# CS & IT ENGINEERING

## Operating System

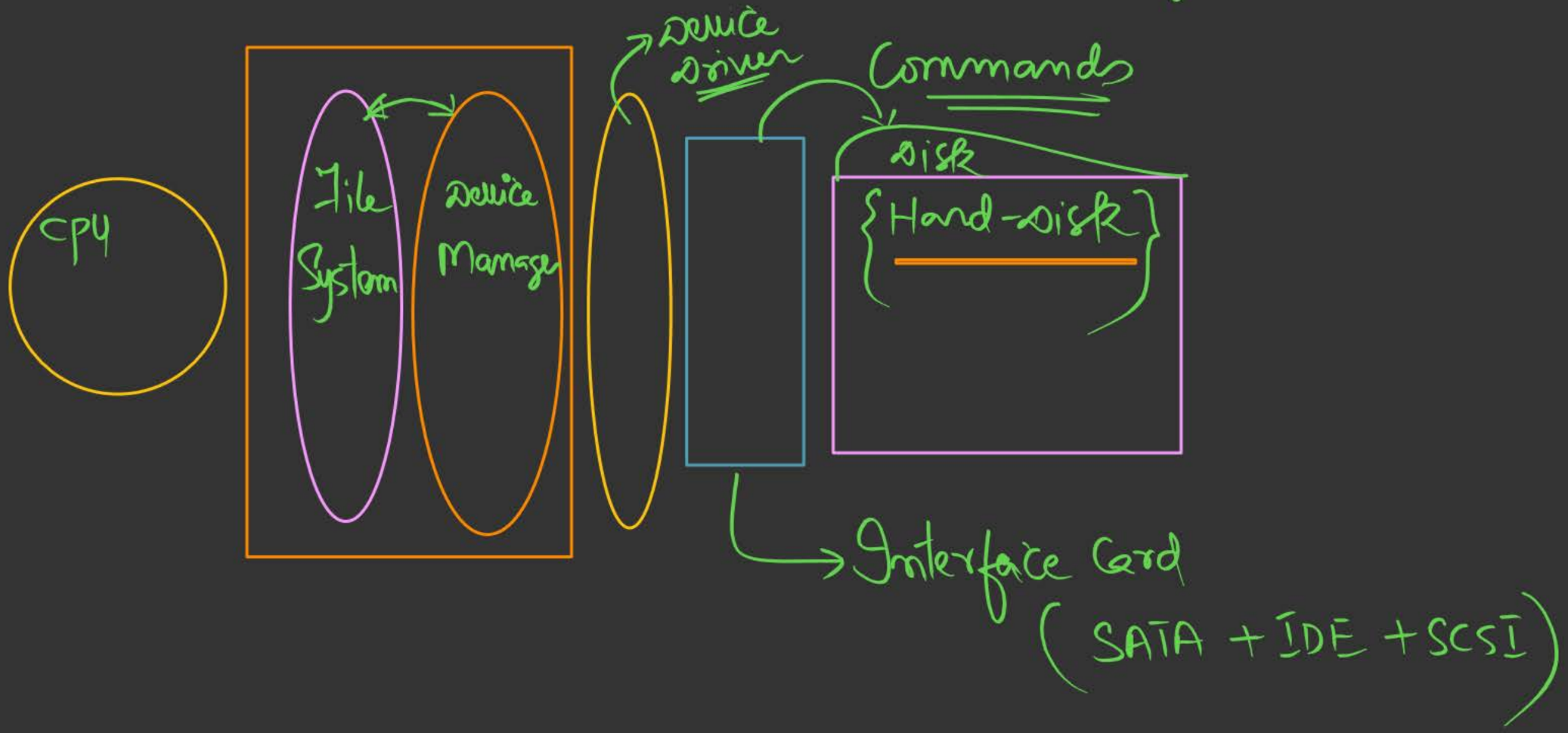
## File System & Device Management

**Lecture No.** 04

By- Dr. Khaleel Khan Sir


- Introduction to File System
- Physical Structure of Disk
- Problem Solving

# FILE SYSTEM : is the visible part of O.S



# Physical Geometry of Disk

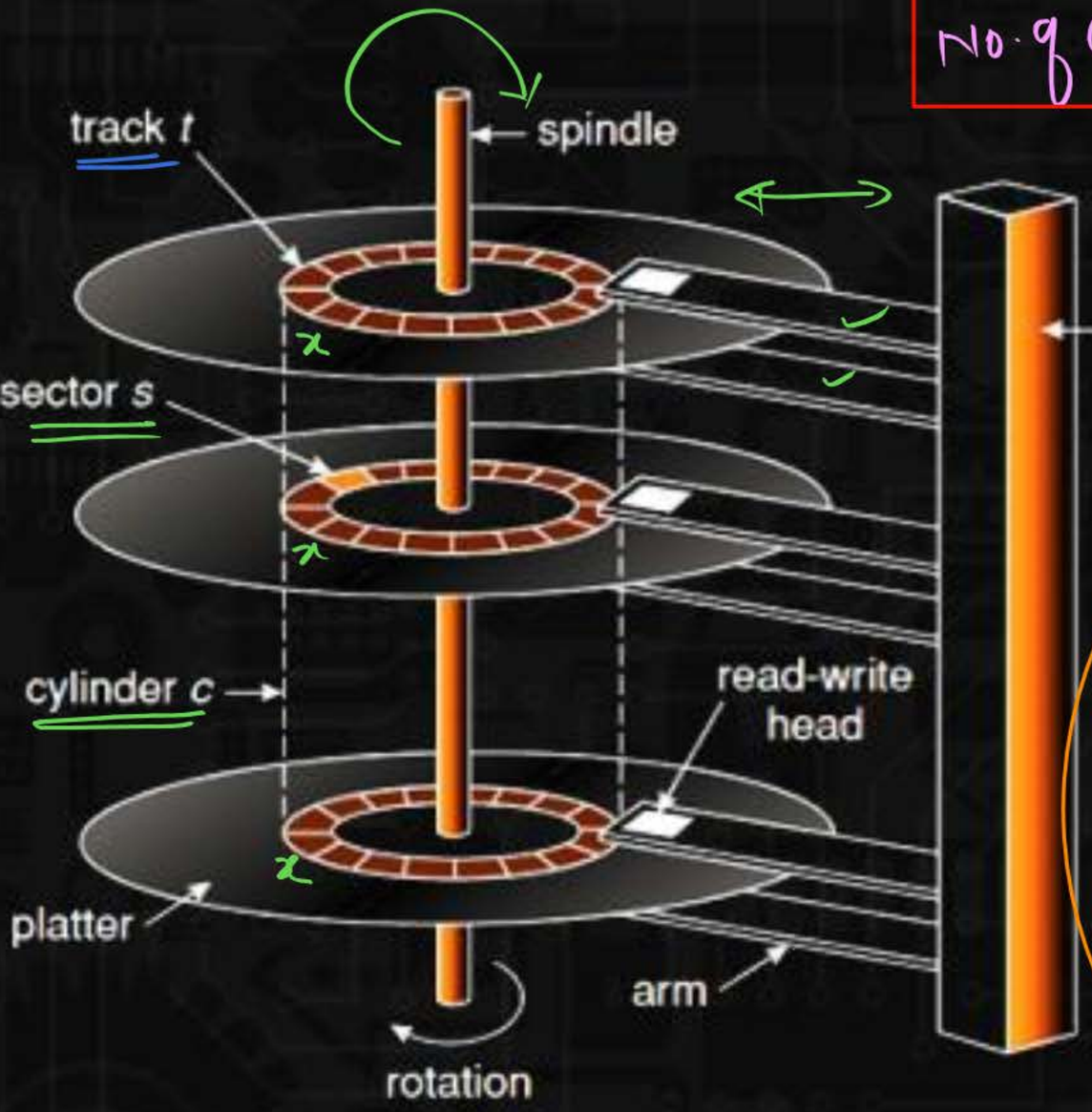

Computer hard drive
spindle
disk
saved file
actuator
read/write head
arm
power port
data cable port
drive configuration port
circuit board

# Block-diagram

No. of cylinders = No. of Tracks

Tracks

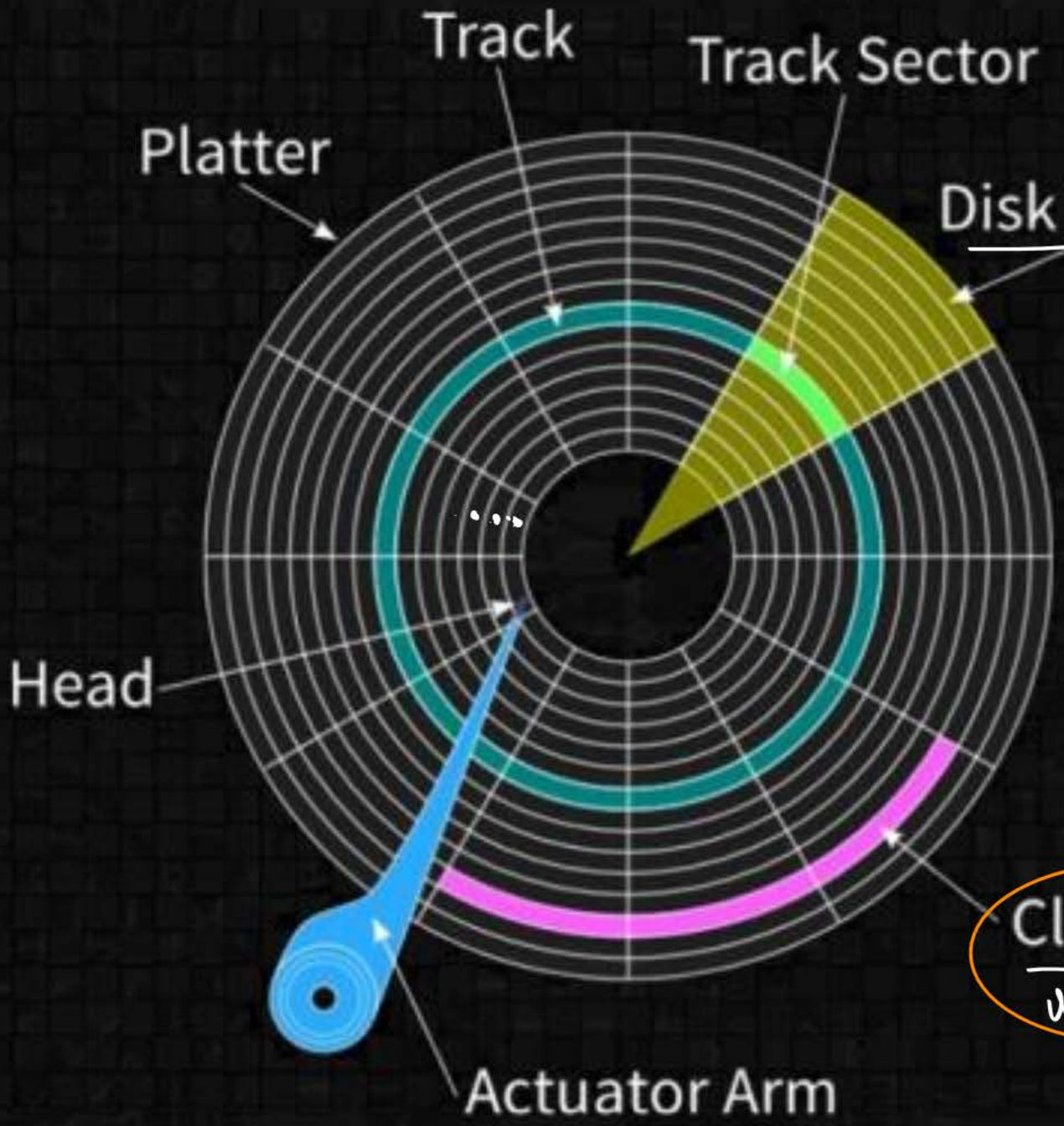
Surface





Sec-No of Sector

Disk
↓
Platters
↓
2 surfaces
↓
Tracks
↓
Sectors (Blocks)
(Unit of Trans fer)


Track
Track Sector
Platter
Disk Sector
Head
Actuator Arm
Cluster
is a group of one/more adj. Sectors
unit of Transfer

## Disk IO Time to read/write a Sector

: Seek Time (ST) + Rotational Latency Time (L.T) + Transfer Time (TT)

: $|x-y| * TTT$ : $\frac{R}{2}$

TTT = Track-Track Time

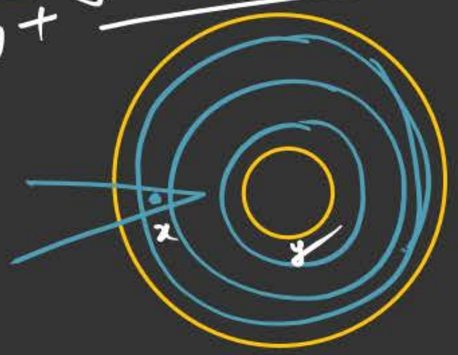
R = Time for one Rotation

(rpm) : 3600

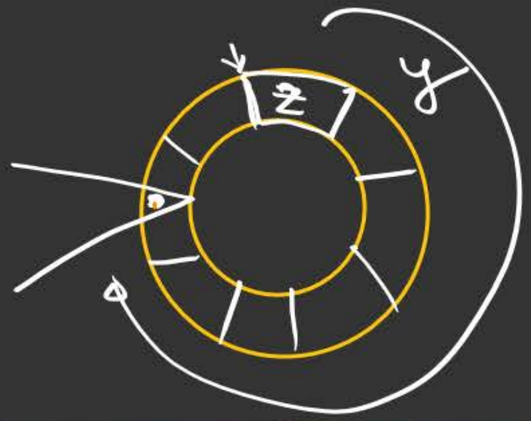
3600 r – 60 s

1 r – ?

$R = \frac{60}{3600} s = 16.6\ ms$








Track Size = 'Z' Bytes

Sector Size = 'K' Bytes

Rot. Time = R ms

$R_{ms} \longrightarrow$ 'Z' Bytes

? $\longleftarrow$ 'K' Bytes

$TT = \frac{K \cdot R}{Z}$

## Data Transfer Rate (DTR) : (B/S)

'Z' Bytes $\longrightarrow R \times 10^{-3}$ S

? $\longleftarrow$ 1 S

$\Rightarrow DTR = \frac{Z}{R} \times 10^{3}$ B/S

# File System Workflow


Hard Drive

Partition

File System

Directory

Files

**Q.**

Consider the following Disk Specifications:

Number of Platters = 16

Number of Tracks/Surface = 512

Number of Sectors/Track = 2048

Sector offset = 12 bits $\Rightarrow$ S.S = $2^{12}$ = 4KB

Average Seek Time = 30 ms

Disk RPM = 3600 $\Rightarrow$ R = 16.6 ms

Calculate the Following:

A. Unformatted Capacity of Disk.

Surf. Capacity = 512 × 2K * 4KB

Disk Capacity = (16 × 2 * 512 * 2K * 4KB)

$= 2^4 \times 2^1 \times 2^9 \times 2^{11} \times 2^{12}$

$= 2^{37} = 128GB$

TS = 2K * 4KB = 8MB

B. IO-Time/Sector : (ST + LT + TT) : 30ms + 8.3ms + x

8MB → 16.6 ms

4KB → ?

$\frac{4KB}{8MB} \times 16.6 = \left(\frac{16.6}{2048}\ ms\right) = x$

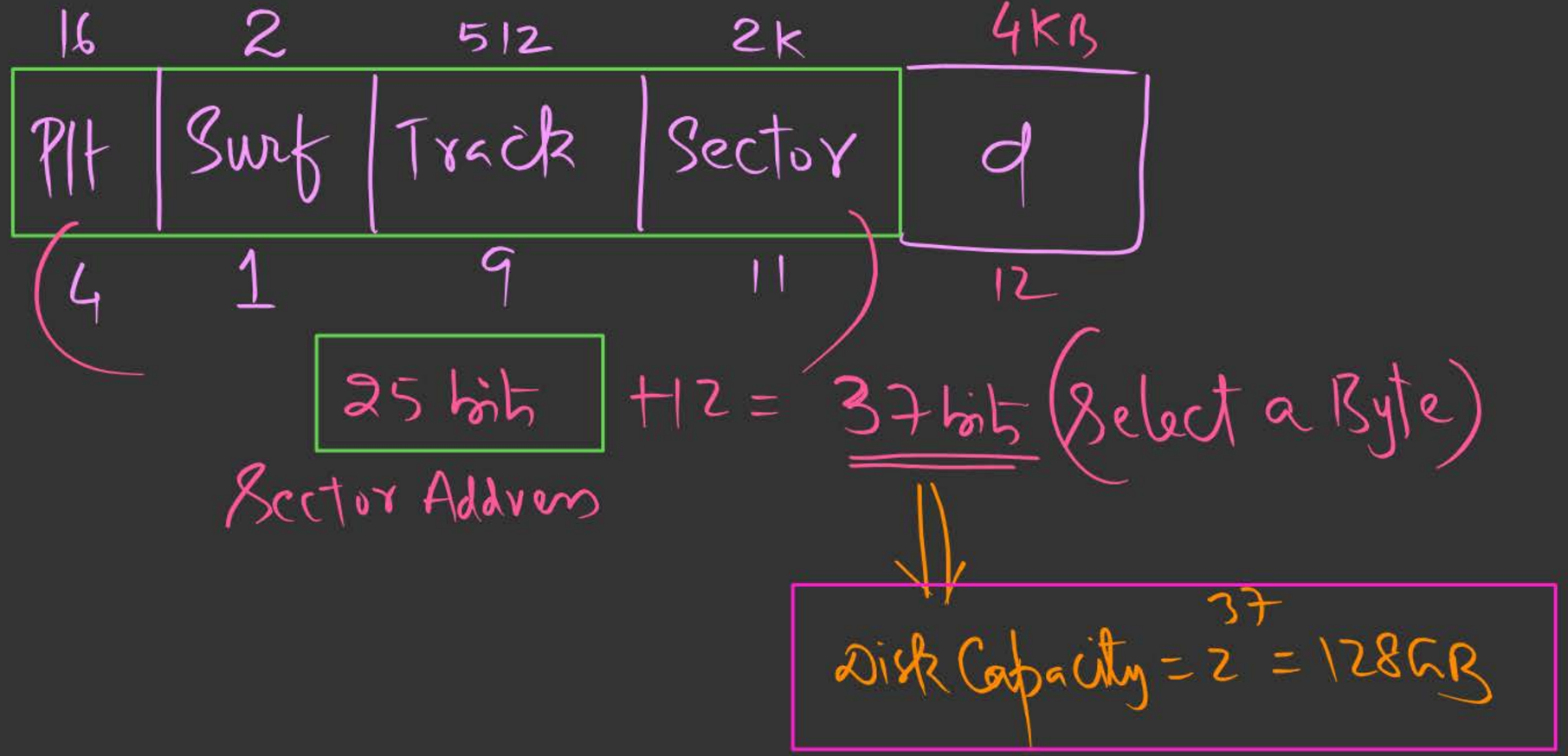
C. Data Transfer Rate

8MB — 16.6 × $10^{-3}$ s

? — 1 s

$= \frac{8MB \times 10^3}{16.6}$ B/s = 0.5GB/s

D. Sector Address


16
2
512
2k
4KB
Plt
Surf
Track
Sector
d
4
1
9
11
12
25 bits
+12 =
37 bits (Select a Byte)
Sector Address
Disk Capacity = 2^37 = 128GB

Q.

G1

Consider a Disk with the following Specifications:

Number of surfaces = 64

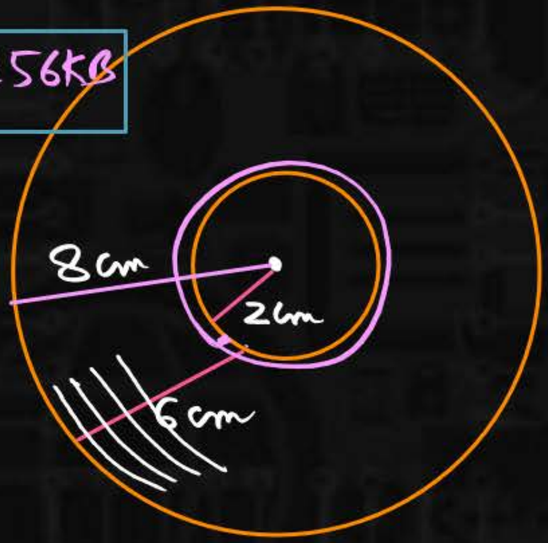
Outer diameter = 16 cm

Inner diameter = 4 cm

Inter Track space = 0.1 mm

Max Density = 8000 bits/cm : Innermost Track

Calculate the Unformatted Capacity of Disk.




$$\frac{\text{No. of Tracks}}{\text{Surface}} = \frac{6\text{cm}}{0.1\times10^{-1}\text{cm}} = 600$$

Track Size = 8000 × 12.56 bits = 1000 × 12.56 B

= 12560 B

= 12.56 KB

Track Size = $2\pi r = 2\times3.14\times2$

$= 4\times3.14$ cm

$= 12.56$ cm

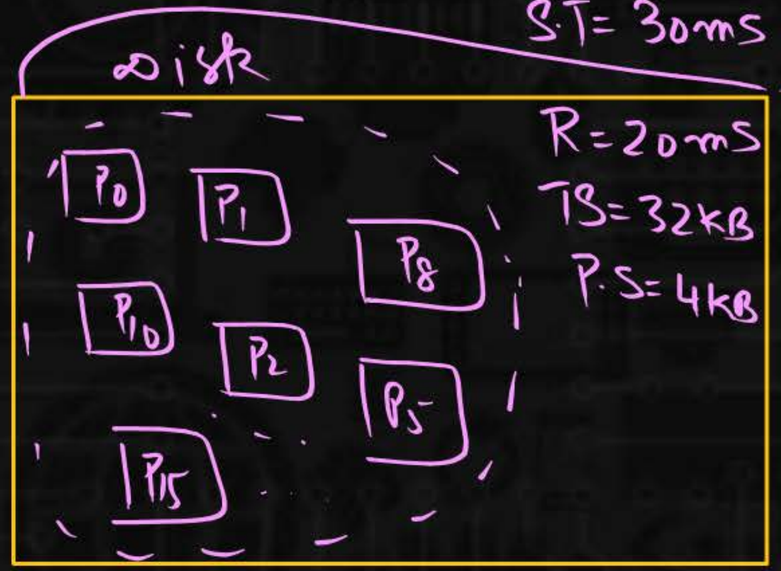
Surf Capacity = 600 × 12.56 KB = 6 × 1256 KB

Disk Capacity = 64 × 6 × 1256 KB

Q.

(*)

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?




32KB — 20ms
4KB — ?

2.5ms

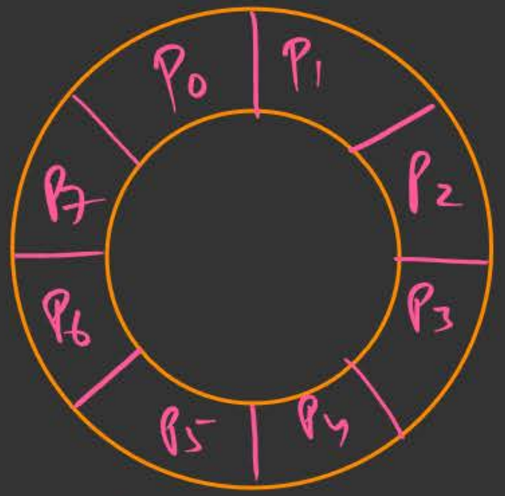
Time to Load Prog = (Time to load a Page) * Npages

= (S.T + L.T + TT)

= (30ms + 10ms + 2.5ms) = (42.5ms)

= 42.5 × 16 ms = 680ms

= 0.685 ✓




$\therefore (30ms + 16ms + 20ms) + 8 \times 42.5ms = 400ms = 0.4s$

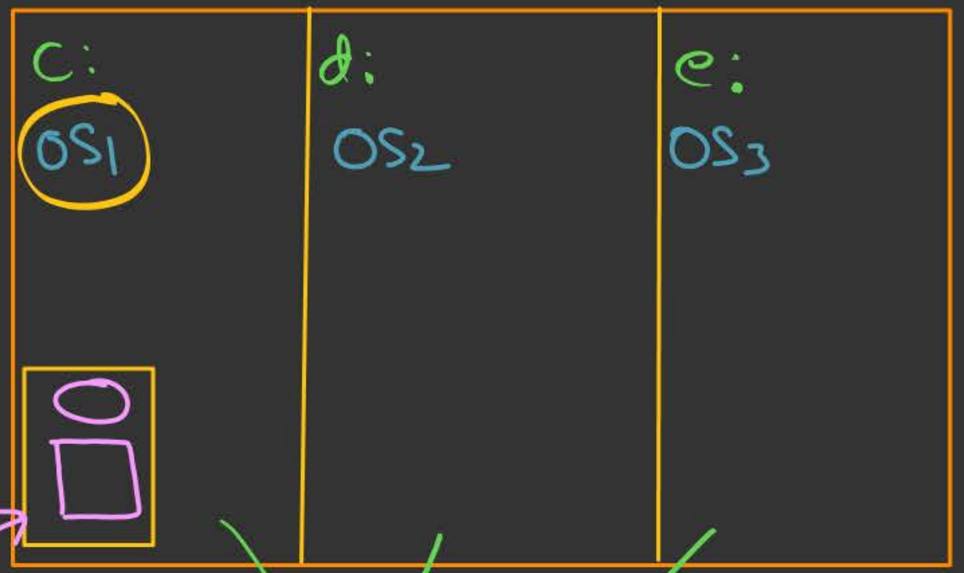
$$\% \text{ Saving} = \frac{0.28}{0.68}$$

Q.

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?

# (fdisk) Logical Structure of Disk [Formatting]




Partitions

Primary (Bootable)
OS + S/W + Data

Extended (Logical Drives)
S/W + Data

Multi-Boot Computer

M.B.R (Master Boot record)
- Partition Table
- Boot loader

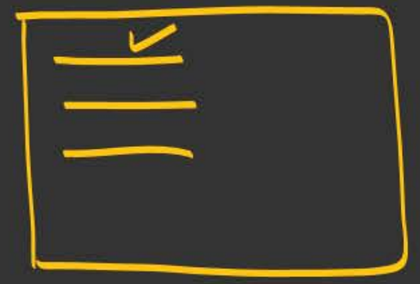
Partitions / volumes / Mini disk

# Booting Process

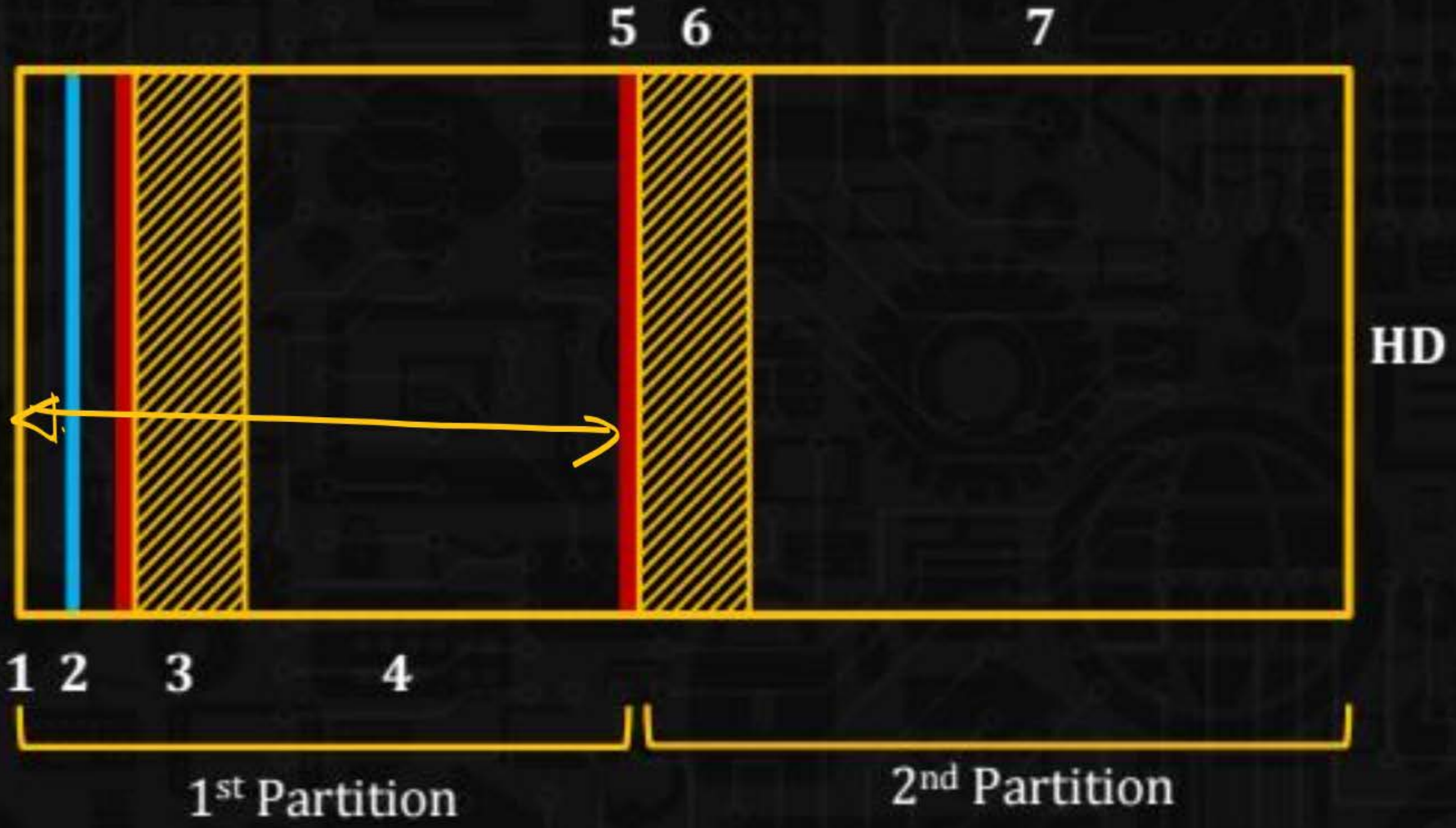
1. Power on
   → POST (Power on Self-Test)
2. BIOS (Rom)
   (Basic Io Sys)
3. Bootstrap (Rom)
   → Load MBR into RAM
   ↓
   (Boot loader)

4) Boot loader

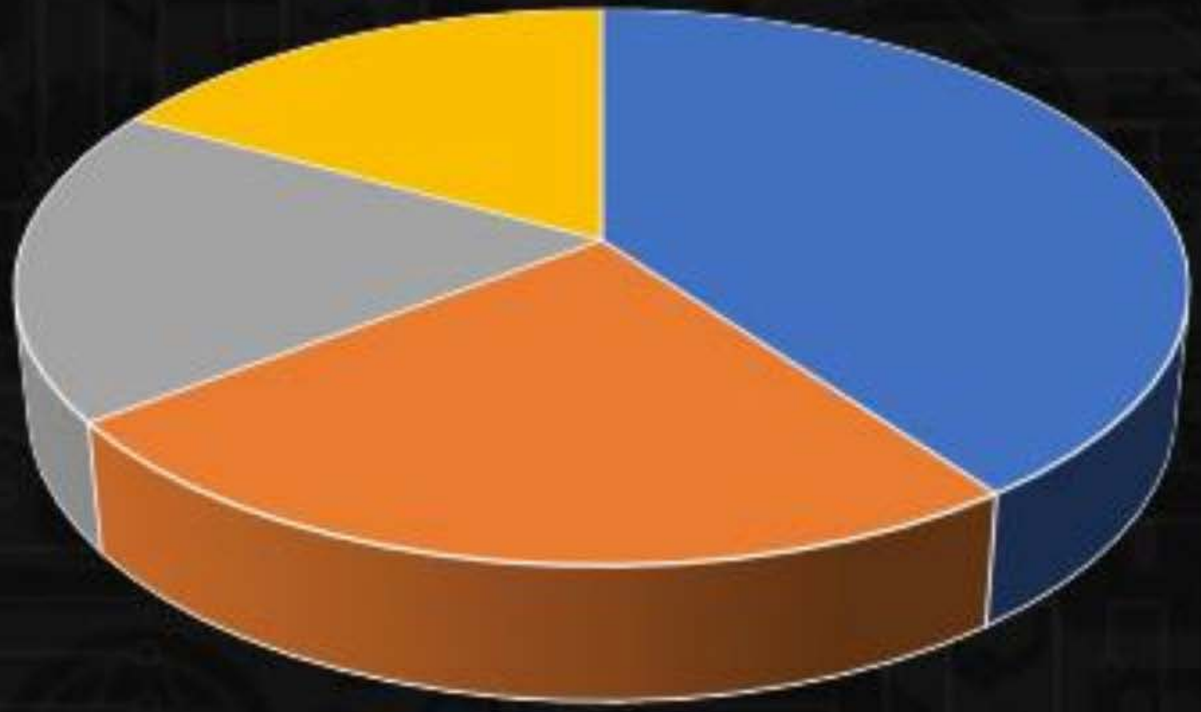
Partition Table

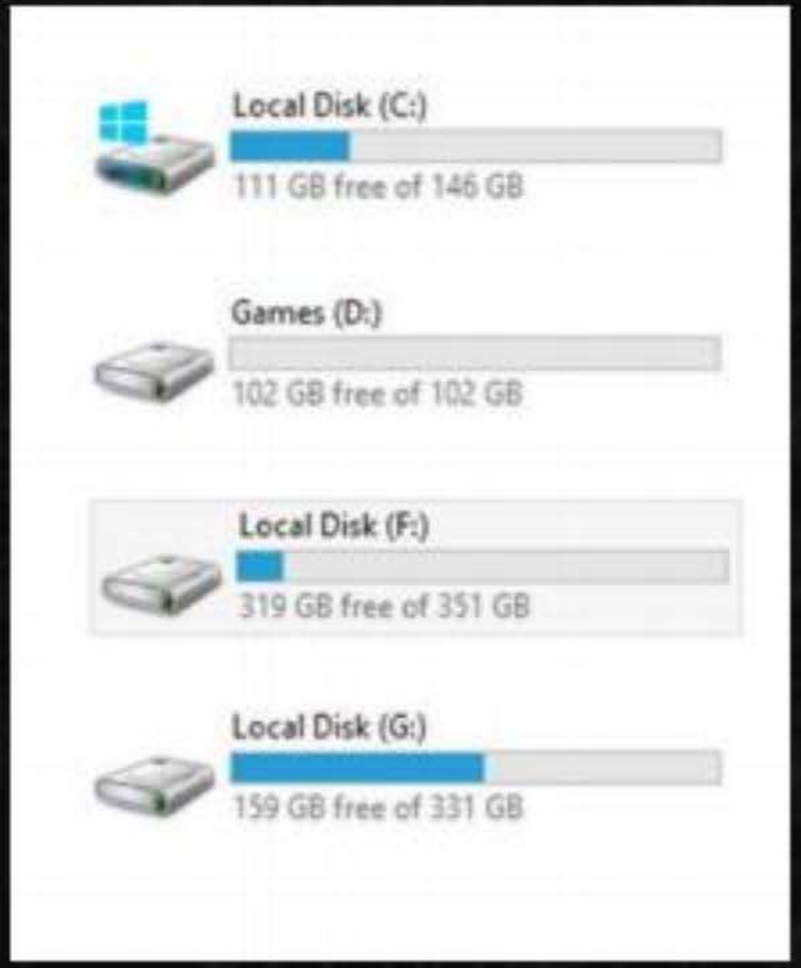


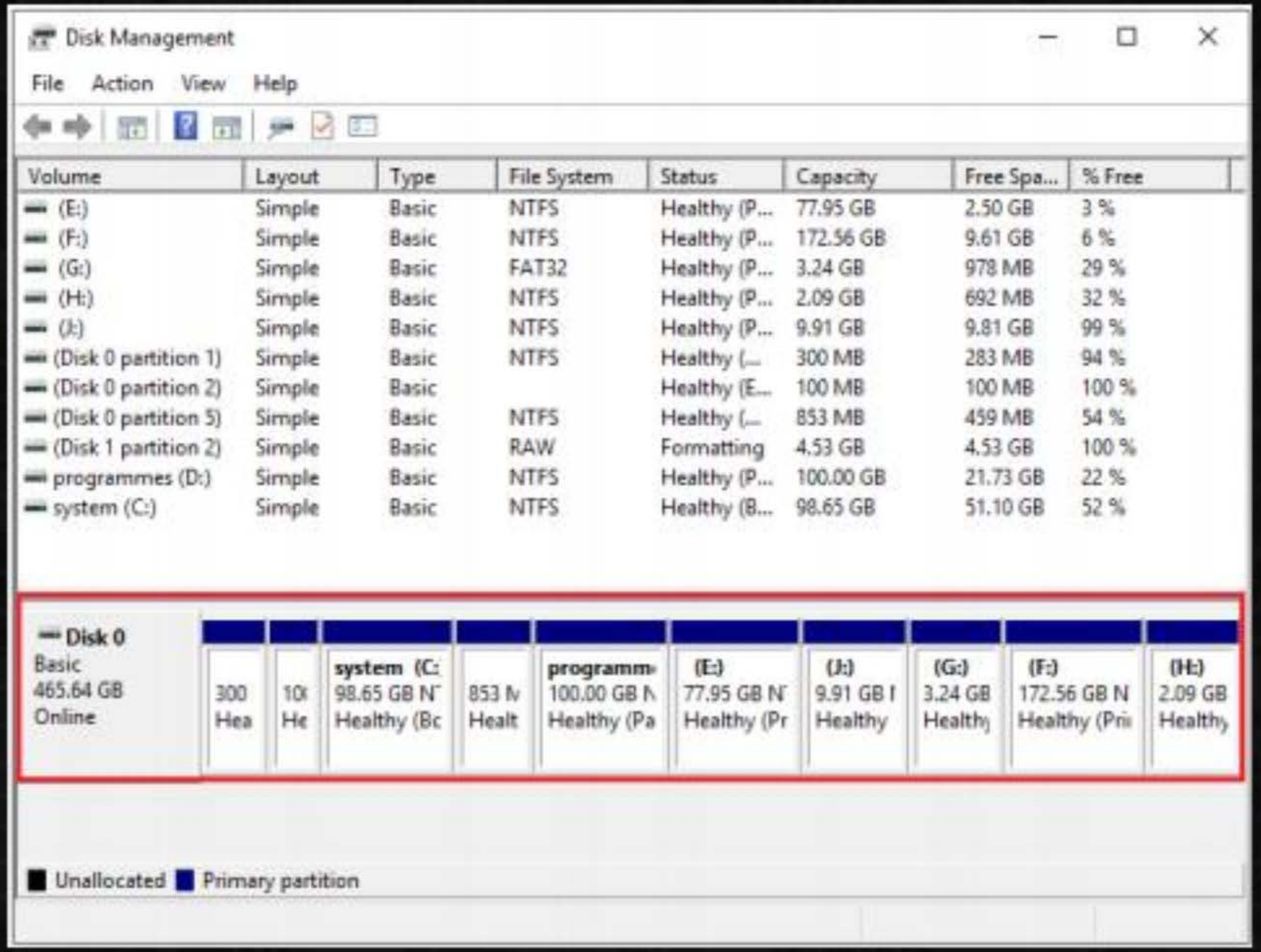
→ Secondary B.L's
   → GRUB
      → LILO
   → NT-LDR



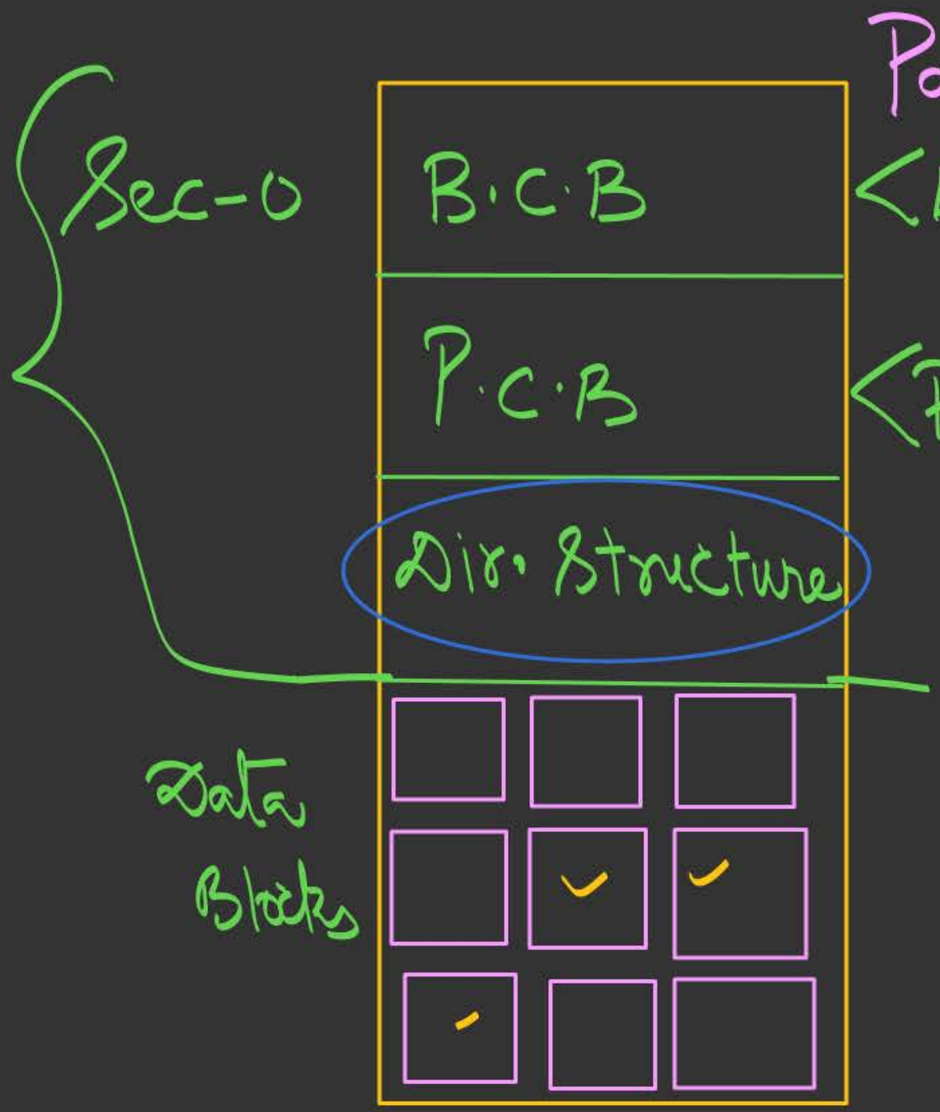

1- MBR
2 - First Partition Boot Sector
3 - File system Area of 1st Partition
4 - Data Area of 1st Partition




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 <due to Formatting with Some File System>




<Boot Control Block>

Firm-OS : init

<Partition Control Block>

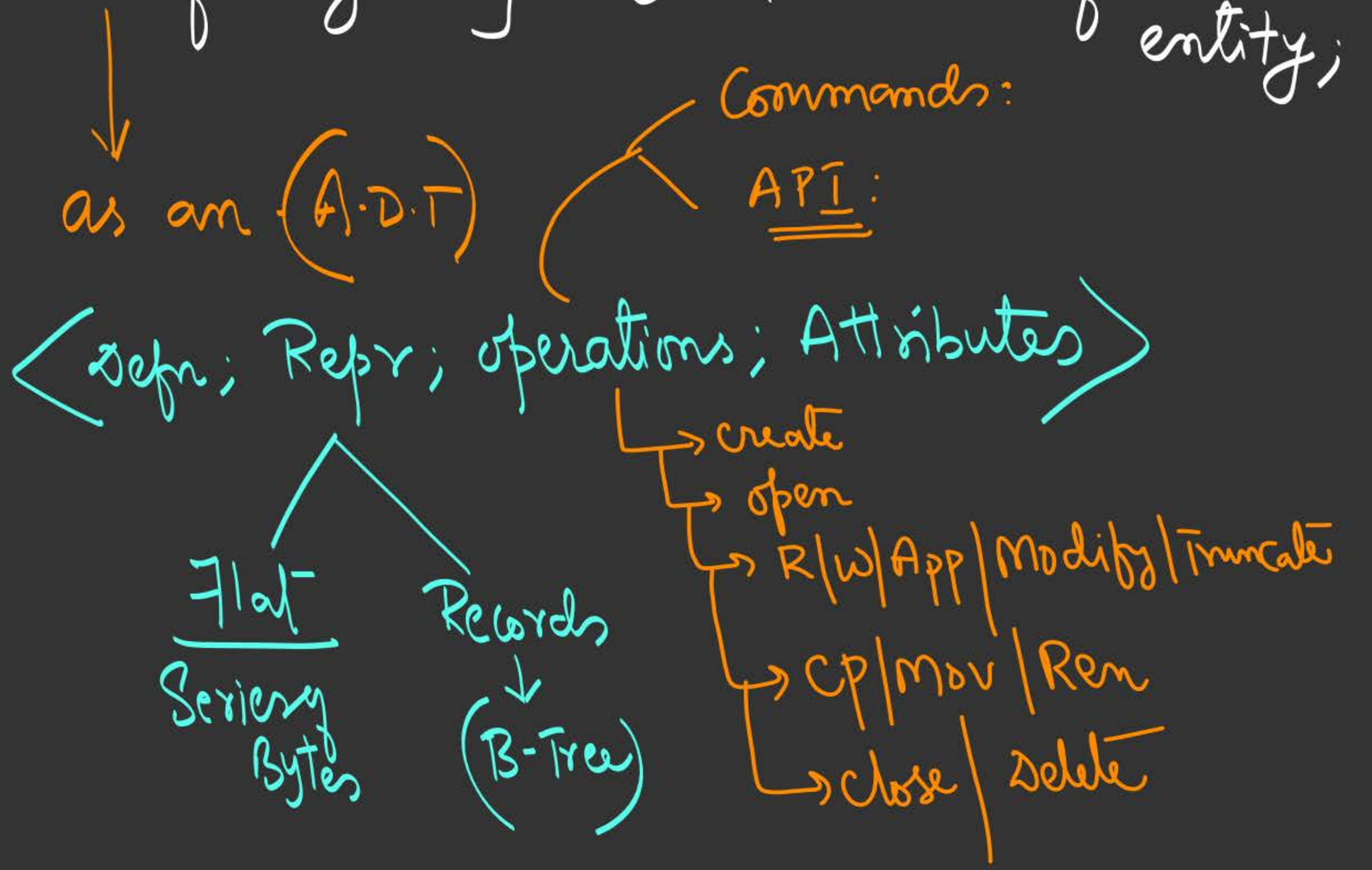
UNIX : SuperBlock

NTFS : Master File Table

FAT-8 : F.A.T

# Files vs Directory : File Interface :

File: is a collection of logically related records of some entity;



# File Attributes :

- Name, Type
- Size, Mode
- owner
- Acc. Rights
- Date & Time Stamps
- Location

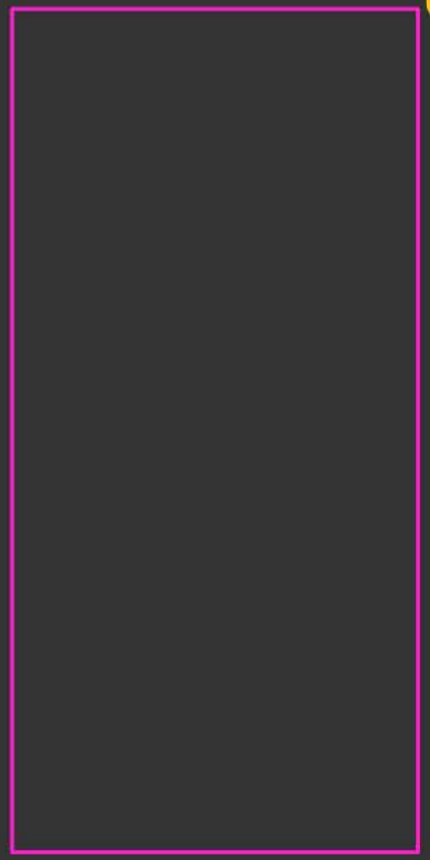
} General

- Blocks in use

} Loc Specific

- UFS : I-Node
- NFFS : Dir-entry

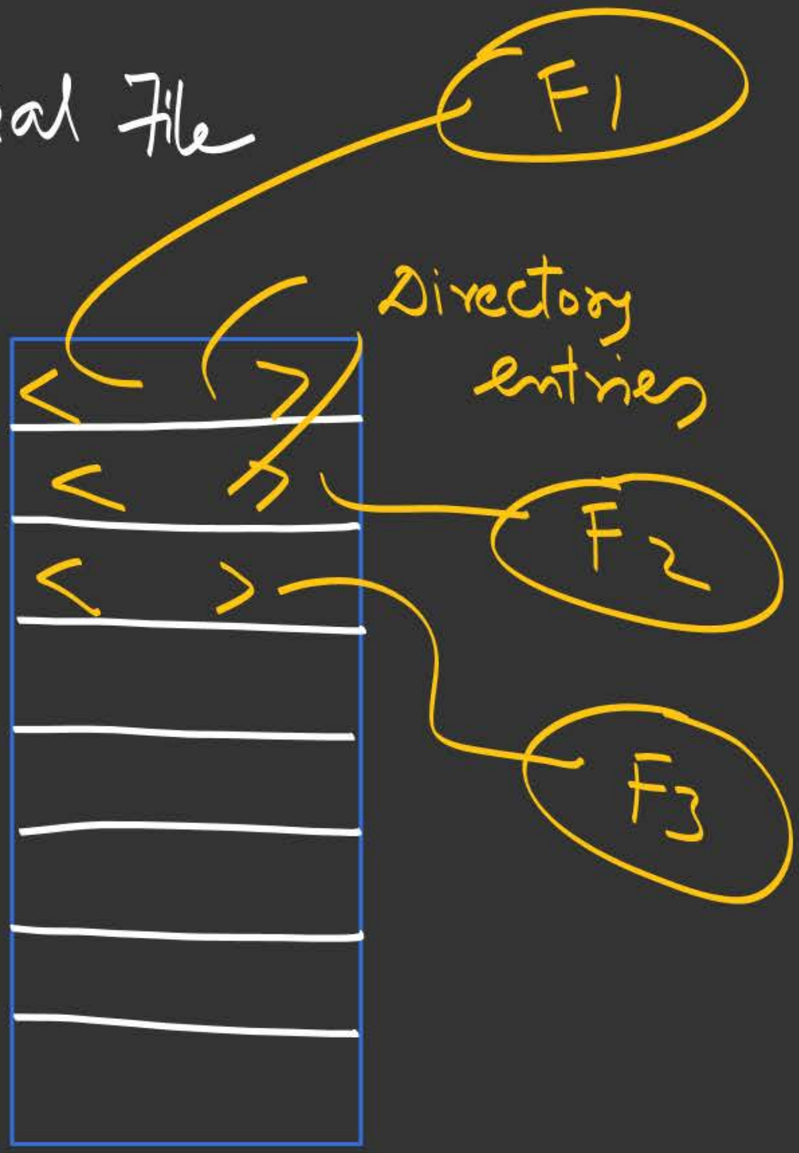
F.C.B < File Control Block >



**Directory:** is a collection of files

↓ is also a special file





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.

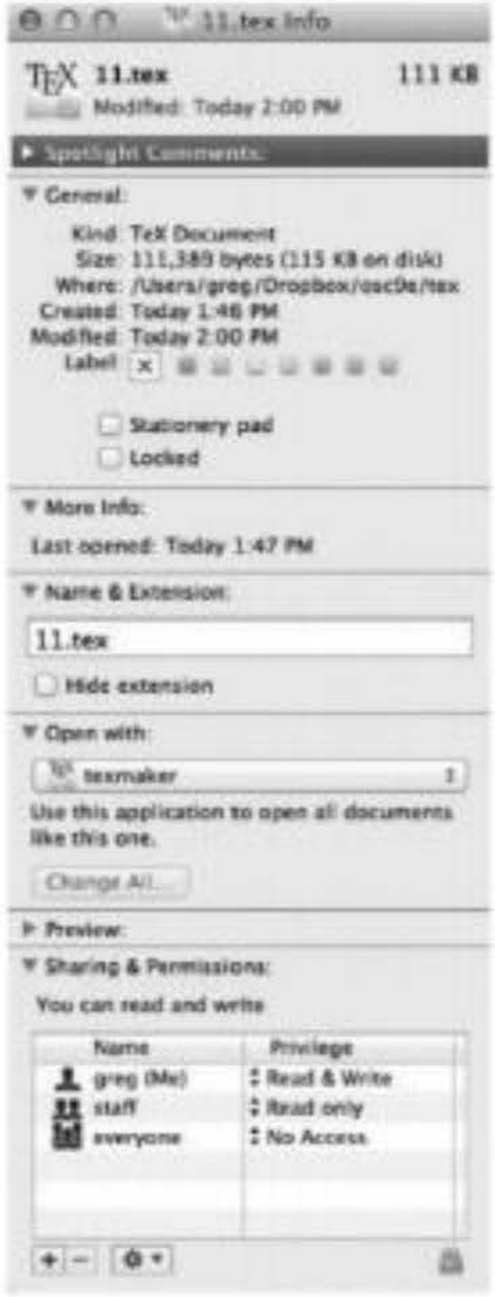



Figure 11.1 A file info window on Mac OS X.

- 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.

- 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 |


THANK YOU GW
SOLDIERS !