COMPUTER SCIENCE

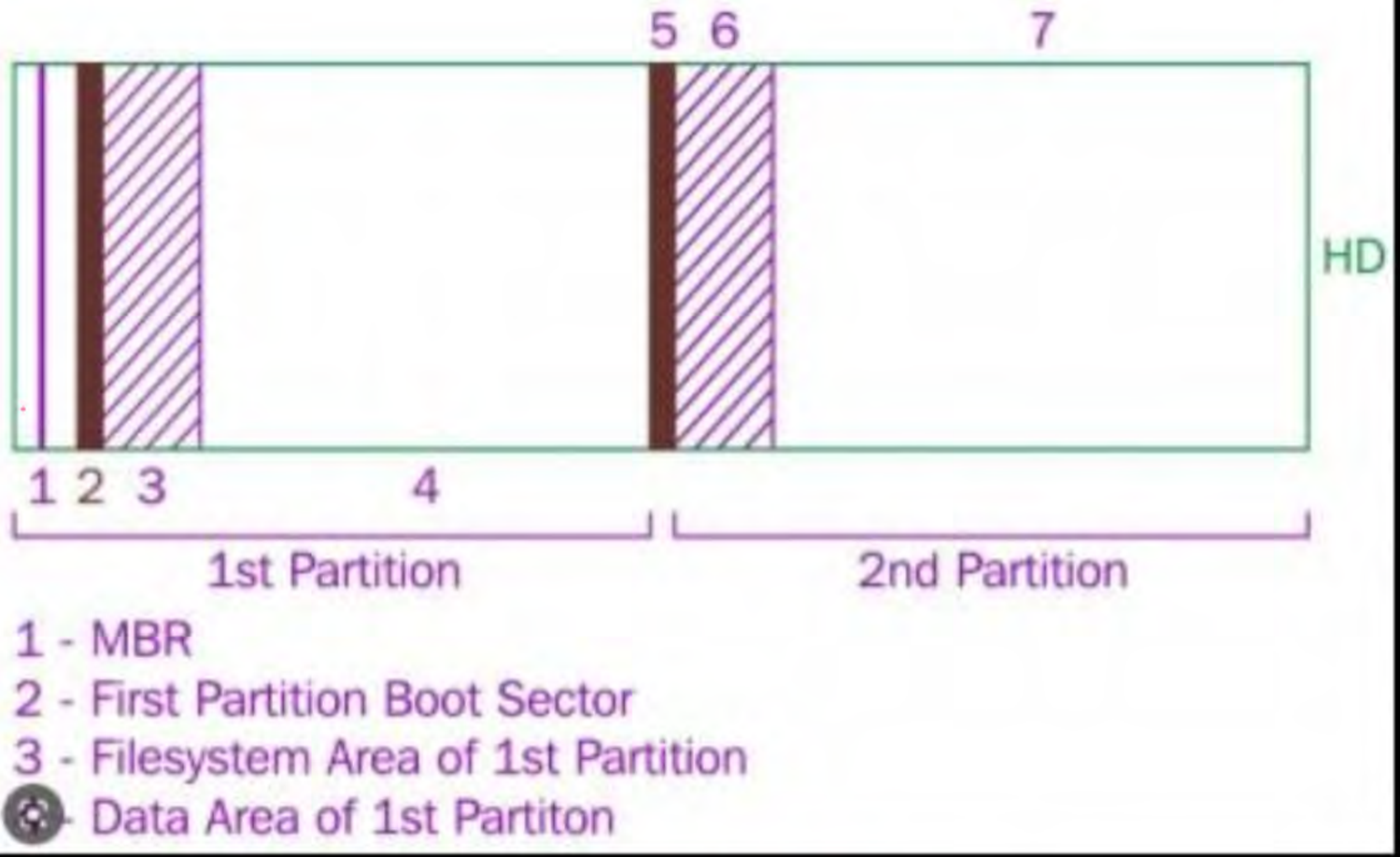
Operating System

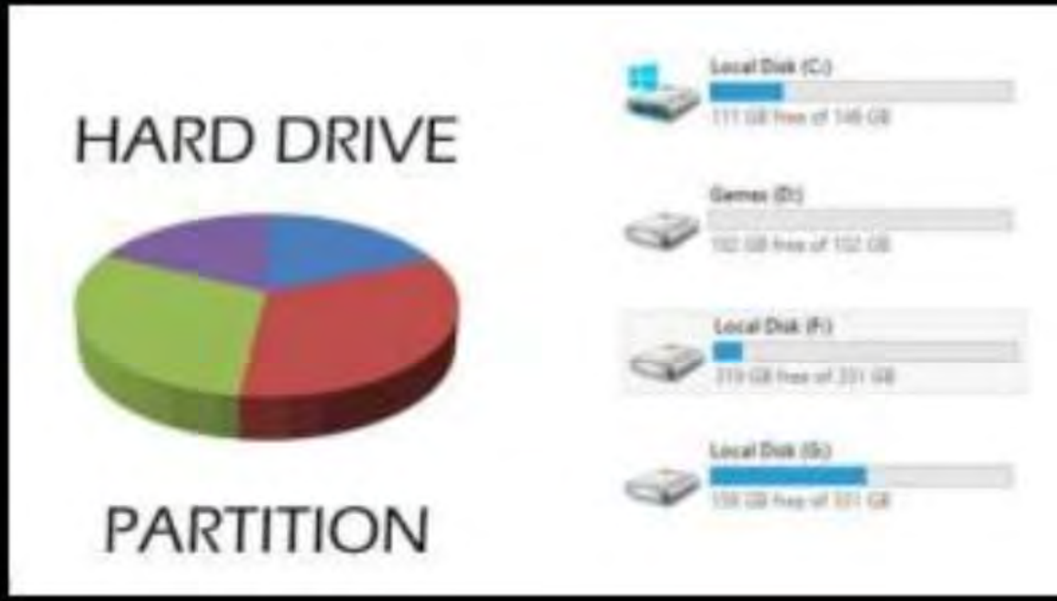
File System & Device Management

Physical structure of disk

Lecture No-01

Physical structure of disk

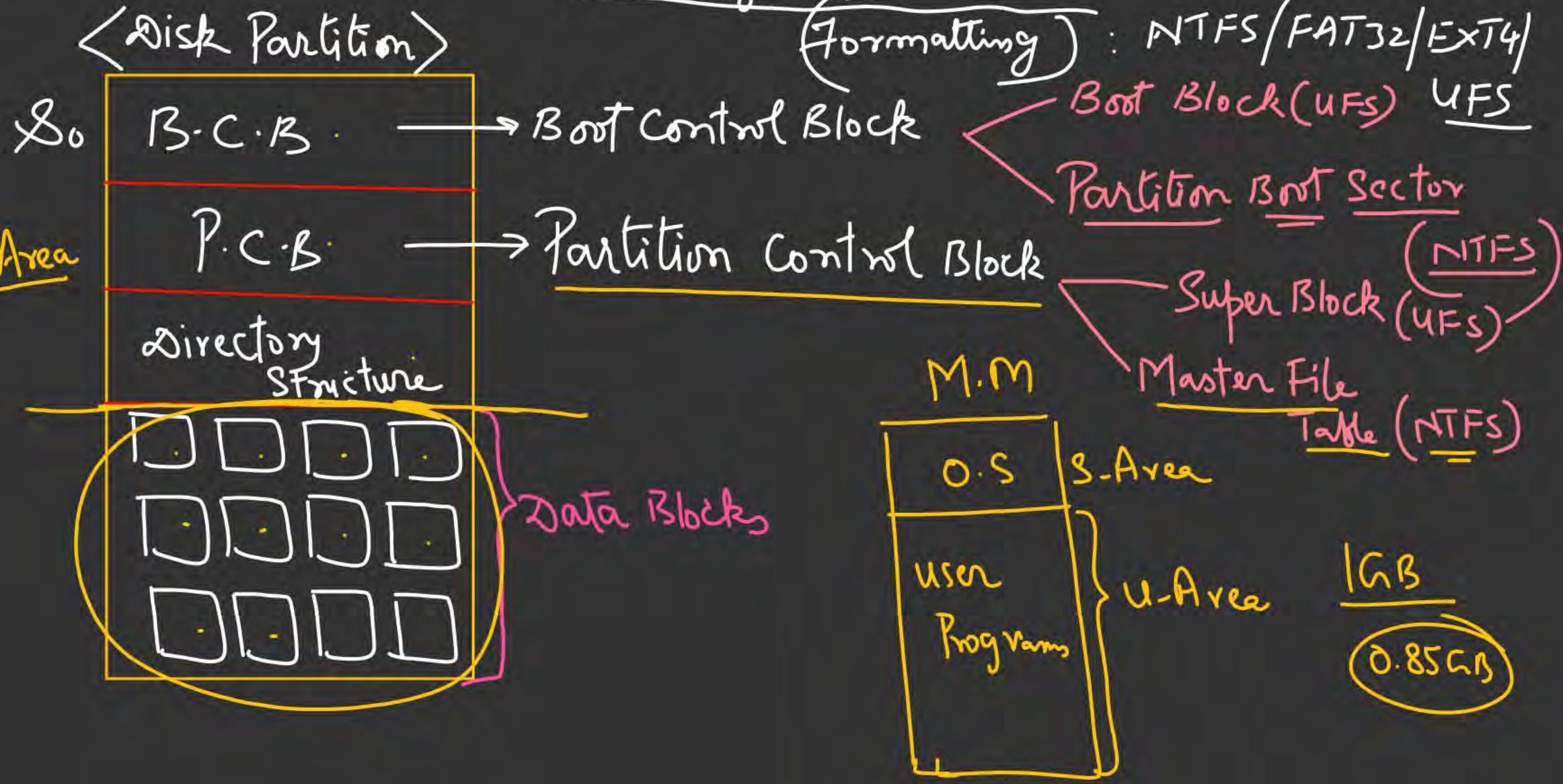




Booting Process

Disk Logical Structure

(Formatting) : NTFS / FAT32 / EXT4 / UFS

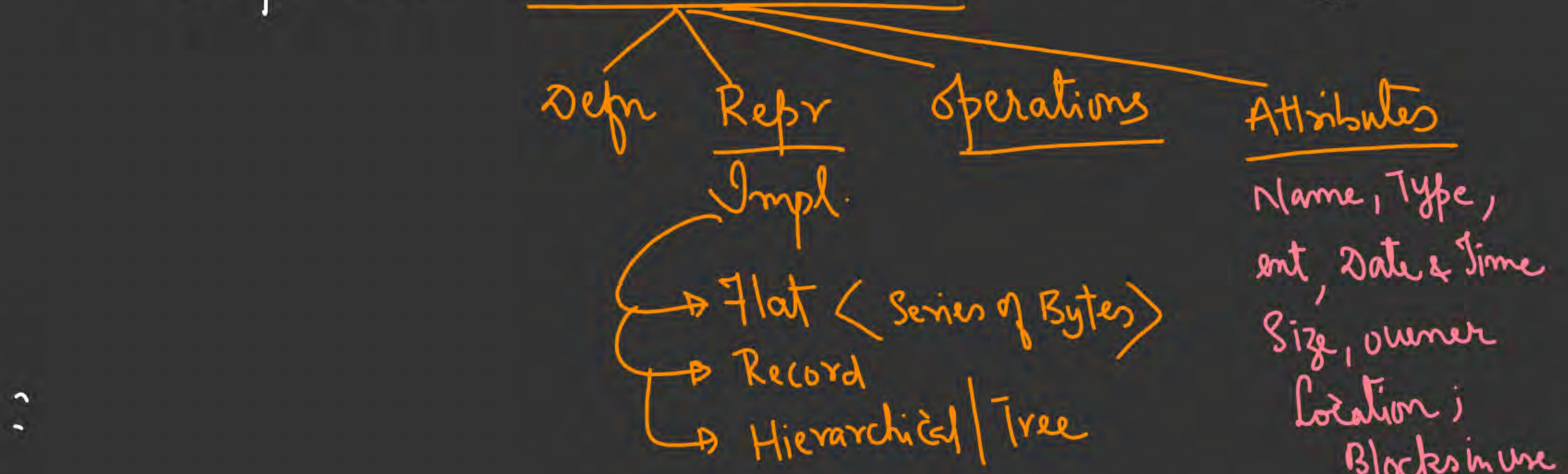


File System Interface

File Concept:

→ is a collection of Logically Related Records / Entity;

Impl-wise Files is an A.D.T

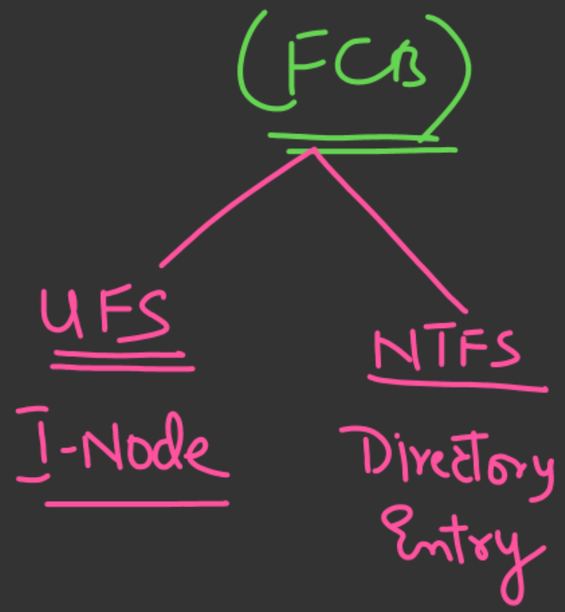


A file's attributes vary from one operating system to another but typically consist of these:

- **Name**. The symbolic file name is the only information kept in human-readable form.
- **Identifier**. This unique tag, usually a number, identifies the file within the file system; it is the non-human-readable name for the file.
- **Type**. This information is needed for systems that support different types of files.
- **Location**. This information is a pointer to a device and to the location of the file on that device.
- **Size**. The current size of the file (in bytes, words, or blocks) and possibly the maximum allowed size are included in this attribute.
- **Protection**. Access-control information determines who can do reading, writing, executing, and so on.
- **Time, date, and user identification**. This information may be kept for creation, last modification, and last use. These data can be useful for protection, security, and usage monitoring.

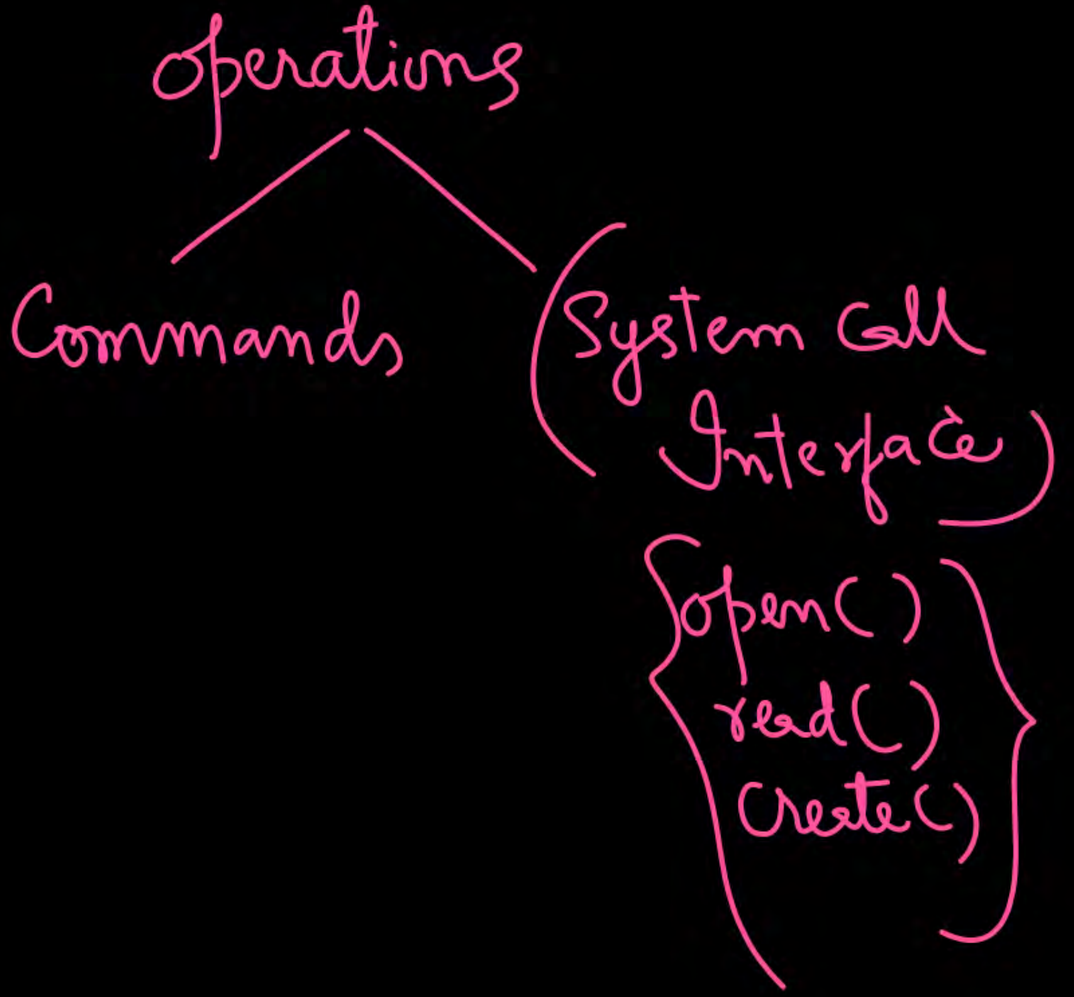
Figure 11.1 A file info window on Mac OS X.

Attributes of a File are Kept (Stored) in File Control Block (FCB).



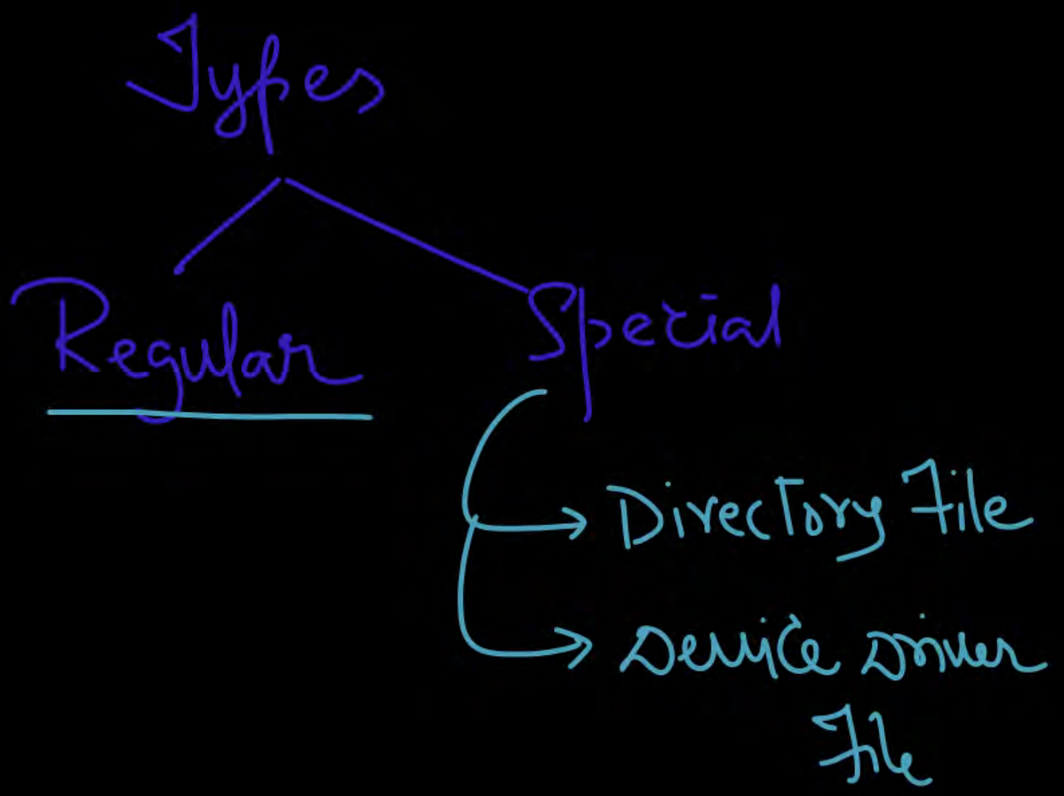
File operations:

- **Creating a file.** Two steps are necessary to create a file. First, space in the file system must be found for the file. We discuss how to allocate space for the file in Chapter 12. Second, an entry for the new file must be made in the directory.
- **Writing a file.** To write a file, we make a system call specifying both the name of the file and the information to be written to the file. Given the name of the file, the system searches the directory to find the file's location. The system must keep a write pointer to the location in the file where the next write is to take place. The write pointer must be updated whenever a write occurs.
- **Reading a file.** To read from a file, we use a system call that specifies the name of the file and where (in memory) the next block of the file should be put. Again, the directory is searched for the associated entry, and the system needs to keep a read pointer to the location in the file where the next read is to take place. Once the read has taken place, the read pointer is updated. Because a process is usually either reading from or writing to a file, the current operation location can be kept as a per-process current-file-position pointer. Both the read and write operations use this same pointer, saving space and reducing system complexity.
- **Repositioning within a file.** The directory is searched for the appropriate entry, and the current-file-position pointer is repositioned to a given value. Repositioning within a file need not involve any actual I/O. This file operation is also known as a file seek.
- **Deleting a file.** To delete a file, we search the directory for the named file. Having found the associated directory entry, we release all file space, so that it can be reused by other files, and erase the directory entry.
- **Truncating a file.** The user may want to erase the contents of a file but keep its attributes. Rather than forcing the user to delete the file and then recreate it, this function allows all attributes to remain unchanged—except for file length—but lets the file be reset to length zero and its file space released.



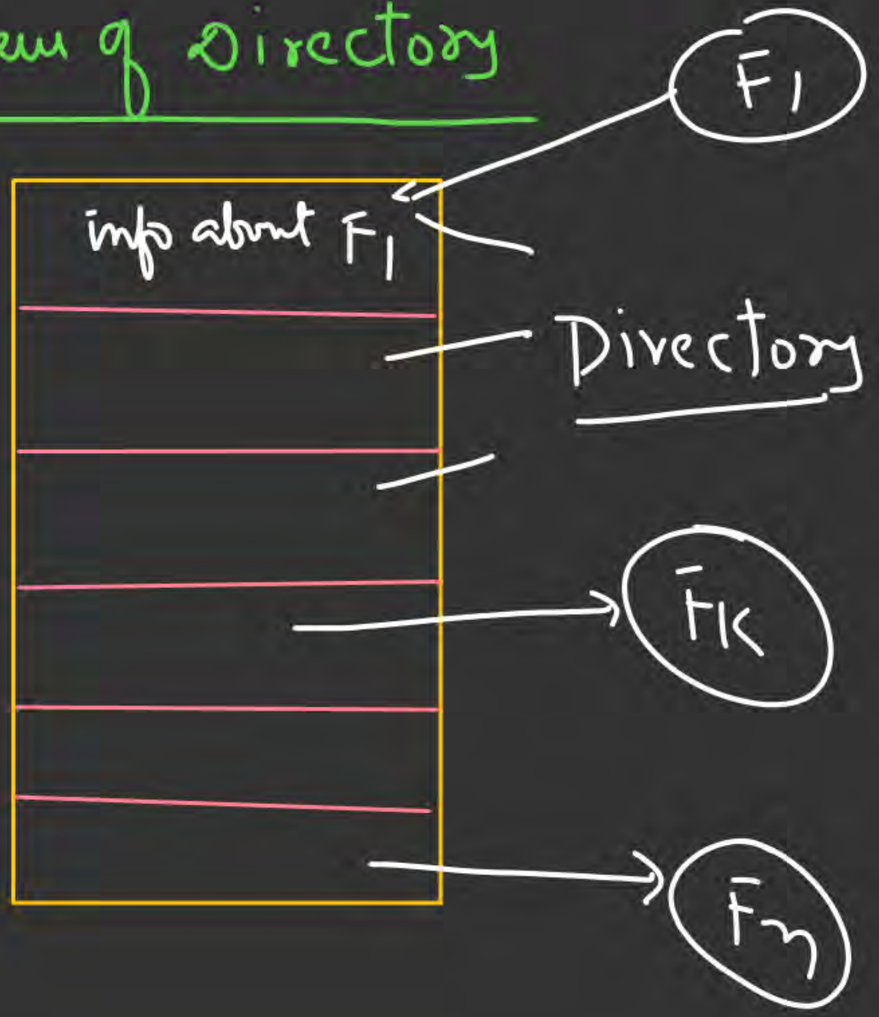
file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, perl, asm	source code in various languages
batch	bat, sh	commands to the command interpreter
markup	xml, html, tex	textual data, documents
word processor	xml, rtf, docx	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	gif, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	rar, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, mp3, mp4, avi	binary file containing audio or A/V information

Figure 11.3 Common file types.



Directory : Contains information about files; (Metadata)

Abs- view of Directory



Linear 1-Dim- Array

[1 entry Per file]

Directory Structures

1) Single level
2) Two - Level
3) Tree
4) DAG

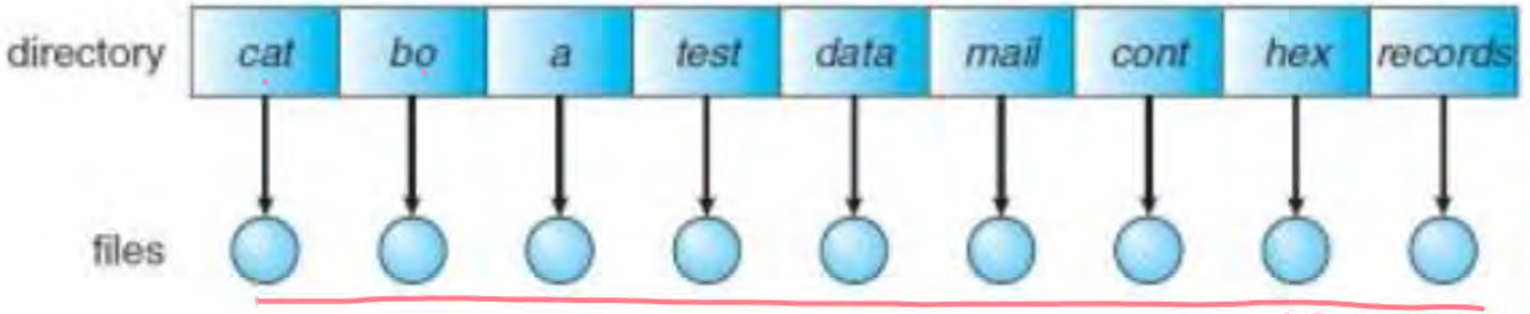


Figure 11.9 Single-level directory.

Simple to Impl.
→ Search Time ;
→ Name Conflict

Files

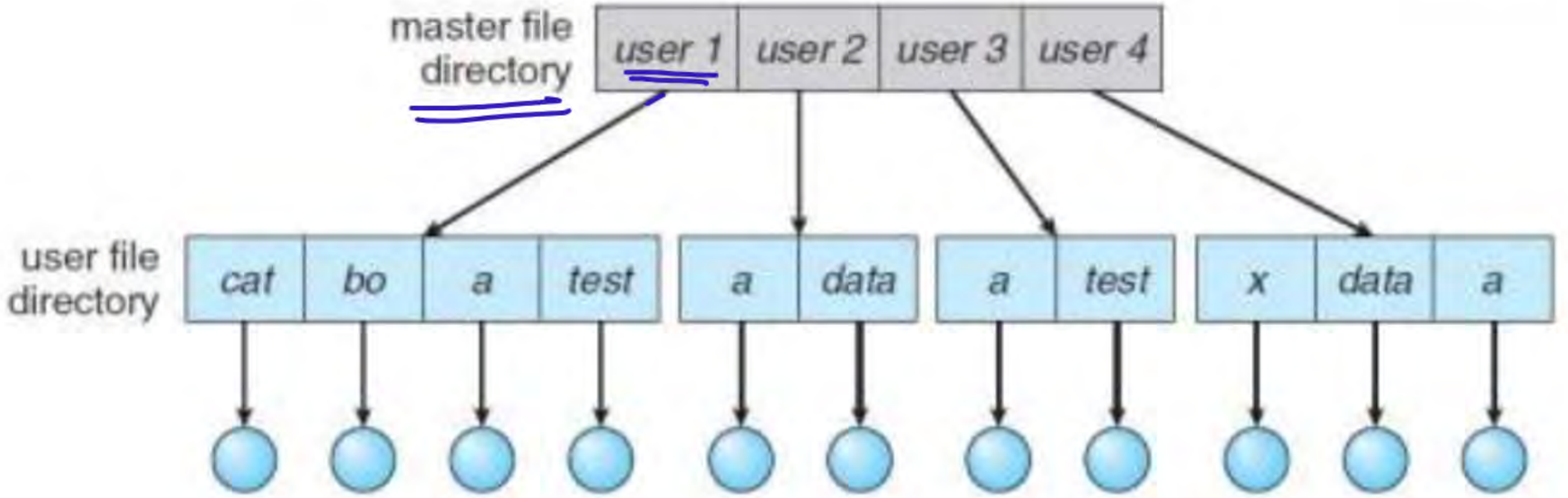


Figure 11.10 Two-level directory structure.

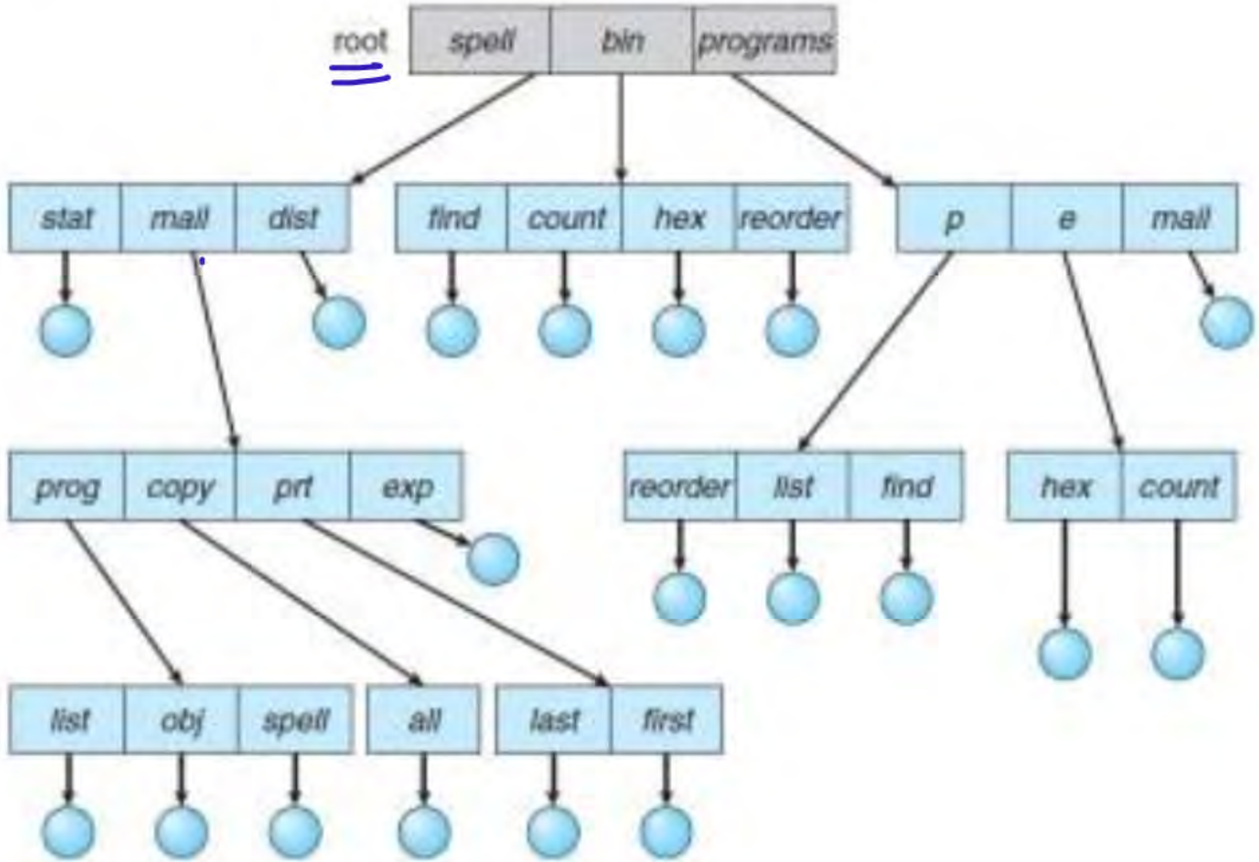


Figure 11.11 Tree-structured directory structure.

File Sharing

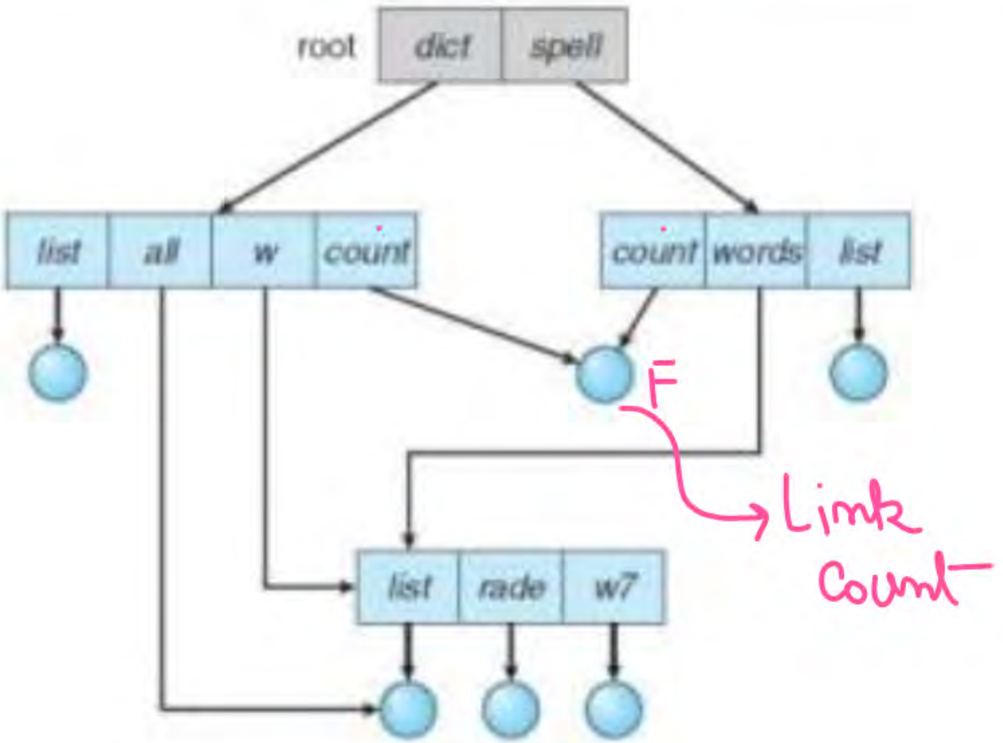


Figure 11.12 Acyclic-graph directory structure.

DAG

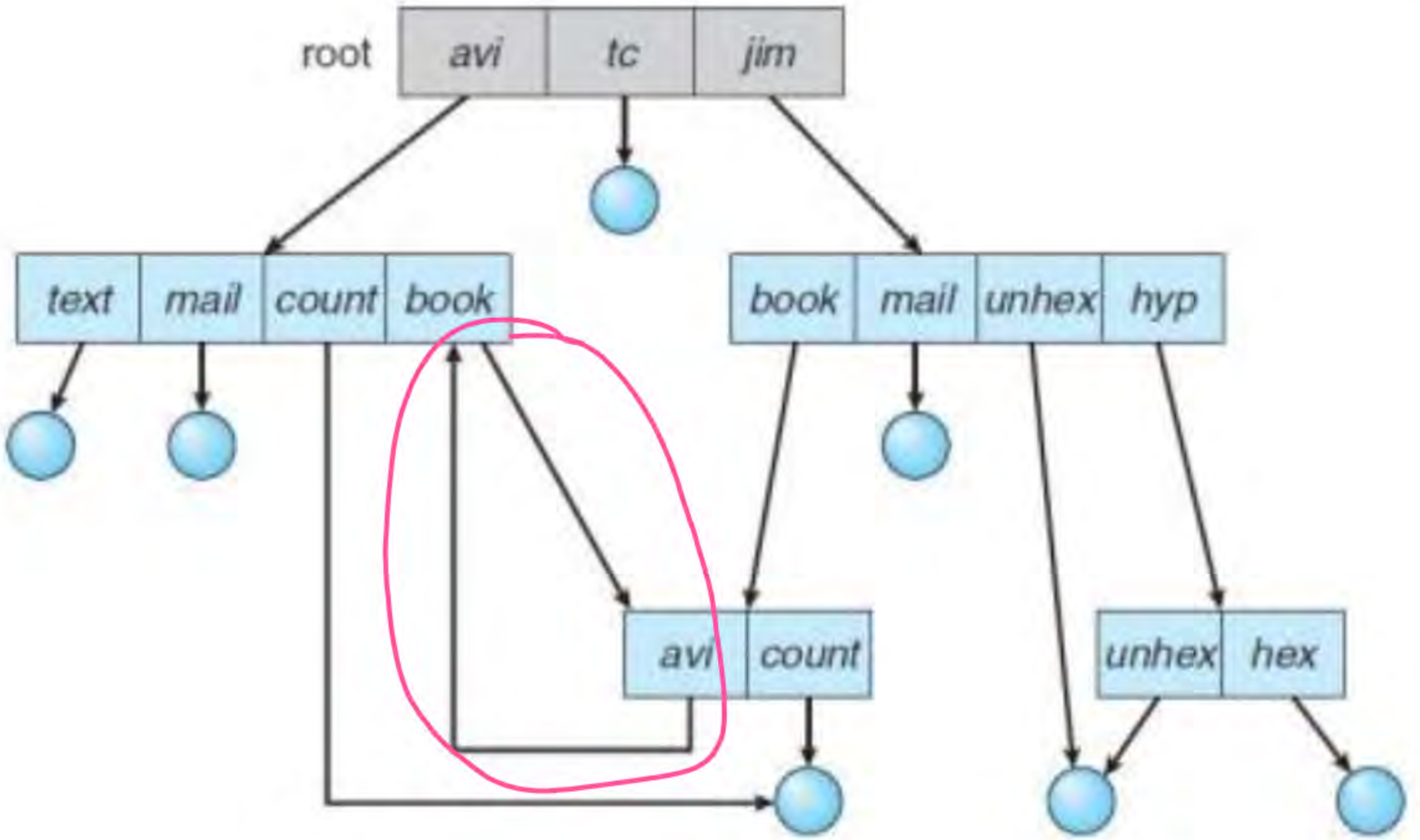


Figure 11.13 General graph directory.

Traversal (opn)
Directory
needs
(Traversal
Mechanism)
DFS
BFS

(**) File System Implementation

(Layers)

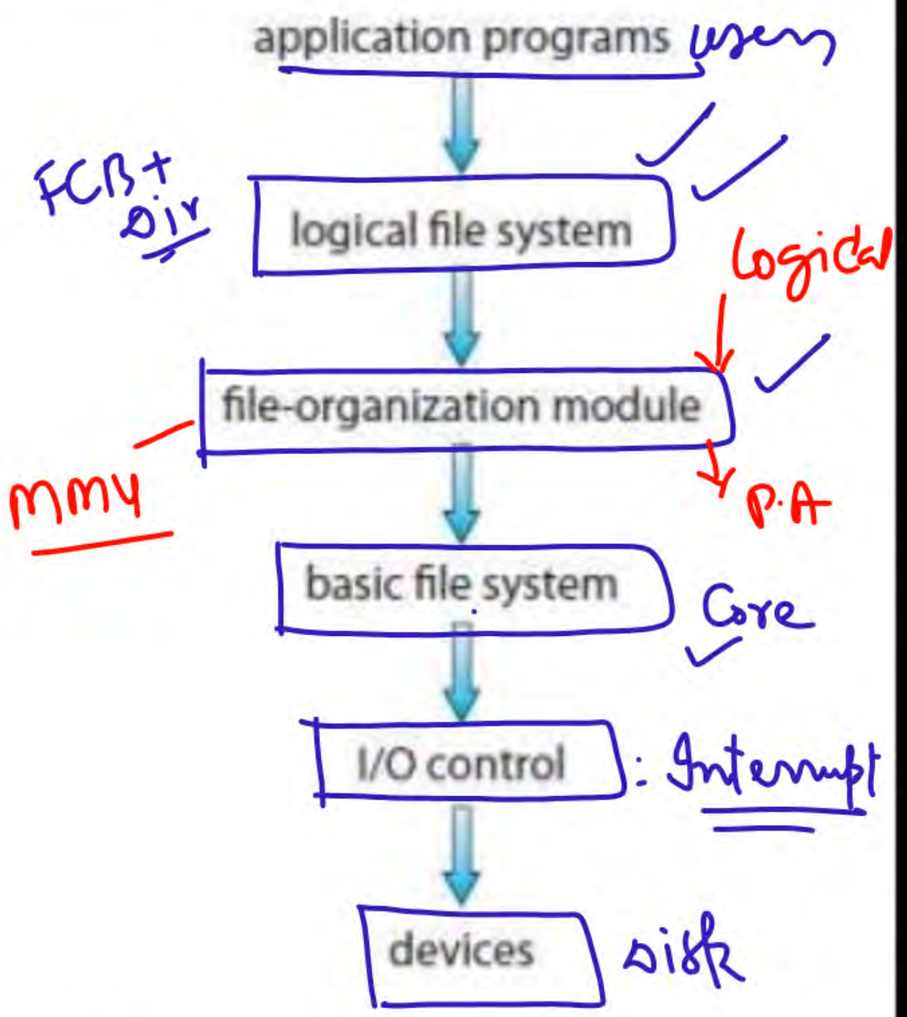


Figure 12.1 Layered file system.

The I/O control level consists of device drivers and interrupt handlers to transfer information between the main memory and the disk system. A device driver can be thought of as a translator. Its input consists of high-level commands such as "retrieve block 123." Its output consists of low-level, hardware-specific instructions that are used by the hardware controller, which interfaces the I/O device to the rest of the system. The device driver usually

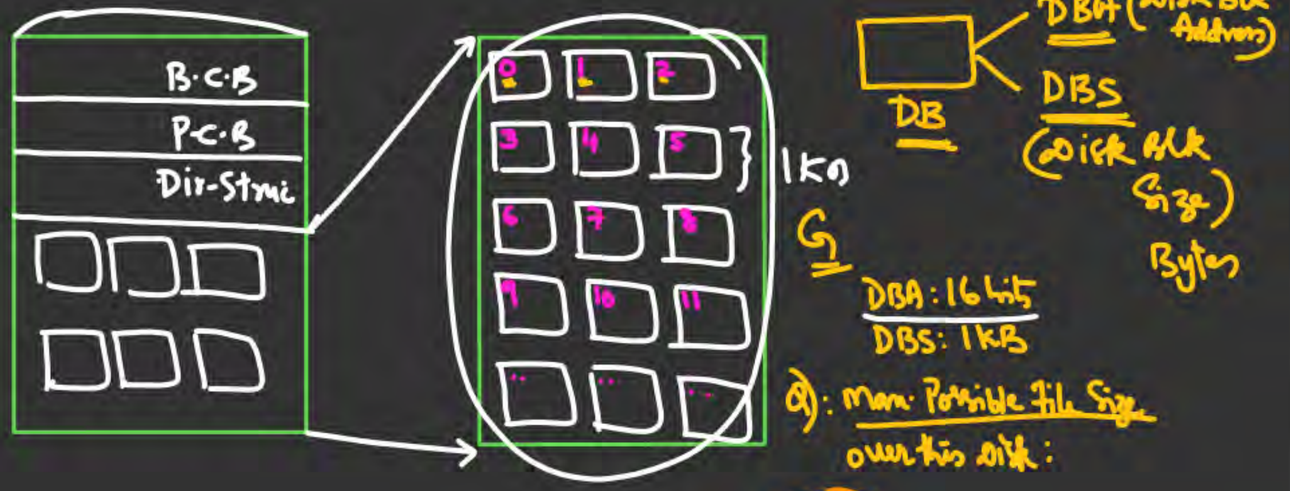
The basic file system needs only to issue generic commands to the appropriate device driver to read and write physical blocks on the disk. Each physical block is identified by its numeric disk address (for example, drive 1, cylinder 73, track 2, sector 10). This layer also manages the memory buffers and caches that hold various file-system, directory, and data blocks. A block

The file-organization module knows about files and their logical blocks, as well as physical blocks. By knowing the type of file allocation used and the location of the file, the file-organization module can translate logical block addresses to physical block addresses for the basic file system to transfer.

a translation is needed to locate each block. The file-organization module also includes the free-space manager, which tracks unallocated blocks and provides these blocks to the file-organization module when requested.

Finally, the logical file system manages metadata information. Metadata includes all of the file-system structure except the actual data (or contents of the files). The logical file system manages the directory structure to provide the file-organization module with the information the latter needs, given a symbolic file name. It maintains file structure via file-control blocks. A file-control block (FCB) (an inode in UNIX file systems) contains information about the file, including ownership, permissions, and location of the file contents. The

1. File Allocation Methods (Disk Space Alloc. Methods)



$64K = 2^{16}$

Actual disk size ≤ Max. Possible Disk Size (given)

DBA: 16 bits

Total # of Blks: $2^{16} = 64K$

DBS = 1KB

Total Blk Size = $64K * 1KB$

$2^{DBA} * DBS$

Max. Possible disk size = 64MB

Max. File Size ~ Max. disk size;

Max. Possible disk = $2^{DBA} * DBS$

File | Drive | OS | S/W ...

What ever is Stored on disk is Stored in the form of Block (unit of Alloc.)

Allocation Methods

(*)

- Contiguous Alloc. (CA)
- Linked Alloc. (NCA)
- Indexed Alloc.

directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Figure 12.5 Contiguous allocation of disk space.

1. Contiguous Allocation (Arrays)

Performance:

1) Int. Frag: ✓
2) External Frag: ✓
3) Increasing File Size: Inflexible
4) Type of Access: Seq ✓ | Random ✓
   - Direct
   - Faster

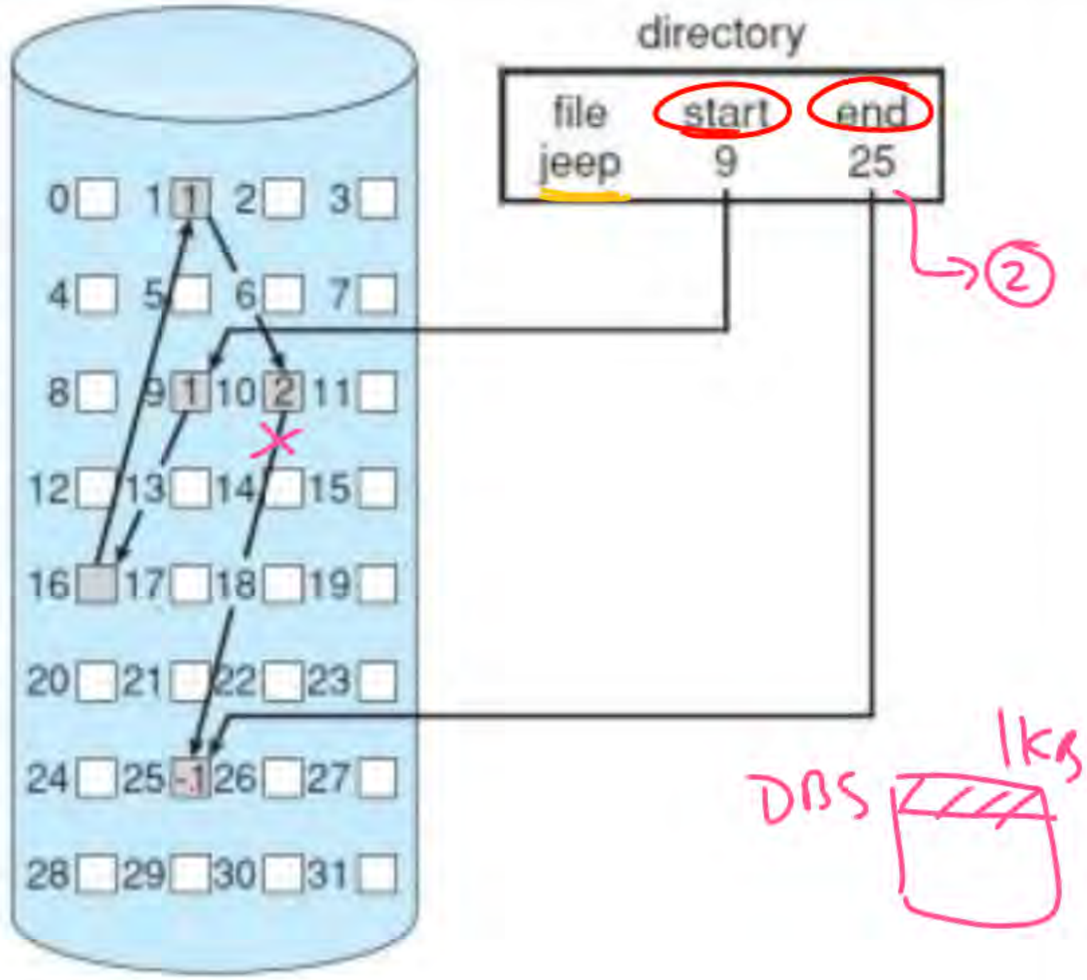


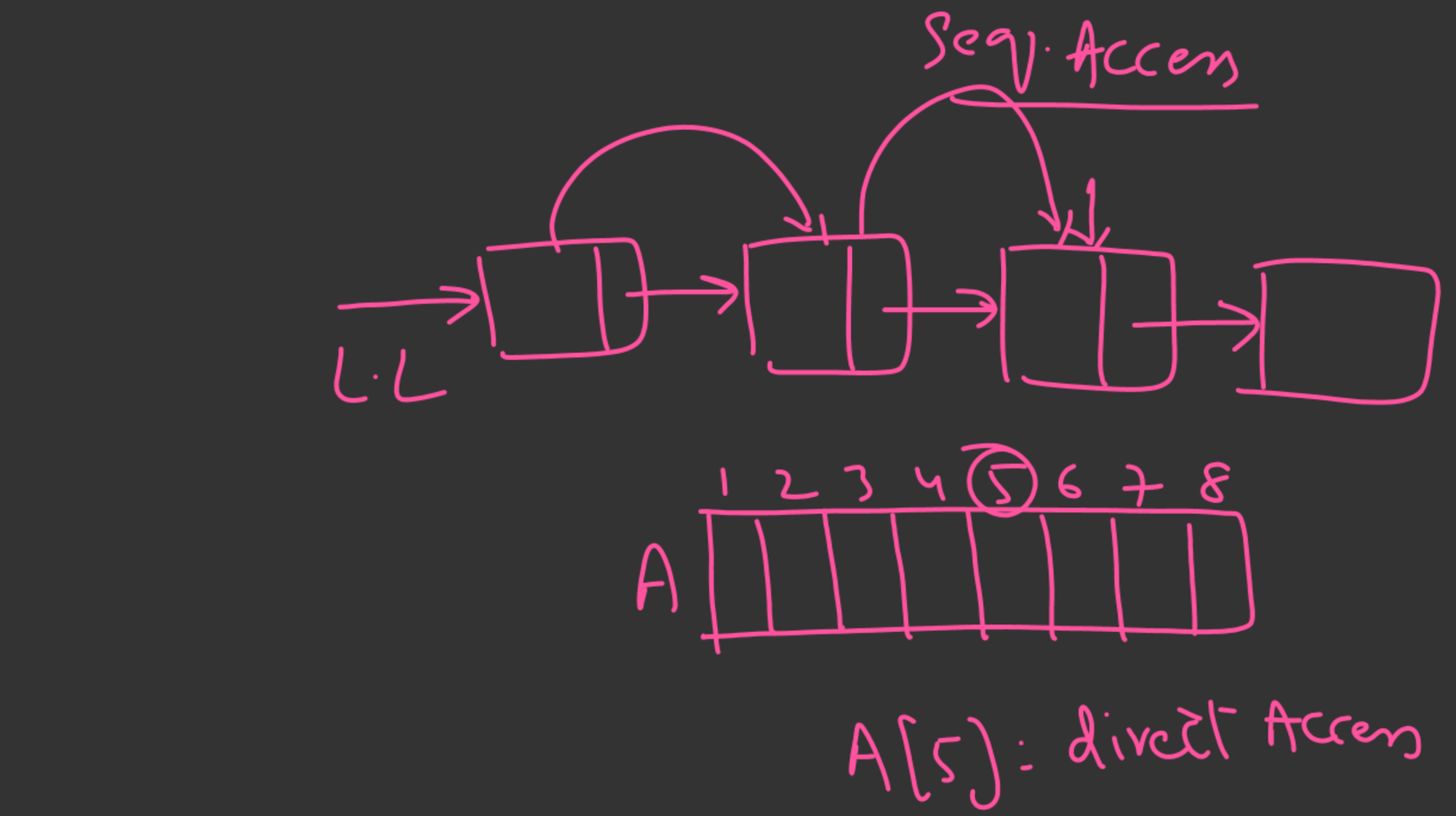
Figure 12.6 Linked allocation of disk space.

II. Linked Allocation

NCG: <A linked list is created among the Blocks>

Performance Issues:

1) Int. Frag: ✓ (happens in last block)
2) External Frag: X
3) Inc. File Size: Flexible
4) Type of access: only Seq. (Slow)
5) Ptrs consume disk space;
6) Vulnerability of ptr; (Truncated)



A[5] : direct Access

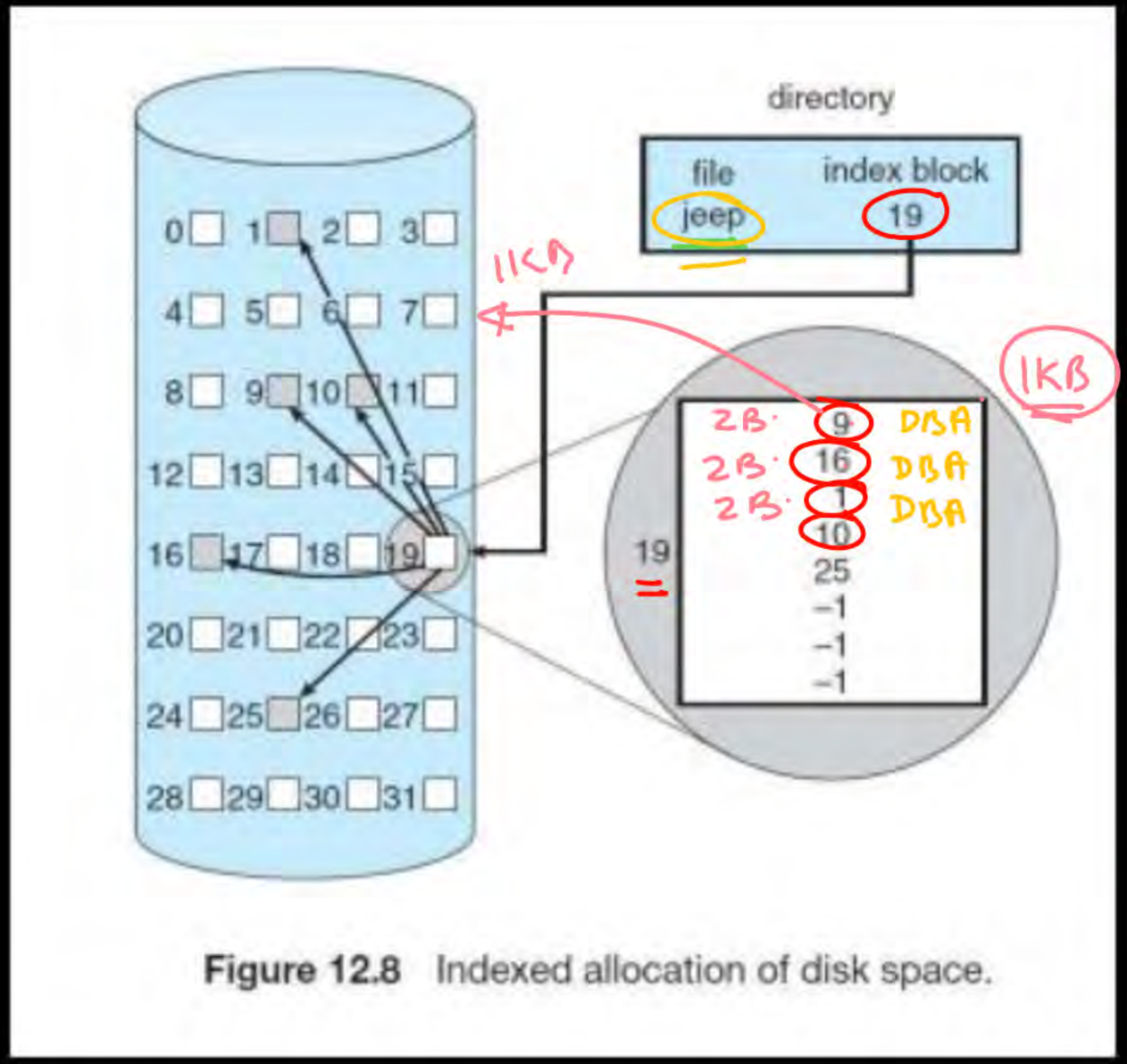


Figure 12.8 Indexed allocation of disk space.

(x) III. Indexed Alloc.

< CG + NCG >

: Each file is associated with an Index Block.

: Index Blk holds Addresses of data Blocks in use by the file;

→ DBA: 16 bits; DBS: 1KB

2B

Max. File Size (1-Index Blk) : No. of Addrs = 1KB / 2B = 512

Total FS = 512 * 1KB = 512KB

Case Study: UNIX | LINUX

I-Node Structure (**)

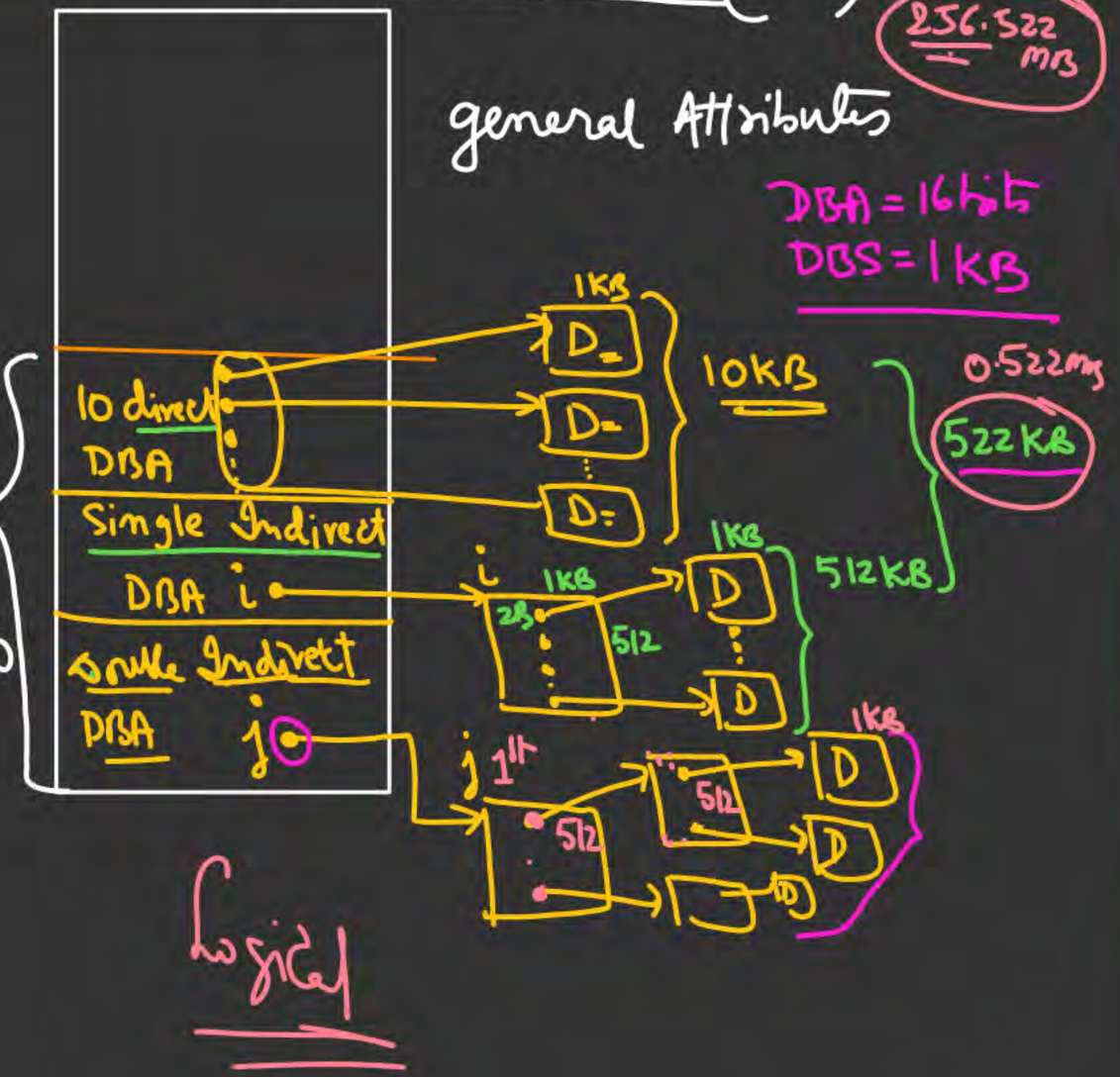
256.522 MB

general Attributes

DBA = 16 bits
DBS = 1 KB

File Name	I-Node NO
KRK	23
AKR	33

Directory Structure.



$2^{28} = 256MB$

$\frac{}{2^1 \times 2^1 \times 2^{10}}$

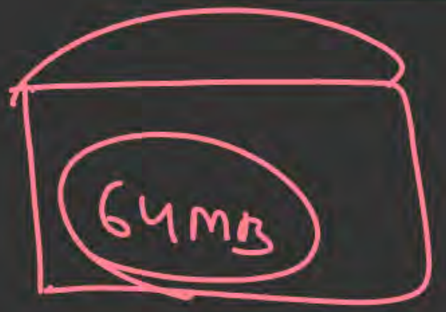
$512 * 512KB$

DBA = 16 bits ✓

DBS = 1KB ✓

Max. O.S : $2^{16} \times 1KB$

$= 64MB$.



Multi-Level Indexed

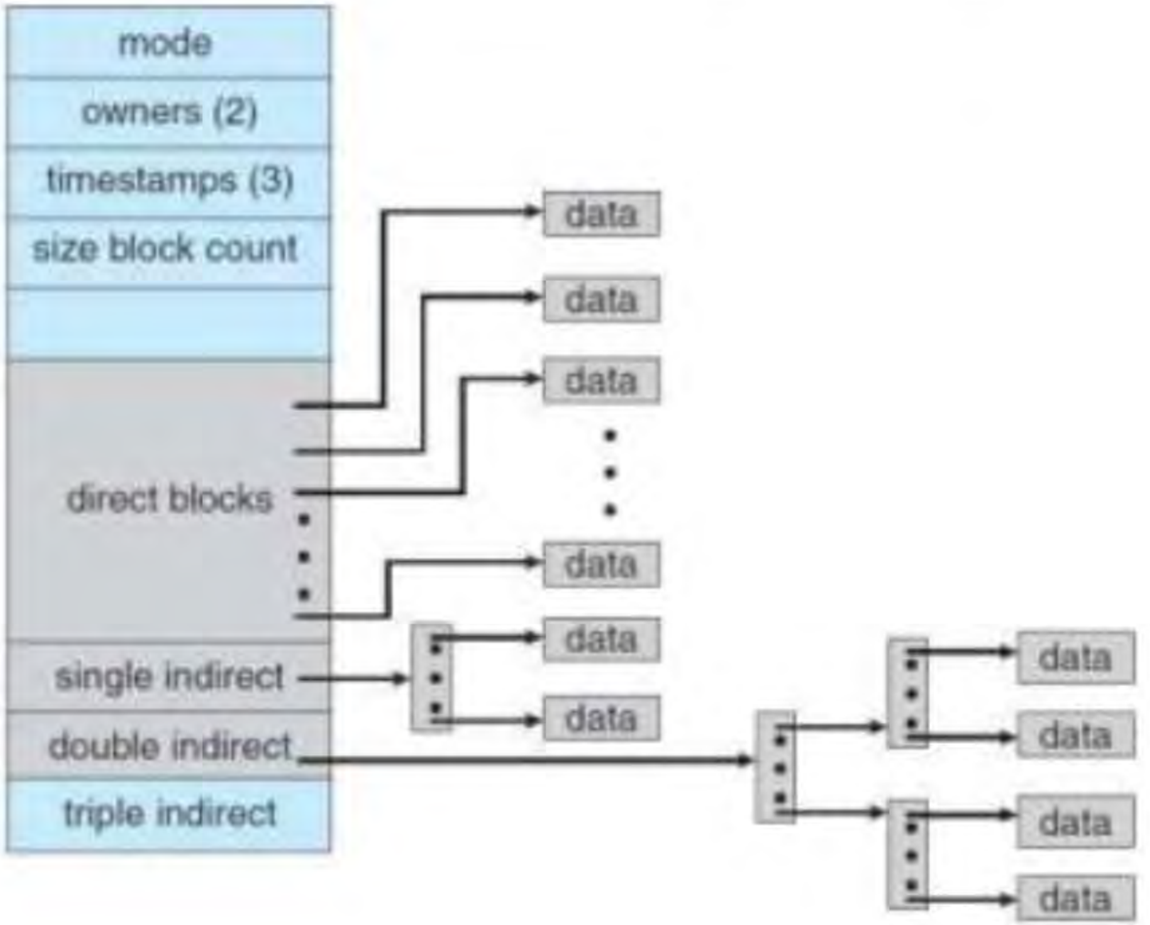
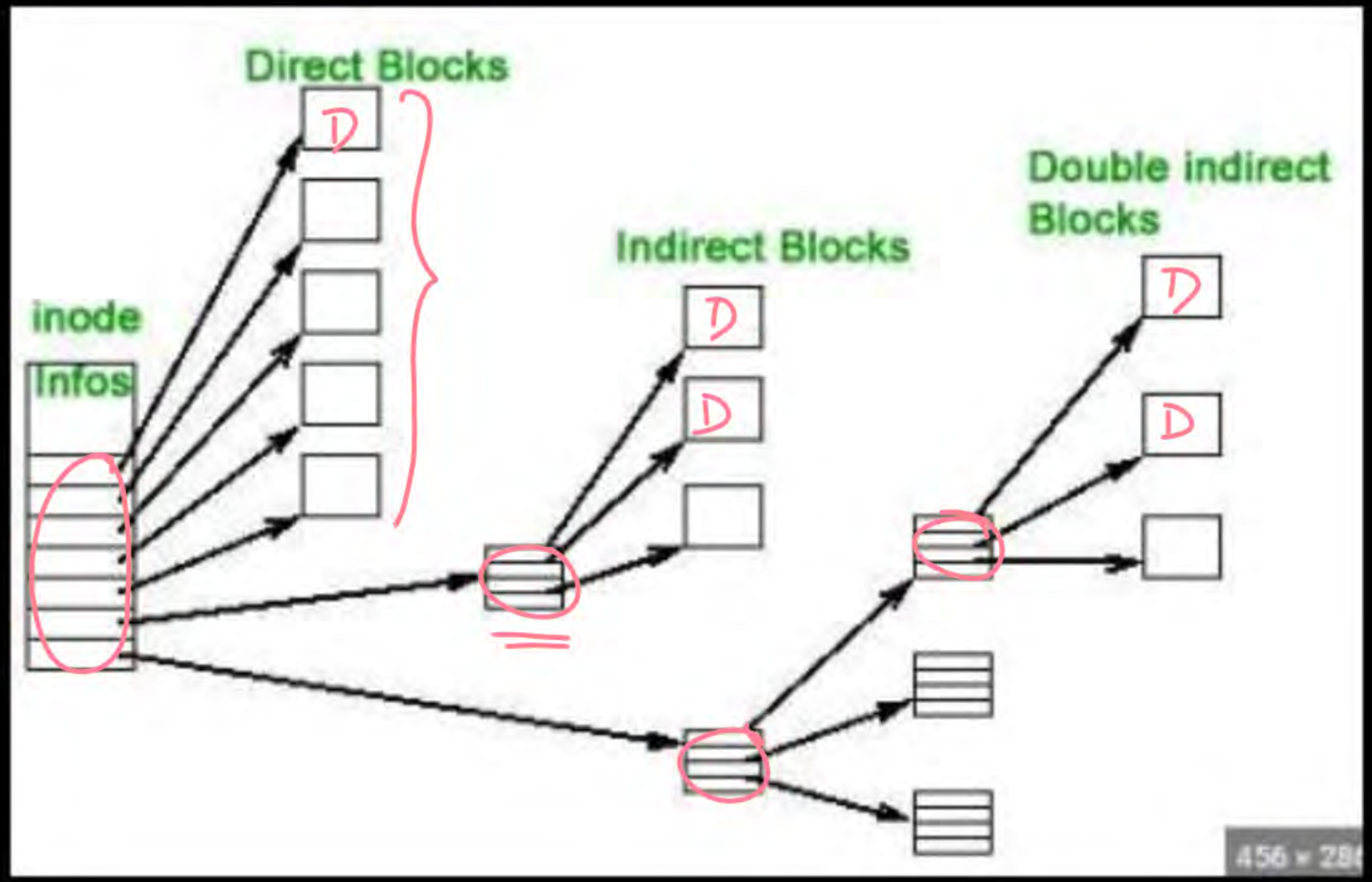


Figure 12.9 The UNIX inode.


Direct Blocks
inode
Infos
Indirect Blocks
Double indirect
Blocks
D
D
D
D
D
D

II: DOS/WINDOWS Impl

: Tabular linked Alloc.

Gen Attributes		
F.Name	. - - - - -	First DBA
KRK		x

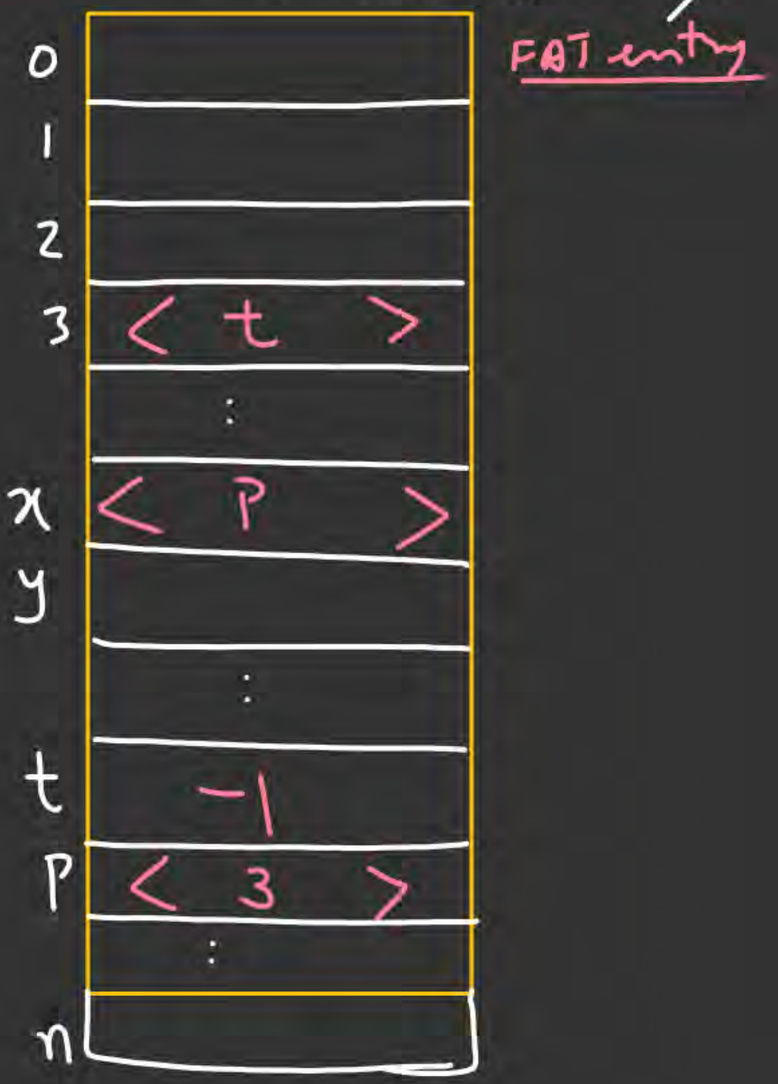
1000 . . 0 . . 999

F.A.T <M.F.T>

FAT entry

Dir. Structure

<x, P, 3, t>



→ FAT is orgniz as a set of entries

→ No. of entries in FAT = No. of Blocks of Disk

→ FAT entry Contains addrss (DBA) of next Disk (Data) Blk in use by File;

→ FAT entry Size ~ DBA

Q.6 In a file allocation system, which of the following allocation scheme(s) can be used if no external fragmentation is allowed?

I. Contiguous

II. Linked

III. Indexed

A. I and III only

B. II only

C. III only

D. II and III only

Q.11 The Data Blocks of a very large file in the Unix File System are allocated using

A. Contiguous allocation

B. Linked allocation

C. Indexed allocation

D. An extension of indexed allocation. ✓

M.L

Q.12 Using a Larger Block size in a Fixed Block Size File System leads to

11:45 am

A. Better Disk Throughput but Poorer Disk Space Utilization.

B. Better Disk Throughput and Better Disk Space Utilization

C. Poorer Disk Throughput but Better Disk Space Utilization

D. Poorer Disk Throughput and Poorer Disk Space Utilization

THANK YOU!