External Storage Structures

CE-2101 Algorithms and Data Structures

Disclaimer / Descargo de Responsabilidad

Esta presentación corresponde a una guía usada por el profesor durante las clases. La misma ha sido modificada para ser utilizado en el modelo de cursos asistidos por tecnología. No es una versión final, por lo que la misma podría requerir todavía hacer algunos ajustes. Para aspectos de evaluación esta presentación es solo una guía, por lo que el estudiante debe profundizar con el material de lectura asignado y lo discutido en clases para aspectos de evaluación.

This presentation corresponds to a guide material used by the professor during classes. It has been modified to be used in the model of technology-assisted courses. It is not a final version, so it may still require some adjustments. For evaluation aspects, this presentation is only a guide, so the student should delve with the assigned reading material and what has been discussed in class.

Introduction

- ➔ Computer storage devices are typically classified into primary or main memory and **secondary or peripheral storage**

- ➔ Primary storage usually refers to Random Access Memory

- ➔ Secondary storage refers to devices such as hard disk drives, SSD, removable USBs...

What are differences between primary and secondary storage?

Hard disk drive

➔ Often shortened as Hard disk, Hard drive or HDD

➔ Non-volatile

➔ Stores data in rapidly rotating rigid platters with magnetic surfaces

Hard disk drive

- HDD is a sealed unit containing a number of platters in a stack

- Electromagnetic read/write heads are positioned above and below each platter

Hard disk drive

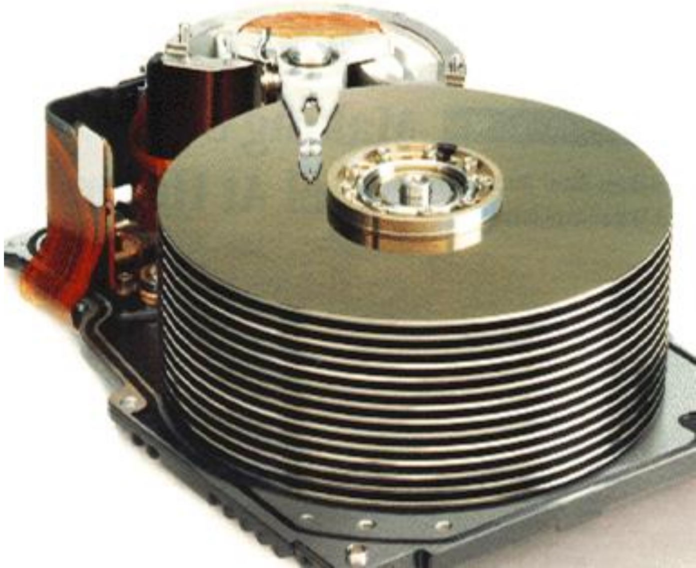
- ➔ Disk drives are often referred to as **direct access** storage devices
- ➔ Meaning it takes almost the same time to access any record in a file
- ➔ Contrast this to a tape...

Hard disk drive

Platters

- Platter is a circular, metal disk that is mounted inside a hard disk drive.

- Several platters are mounted on a fixed spindle motor to create more storage surfaces.

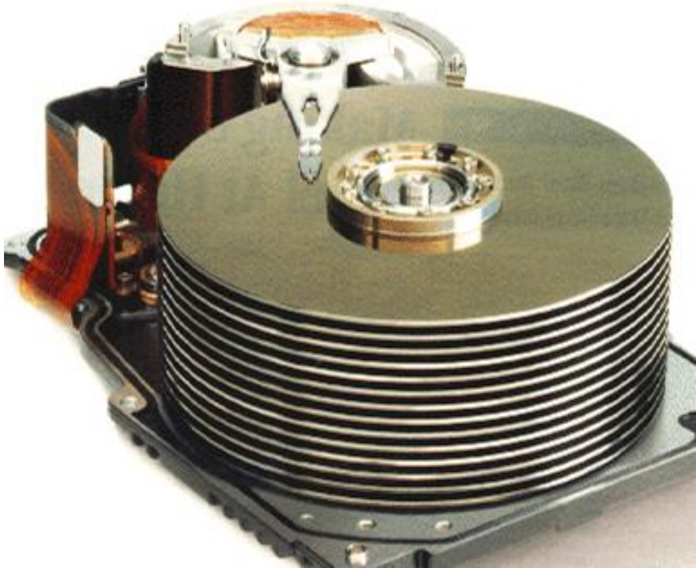


Hard disk drive

Platters

- ➔ The platters is made up of aluminium or glass substrate

- ➔ Covered with a layer of Ferric Oxide or cobalt alloy



Hard disk drive

Tracks

- ➔ Each platter is divide into thousands concentric circles
- ➔ These tracks resemble the structure of annual rings of a tree.
- ➔ All the information is stored in tracks

Hard disk drive

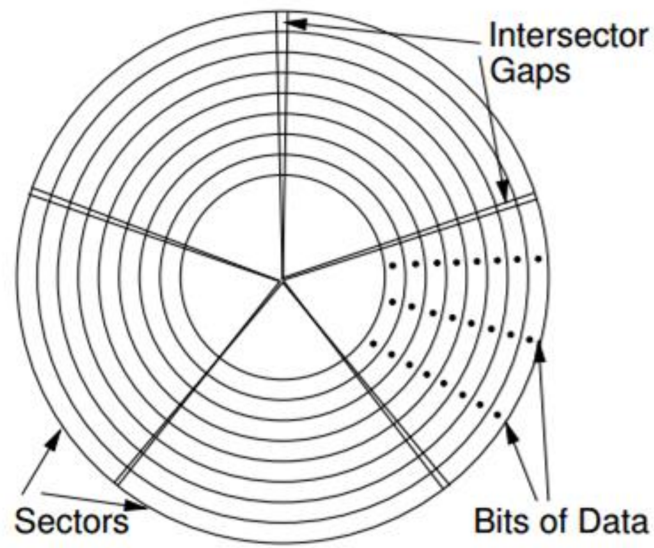
Tracks

- ➔ Outer track is the track 0
- ➔ Each track can store large amount of data counting to thousands of bytes.
- ➔ There are tracks in both sides of the platter

Hard disk drive

Sectors

- Each track is divide in small units called sectors.
- It is the basic unit of data storage on a HDD
- A single track can have thousands of sectors and each sector can hold more than 512 bytes.



Hard disk drive

Sectors

- ➔ A few additional bytes are needed to keep control structures and error detection and correction.

- ➔ Sectors are grouped in **clusters**

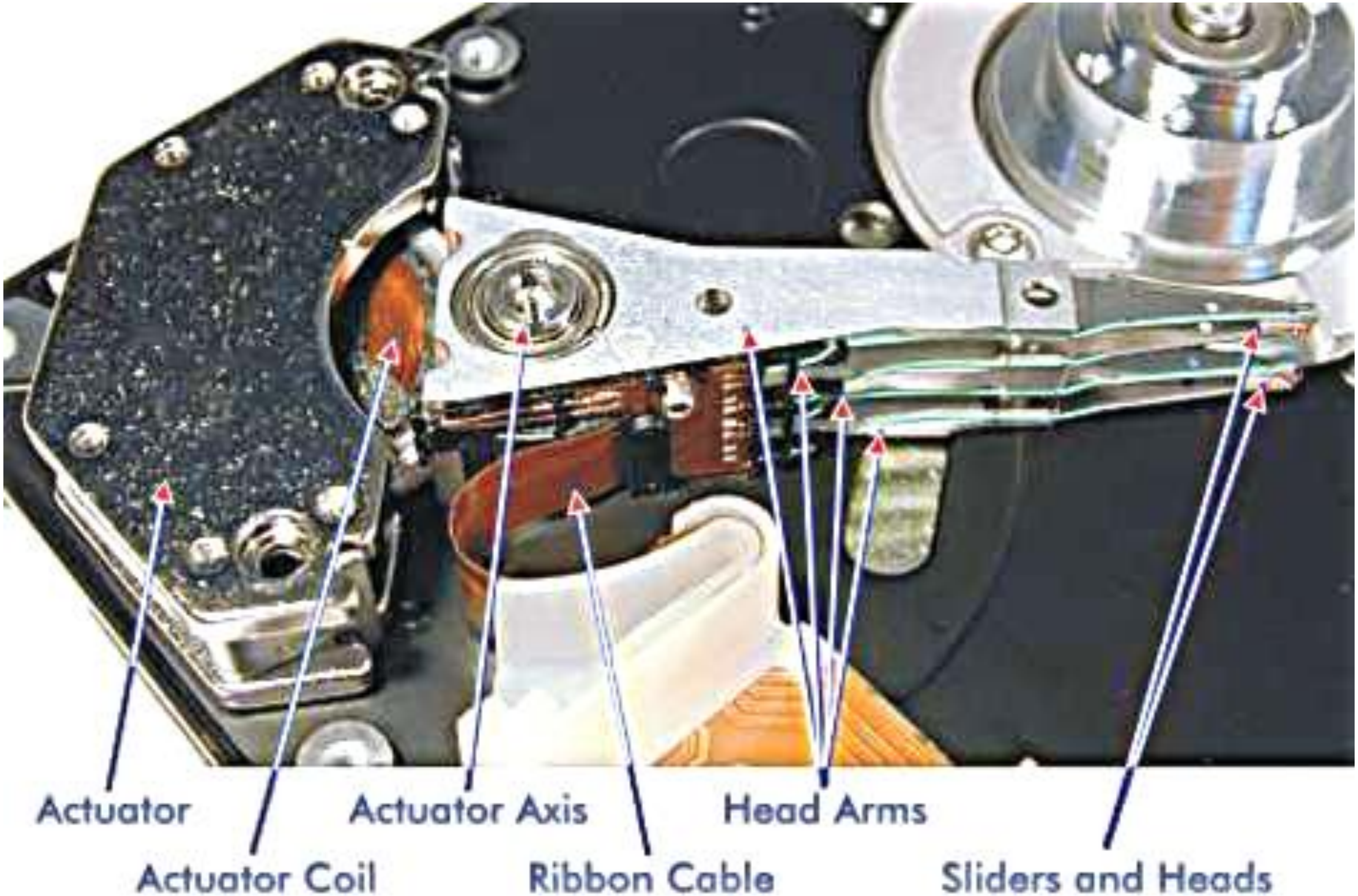
- ➔ Intersector gaps allows to detect the start and end of a sector

Hard disk drive

Read / Write Heads

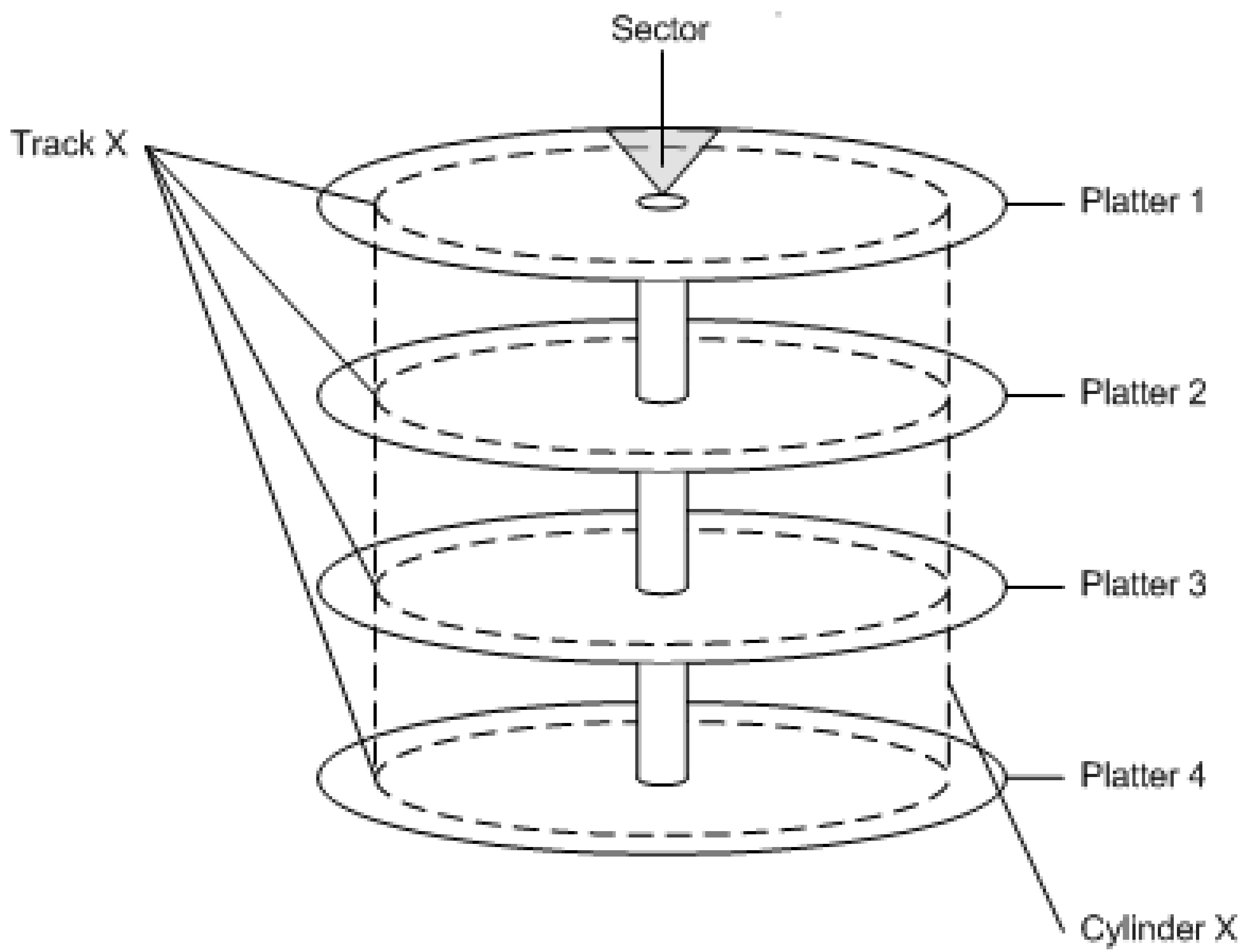
- ➔ Interface between the magnetic field where the data is stored and electronic components in the hard disk.
- ➔ Converts the bits to magnetic pulses when it is to be stored on the platter.
- ➔ Performs the reverse process to read information from the disk

Hard disk drive

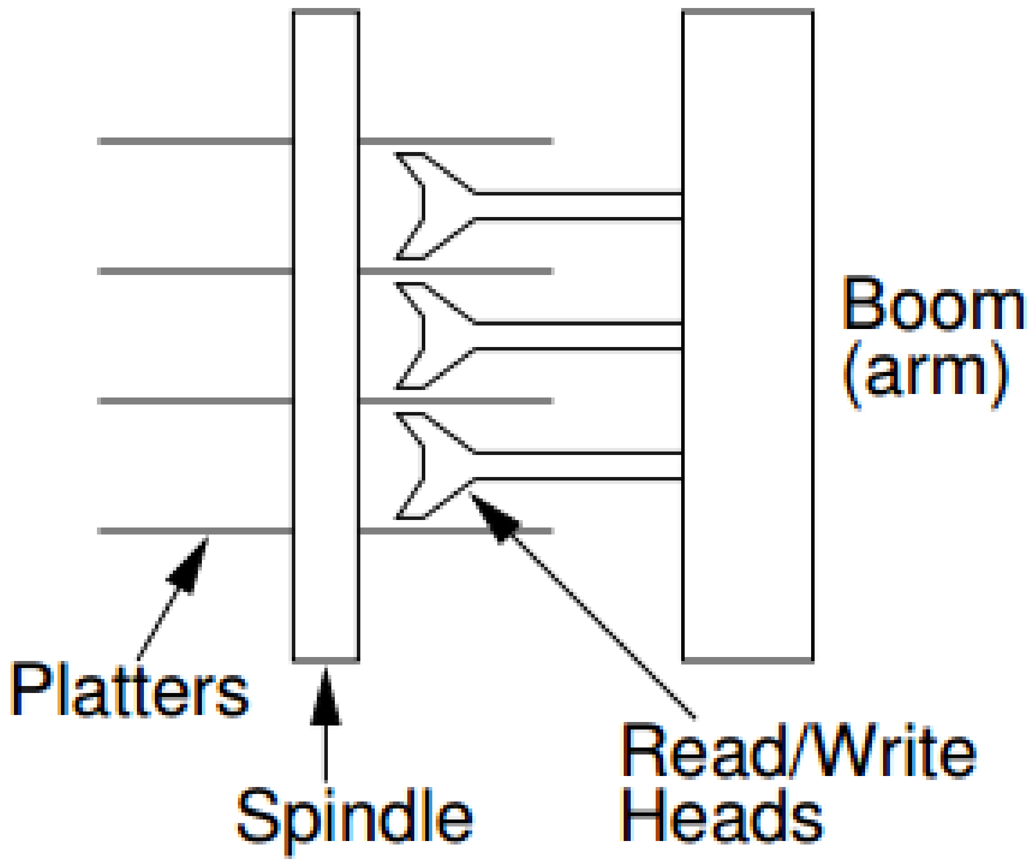


Hard disk drive

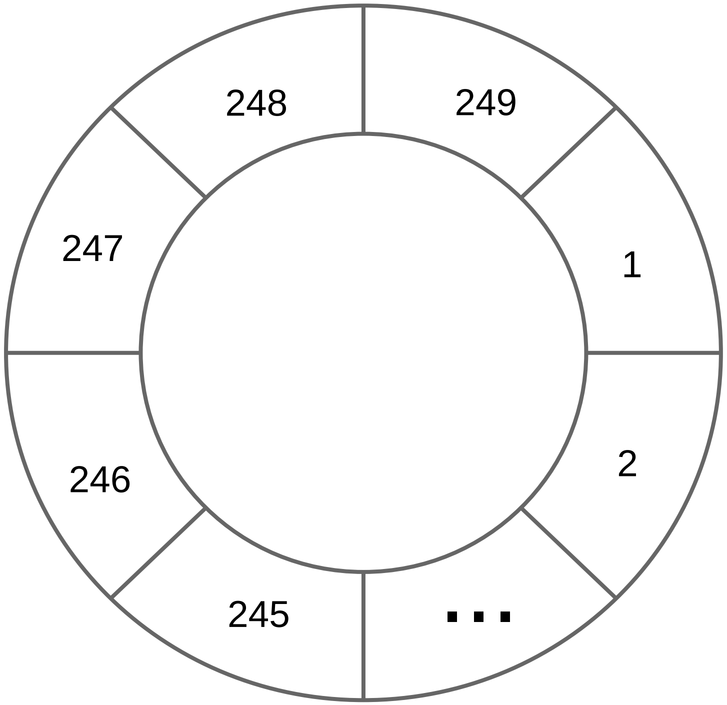
Hard disk drive



Hard disk drive



Cylinder Method



Track 40, Surface 2

Sector Address = track number
+ cylinder
number
+ surface number
+ register
number

Hard disk drive

Cylinder Method

- If the platter has 200 tracks in each side, the disk has 200 cylinder

- Each platter has two sides

- If the disk has 11 platters, the surfaces are enumerated from 0 to 19

Hard disk drive

Sector Method

- ➔ Each track is divided in sector, each sector has a fixed storage space.
- ➔ The address of the register is the address of the sector.

Hard disk drive

Disk Access Time

Moving Head

Access Time = Positioning Time
+ Head Activation Time
+ Rotational Delay
+ Transfer Time

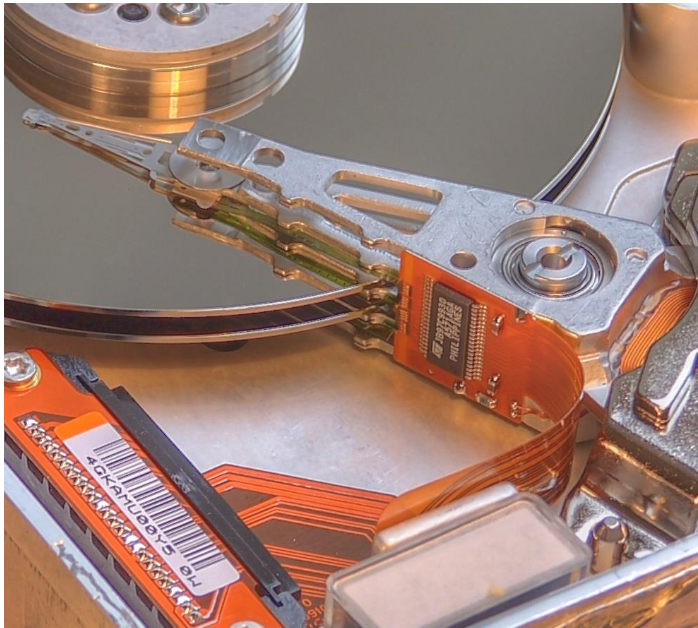
Disk Access Time

Static Head

➔ There is a head per track

➔ Access Time = Head Activation Time

+ Rotational Delay (sector selection)
+ Transfer Time



Hard disk drive

Types

- ➔ **IDE** stands for Integrated Drive Electronics and it was a term used to differentiate a new hard drive technology

- ➔ The IDE specification included the ATA interface. **ATA** stands for Advanced Technology Attachment

Types

- ➔ When the **Serial-ATA** (SATA) was introduced, the Old ATA term was changed to **PATA** (parallel-ATA)

- ➔ The IDE term is obsolete

Hard disk drive

Types

- ➔ SCSI (Small Computer System Interface) is a set of standard to connect a computer with a device

- ➔ Current versions of SCSI is SAS (Serial attached SCSI) which is a serial version of SCSI and is still being used

Disk Arrays

Data Striping

➔ "... is a technique of divide logically sequential data (such a file) ... "

Mirroring

➔ "... is the replication of logical disk volumes onto separate physical HDD in real time to ensure continuous availability ..."

Disk Arrays

Parity Bit / Check Bit

➔ Is a bit added to the end of a string of a binary code that indicates if the number of bits in the string is even (1) or odd (0)

➔ Simplest form of **error detecting code**

Parity Bit / Check Bit

- → **Even parity bit**: 1 if the number of ones of a given string of bits is odd
- → **Odd parity bit**: 1 if the number of ones is even.

Parity Bit / Check Bit

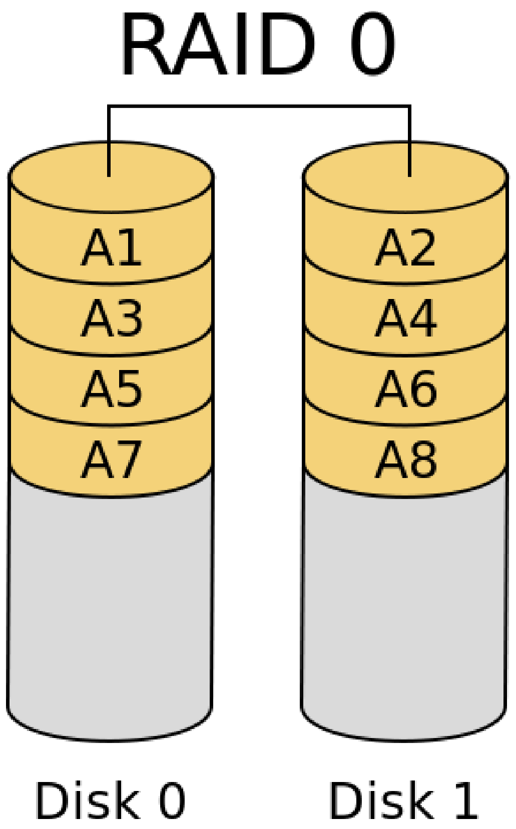
Parity Bit/Check Bit

Data	Even	Odd
11001100	11001100**0**	11001100**1**
11111011	11111011**1**	11111011**0**
11001111	11001111**0**	11001111**1**
00110011	00110011**0**	00110011**1**
01111111	01111111**1**	01111111**0**
01010101	01010101**0**	01010101**1**

Disk Arrays

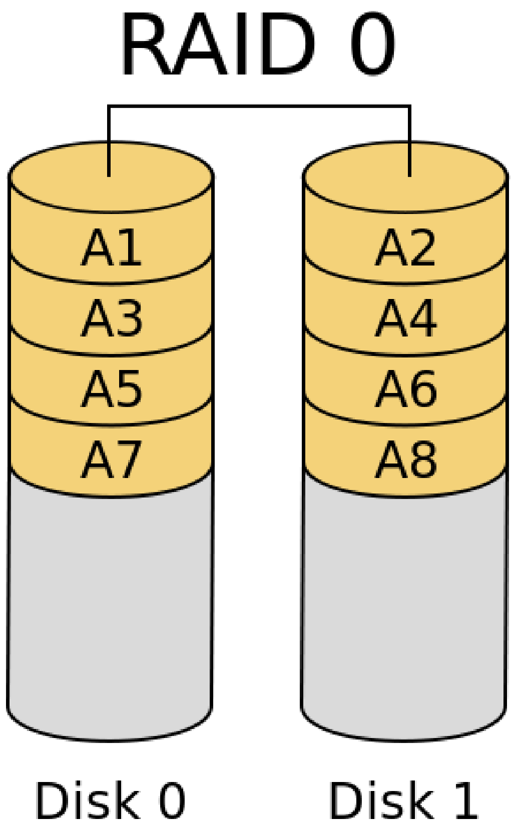
RAID 0

- ➔ Redundant Array of Inexpensive Disks
- ➔ Block level striping
- ➔ No parity
- ➔ No mirroring
- ➔ No redundancy
- ➔ Performance Improvement
- ➔ Additional Storage



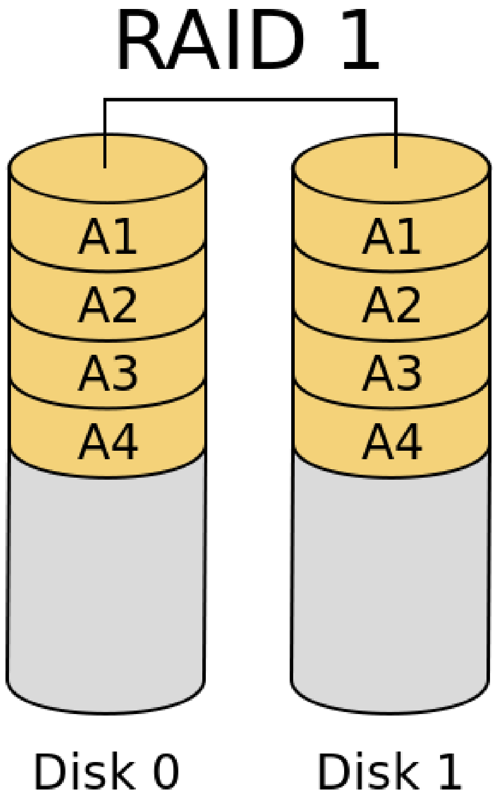
RAID 0

- ➔ No fault tolerance
- ➔ If one disk fail, destroy the entire array
- ➔ Parallel Access
- ➔ Minimum 2 disks



RAID 1

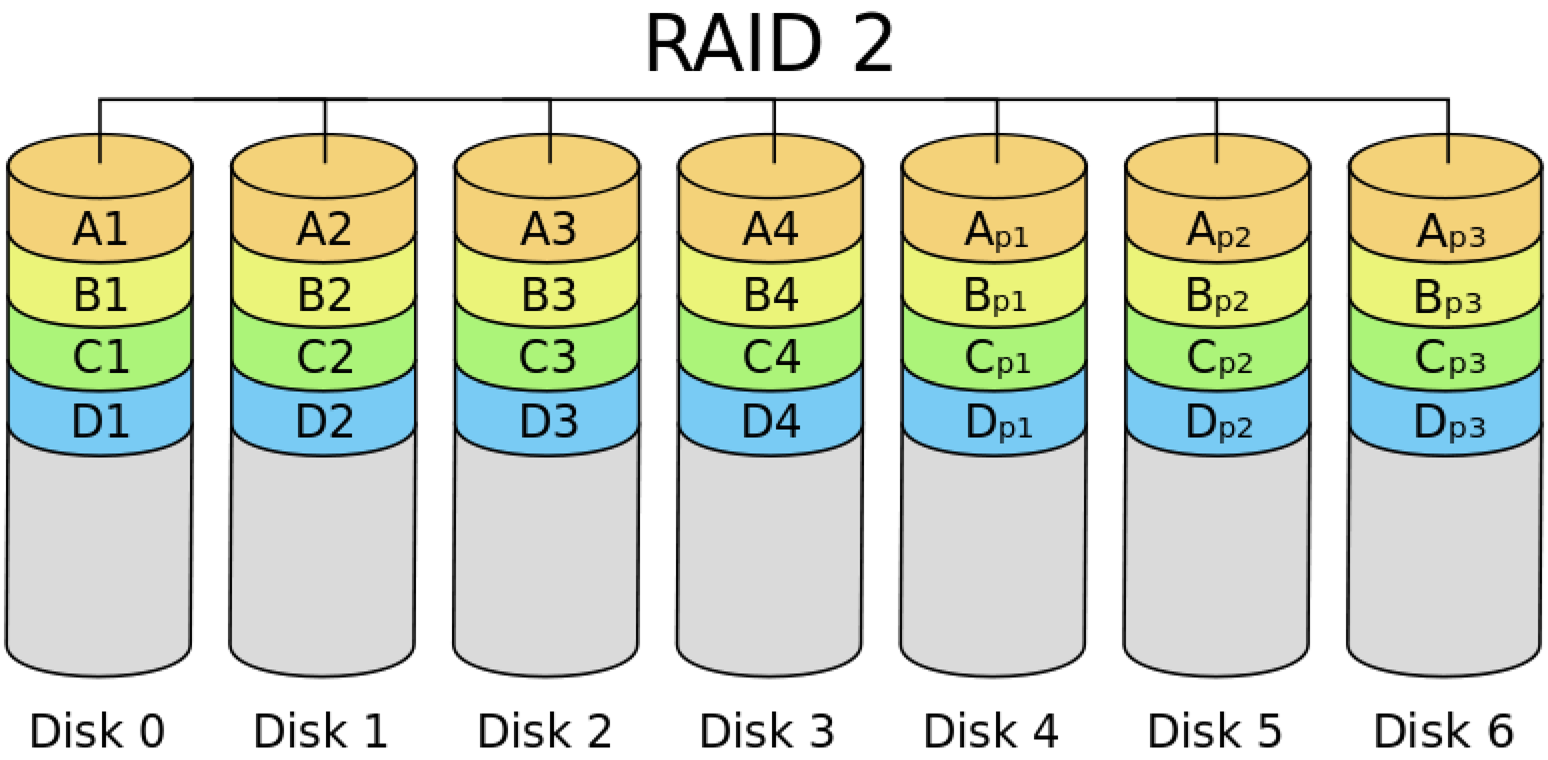
- ➔ Mirroring
- ➔ No parity
- ➔ No Stripping
- ➔ Data is written identically to two drives
- ➔ At least two disks
- ➔ Redundancy



RAID 2

- ➔ Not commonly used in practice
- ➔ Implementations exists
- ➔ Bit level striping
- ➔ All disks spindle rotation is synchronized
- ➔ The parity bit is calculated across corresponding set of bits and stored on at least one parity drive.

Disk Arrays
RAID 2



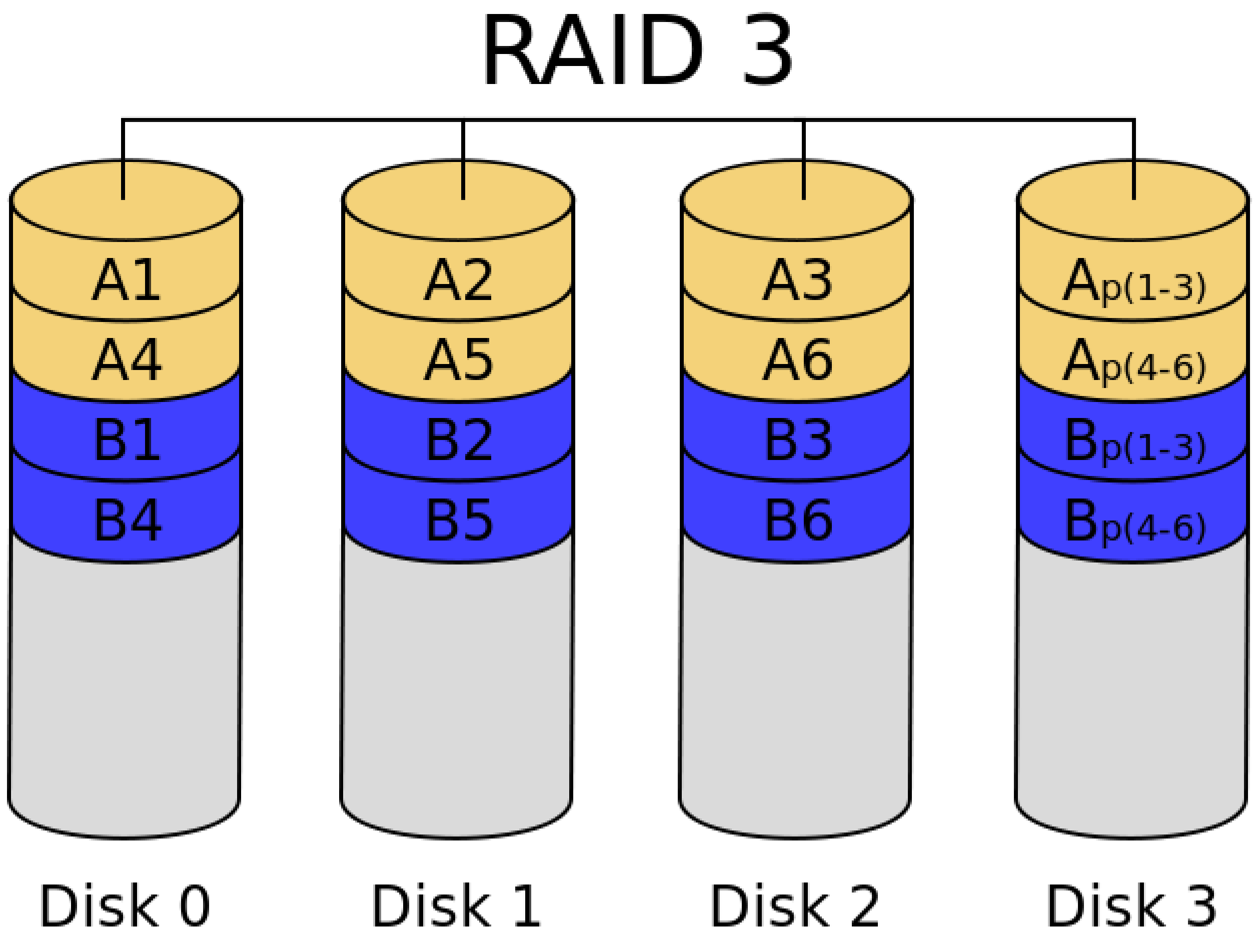
Disk Arrays

RAID 3

- ➔ Byte level striping
- ➔ Dedicated parity
- ➔ All disk spindle rotation is synchronized
- ➔ Parity is calculated across corresponding bytes and stored on a dedicated parity drive.
- ➔ Not commonly used in practice
- ➔ Implementations exists

Disk Arrays

RAID 3



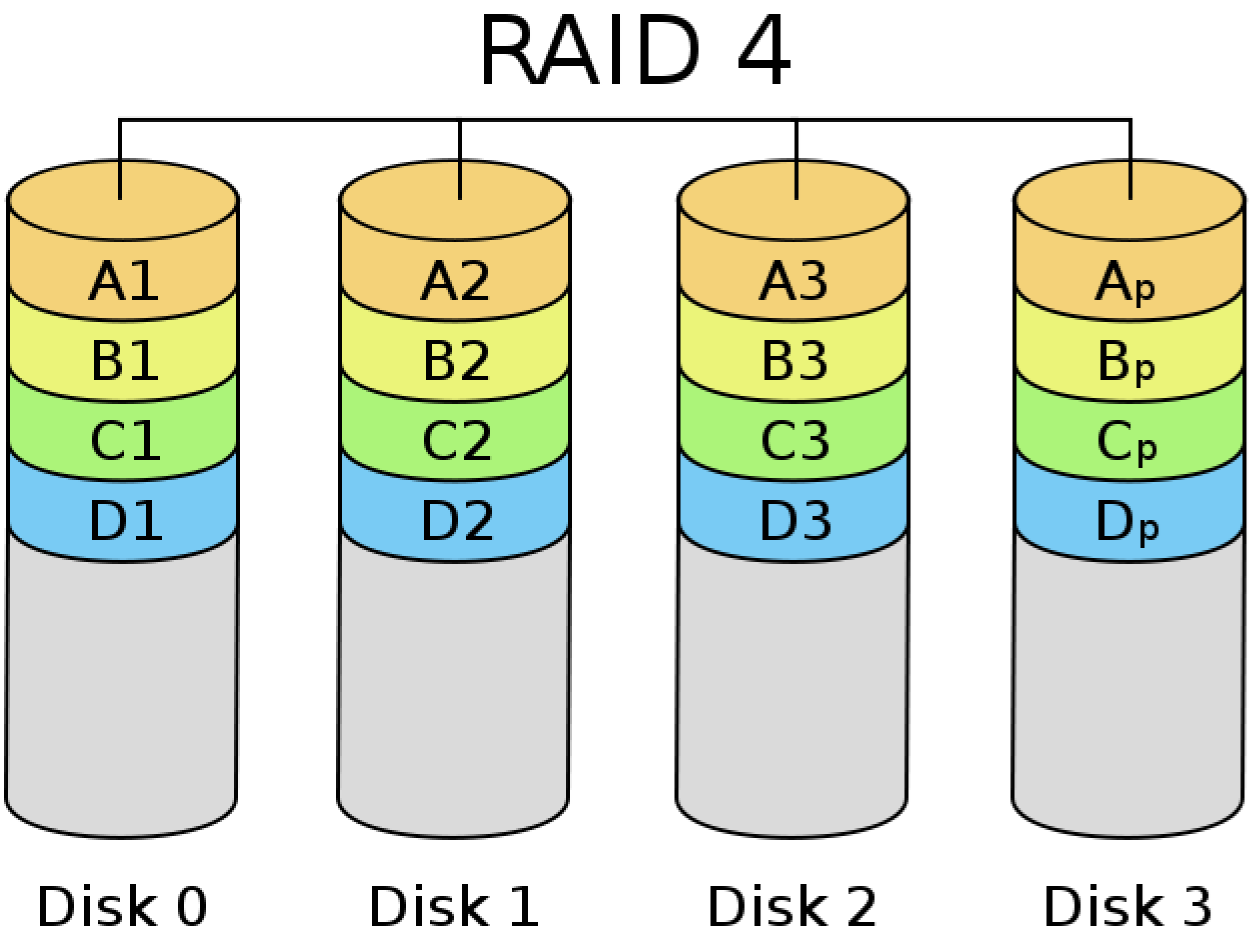
Disk Arrays

RAID 4

- ➔ Block level striping
- ➔ Dedicated parity
- ➔ Is equivalent to Raid 5, with the difference that all the parity data is stored on a single drive.
- ➔ Files are distributed between drives
- ➔ Each drive operates independently
- ➔ I/O request can be performed in parallel

Disk Arrays

RAID 4



RAID 5

- ➔ Block level striping
- ➔ Distributed parity
- ➔ Provides fault tolerance even if two disks fail
- ➔ Distribute parity and data across all the drives

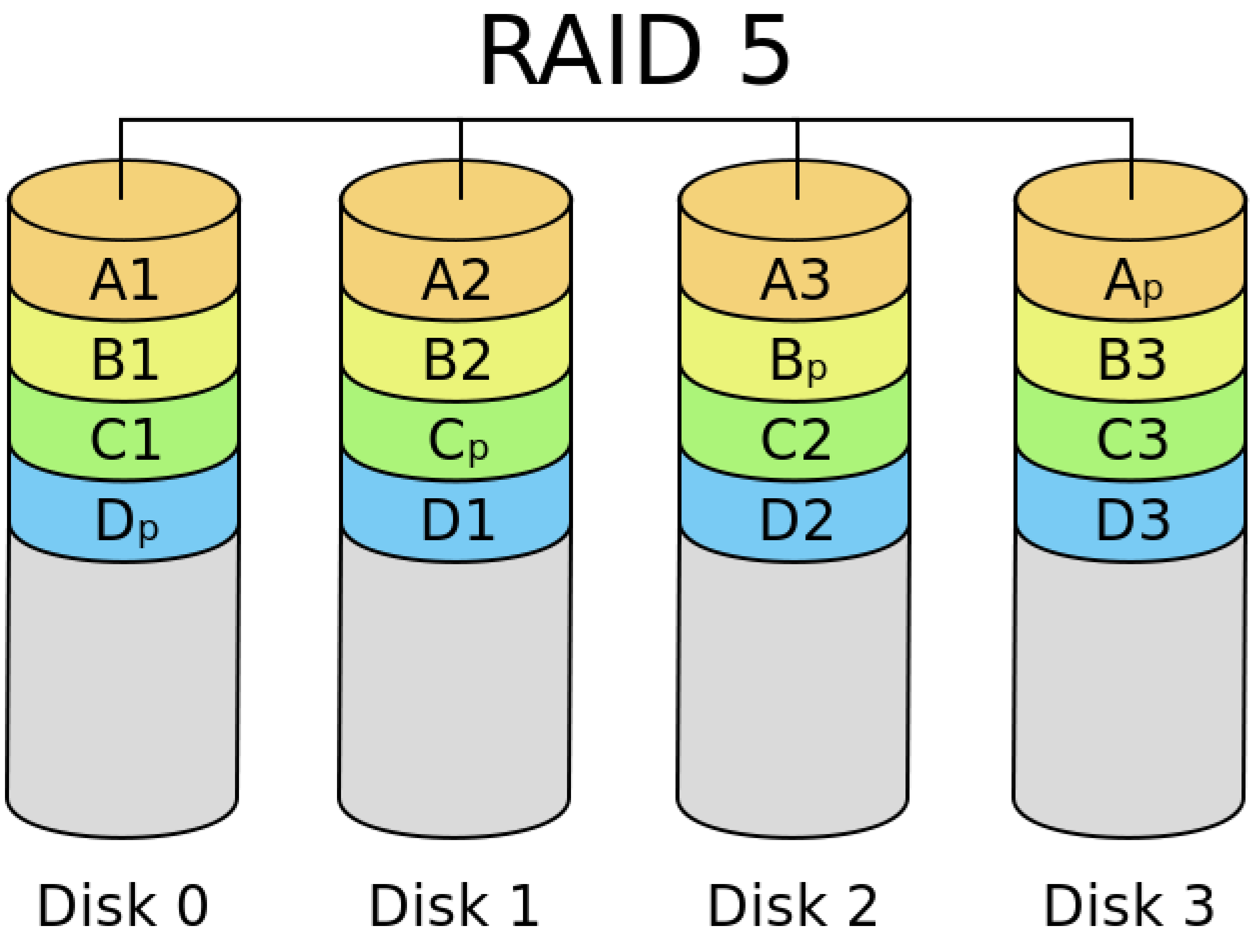
Disk Arrays

RAID 5

- ➔ Array is not destroyed by a single drive failure
- ➔ Requires at least two disks
- ➔ Good performance, good redundancy
- ➔ Fast reads slow writes

Disk Arrays

RAID 5



Disk Arrays

RAID 5

- Total space available is ((`number of drives -1) * size of smallest drive`)
- The more drives you use, the more efficient your storage space become, without losing any redundancy
- The array can survive without a drive, but it is slow, the data must be rebuild on the fly.
- Uses XOR to rebuild data and creates the parity strings

RAID 5

XOR

Input		Output
0	0	0
0	1	1
1	0	1
1	1	0

Disk Arrays

RAID 5

	Drive 1	Drive 2	Drive 3	Drive 4
Stripe 1	0100	0101	0010	0011
Stripe 2	0010	0000	0110	0100
Stripe 3	0011	0001	1010	1000
Stripe 4	0110	0001	1101	1010

Disk Arrays

RAID 5

	Drive 1	Drive 2	Drive 3	Drive 4
Stripe 1	0100	A	0010	0011
Stripe 2	0010	B	0110	0100
Stripe 3	0011	D	1010	1000
Stripe 4	0110	E	1101	1010

How can we rebuild the lost disk?

Disk Arrays

RAID 5

- ➔ Just apply XOR with the remaining blocks
  - ◆ For example to rebuild the streap 1 of the disk two:
    - `(stripe 1 disk 1) XOR (stripe 1 disk 3) XOR (stripe 1 disk 4) = (stripe 1 disk 2)`
    - `0100 XOR 0010 XOR 0011 = 0101`

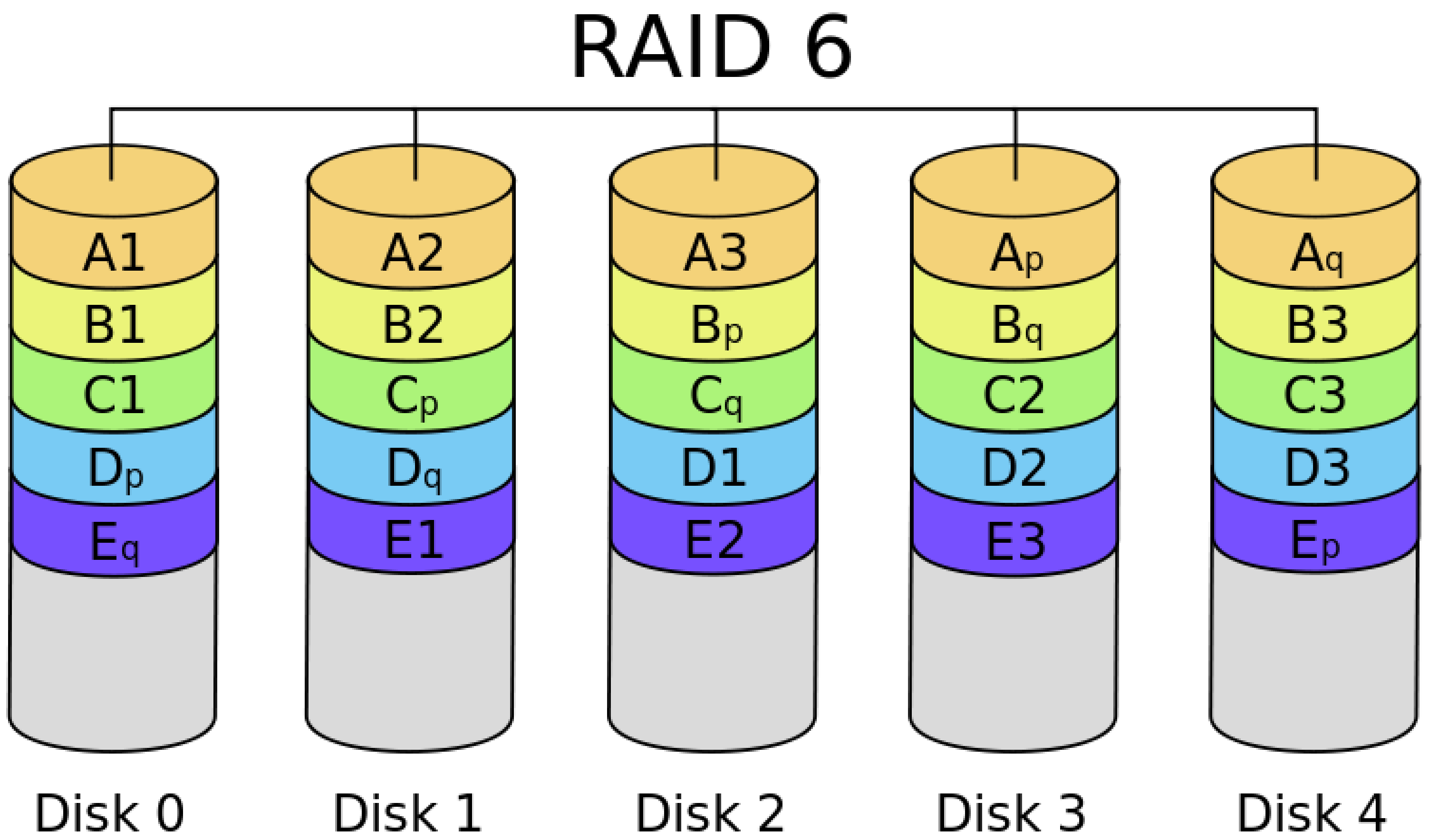
Disk Arrays

RAID 6

- ➔ Block level striping
- ➔ double distributing parity
- ➔ fault tolerance up to failed disks

Disk Arrays

RAID 6



RAID 10

- ➔ Often referred to RAID 1+0, stripe of mirrors
- ➔ Mirroring and Striping
- ➔ Data is written in stripes across primary disks
- ➔ The primary disks have been mirrored to the secondary disks.

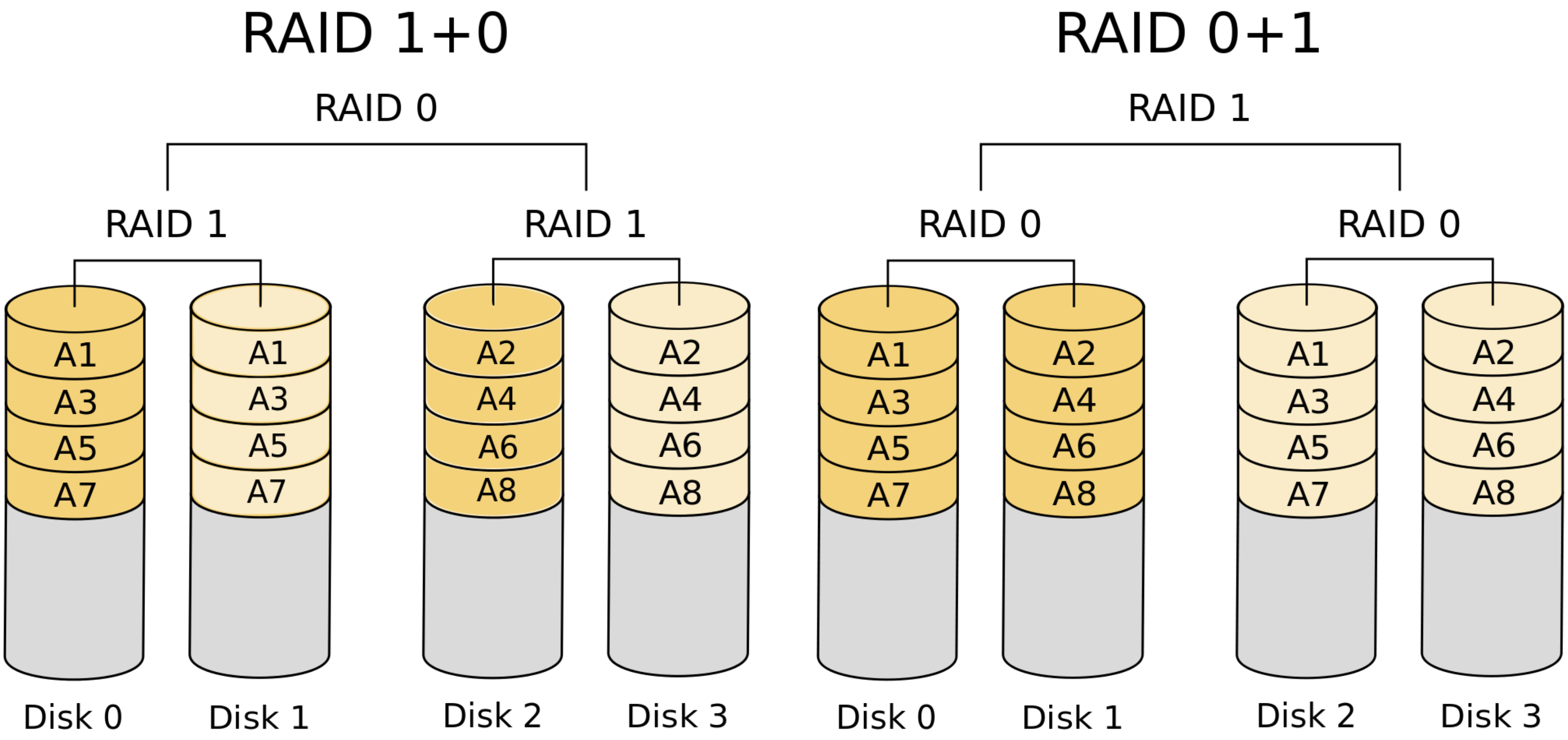
Disk Arrays

RAID 10

- ➔ Minimum 4 disks
- ➔ Excellent Redundancy
- ➔ Excellent performance
- ➔ Best option to mission critical applications

Disk Arrays

RAID 10



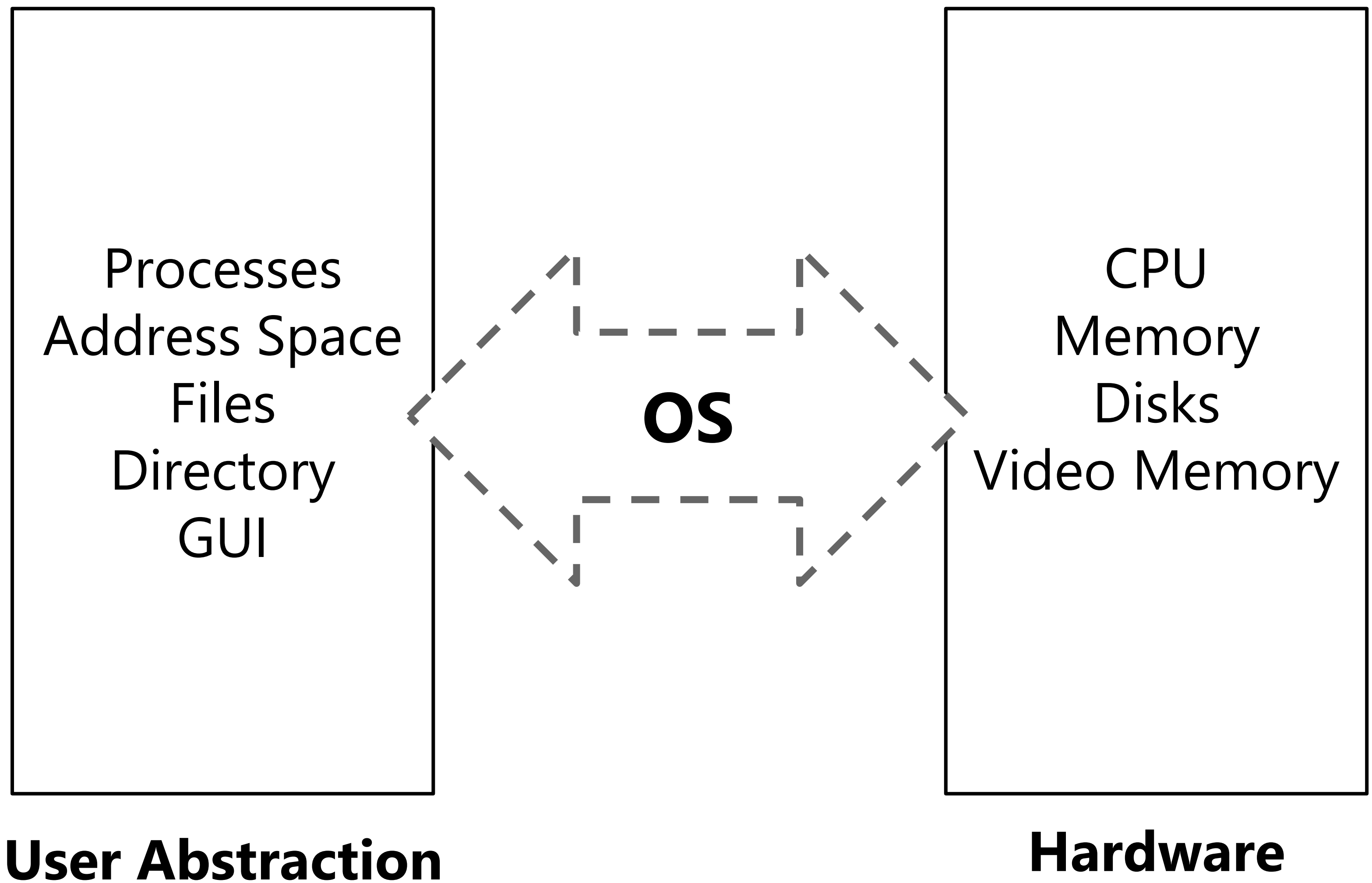
File Systems

➔ We have learned about external storage from a hardware point of view

➔ From now on, we will think of disks as linear sequence of blocks and supporting two operations:
  - Read block *k*
  - Write block *k*

File Systems

- ➔ But, if all we have are these two operations:
  - How do we find information?
  - How do keep one user from reading another's user data?
  - How do you know which blocks are free?


Processes
Address Space
Files
Directory
GUI
OS
CPU
Memory
Disks
Video Memory
User Abstraction
Hardware

File Systems

- ➔ Just like with any other resources of the computer, the operating system provides an abstract representation: **Files**
- ➔ Files are logical units of information created by processes
- ➔ A disk may contain thousands of them, each one **independent from the other**

File Systems

- ➔ Information in files must be **persistent**: not affected by process creation or termination
- ➔ The part of the operating system dealing with files is known as the file system

File Systems

- ➔ There are many types of file systems:
  - ◆ Ext2, Ext3 (Linux)
  - ◆ NTFS (Windows)
  - ◆ FAT, FAT32 (DOS, Windows)
  - ◆ CDFS (CD Rom)
  - ◆ Many others

- ➔ Programmers use an API to interact with the file system

File Systems

Requirements

- ➔ Persistence
- ➔ Performance
- ➔ Size
- ➔ Sharing
- ➔ Protection
- ➔ Ease of use
- ➔ Random Access

Terminology

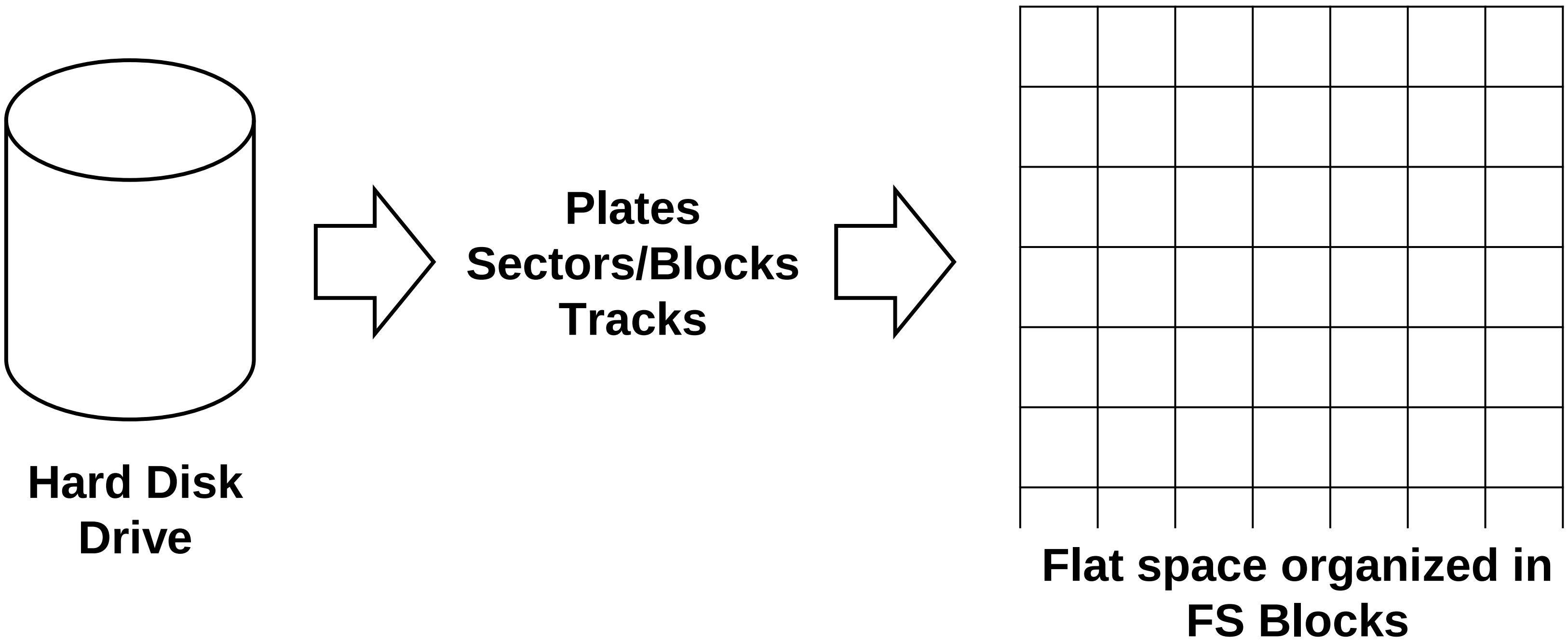
- ➔ **Disk**: permanent storage.
- ➔ **Block/Sector**: The smallest writable unit by a disk or FS. The FS block is always the same size as or larger than the disk block size.
- ➔ **Partition**: Subset of blocks

Terminology

- ➔ **Volume**: Name of collection of blocks. Disk or partition initialized with the FS.
- ➔ **Superblock**: where the FS stores its critical information (size, name, etc)

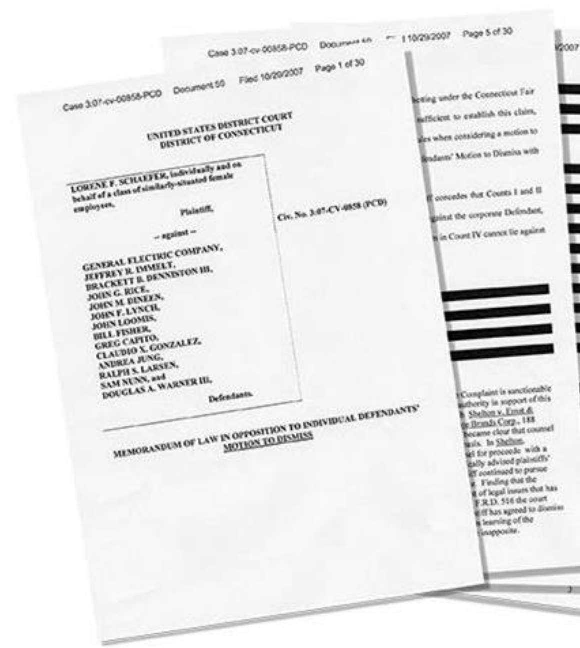
File Systems

Terminology



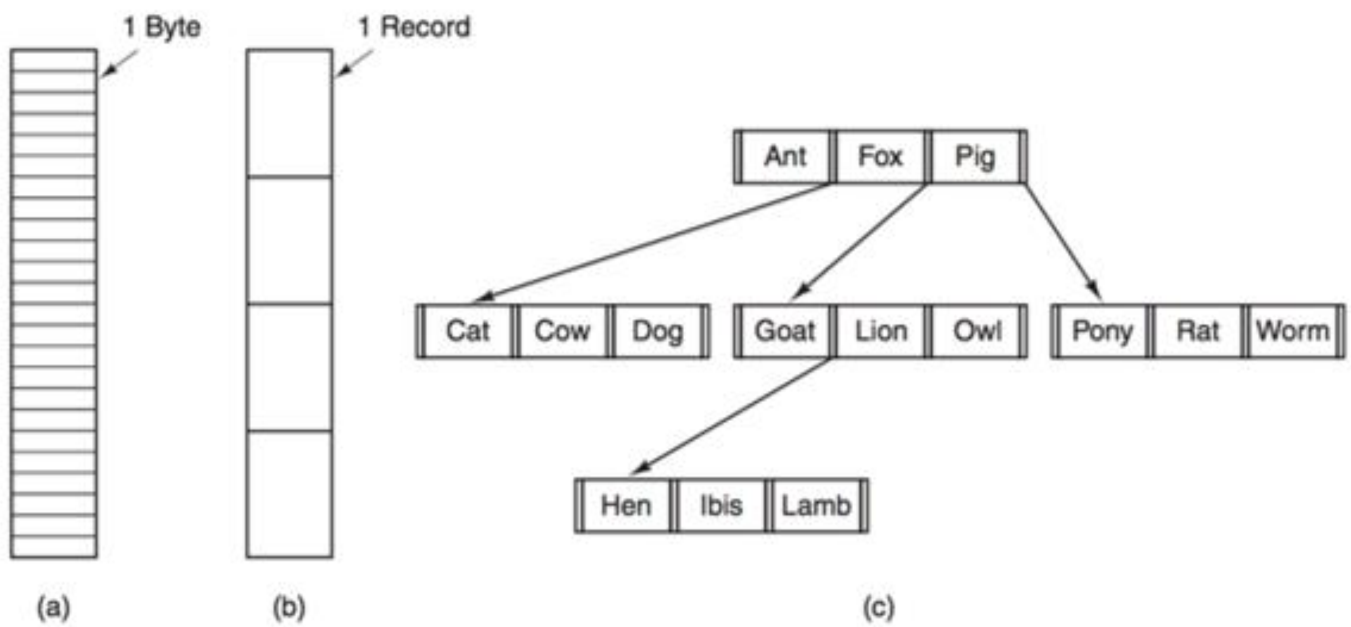
Files

- ➔ A file is an abstraction mechanism
  - ◆ Provides a way to store information on a disk and read it back later
  - ◆ Shields the user from the details of how and where the information is stored and how the disks actually work



File Structure

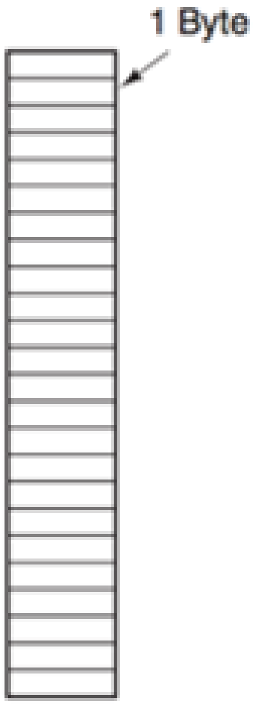
➔ There are three common possibilities for file structure:



File Structure

Byte Sequence

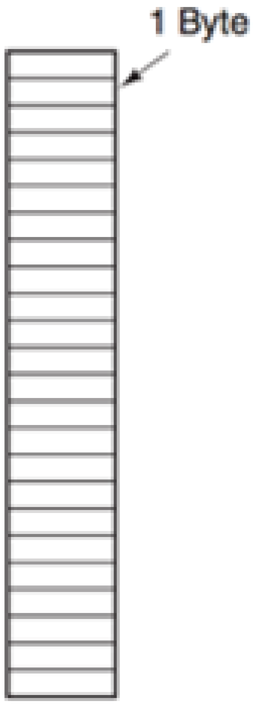
- ➔ The operating system does not know or care what is in the file, only sees bytes.
- ➔ Unix and Windows uses this approach
- ➔ Provides maximum flexibility, the OS doesn't help but also doesn't get in the way



(a)

Byte Sequence

→ These type of files are usually called **plain files**



(a)

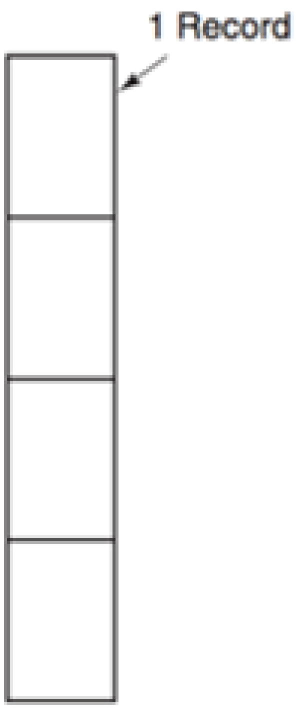
File Structure

Fixed-length records

➔ A file is a sequence of **fixed-length records** each with some internal structure

➔ Read operation returns a complete record and Write operation overwrites or appends one record

➔ Used in mainframe/midrange operating systems



(b)

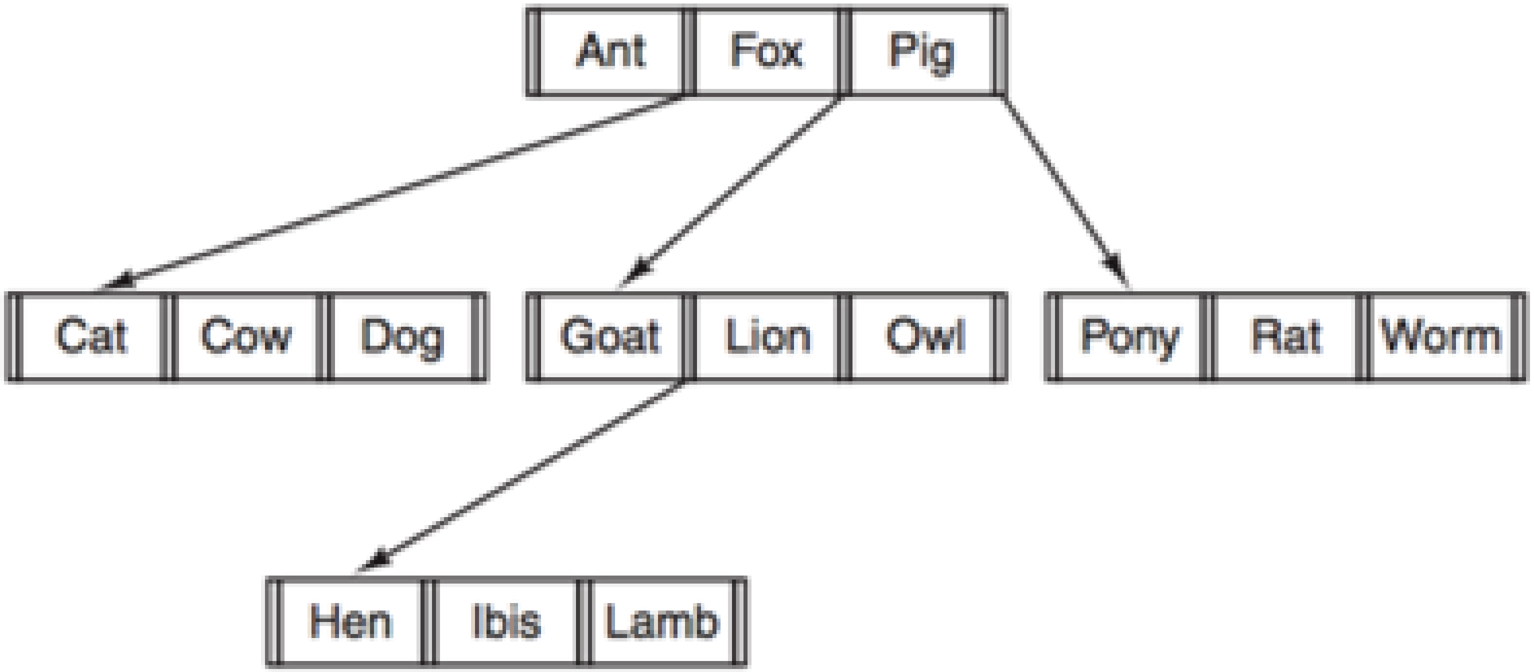
File Structure

Tree of records

- ➔ A file consists of a **tree of records**, not necessarily of the same length, each containing a key field
- ➔ The **tree is sorted by the key field** to allow rapid searching for a particular key
- ➔ The basic operation is not getting the next record, but a record with a specific key

File Structure

Tree of records



(c)

File Types

- ➔ **Regular files** contain user information
- ➔ **Directories** are system files for maintaining the structure of the file system
- ➔ **Character special files** are related to I/O and used to model devices such as printers, terminals and networks

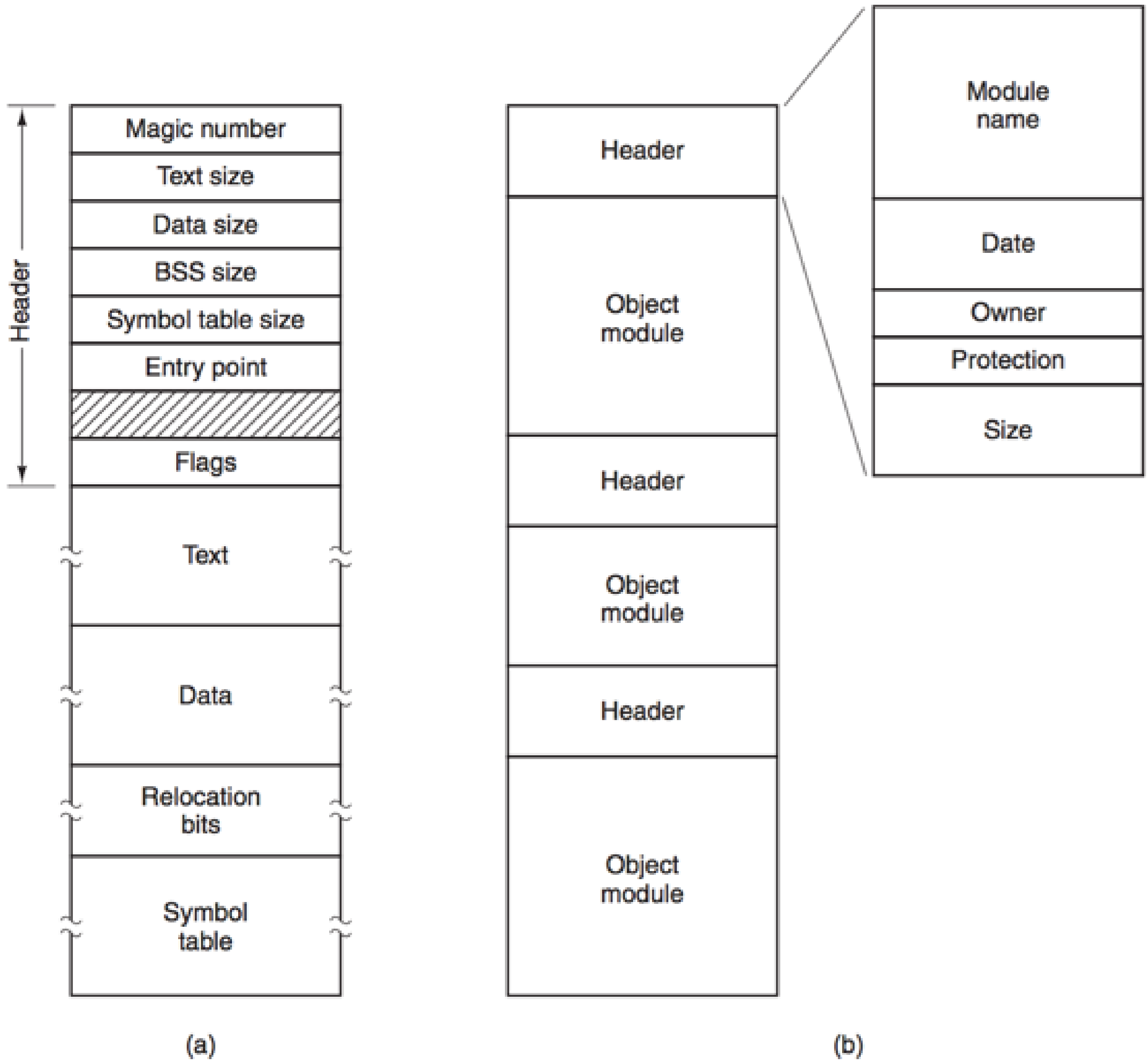
File Types

➔ **Block files** are used to model disks

➔ **Binary files** are executable programs

➔ **Archive files** are module libraries

File Types

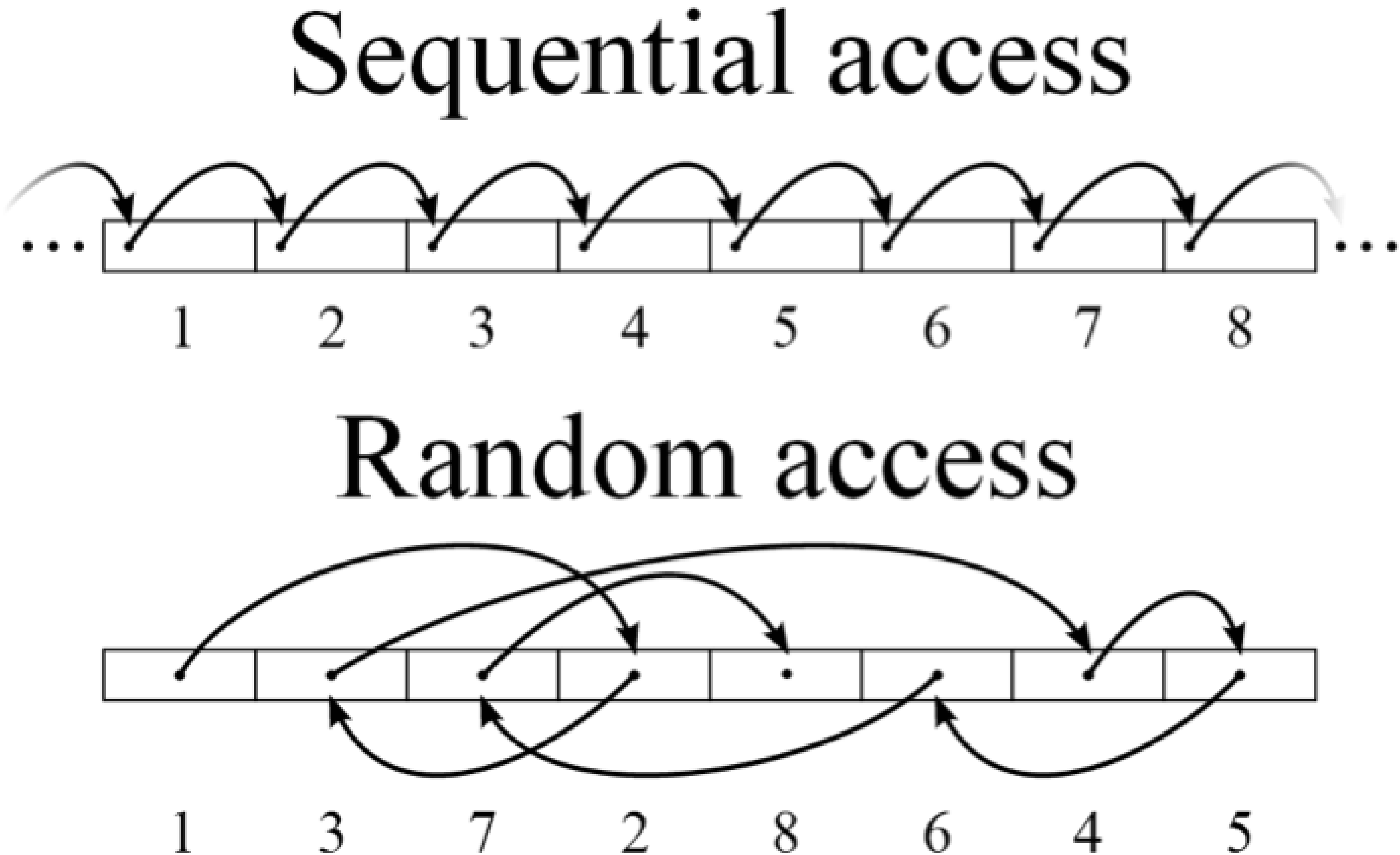

Header
Magic number
Text size
Data size
BSS size
Symbol table size
Entry point
Flags
Text
Data
Relocation bits
Symbol table
(a)
Header
Object module
Header
Object module
Header
Object module
Module name
Date
Owner
Protection
Size
(b)

Sequential Access

- Provided in early operating systems
- A process could only read all the bytes or records in a file i**n order, starting at the beginning**, but could not skip around and read as often as needed
- Used when the storage was magnetic tape

Random Access File

- ➔ Introduced with the usage of disks
- ➔ Allow to **read bytes or records of a file out of order** or to access records by key rather than by position
- ➔ Essential for many applications, for example database systems


Sequential access
... ...
1 2 3 4 5 6 7 8
Random access
1 3 7 2 8 6 4 5

File Metadata

Attribute	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 for normal files; 1 for system file
Archive flag	0 for has been backed up; 1 for needs to be backed up
ASCII/binary flag	0 for ASCII file; 1 for binary file
Random access flag	0 for sequential access only; 1 for random access
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was created
Time of last access	Date and time the file was last accessed
Time of last change	Date and time the file was last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

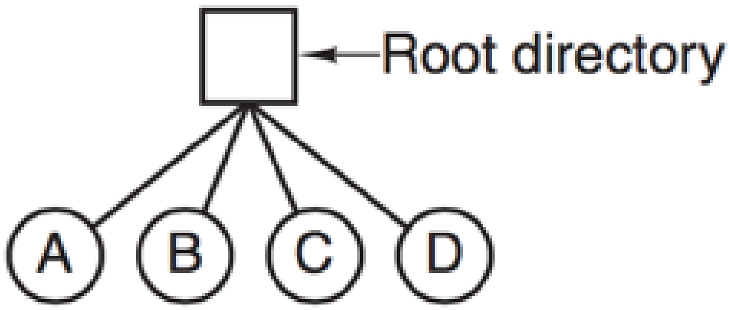
File System calls

```
int main(int argc, char *argv[])
{
        int in_fd, out_fd, rd_count, wt_count;
        char buffer[BUF_SIZE];

        if (argc != 3) exit(1);                      /* syntax error if argc is not 3 */

        /* Open the input file and create the output file */
        in_fd = open(argv[1], O_RDONLY);             /* open the source file */
        if (in_fd < 0) exit(2);                      /* if it cannot be opened, exit */
        out_fd = creat(argv[2], OUTPUT_MODE);        /* create the destination file */
        if (out_fd < 0) exit(3);                     /* if it cannot be created, exit */

        /* Copy loop */
        while (TRUE) {
                rd_count = read(in_fd, buffer, BUF_SIZE); /* read a block of data */
                if (rd_count <= 0) break;            /* if end of file or error, exit loop */
                wt_count = write(out_fd, buffer, rd_count); /* write data */
                if (wt_count <= 0) exit(4);          /* wt_count <= 0 is an error */
        }

        /* Close the files */
        close(in_fd);
        close(out_fd);
        if (rd_count == 0)                           /* no error on last read */
                exit(0);
        else
                exit(5);                             /* error on last read */
}
```

Directories

➔ Special **files** used to keep track of files

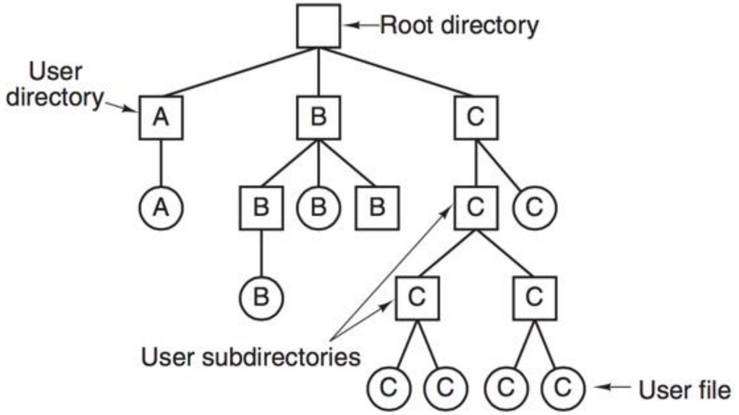
➔ The simplest form of directory system is having one directory containing all files



Directories

- ➔ Having all files in a single directory was not useful for general applications

- ➔ A hierarchical directory system (tree of directories) allowed to have as many directories as needed grouped in a suitable way



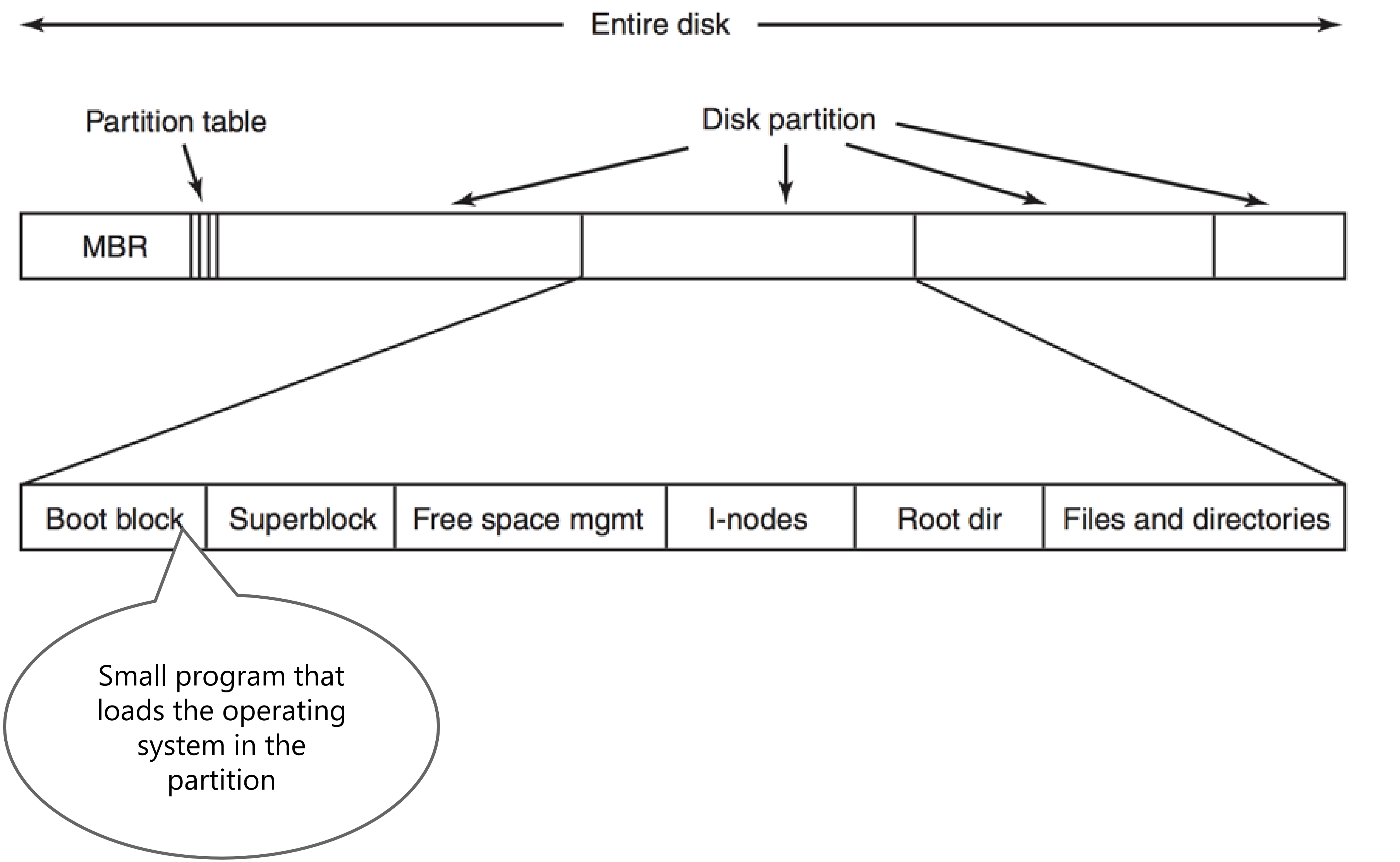
File System implementation

Layout

- ➔ File systems are stored on disks
- ➔ Sector 0 of the disk is called **Master Boot Record** (MBR) and is used to boot the computer
- ➔ At the end of the MBR is the partition table

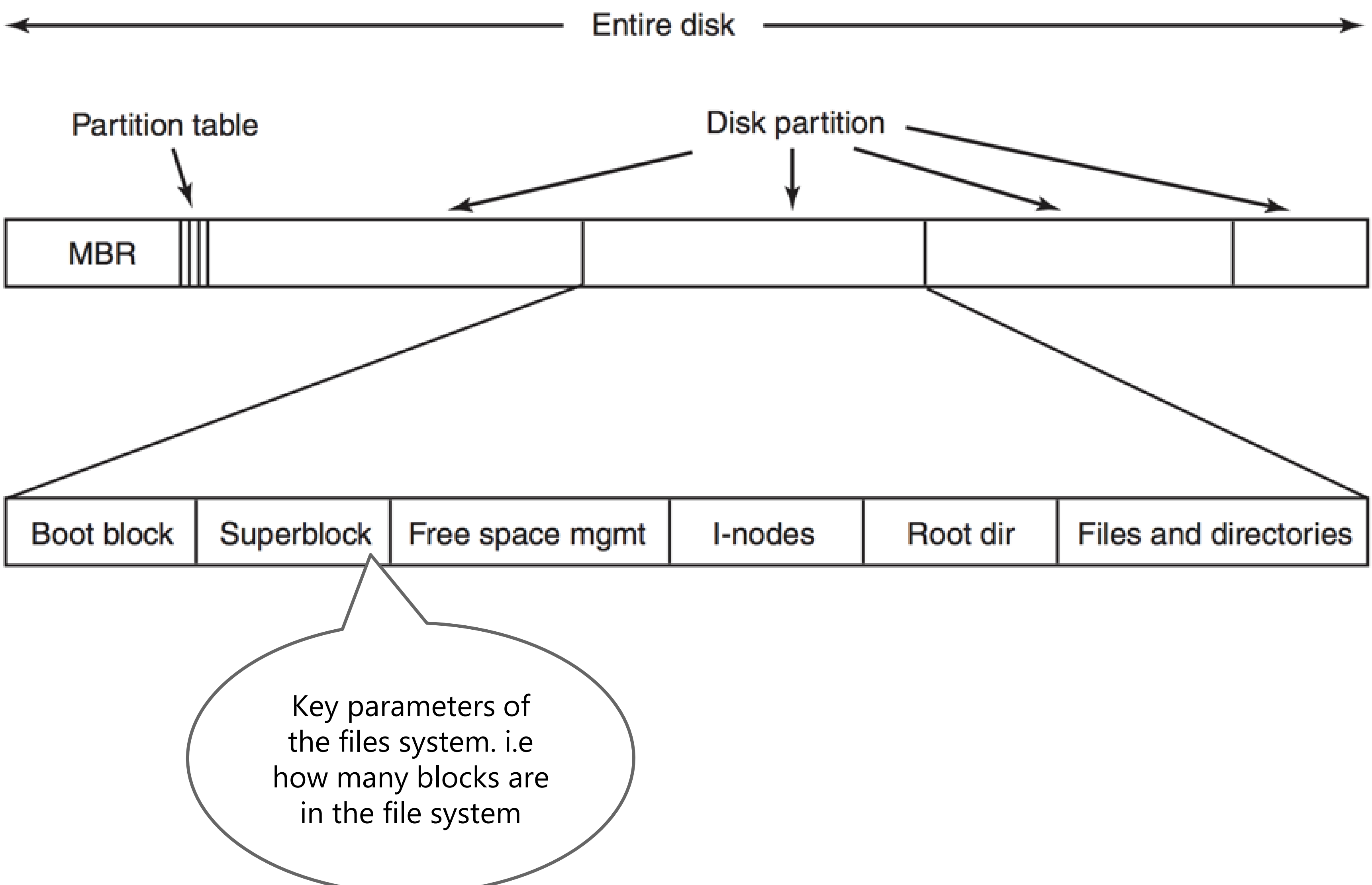
File System implementation

Layout



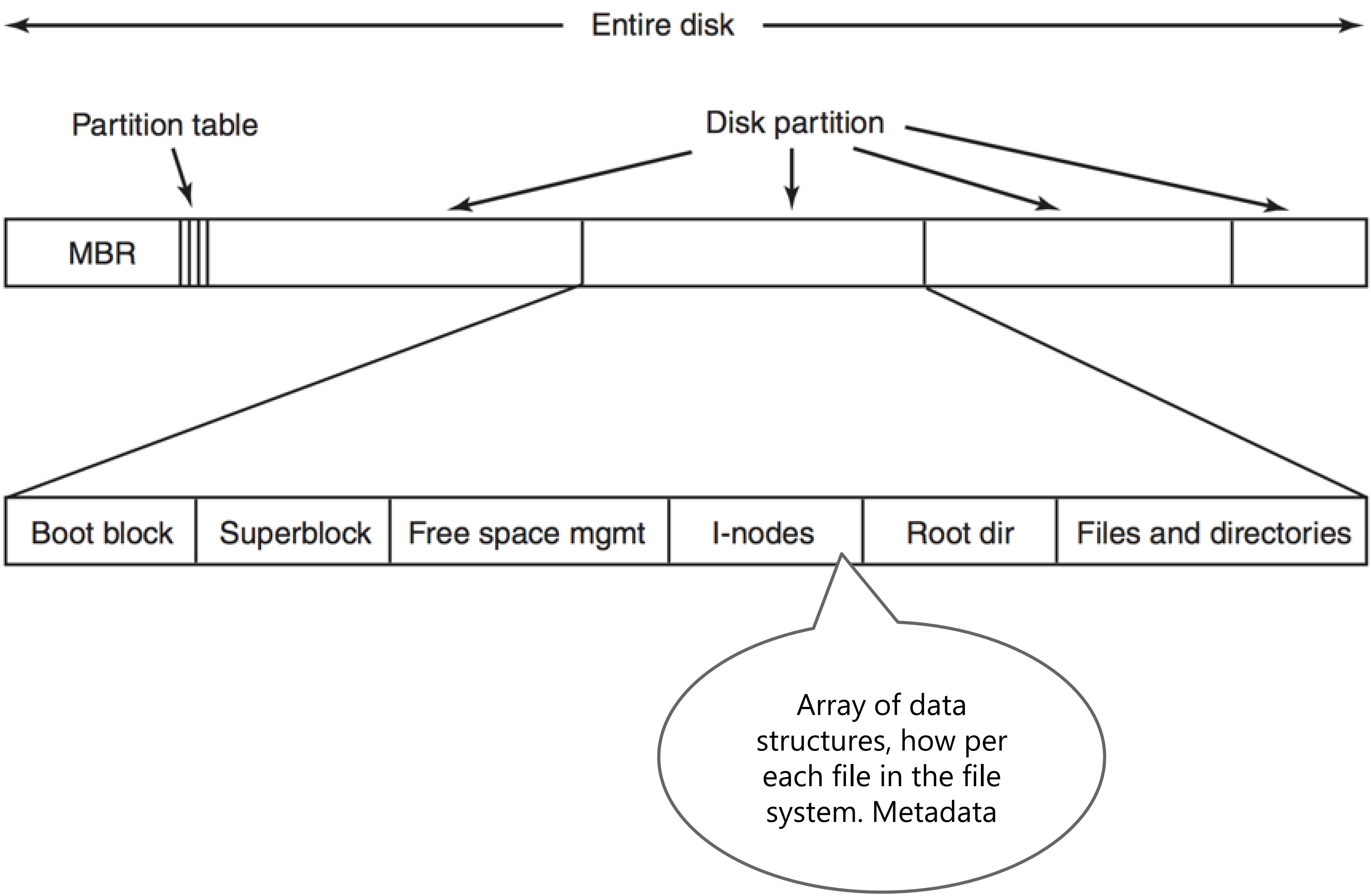
File System implementation

Layout



File System implementation

Layout

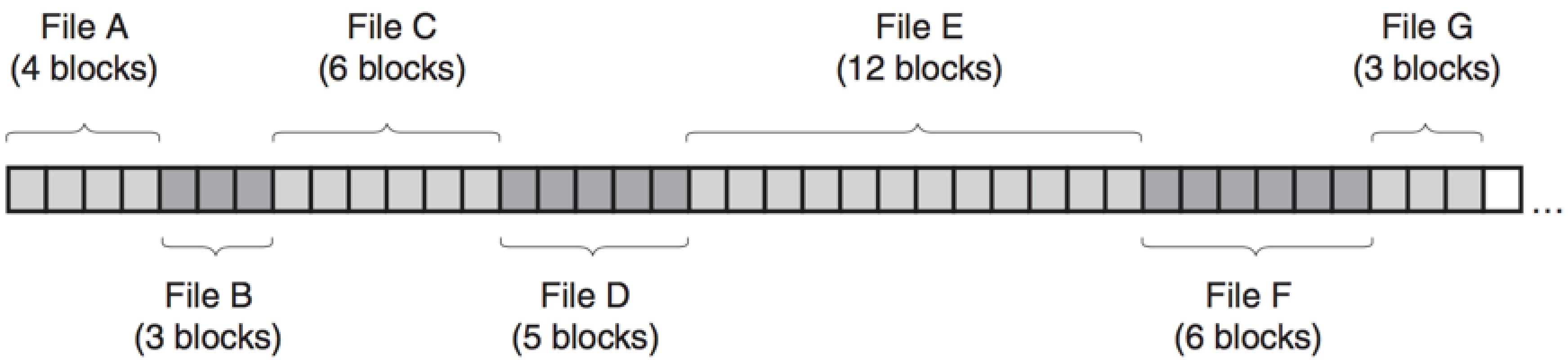


File System implementation

Files

➔ How to keep track of which blocks go with which file?

➔ **Contiguous Allocation**: all blocks of a file are physically next one of each other



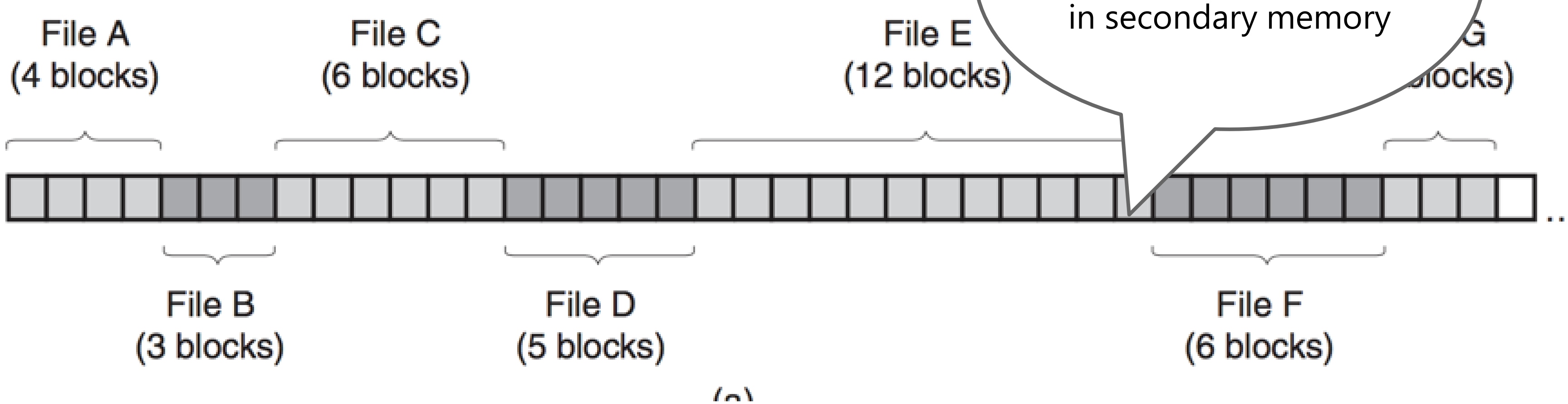
(a)

Files

➔ How to keep track of which blocks go with which file?

➔ **Contiguous Allocation**: all bloc f a file are physically next one of ea

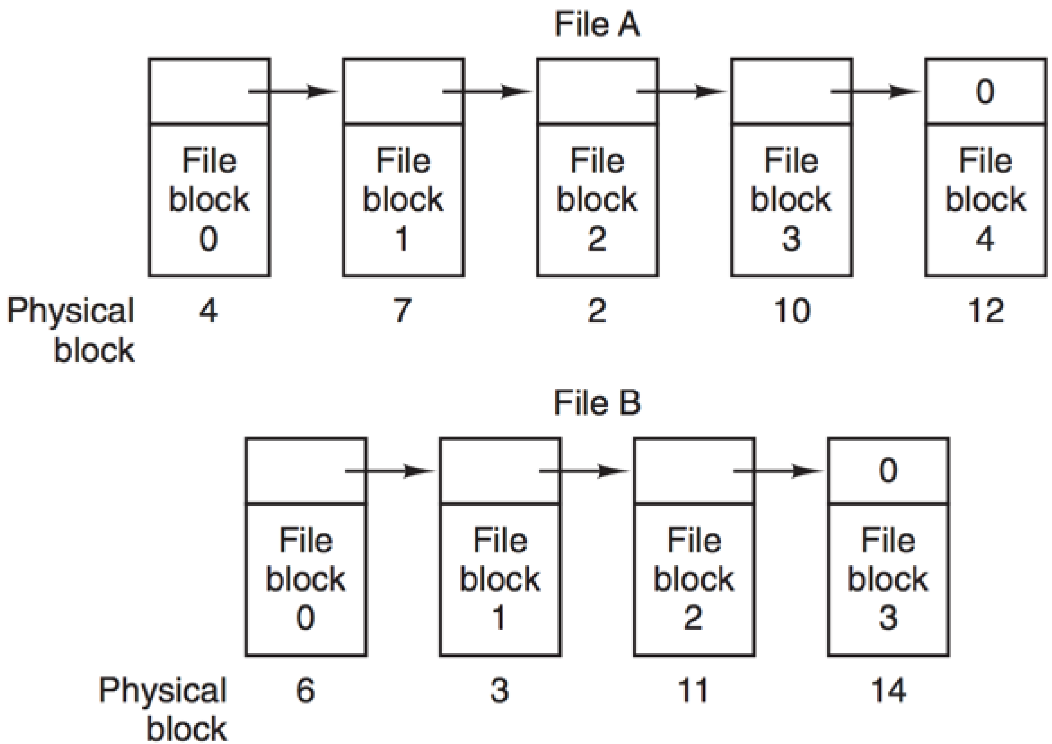
De-fragmentation is extremely expensive in secondary memory



File System implementation

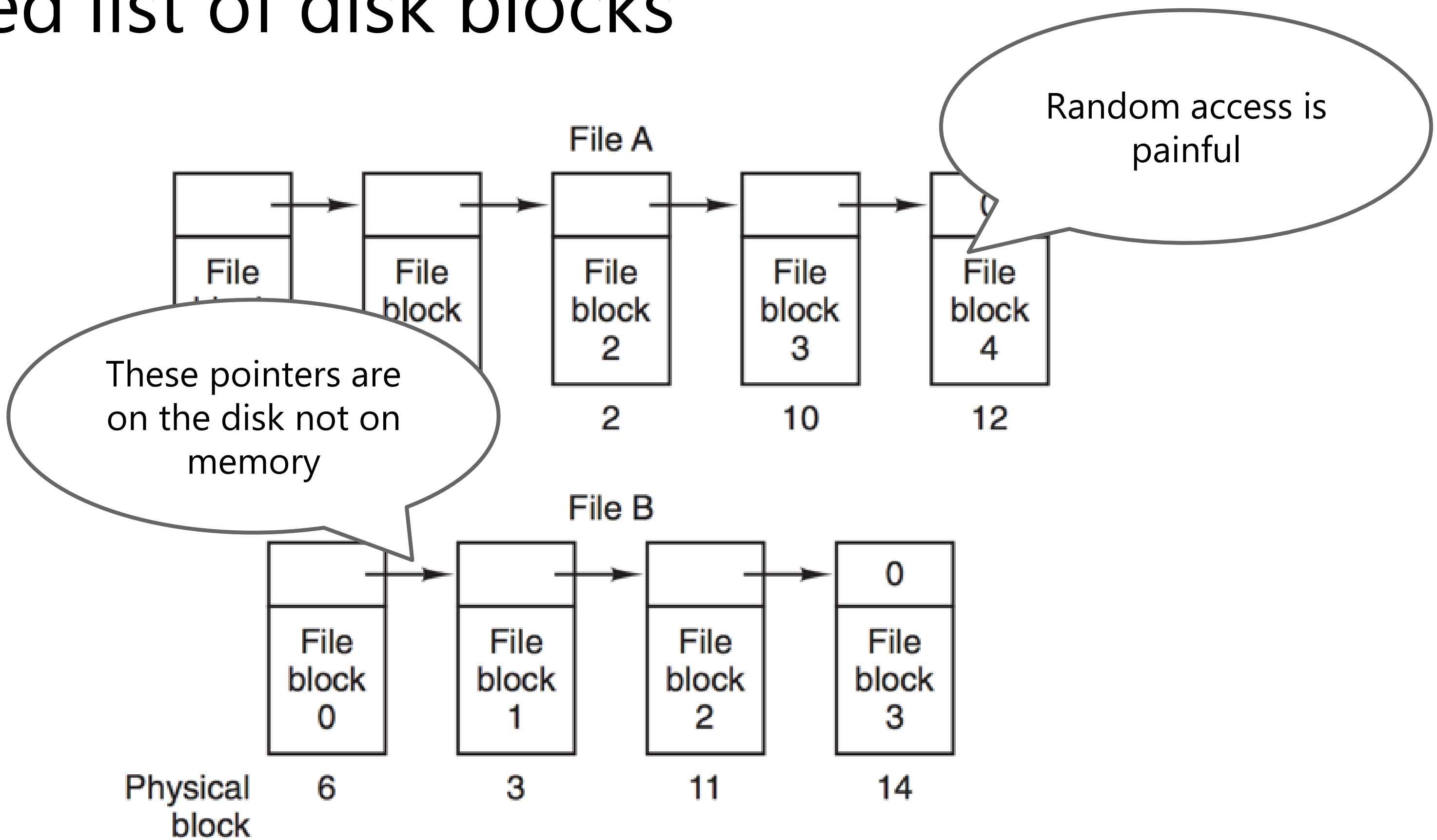
Files

→ **Linked-List Allocation**: keep each file as a linked list of disk blocks



Files

➔ **Linked-List Allocation**: keep each file as a linked list of disk blocks

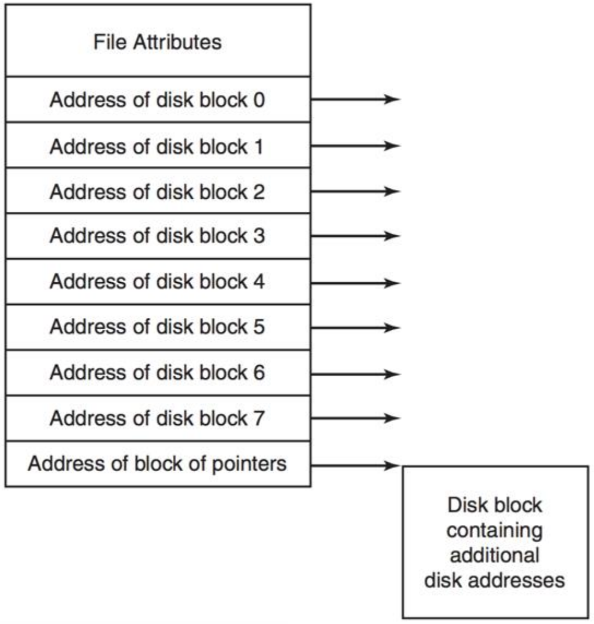


File System implementation

Files

➔ **i-Nodes (index-node)**

associate each file with a data structure which lists the attributes and disk addresses of the file' s block

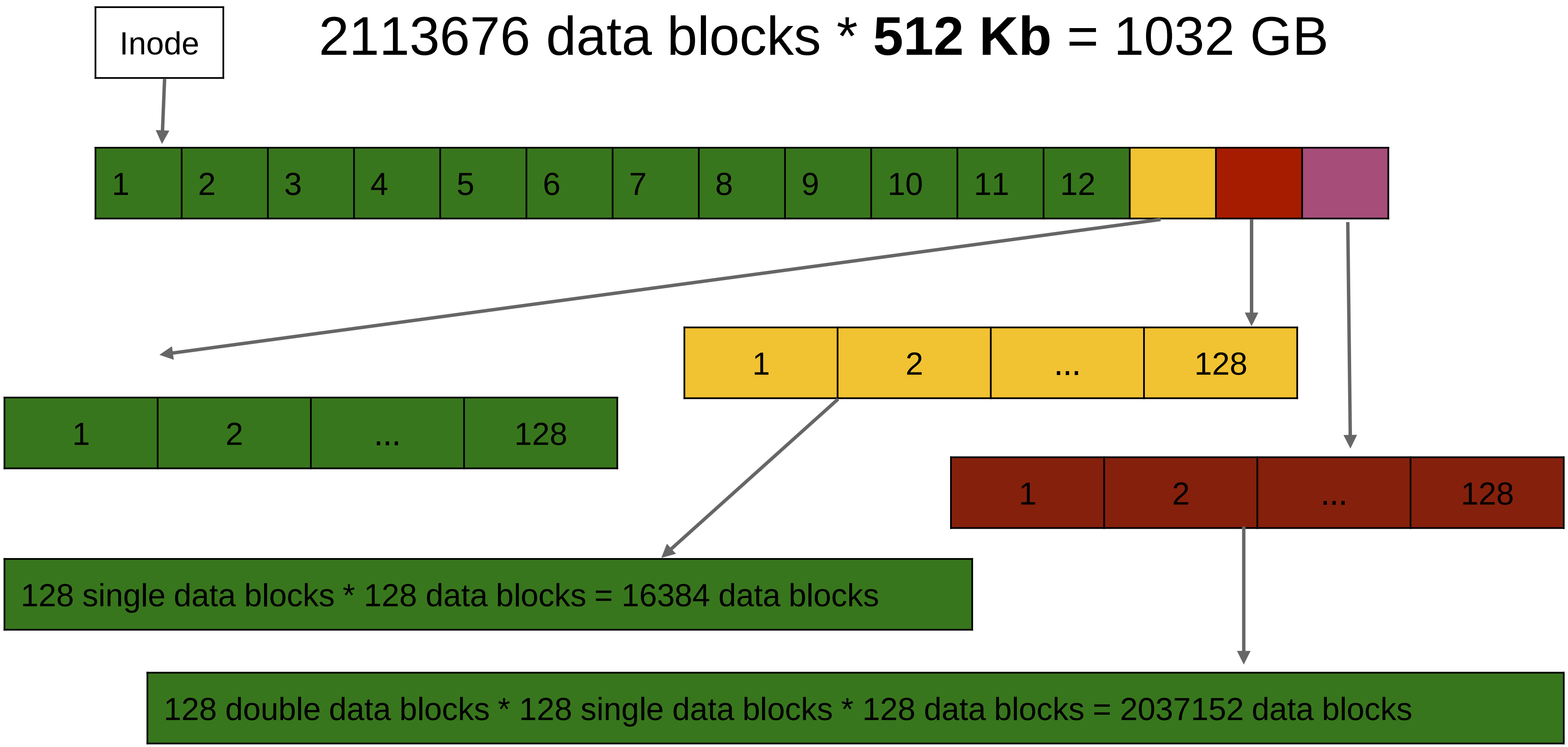


Inodes

- 15 pointers
  - 12 pointers to data blocks
  - 13th indirect block, block containing pointers to data blocks.
  - 14th doubly-indirect data blocks (128 pointers).
  - 15th triply-indirect data blocks

File System implementation
Inodes

Maximun File Size



File System implementation

Files

- ➔ The main advantage of this scheme is that the i-node need to be in memory only when the corresponding file is opened
  - ◆ Less overhead

Important

- Hotspare
- Rebuild drive
- Raid Controller
- Software Raid
- Hardware Raid
- Nested Raid Levels
- Direct Memory Access

External Storage Structures

CE-2101 Algorithms and Data Structures