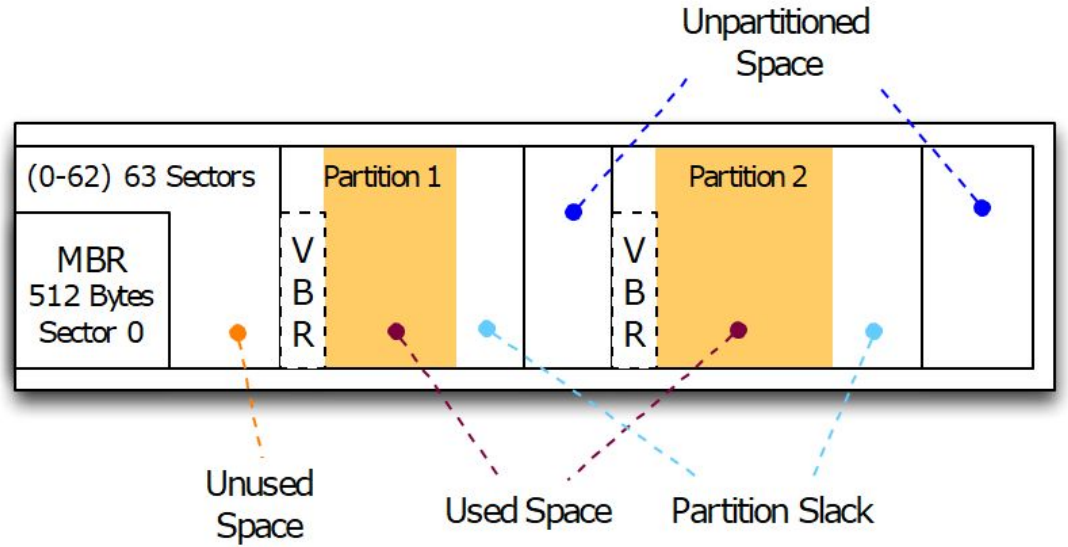
Partitions are sequentially allocated

Partitions can be extended and in some cases shrinked



Storage - Partitions and data layout - slack space

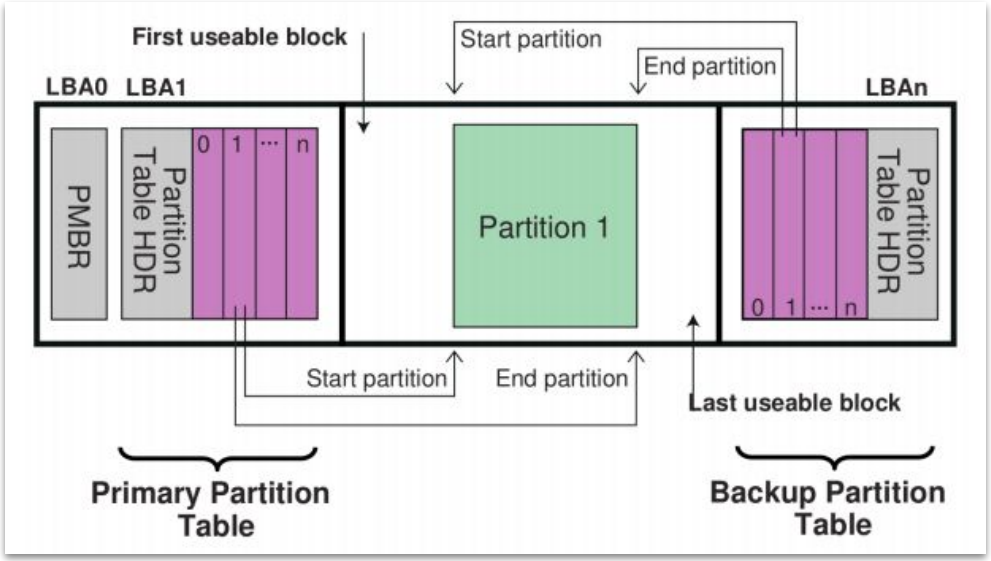
Slack Space – is a space on HDD which is not accessible by typical OS means. There might be unused space on the disk or space that cannot be used due to different reasons



Storage - Partitions and data layout - New Design: GUID Partition Table

Main Features:

- Logical Block Addresses (LBAs) are 512 bytes long
- Supports up to 128 primary partitions per disk
- Provides both a primary and backup partition table for redundancy
- Uses CRC32 fields for improved data integrity
- Defines a GUID for uniquely identifying each partition



Storage - Partitions and data layout - New Design: GUID Partition Table

Slack Space in GPT?

- Reserved by the system areas (for example between GPT Primary Partition Table and the First Partition)
- ((2048* 512 bytes) - (1 {Safety Table} + 1 {GPT Header} + 32 {Partition Table}) * 512 bytes) = 1031168 bytes = 1Mb

```
root@linuxPC:~# mmls /dev/sda
GUID Partition Table (EFI)
Offset Sector: 0
Units are in 512-byte sectors

      Slot      Start        End          Length       Description
000:  Meta      0000000000   0000000000   0000000001   Safety Table
001:  -------   0000000000   0000002047   0000002048   Unallocated
002:  Meta      0000000001   0000000001   0000000001   GPT Header
003:  Meta      0000000002   0000000033   0000000032   Partition Table
004:  000       0000002048   0001050623   0001048576   EFI System Partition
005:  001       0001050624   0042037247   0040986624
006:  -------   0042037248   0042038335   0000001088   Unallocated
```

Storage - Partitions and data layout - New Design: GUID Partition Table

Partition type GUID vs UUID vs Partition UUID

- matters when working with multiple disks and different types of partitions
- e.g. usage in fstab

EFI System partition	C12A7328-F81F-11D2-BA4B-00A0C93EC93B

```
root@linuxPC:~# blkid /dev/sda1
/dev/sda1: UUID="B5C8-785B" BLOCK_SIZE="512" TYPE="vfat" PARTLABEL="EFI System Partition
" PARTUUID="a25ba309-874e-4739-b458-8889ec820220"
```

```
root@linuxPC:~# sudo lsblk -f | grep -v loop
NAME   FSTYPE   FSVER LABEL UUID                                 FSAVAIL FSUSE% MOUNTPOINTS
sda
├─sda1 vfat     FAT32       B5C8-785B                            505.7M     1% /boot/efi
└─sda2 ext4     1.0         1b401010-230c-4042-a9db-fb6febda3a0c    7.5G    55% /
sr0
```

Partition entries (LBA 2–33) [edit]

GUID partition entry format

Offset	Length	Contents
0 (0x00)	16 bytes	Partition type GUID (mixed endian[7])
16 (0x10)	16 bytes	Unique partition GUID (mixed endian)
32 (0x20)	8 bytes	First LBA (little endian)
40 (0x28)	8 bytes	Last LBA (inclusive, usually odd)
48 (0x30)	8 bytes	Attribute flags (e.g. bit 60 denotes read-only)
56 (0x38)	72 bytes	Partition name (36 UTF-16LE code units)

src: wiki

Filesystems (FAT 32, NTFS, EXT4, etc)

File - data object that is stored and operated by the system

FS metadata - store all information about files (but does not store the files content)

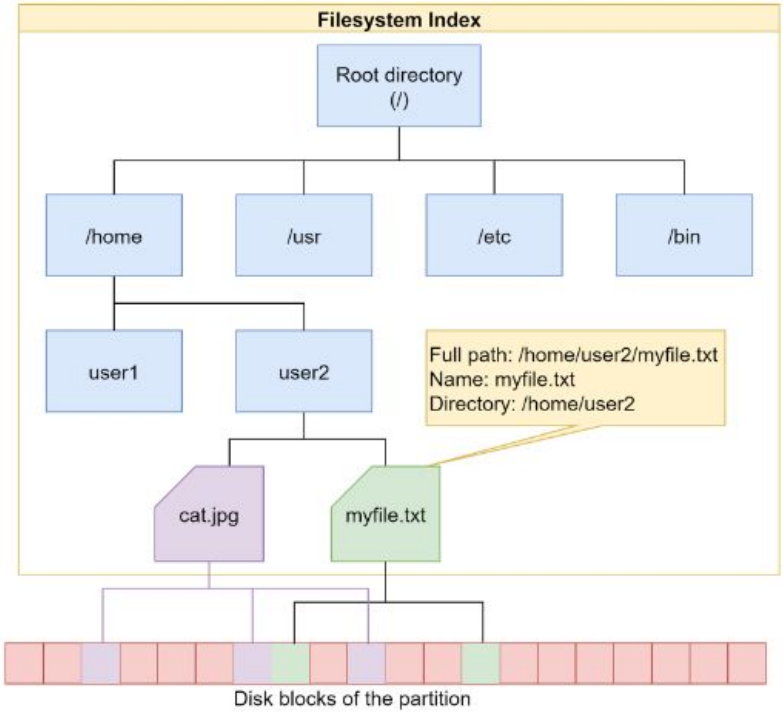
Hard link ("normal file") - points to the actual data

Symbolic link - points to another file object

Directory - no data, but can be "parent" for others

Path - string contain the full address of a filesystem object

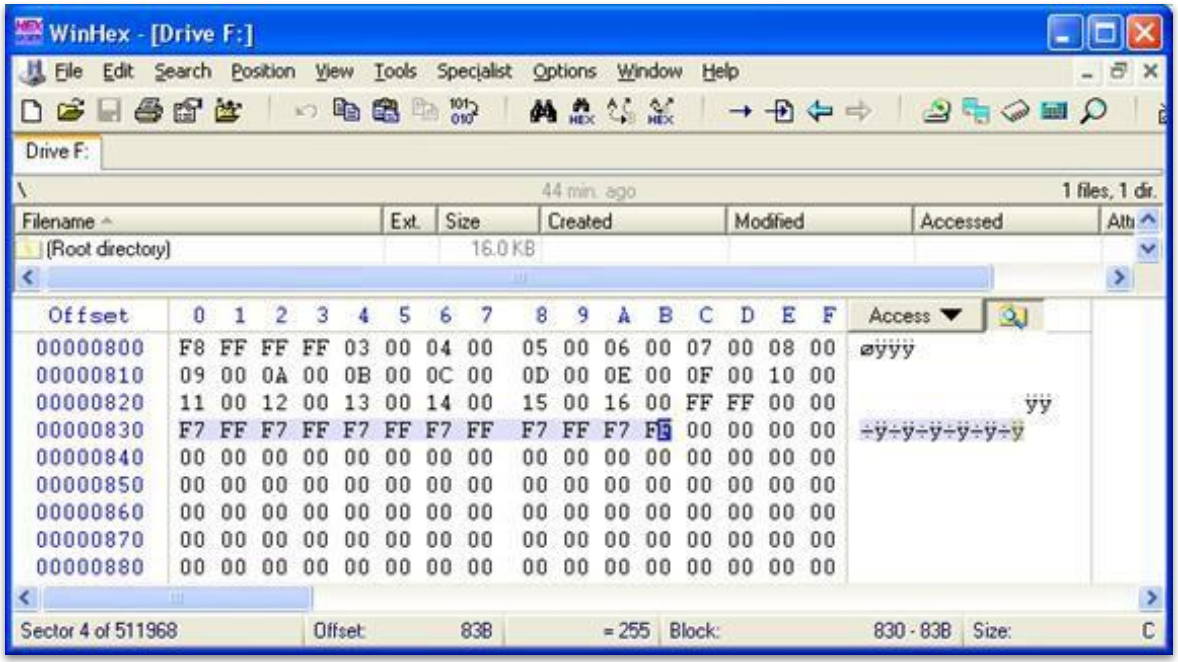
Filesystems **are mounted** to certain directories in the **OS paths hierarchy**



Filesystems (FAT)

File Allocation Table (FAT)

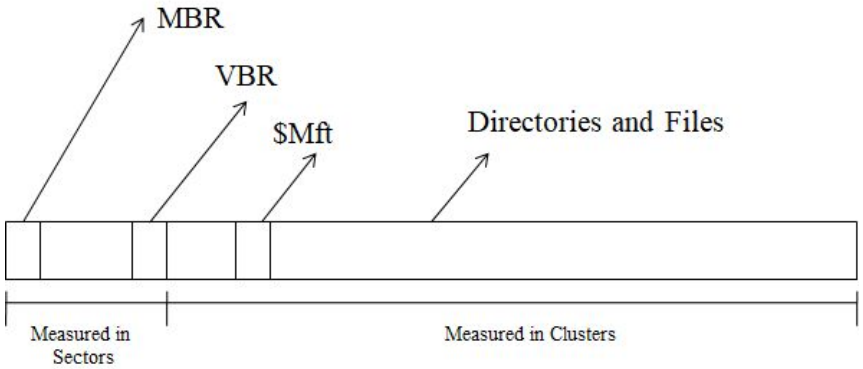
- Unused (0x0000)
- Cluster in use by a file (0x___)
- End of file (0xFFFF)
- Bad cluster (0xFFF7)



Filesystems (NTFS)

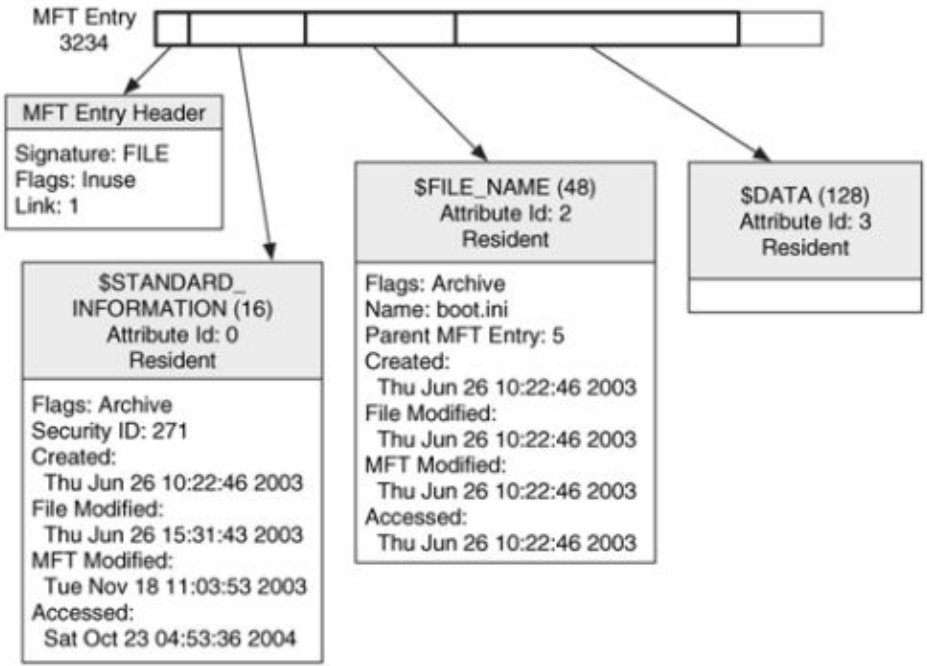
NTFS

- All important data are allocated to files ("everything is a file")
  - including file system administrative data that can be located anywhere in the volume
  - no specific layout
- First sectors contain the boot sector and boot code
- Any sector beyond those can be allocated to a file
- **Master file table (MFT)** contains info about all files and directories



Filesystems (NTFS)

MFT entry and basics 3 file attributes

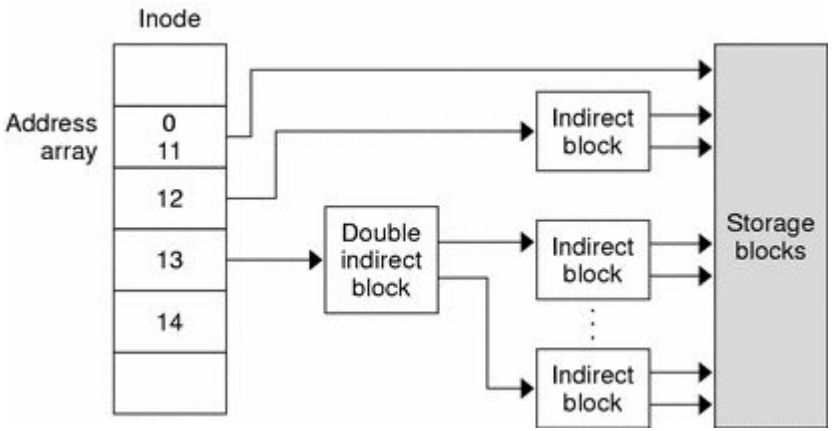


Filesystems (EXT4)

Inode - file data structure that stores information about any Linux file except its name and data.

Information contained in an inode:

- File size
- Device on which the file is stored
- User and group IDs associated with the file
- Permissions needed to access the file
- Creation, read, and write timestamps
- Location of the data (though not the file path)



Indirect pointer - when file exceeds the size of your inode, your original inode will have to point to another inode in order to fully encompass the file's metadata.

Filesystems (EXT4)

What can go wrong with inodes ?

Once inodes are 100% used, you'll begin to notice:

- Data loss
- Applications crashing
- OS restarting
- Processes don't restart
- Periodic tasks not working

```
root@linuxPC:/tmp# df -i
Filesystem      Inodes  IUsed   IFree IUse% Mounted on
tmpfs           500607    940  499667    1% /run
/dev/sda2      1281120 215772 1065348   17% /
tmpfs           500607      1  500606    1% /dev/shm
tmpfs           500607      4  500603    1% /run/lock
/dev/sda1            0      0       0     - /boot/efi
tmpfs           100121    151   99970    1% /run/user/1000
root@linuxPC:/tmp# ls -i /bin/bash
914049 /bin/bash
```

Use *df* to control inode capacity.

Virtual file systems

Virtual files represent objects that are not files, but can be managed like files. Examples:

- named pipes
- unix socket files

Virtual file systems are used to represent object hierarchies using virtual files. Examples:

- */proc/self/fd/0* - standard input
- */dev/stdin* - standard input

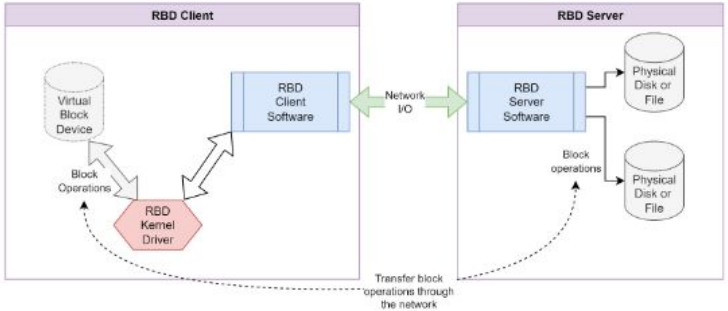
Main Principle - "Everything is a file"

- Sysfs(/sys) - kernel parameters (sysctl)
- Procfs(/proc) - processes and process management related parameters
- Devfs(/dev) - devices as files
- Named pipes and sockets

Network storage

Remote/Network block devices

- Virtual disk that can be used through the network
- Operates with raw devices and blocks (read block, write block, erase block)
- Can have filesystem, but its not mandatory
- Examples:
  - iSCSI, DRBD, GlusterFS block model, Ceph RBD



Network file systems

- Virtual file system that is operated through the network
- Uses filesystem operations (open file, read/write to file, close file, list directory content)
- Usually emulates local FS. Apps should not see the difference
- Examples:
  - NFS, SMB, GlusterFS, Ceph FS