Operating Systems

Filesystems

Sven Dziadek

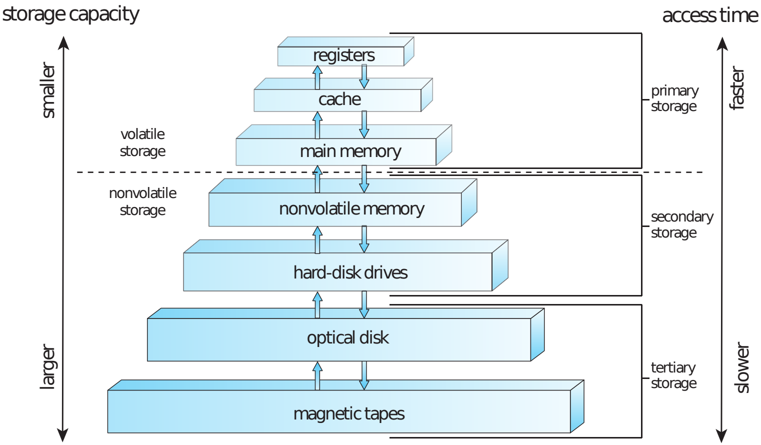
October/November 2022

EPITA

Storage devices

Recall: Storage Types

Today, we are talking about secondary storage:



Storage devices

Also called mass storage

Through the ages, lots of different norms & physical connections
The main ones:

- ATA
- Parallel ATA (IDE)
- SCSI
- ATAPI
- SATA
- NVMe also over PCIe

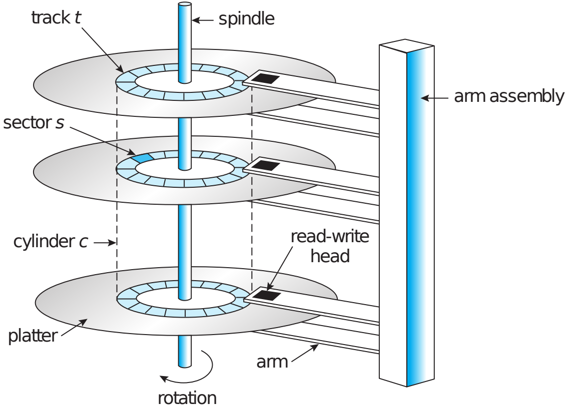
Hard Disk Drive

Seek time

Time to move the disk arm to desired cylinder

Sequential access preferred over random access

Note: Try storing data sequentially



Non-Volatile Memory (NVM)

Examples

SSD, flash memory storage (smartphone)

Access of data is **electronically** compared to hdd (electromechanically)
$\Rightarrow$ fast random access performance

Data cannot be overwritten
$\Rightarrow$ Copy changed data to new cell and mark old cell as unused

Garbage collection

Larger units of unused NAND cells are erased

Note: Tell SSD controller when deleting files (TRIM)

Some Similarities

Block Devices

Divided into blocks of fixed size

For all operations: specify the Logical Block Address (LBA)

Controllers

Kernel never talks with the drives directly

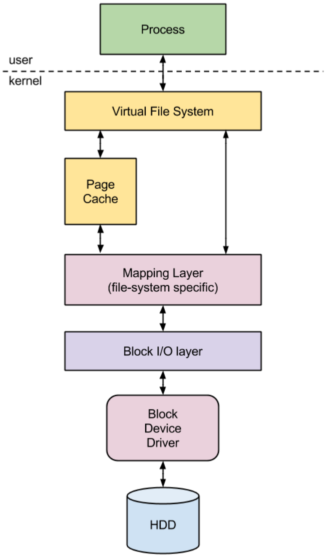
Give orders to controller

Kernel Abstraction

Block layer

Abstraction

to interact with block devices generically



Filesystems concepts

Filesystems

Filesystem

Method and data structure for secondary storage

Lots of different filesystems.

- Each has different features, purposes, limits and caveats, target OS.

Organization in the form of files & directories,
features like Access-Control-Lists (ACLs) (for permissions) etc. . .

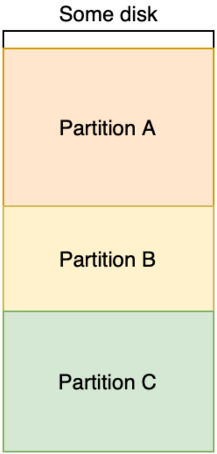
Partitions

Disk partitions allow multiple filesystems

Depending on machine's firmware (`BIOS` / `EFI`)

Contain:

- location (start LBA)
- size (in disk blocks)
- (filesystem) type
- etc



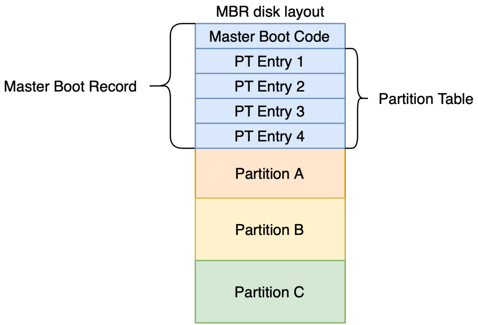
Partitions layout - Master Boot Record (MBR)

Master Boot Record (MBR)

Legacy boot sector format

- Uses first sector of the disk (512 bytes)
- Loaded by the `BIOS`
- Boot routine (bootloader)
- 4 primary partitions table entries
- Partitions maximum size of 2 TiB ($2^{32} \times 512$ bytes)

Extended partitions:
pointer to another partition table

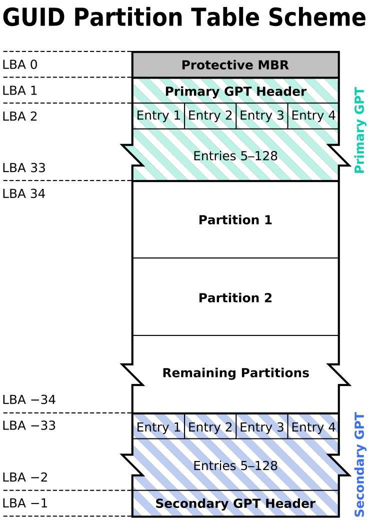


Partitions layout - GUID Partition Table (GPT)

GUID Partition Table (GPT)

Modern boot sector format

- Used with `EFI` firmwares
- Up to 128 entries
- Duplicated tables for more robustness
- Can boot full binaries in `EFI System Partitions (ESP)`
- Partition maximum size of 8 ZiB



Filesystem - Superblock

Superblock

Metadata of the filesystem

- Contains, e.g.,
  - Size
  - Block size
  - Number of free blocks
  - State
  - Root directory entry

Note: Corruption of the superblock makes whole filesystem unreadable

$\Rightarrow$ usually **duplicated** a few times on the disk

Filesystem - Inodes

Linux: Inode

also called **file-control block**

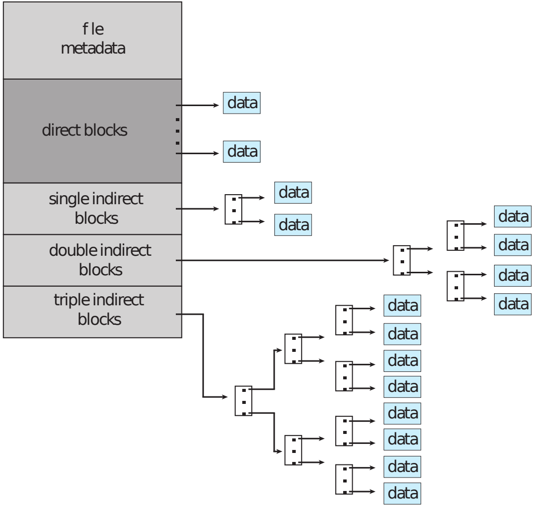
Metadata of a file:

- Access, write, create time
- ACLs, owner, group
- File size
- More depending on the filesystem

Space for inodes is

pre-allocated on filesystem creation

Note: Inode does **not** contain file names!



Filesystem - Directories

Directories

Relate file names to inode

Main solutions:

	Linear list	**Hash table**
	Simple and small	Fast
	Linear search is slow	Not easily adjust to size increase

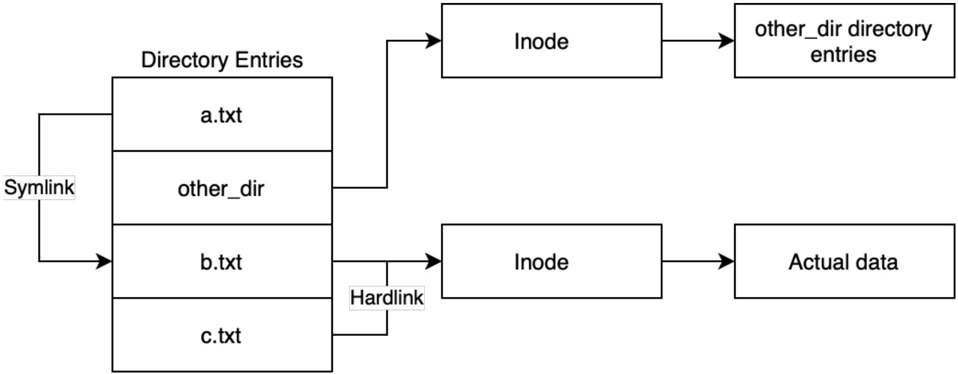
Hard link

Multiple directory entries pointing to one inode

Unix typically stores also type in directory entry

Directories are files $\Rightarrow$ they have an inode, directory entries are in file data

Filesystem - Summary



Demo: Inodes

Use `ln` or `ln -s` to create hard or symbolic links.

Compare their inodes with `ls -il` and `stat`.

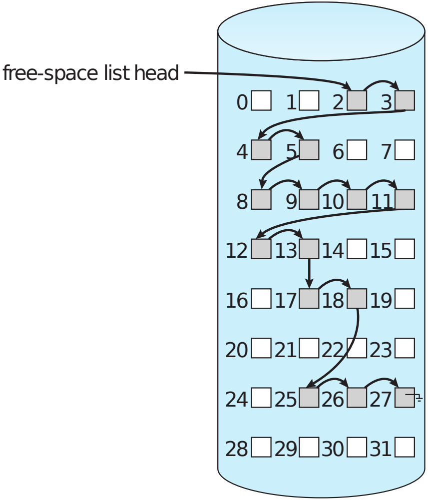
Also compare inode of a directory with the inode of `.` in this directory.

Free-Space List

Free blocks can be kept in **bit vector** but slow for large disks

Linked list give first free block fast

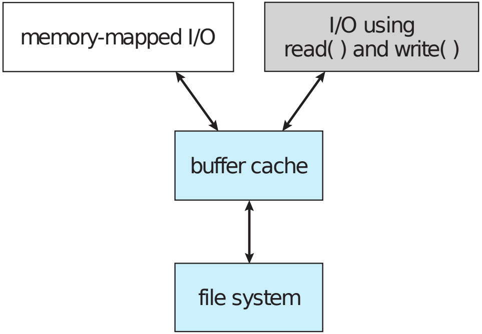
For larger allocation:
grouping (link to *n* free blocks)
counting (add counter to links if multiple contiguous blocks are free)



Caching

Keep important parts in memory:

- currently used directories
- free-space list
- used blocks
- for sequential access:
  subsequent blocks



Journaling

After a crash, filesystem must be checked for inconsistency

To simplify and speed up recovery: use log

- all metadata changes are written to log
- replay log in real fs structures
- mark changes in log as done
- After system crash: complete unfinished transactions

Advanced Features

Some filesystems like `ZFS` or `btrfs` go into volume management territory with:

- Logical volumes
- Checksuming
- Copy-on-write & snapshoting
- Software RAID

Unix way of life

Virtual Filesystem (VFS)

Everything is a file

Many things abstracted as file-like objects

Handled as files:

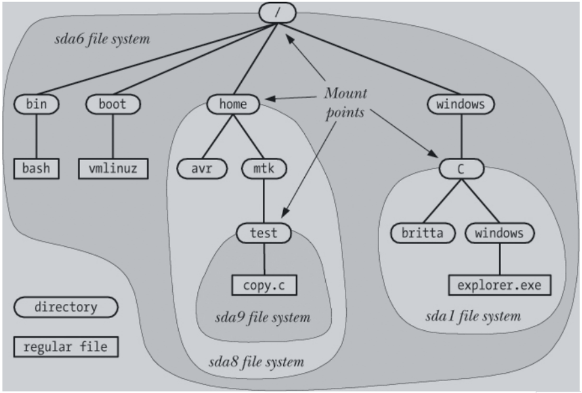
- Actual files
- Unix Sockets, Named pipes
- Devices (`char devices` & `block devices`)
- Partitions
- Kernel interfaces (`sysfs` and `procfs`)

Virtual Filesystem (VFS)

Direct interface to user applications:

Filesystem implementation completely abstracted

Filesystems are mounted on the `VFS` (e.g your **rootfs** is mount on /)



Proc Filesystem: Demo

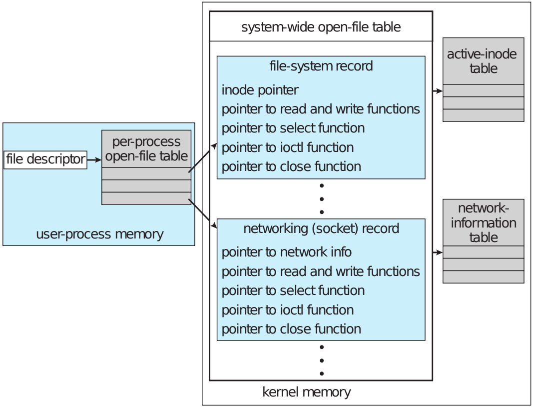
```
cat /proc/self/status
```

For more info:

https://www.kernel.org/doc/html/latest/filesystems/proc.html

File descriptors

- Kernel keeps a table of all opened file descriptors
- Each process has its own table: the File Descriptor Table
- Entries of those tables represent the file, and its operations
  Syscalls `read(2)`, `seek(2)`, ... are linked to those operations
- `open(2)`, `socket(2)`, `pipe(2)`, ... create new entries
- The file descriptor integers in user space are the **index in the table**



Device mapper

- Devices in the `VFS` are logical mappings to physical devices
- This is managed through the device mapper subsystem
- This gives a lot of flexibility on how you manage block devices
- Foundation for other subsystems:
  - `LVM`: for logical volumes management
  - `dm-crypt`: for disk encryption
  - Software RAID

Special files - `ioctl(2)`

- Some **files** in the `VFS` are special files - e.g **char devices**
- They can have special operations
  - For example, changing the baudrate of a serial tty
- Those operations are done through the `ioctl(2)` syscall
- A bit of a **do anything** kind of syscall
- The available commands depend on the file and the underlying driver

Analyse de l’enseignement

Please take the time to fill the following form:

`https://forms.forge.epita.fr/`

Thank you for your feedback!