CS & IT ENGINEERING

Operating System

File System & Device Management

Lecture No. 2

By- Dr. Khaleel Khan Sir

- Disk Logical Structure
- File System Interface
- Problem Solving

Q. 2

174Q

Consider a Disk with the following Specifications:

- Number of surfaces = 64
- Outer diameter = 16 cm
- Inner diameter = 4 cm
- Inter Track space = 0.1 mm ✓
- Max Density = 8000 bits/cm

Calculate the Unformatted Capacity of Disk.

Disk Cap = 64 × 600 × 12.56 KB
= 64 × 6 × 1256 KB
= 64 × 6 × 1.256 MB
= 482.3 MB = 0.48 GB

No. of Tracks: $\frac{6\text{ cm}}{0.1\text{ mm}} = \frac{60}{0.1} = 600$

8000 bits — 1 cm
? — 12.56 cm

Track size = $\frac{8000 \times 12.56 \text{ bits}}{8}$
= 1000 × 12.56 B
= 12560 B
(12.56 KB)

Inner Track size = $2\pi r$
= $\pi \cdot d$
= 3.14 × 4 cm
= 12.56 cm

Surf Cap = 600 × 12.56 KB

Q. 3

How long does it take to load a 64 Kbytes Program from a disk whose Average Seek time is 30 ms Rotation time is 20 ms, Track Size is 32 Kbytes, Page Size is 4 Kbytes. Assume that Pages of the Program are distributed randomly around the disk. What will be the % saving in time if 50% of the Pages of program are Contiguous?

ST = 30ms; R = 20ms;
TS = 32KB
PS = 4KB

$N = \frac{64KB}{4KB}$

N = 16

a) Time to Load Program = (Time to Load a Page) * N

Time to Load a Page = S.T + L.T + T.T
= (30ms + 10ms + 2.5ms)
= 42.5ms

Prog-loading Time = 42.5 * 16 = 680ms = 0.68s

32KB – 20ms
4KB – ?

$\frac{4KB}{32KB} * 20$

$\frac{2.5}{8}$

When Pages are CG then only one Seek & one Latency

32KB; P.S=4KB

$$\therefore \frac{\left(30ms + 10ms + \frac{2.5 \times 8}{CG}\right) + 8 \times 42.5ms}{NCG}$$

$$\therefore 60ms + 340.0$$

$$\therefore (400ms) = 0.4s$$

(i) 0.68 s

(ii) 0.4 s

$= 0.28s$

$\therefore \text{Saving} = \left(\frac{0.28}{0.68}\right)$

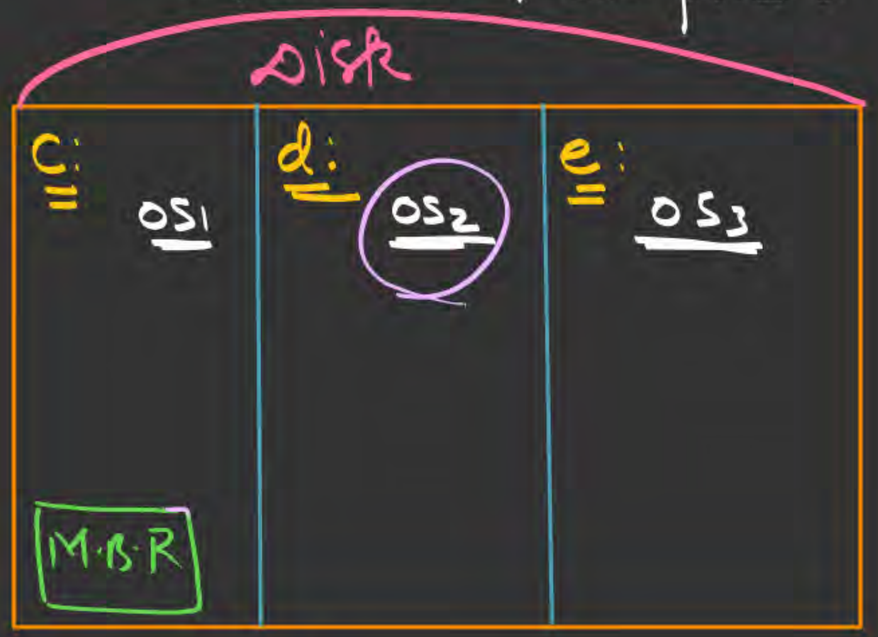
Q. 4

H/W

An Application requires 100 libraries at startup. Each library requires 1 disk access. Seek Time is 10 ms, Disk RPM is 6000. All 100 libraries are at random locations. 50% of Libraries requires transfer time of ½ Rotation, while for the remaining 50% it is negligible. How long does it take to load all 100 libraries?

II. Logical Structure of Disk [Formatting Process]

Multi-Boot Computer



MBR (Master Boot record)

- Partition Table
- Boot Loader

(Placing Infrastructure onto Disk)

(Ram Disk)

→ Effective Storage & Faster retrieval

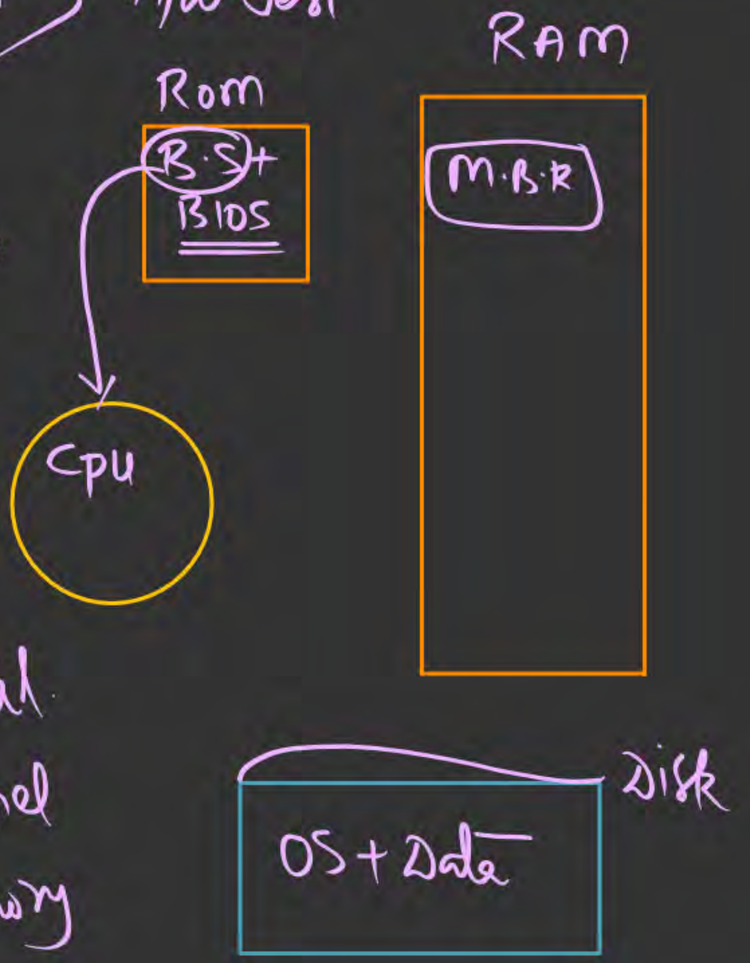
Disk

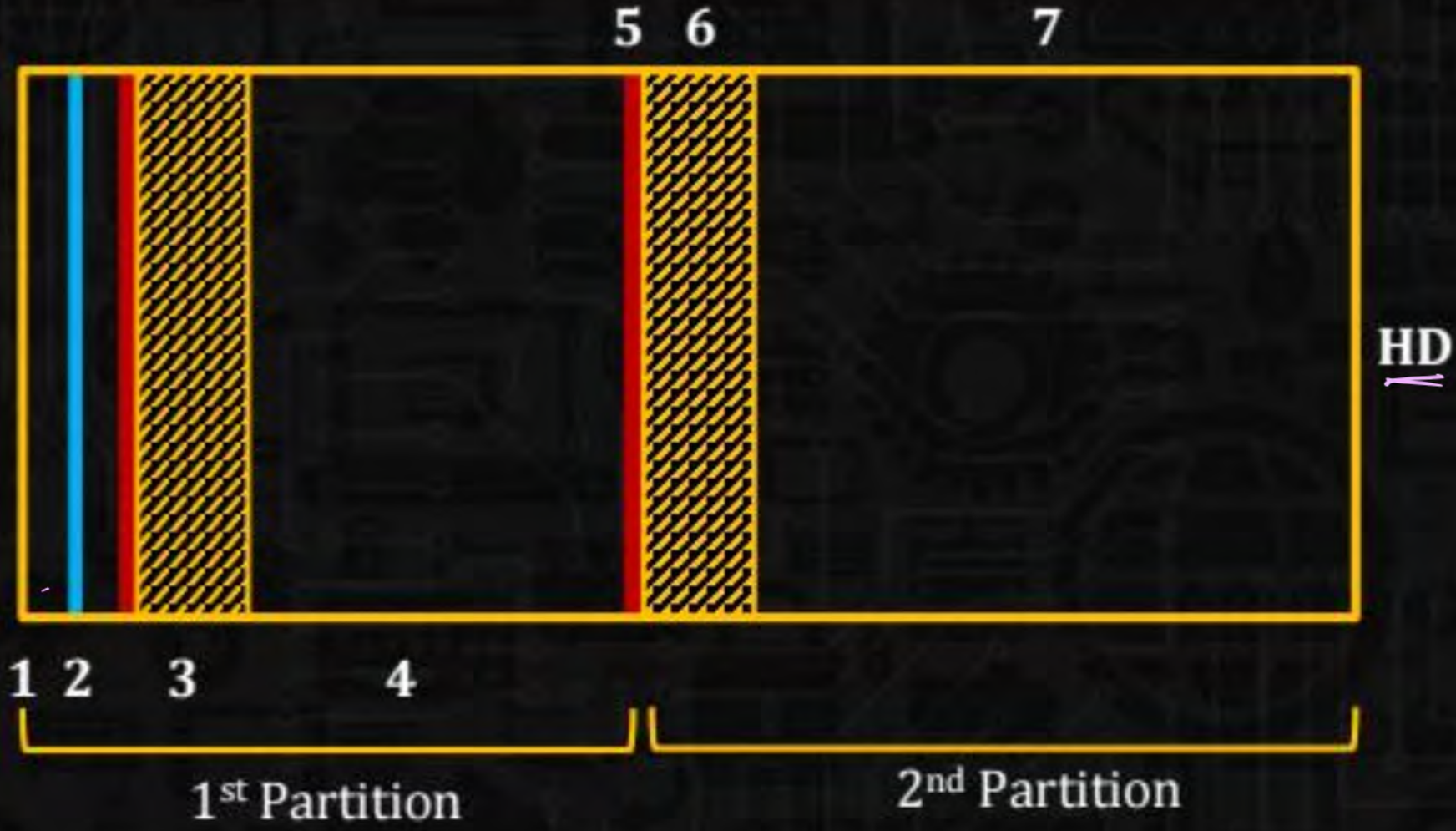
→ divided into Partitions (volumes / Minidisk)

- Primary (OS) (Bootable) + Data + s/w
- Extended / Logical drives (Non-Bootable) : Data + s/w

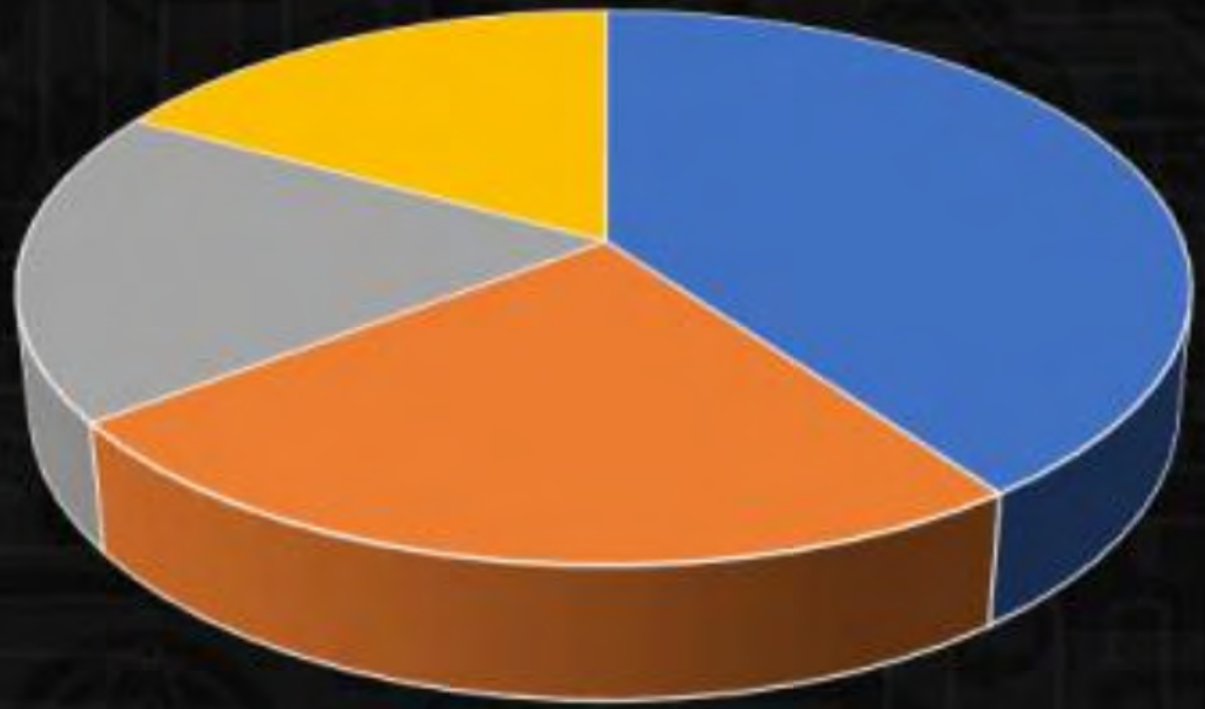
Steps in Booting Process

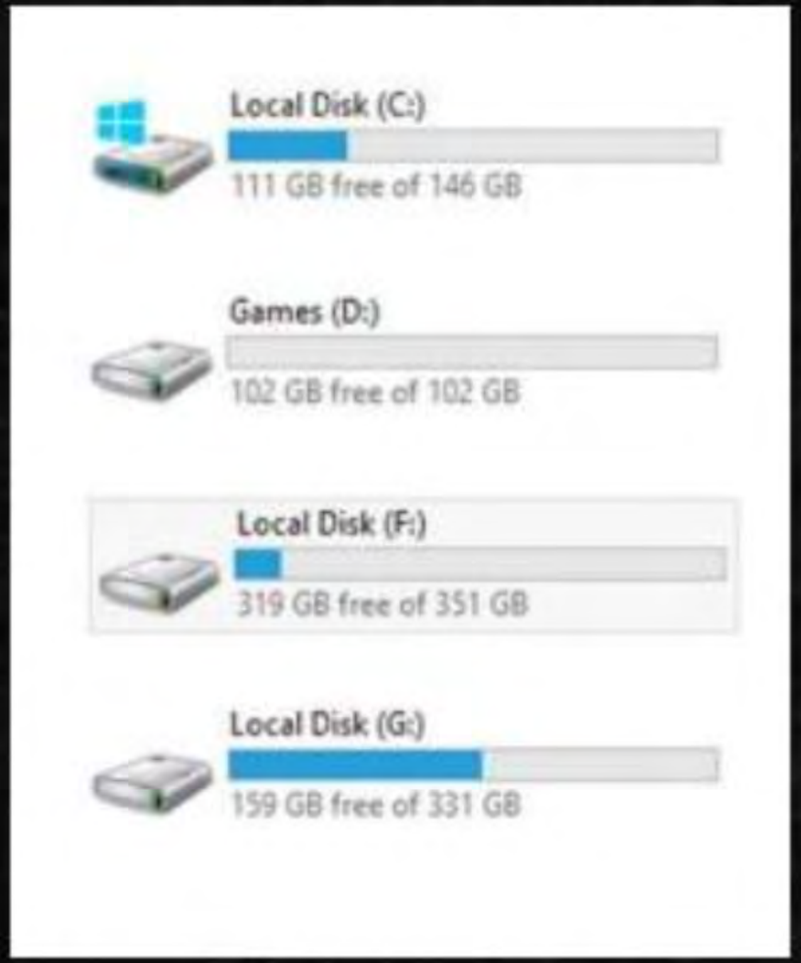
1. Switch on
2. POST <Power on Self - Test> H/w Test
3. BIOS (Basic IO System)
4. Bootstrap loads MBR into RAM and hands over the control to Boot loader
5. Boot loader reads Partition Table & display options if Avail.
6. Boot Loader will load the Kernel Program from the disk into memory

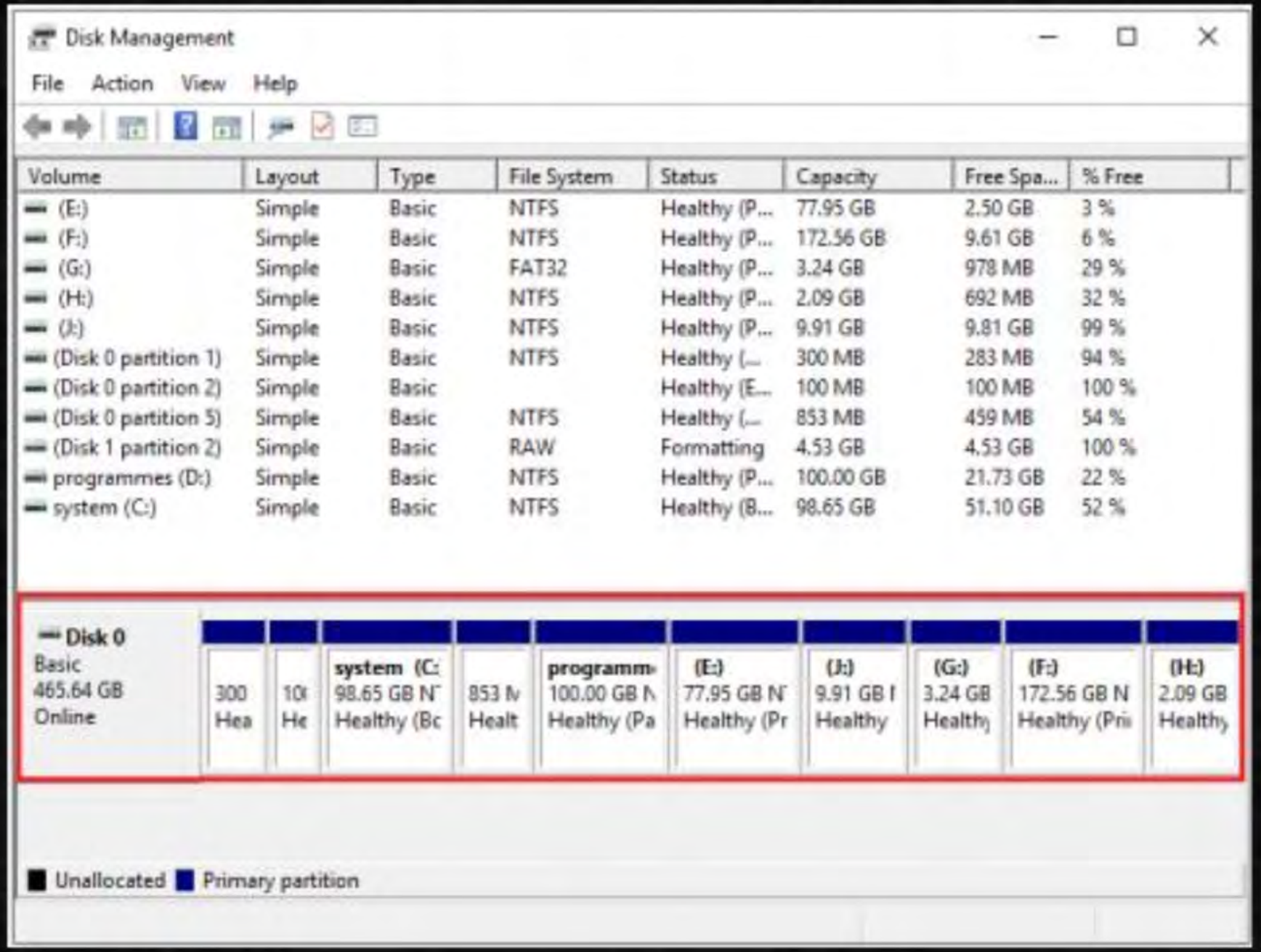




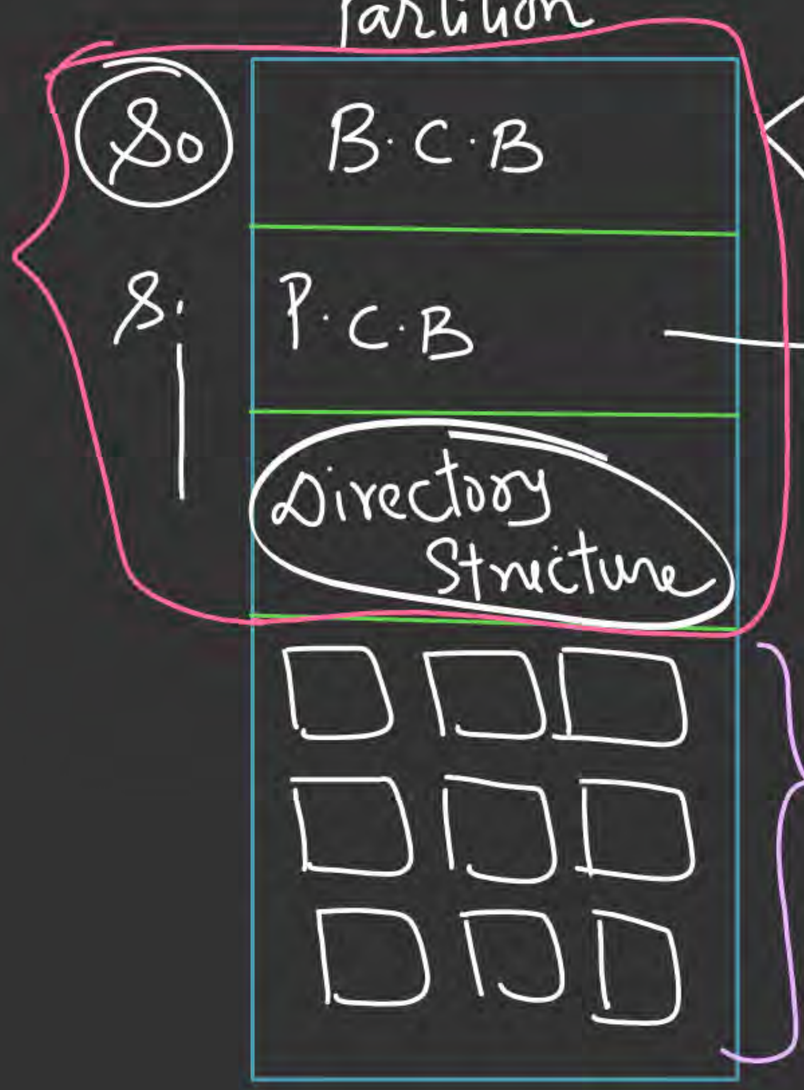
1- MBR
2 - First Partition Boot Sector
3 - File system Area of 1st Partition
4 - Data Area of 1st Partition




Local Disk (C:)
111 GB free of 146 GB
Games (D:)
102 GB free of 102 GB
Local Disk (F:)
319 GB free of 351 GB
Local Disk (G:)
159 GB free of 331 GB


Disk Management
File Action View Help
Volume Layout Type File System Status Capacity Free Spa... % Free
(E:) Simple Basic NTFS Healthy (P... 77.95 GB 2.50 GB 3 %
(F:) Simple Basic NTFS Healthy (P... 172.56 GB 9.61 GB 6 %
(G:) Simple Basic FAT32 Healthy (P... 3.24 GB 978 MB 29 %
(H:) Simple Basic NTFS Healthy (P... 2.09 GB 692 MB 32 %
(J:) Simple Basic NTFS Healthy (P... 9.91 GB 9.81 GB 99 %
(Disk 0 partition 1) Simple Basic NTFS Healthy (... 300 MB 283 MB 94 %
(Disk 0 partition 2) Simple Basic Healthy (E... 100 MB 100 MB 100 %
(Disk 0 partition 5) Simple Basic NTFS Healthy (... 853 MB 459 MB 54 %
(Disk 1 partition 2) Simple Basic RAW Formatting 4.53 GB 4.53 GB 100 %
programmes (D:) Simple Basic NTFS Healthy (P... 100.00 GB 21.73 GB 22 %
system (C:) Simple Basic NTFS Healthy (B... 98.65 GB 51.10 GB 52 %
Disk 0
Basic
465.64 GB
Online
300
Hea
10
He
system (C:
98.65 GB N
Healthy (Bc
853 M
Healt
programm
100.00 GB N
Healthy (Pa
(E:)
77.95 GB N
Healthy (Pr
(J:)
9.91 GB I
Healthy
(G:)
3.24 GB
Healthy
(F:)
172.56 GB N
Healthy (Pri
(H:)
2.09 GB
Healthy
Unallocated
Primary partition

* Partition Structure [What D.S (Infrastructure) are embossed on the Disk/Partition after Formatting]



<Boot Control Block> → First OS Program (init/kernel)

- Boot Block: UNIX/LINUX
- Partition Boot Sector: NTFS/FAT32

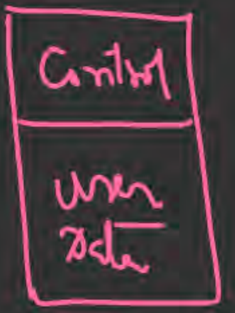
<Partition Control Block>

- SuperBlock: UNIX/LINUX
- Master File Table: NTFS/FAT32

F.A.T

Data Blks

<Contain Data of applications + other OS files..>



Files vs Directory

File: is a collection of Logically related records of entity;

↳ is an A.D.T <defn; Repr / Structure ; operations; Attributes>

opn's

→ (create; open; Read, write, Truncate, Seek close, copy, mov, Rename, delete)

Repr / Structure

- Flat (Series of Bytes)
- Record (Series of records)
  - Hierarchical Tree <B+; B-Trees>

Attributes → Name, ext type, owner Size; Date & Time, Permissions, mode, (Blocks in use)

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 (in bytes, words, of 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 stored/kept in File Control Block; (FCB)

UNIX / LINUX (I-Node)

WINDOWS (Directory entry)

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. Second, an entry for the new file must be made in the director).

- **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.

Seek

- 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 director)' 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 l0ts the file be reset to length zero and its file space released.

File Types – Name, Extension

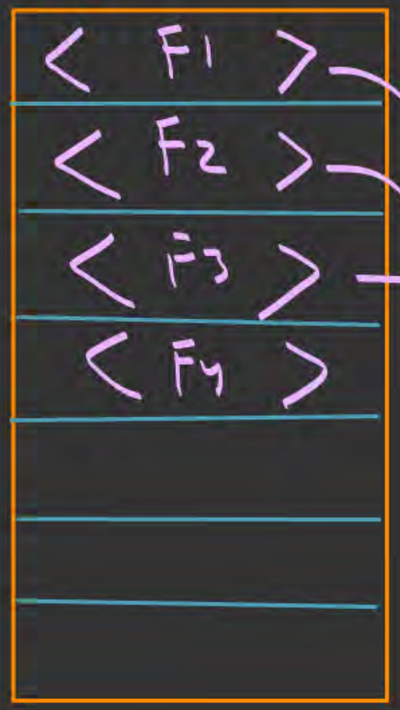
File type	Usual extension	Function
Executable	exe, com, bin or none	Ready-to-run machine-language program
Object	obj, o	Compiled, machine language, no linked
Source code	c, cc, java, pas, asm, a	Source code in various languages
Batch	bat, sh	Commands to the command interpreter
Text	txt, doc	Textual data, documents
Word processor	wp, tex, rtf, doc	Various word processor formats
Library	lib, a, so, dll	Libraries of routines for programmers
Print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
Archive	arc, zip, tar	Related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	Mpeg, mov, rm, mp3, avi	Binary file containing audio or A/V information

Directory:

: is a Collection of Files:

: is also a Special File (Data about other Files) (Meta Data)

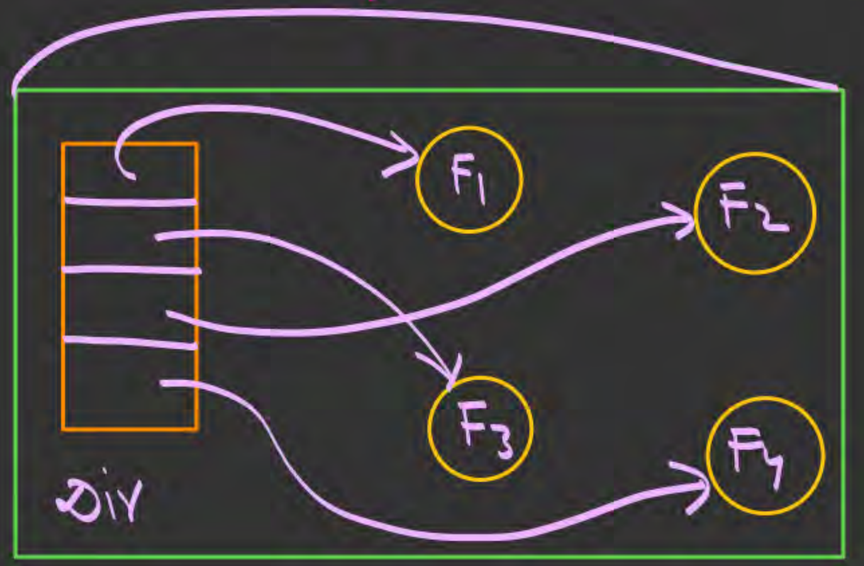
Abstract Directory



(linear 1-D array)

Directory entry

< Dir contains Meta Data of Files >



A single directory for all users

Directory Structures

- No Sub directory

→ Simple to Implement

→ Search Time

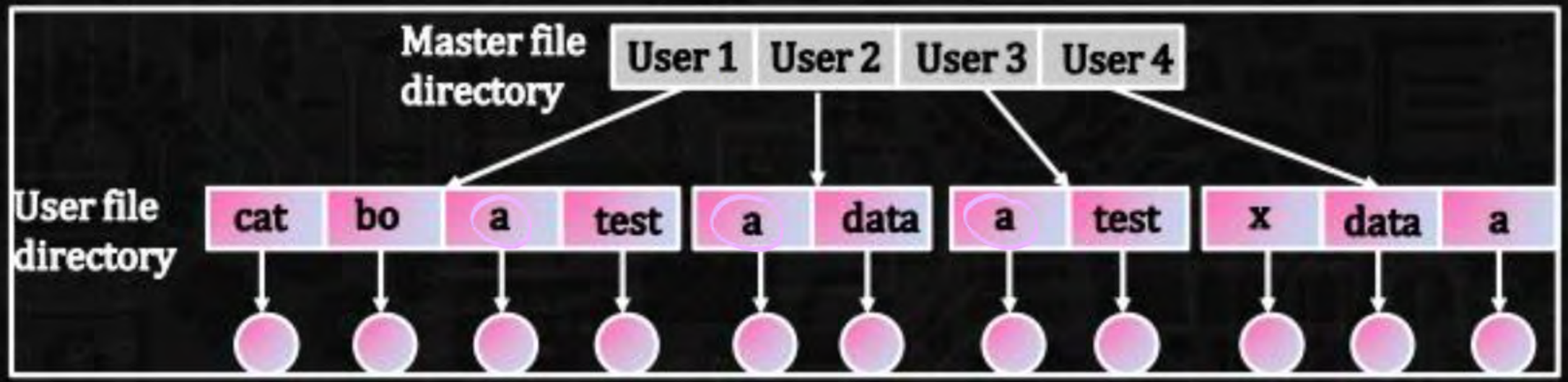
→ Name conflict

Naming problem

Grouping problem

Two-Level Directory

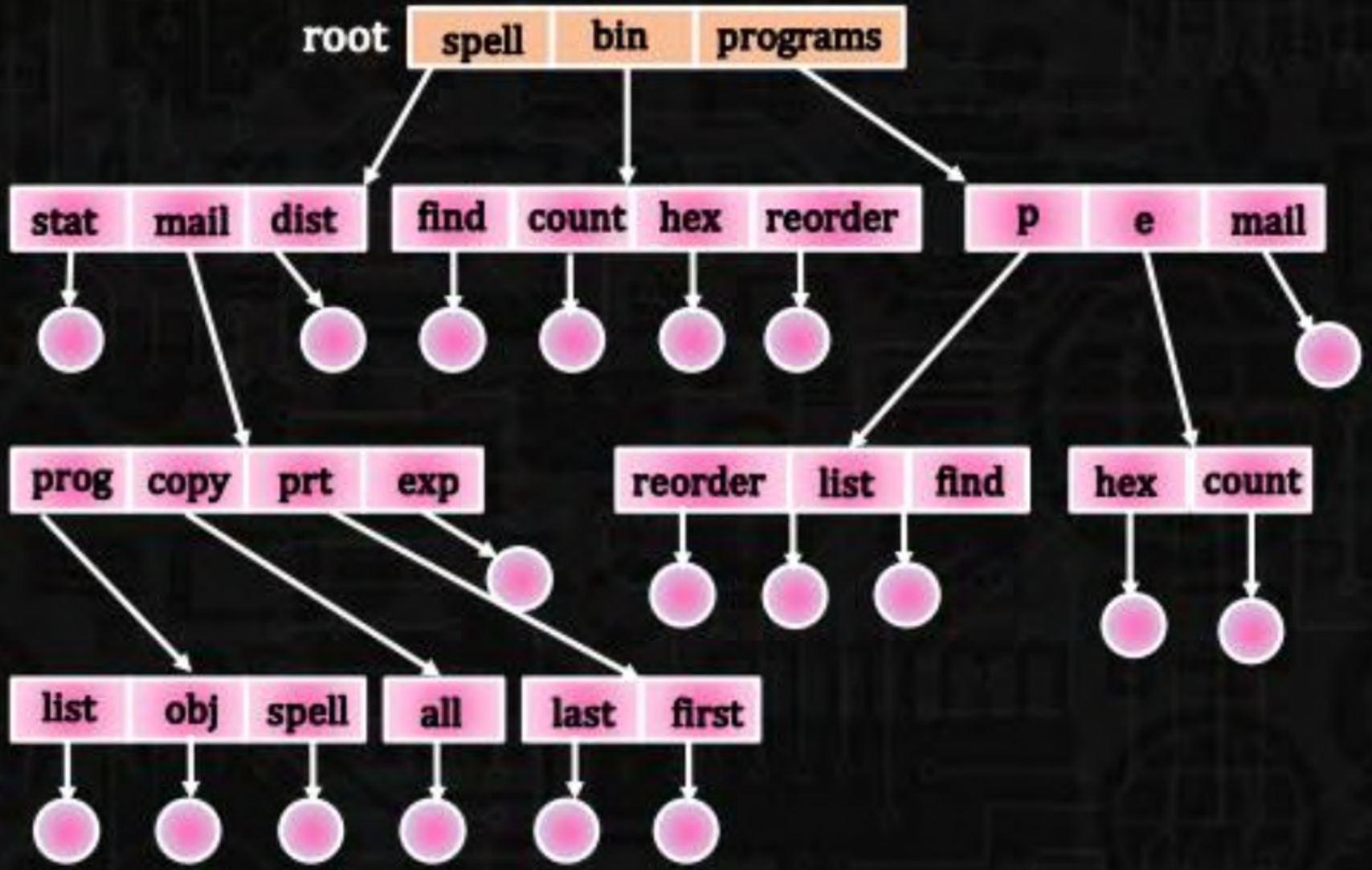
- Separate directory for each user



- Path name
- Can have the same file name for different user
- Efficient searching
- No grouping capability

Tree-Structured Directories

Multi - Level

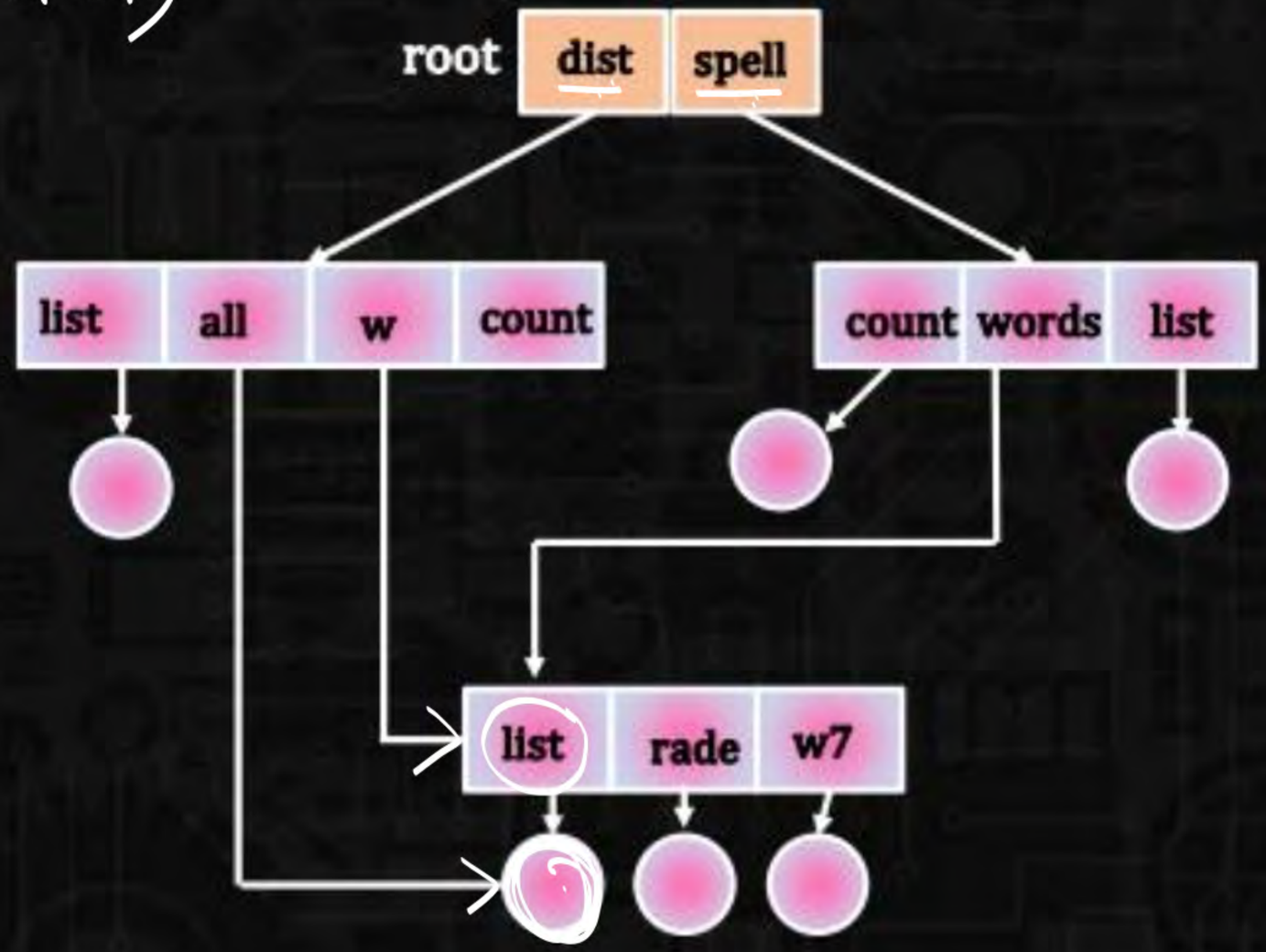


Acyclic-Graph Directories

(D.A.G)

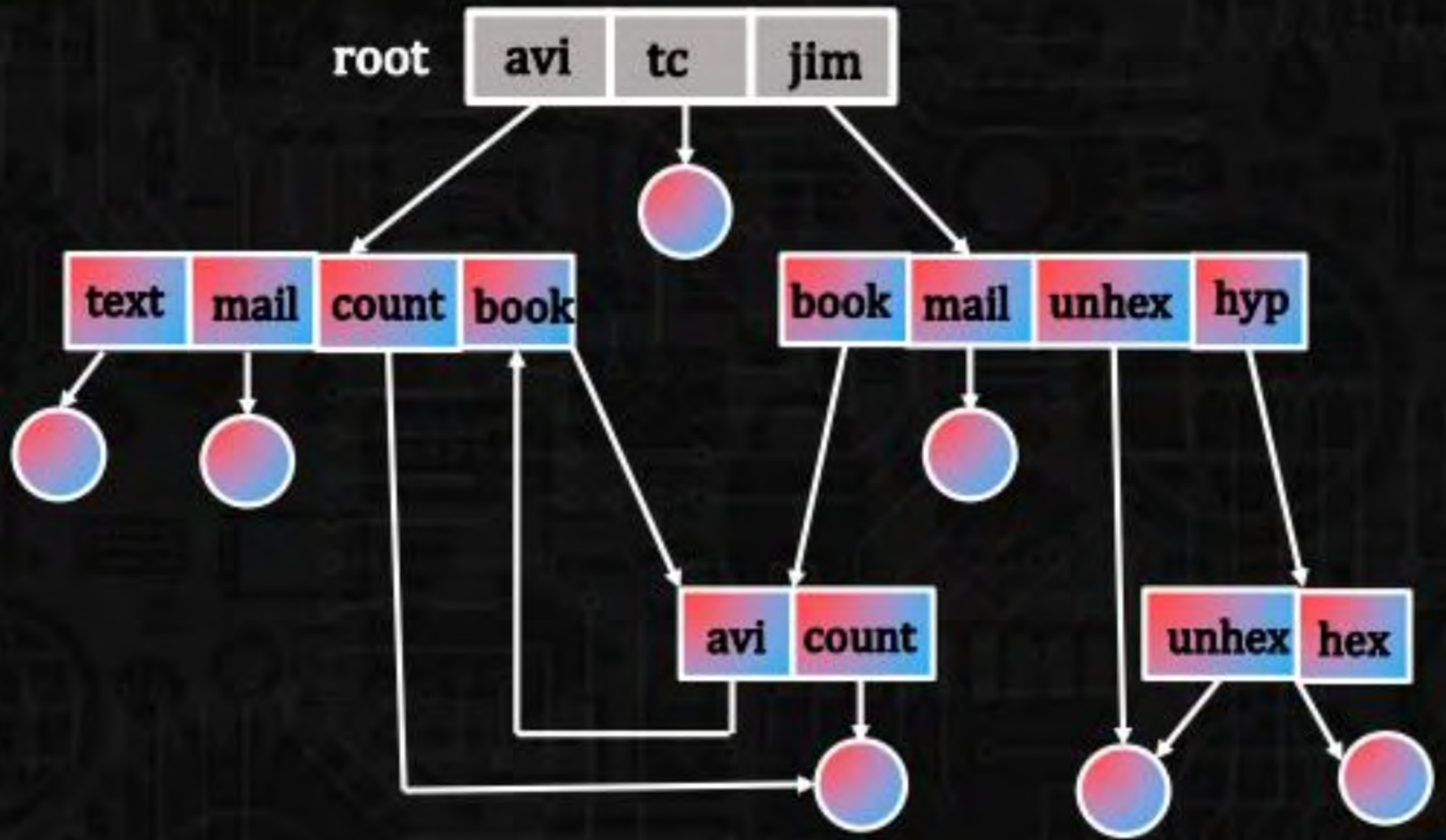
File Sharing:

Link Count = 2



General Graph Directory

: (cycles)



To Search a file we may have to Traverse the directory, for that we may use Graph Traversals like
- → DFS
- → BFS

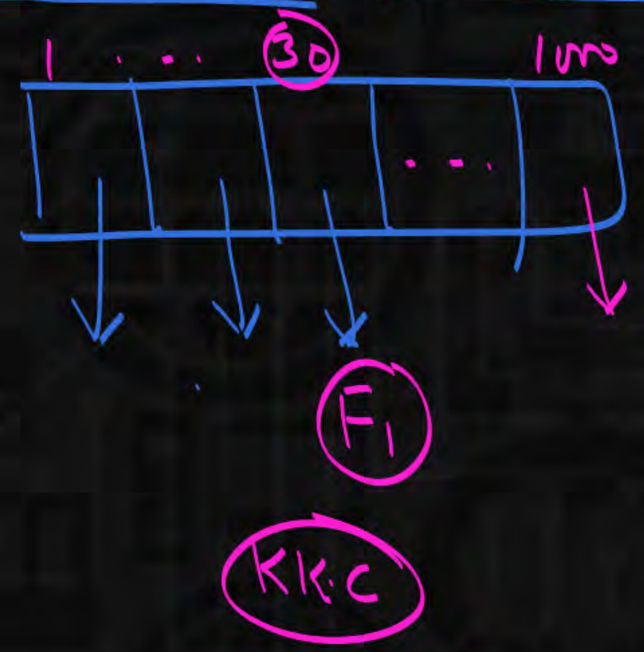
Dir. as a file also Support ops's, open, Read, write ...

but there is a spl. op on directory called Traverse

(MSQ) <B C>

Q. 5

Consider a linear list based directory implementation in a file system. Each directory is a list of nodes, where each node contains the file name along with the file metadata, such as the list of pointers to the data blocks. Consider a given directory foo. Which of the following operations will necessarily require a full scan of foo for successful completion?



A. Opening of an existing file in foo ✗

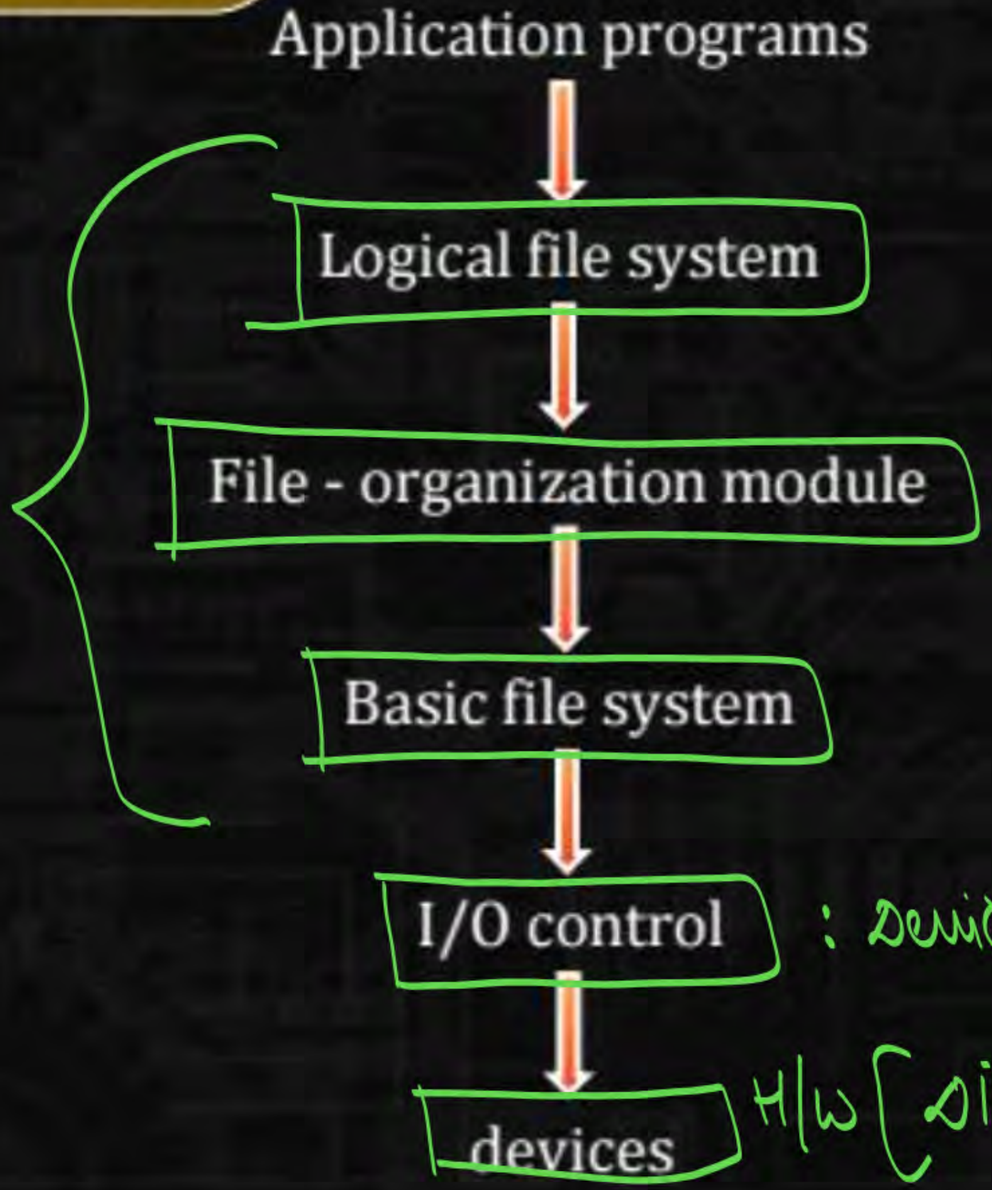
B. Creation of a new file in foo ✓

C. Renaming of an existing file in foo ✓

D. Deletion of an existing file from foo ✗

1. Physical Geometry of Disk
2. Logical Geometry of Disk (formatting)
3. Directory Structure
4. F.C.B ;

Layered File System



File System S/W is Implemented as a Layered S/W

I/O control : Device driver + Interrupt Handlers

devices H/W [Disk] + Controller Cards

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.

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.
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.

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 anode in UNIX file systems) contains information about the file, including ownership, permissions, and location of the file contents.

User

Logical 7.5 : user-level

File orgnz Module

(0.5) Basic File System


THANK YOU GW
SOLDIERS !