Computers
File System

Grau en Ciència i Enginyeria de Dades

Xavier Martorell, Xavier Verdú

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC)

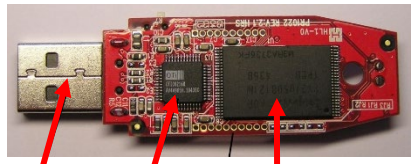
2020-2021 Q2

Creative Commons License

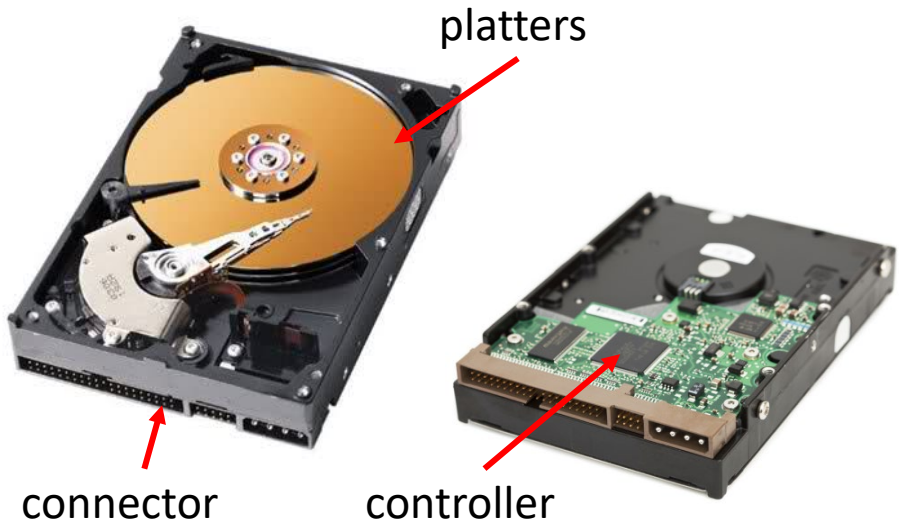
Table of Contents

Physical Storage Devices

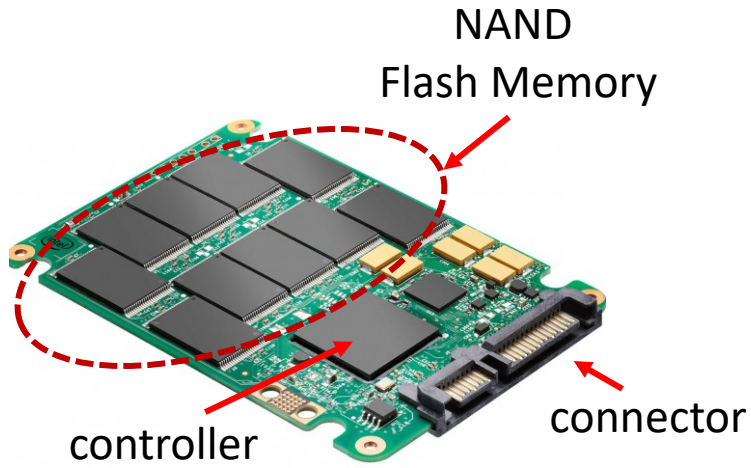
- Non-volatile memory to save data
- Similar vs Different components
  - Impact on performance and capacity



USB Drive



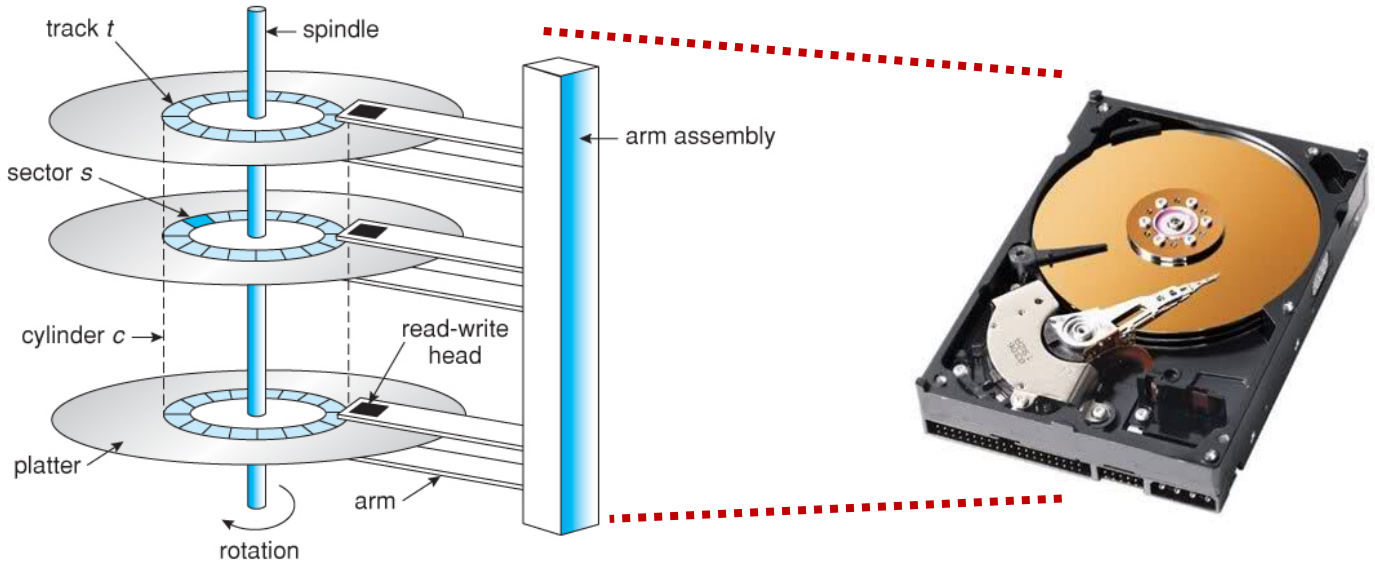
Hard Disk



Solid State Drive (SSD)

How are data physically saved?

- Any storage device needs to organize the pool of memory
  - E.g.: DVD, hard-disk, pen-drive, etc.
- Sector: The smallest unit of data that can be read/written
  - **Defined by the hardware**
  - Fixed size (tipically 512 Bytes)



Some parameters that impact on performance

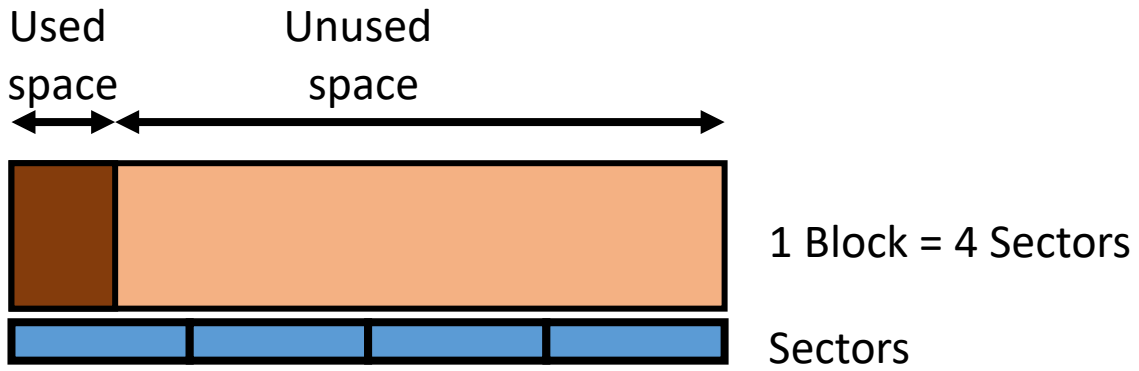
Speed: rpm

Connector Bandwidth: Gbps

How is a storage device organized?

- Block: A group of sectors (the smallest unit to allocate space)
  - **Defined by the OS (when formatting the device)**

- But, what is the best block size????
  - If it is likely to use large files…
    - Large blocks
  - If it is likely to use short files…
    - Short blocks



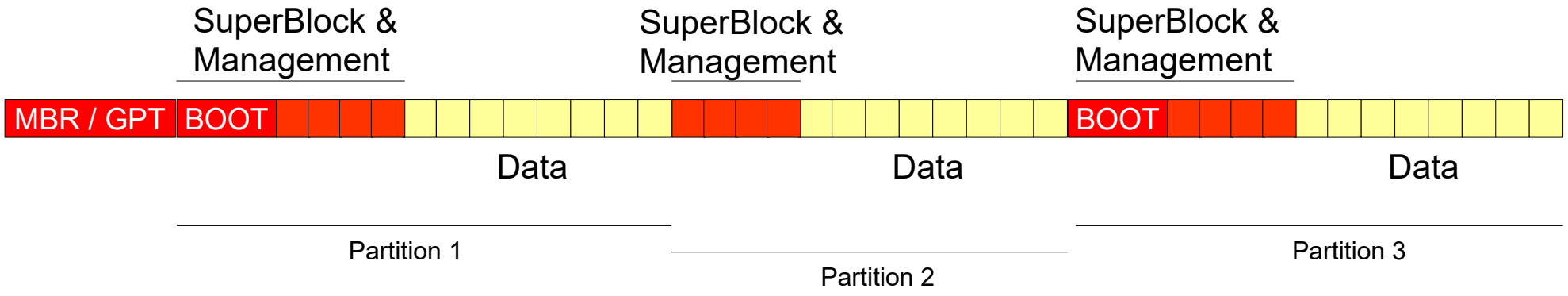
- What is the impact of a bad block size selection???
  - Too large: fragmentation (waste of space)
  - Too short: degrade performanace too many accesses to the device

Partition/Logical Drives

- Partition: a group of consecutive sectors with an unic identifier
  - It can be managed as an independent **logical drive**
    - Windows: C: D: Linux: /dev/sda1 /dev/sda2
  - A single physical storage device can be split into several partitions
- Management sectors
  - MBR: Master Boot Record (up to 2TB physical devices)
  - GPT: GUID Partition Table (replace MBR)

  } To find partitions and the OS to be booted

  - Superblock & management: Critical information to manage the partition
  - Boot Sector: Few bytes to boot the installed OS

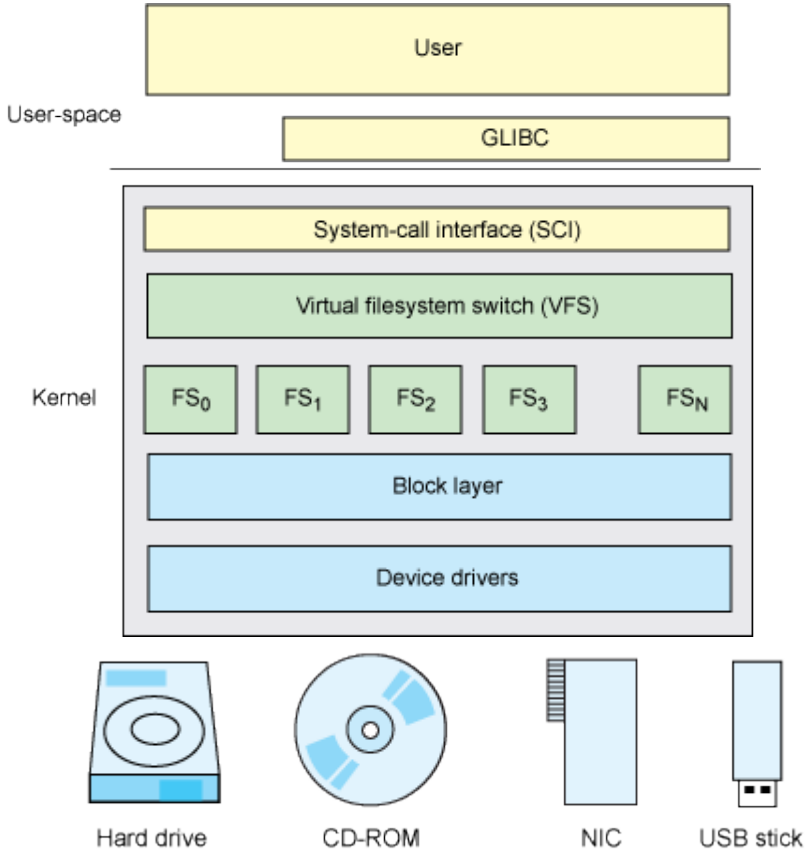


Virtual File System (VFS)

- An abstraction layer to manage different types of file system
  - It provides a single system call interface for any type of file system
  - VFS for UNIX/Linux. Other Oses use similar approaches

- Types of File System
  - FAT (File Allocation Table)
    - Removable drives
  - exFAT (Extended FAT)
    - Removable drives larger tan 4GB
  - NTFS (New Technology Transfer)
    - Windows
  - **I-node based file systems (UNIX/Linux)**
    - Ext3, ext4, Reiser4, XFS, F2FS
  - Cloud File System
    - GlusterFS, Ceph, HadoopFS, ElasticFileSystem (Amazon)

https://en.wikipedia.org/wiki/Comparison_of_file_systems



File Systems

- Swap space: in UNIX/Linux OS is a special file system that extends main memory
  - Windows implements swap space in a single resizable file

- FUSE: Filesystem in Userspace
  - Let’s non-priviledge users implement their own file system without modifying the kernel (it is executed in user space rather tan kernel space)
    - E.g.: GDFS (Google Drive), WikipediaFS, propietary File Systems

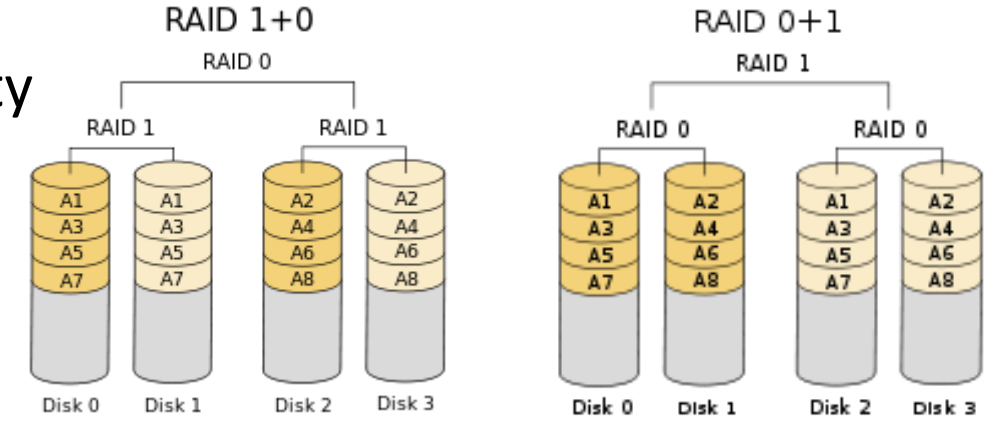
- Different File Systems offer different features that impact on performance, reliability, resilience, security, etc

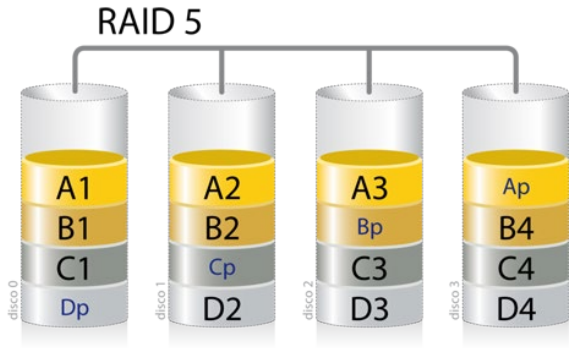
Journaling

- Transaction based File System
  - Keep track of changes not yet committed to the file system
  - It records the changes in a “journal” file
    - It has a dedicated area in the file system
- In case of system failure or outage…
  - The file system can be brought back fast and with lower likelihood of errors
    - E.g.: After a crash, replay the last updates from the “journal”
- Some File Systems that implement Journaling
  - Ext3, ext4, ReiserFS, XFS, JFS

RAID: Redundant Array of Independent Disks

- Storage virtualization technology that combines multiple physical storage drives into a single logical unit
  - Software driver vs hardware controller
  - Impact on performance and effective capacity
- Several approaches (can be combined)
  - RAID 0: Stripping → distributed data
  - RAID 1: Mirroring → replicated data
    - RAID 1+0 vs RAID 0+1
  - RAID 5: with distributed parity blocks





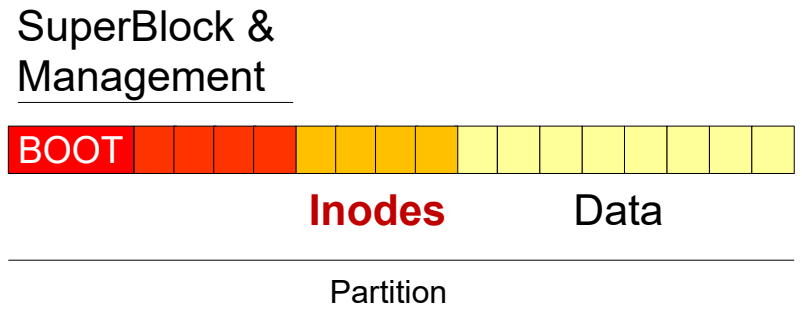
Performance Impact: handling blocks

- Every file system has its own mechanism to handle…
  - Occupied/free blocks
  - The blocks of a given file (i.e. how to access the contents of a given file)

- Depending on the file system, the implementation changes
  - For example:
    - FAT: has a global table with as many entries as blocks has the drive
      - Linked-list based file access
    - I-node based: has a structure called I-node to hold all the information to manage a file
      - Index based file access (a.k.a. multi-level index). The index is hold by every I-node

I-Node Based File System

- Some fields of the I-Node:
  - I-Node ID
  - Size
  - Type of file (regular file, directory, named pipe, socket, etc.)
  - Protection (Read / Write / Execute (RWX) for Owner, Group, Others)
  - Ownership
  - Timestamps
  - Number of Links (# direct relations between a symbolic name and I-Node ID)
  - Pointers to blocks of data (multi-level index). It use to has 12-13 pointers.



Directories: Organizing files

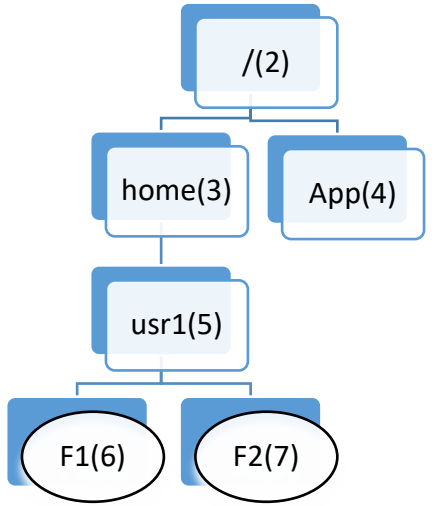
- **Directory:** Logical structure to organize files
  - Pathname: Relative (from any folder) vs Absolute (from the root folder)
- It is a particular type of file managed by the OS
  - "/" is the root folder
    - Every partition has its own root folder **(I-node ID 2)**
  - "." and ".." are mandatory entries in any directory
    - Even though the directory has no files
- **Hardlink:** A direct relation between name and I-node ID

Name	I-node
.	2
..	2
home	3
App	4

Directory: /

Name	I-node
.	5
..	3
F1	6
F2	7

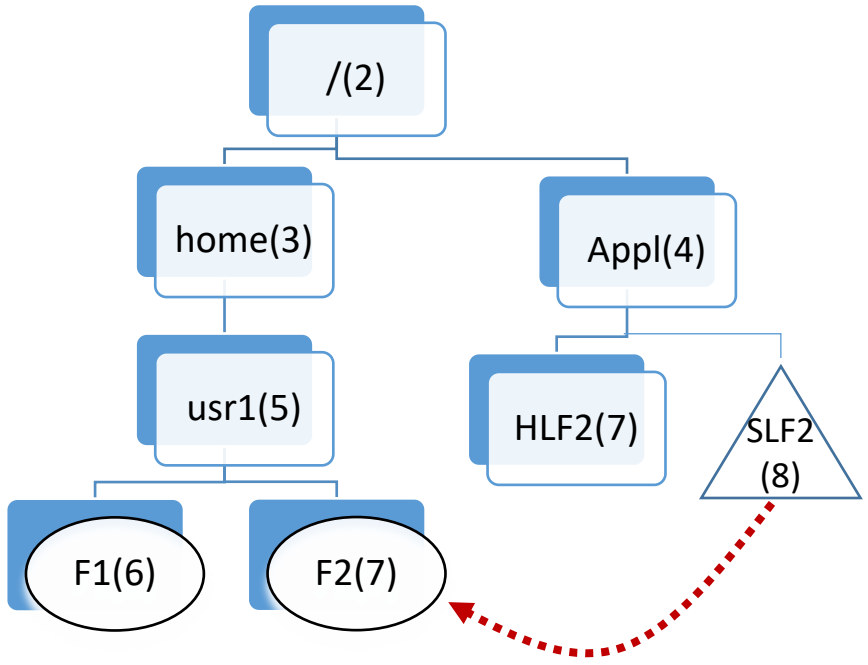
Directory: /home/usr1



Directories: Organizing files

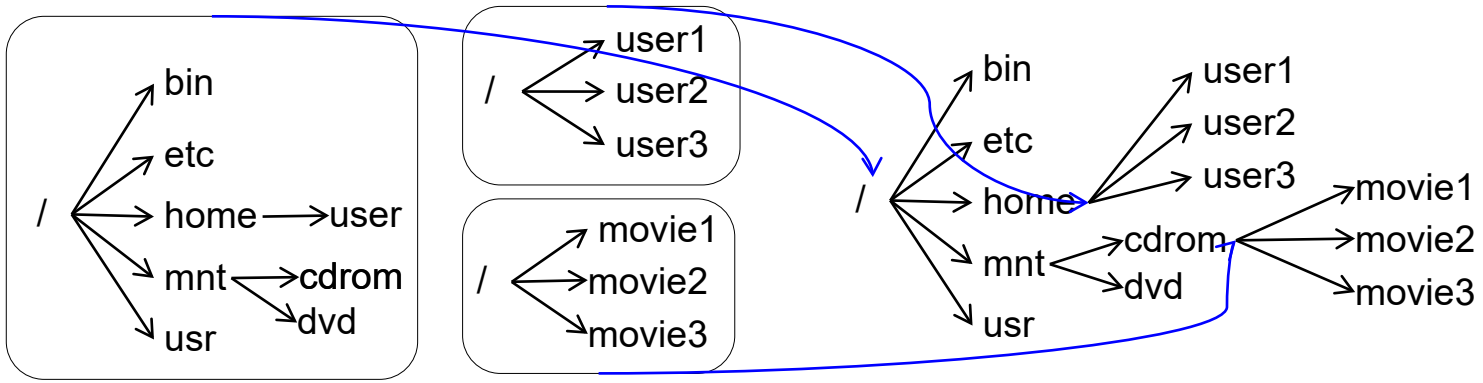
- Directories are organized as graphs
  - A given file can be accessed from different directories

- Sharing files
  - Hardlinks
    - It only needs a new entry in a directory (name→I-node ID)
  - **Softlinks**
    - A **new file** that comprises a pathname
      - Similar to shortcuts in Windows
  - Pros/Cons/restrictions lead to use one or the other



Mount

- Publishing the contents of a disk partition on the file system
  - E.g.: mount -t ext4 /dev/hda1 /home //mounting the home partition
    unmount /dev/hda1 //unmounting the home partition



Internal Structures of the OS

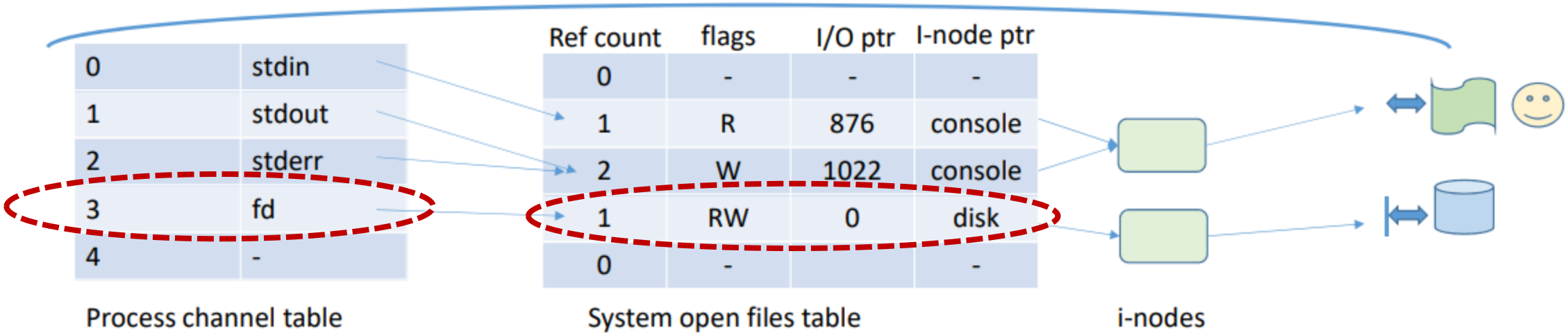
- To manage I/O and reduce accesses to disk
- Open File Table
  - It holds the current position to perform I/O access per open file
- I-node Table
  - It holds the I-node of every open file (like a cache of I-nodes)
- Buffer Cache
  - Memory zone to hold any I-node and data block transfer from/to the disk
  - If the requested I-node or block is in the cache, the access to the disk is not performed

System Calls

- Basic I/O System Calls
  - Fd = open(path, flags[, permissions]);
    - Flags can be combined: O_WRONLY|O_CREAT|O_TRUNC
    - Permissions: access control for the new file
  - close(fd);
  - Bytes = read(fd, @ref, bytes);
  - Bytes = write(fd, @ref, bytes);
  - Newfd = dup(fd);
  - Newfd = dup2(fd, newfd);
  - *New_offset = lseek(fd, offset, whence);*
    - *Whence: SEEK_SET, SEEK_CUR, SEEK_END*

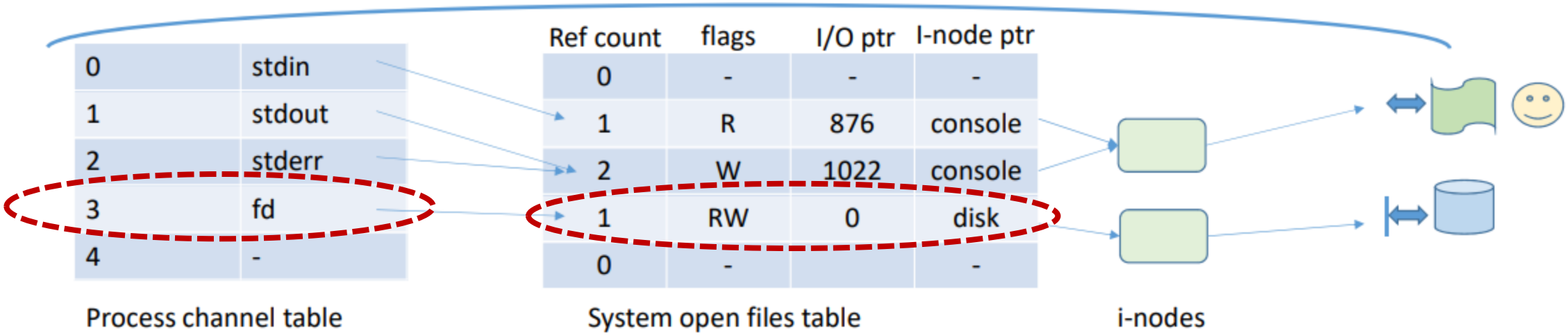
Syscalls

- Fd = open (PATH, mode [, permissions]);
  - New file descriptor → new entry in Open File Table → I-node in I-node Table
  - Accesses to the disk to decode the Pathname
    - The more subdirectories and/or softlinks, the more accesses



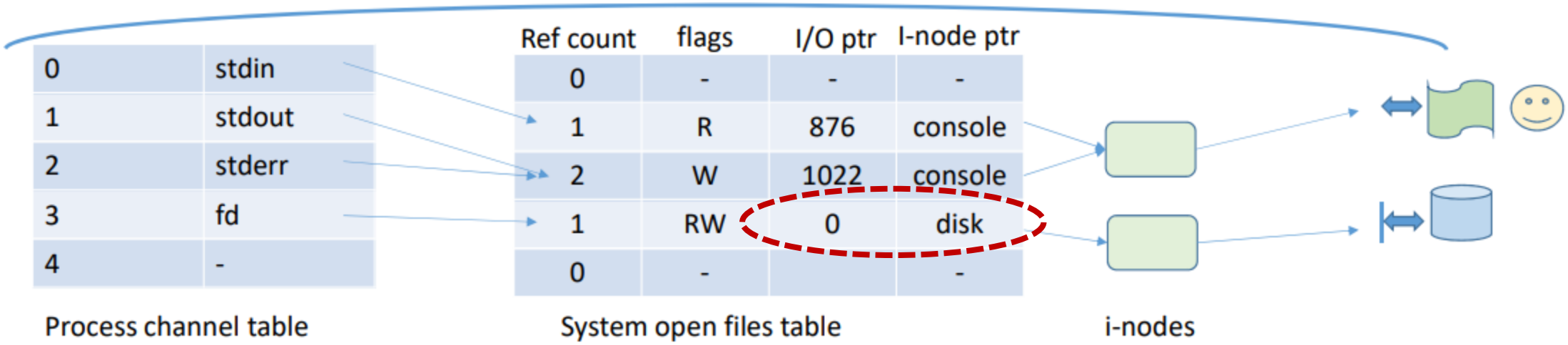
Syscalls

- Close(fd);
  - Release the file descriptor → maybe release entry in Open File Table → maybe release I-node in I-node Table
  - Update the I-node from I-node Table to the disk



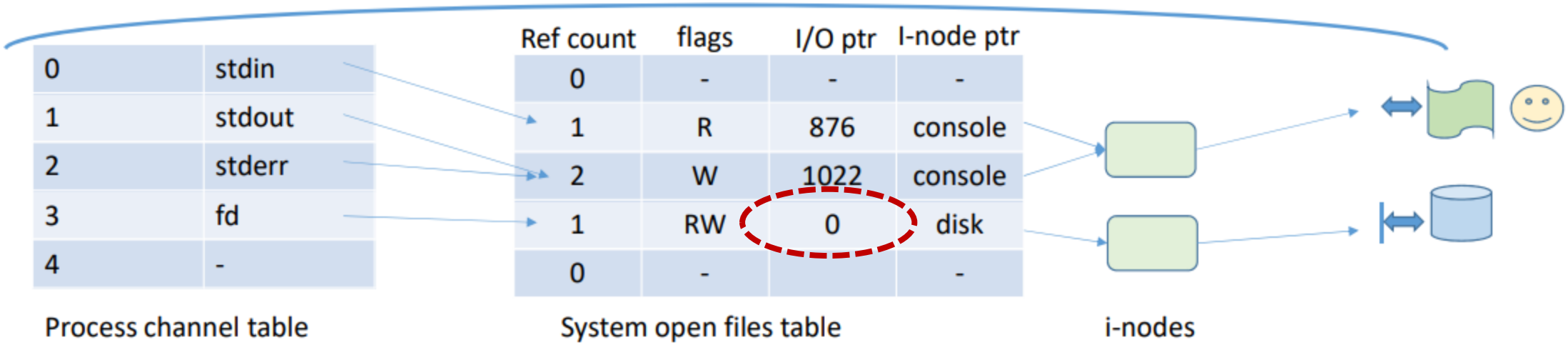
Syscalls

- #bytes = Read/write(fd, ptr, #bytes);
  - Accesses to disk to read/write the requested blocks
  - Automatically updates the I/O ptr (offset)



Syscalls

- New Offset = lseek(fd, offset, whence);
  - NO accesses to disk→only update I/O ptr (offset)
  - Whence: SEEK_SET, SEEK_CUR, SEEK_END



Bibliography

- Operating system concepts (John Wiley & Sons, INC. 2014)
  - Silberschatz, A.; Galvin, P.B.; Gagne, G
  - http://cataleg.upc.edu/record=b1431631~S1*cat

- Operating systems: internals and design principles (Prentice Hall, 2015)
  - Stallings, W
  - http://cataleg.upc.edu/record=b1441252~S1*cat