Operating Systems

Quiz

Q. Which of the following memory allocation method is faster?

A. best fit

B. first fit

C. worst fit

D. none of the above

Q. Memory remains unused which is internal to the partition then it is referred as _____.

a. an external fragmentation

b. an internal fragmentation

c. both options A & B

d. none of the above

QUIZ

Q. The associatively mapped virtual memory makes use of _____
a) TLB
b) Page table
c) Frame table
d) None of the mentioned

Q. The operating system maintains a _____ table that keeps track of how many frames have been allocated, how many are there, and how many are available.
a) memory
b) mapping
c) page
d) frame

QUIZ

Q. _____ is used to implement virtual memory organisation.

A. Page table

B. Frame table

C. MMU

D. None of the mentioned

Q. Which of the following section contains the magic number in an executable file?

A. bss section

B. code section

C. primary header /exe header

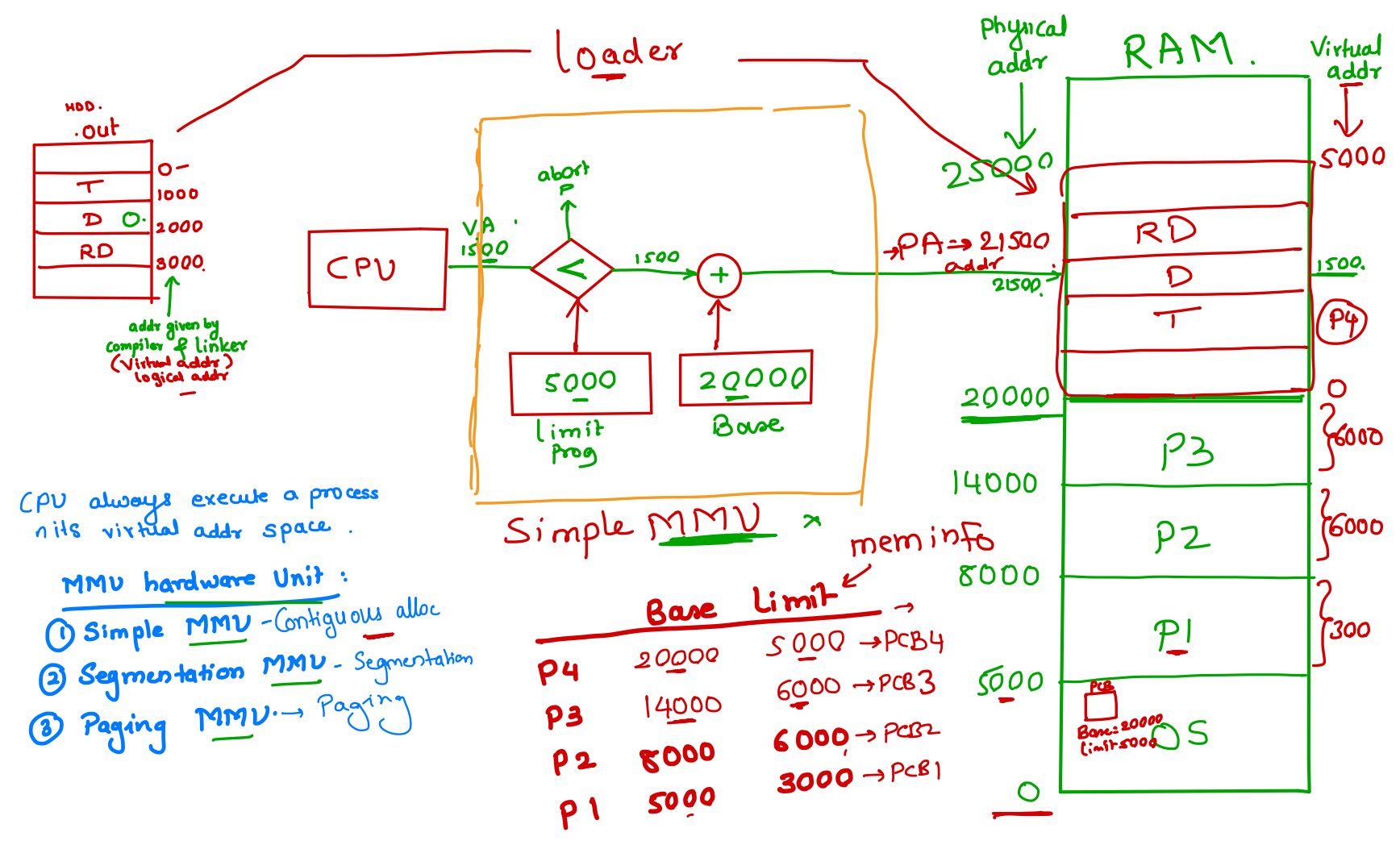
D. symbol table

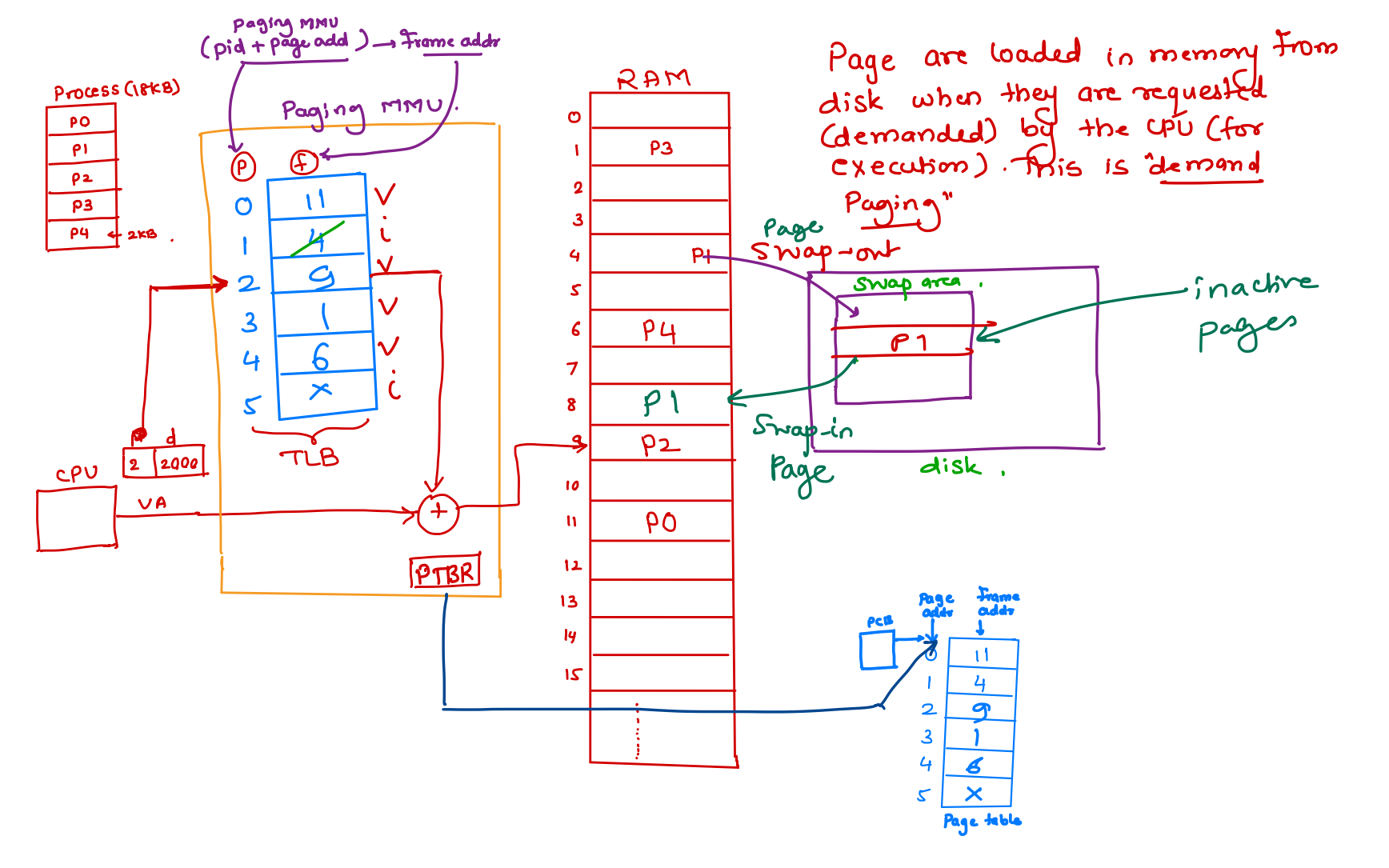
Quiz

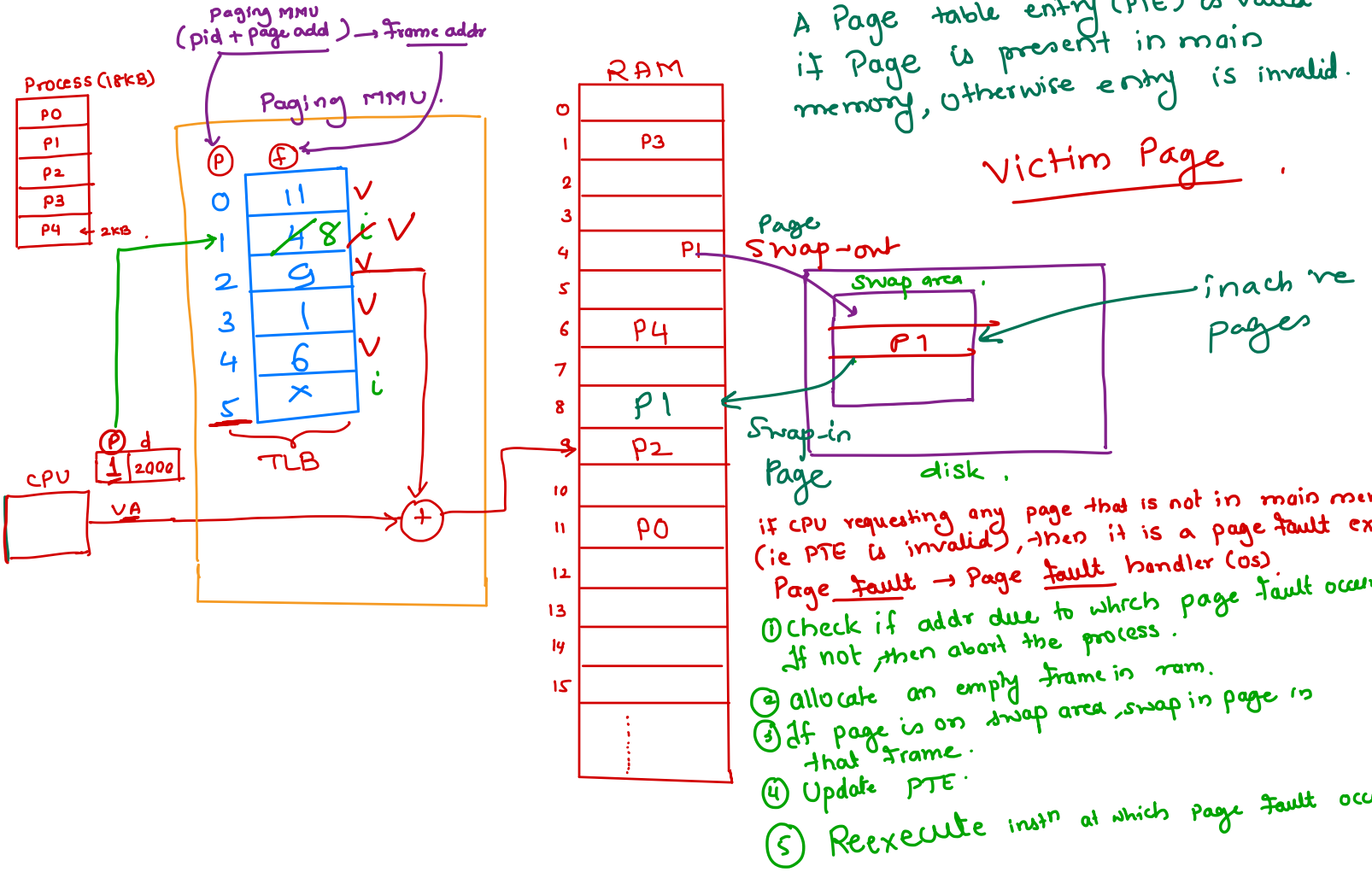
Q. MMU is a _____ that converts logical address into the physical address.

a. system program
b. application program
c. firmware
d. all of the above
e. none of the above

MMU is a.
hardware
Software
hardware / software


Loader
HDD.
.out
0
T
1000
D
2000
RD
3000
addr given by compiler & linker
(Virtual addr)
logical addr
CPU
VA
1500
abort
<
1500
+
5000
limit
Prog
20000
Base
Simple MMU
PA => 21500
addr
21500
physical
addr
25000
RAM.
Virtual
addr
5000
RD
D
T
1500.
P4
0
20000
P3
6000
14000
P2
6000
8000
P1
300
5000
PCB
Base=20000
limit=5000
OS
0
CPU always execute a process
with virtual addr space.
MMU hardware Unit :
1. Simple MMU - Contiguous alloc
2. Segmentation MMU - Segmentation
3. Paging MMU → Paging
meminfo
Base Limit →
P4 20000 5000 → PCB4
P3 14000 6000 → PCB3
P2 8000 6000 → PCB2
P1 5000 3000 → PCB1


Paging MMU
(pid + page add) → frame addr
Process (18KB)
P0
P1
P2
P3
P4 ← 2KB
Paging MMU.
P
f
0
1
2
3
4
5
11
4
9
1
6
X
V
i
V
V
V
i
TLB
p
d
2
2000
CPU
VA
+
PTBR
RAM
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
P3
P1
P4
P1
P2
P0
Page Swap-out
Swap area.
P1
inactive pages
Swap-in Page
disk.
Page are loaded in memory from disk when they are requested (demanded) by the CPU (for execution). This is "demand Paging"
PCB
Page addr
Frame addr
0
1
2
3
4
5
11
4
9
1
6
X
Page table



A Page table entry (PTE) is valid if Page is present in main memory, otherwise entry is invalid.

Victim Page

if CPU requesting any page that is not in main memory (ie PTE is invalid), then it is a page fault exception

Page fault → Page fault handler (os).

① Check if addr due to which page fault occur is valid If not, then abort the process.

② allocate an empty frame in ram.

③ If page is on swap area, swap in page in that frame.

④ Update PTE.

⑤ Reexecute instn at which page fault occured.

Page receplacement Algorithm..

1. FIFO ✓
2. Optimal ✓
3. LRU ✓
4. MRU
5. LFU
6. MFU
7. Second Chance LRU

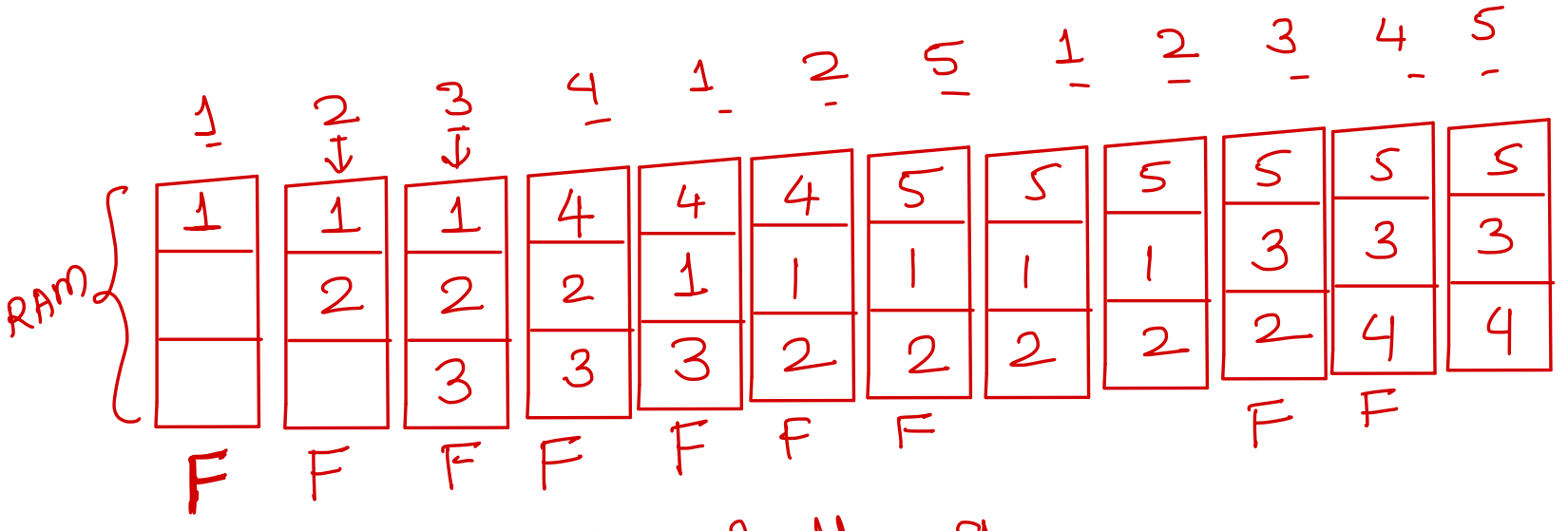
Paging

- If no frame is empty, then OS needs to swap-out some page from memory this is called as “victim page”
- Page replacement algorithm also to decide victim page.

- **Page replacement algorithm**

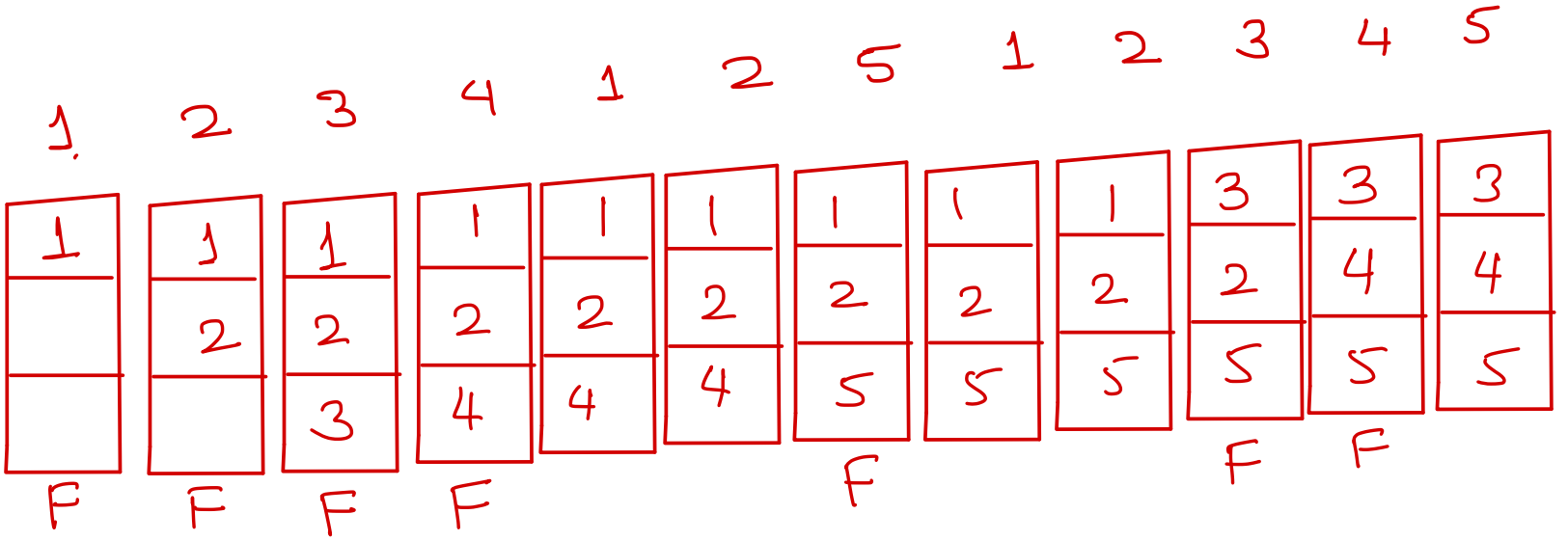
1. FIFO – First in First out
2. Optimal
3. LRU – ~~List~~ Least Recently used
4. MRU - Most Recently used
5. LFU - ~~List~~ least Frequently used
6. MFU - Most Frequently used

FIFO ✓



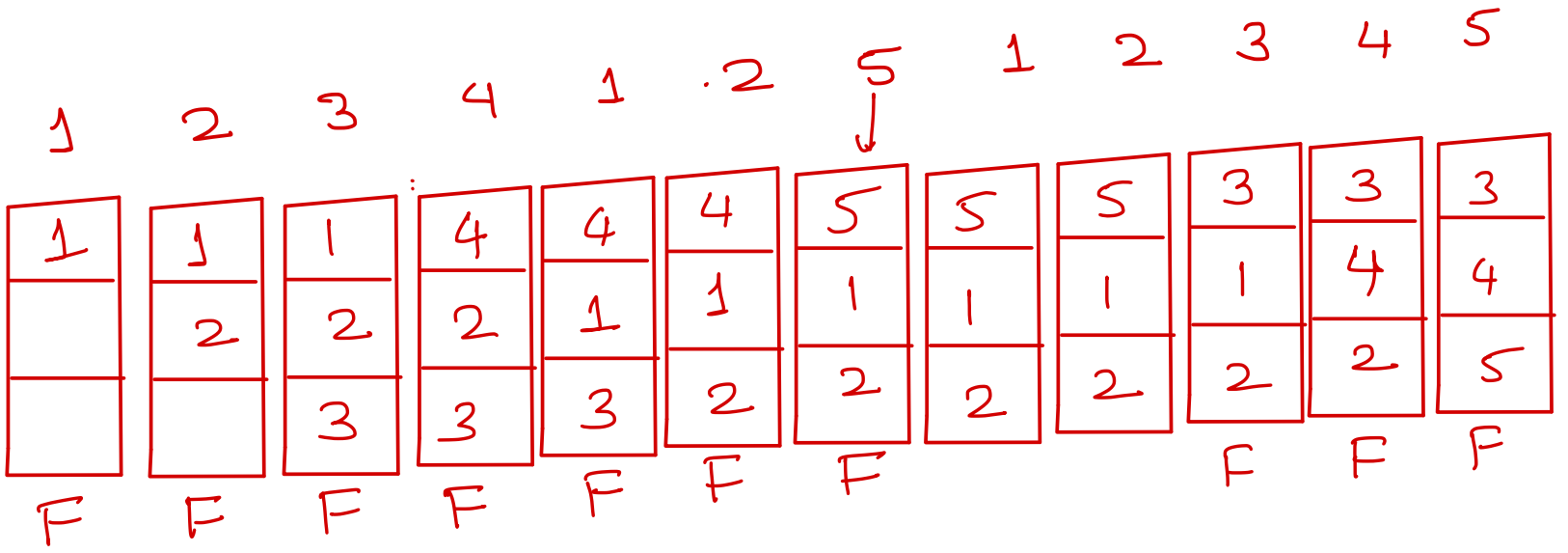
No of Page fault = 9

Optimal Page Replacement.



Fault Fault – 7

LRU Page Replacement



Thrashing

- If a process does not have “enough” pages, the page-fault rate is very high
  - low CPU utilization
  - OS thinks it needs increased multiprogramming
  - adds another process to system
- *Thrashing* is when a process is busy swapping pages in and out

Thrashing reduce → increase Ram size.

Operating System Concepts

 # UNIX Operating System:

- UNIX: UNICS – **Uniplexed Information & Computing Services/System.**

- UNIX was developed at **AT&T Bell Labs** in US, in the decade of 1970's by **Ken Thompson, Denies Ritchie** and team.

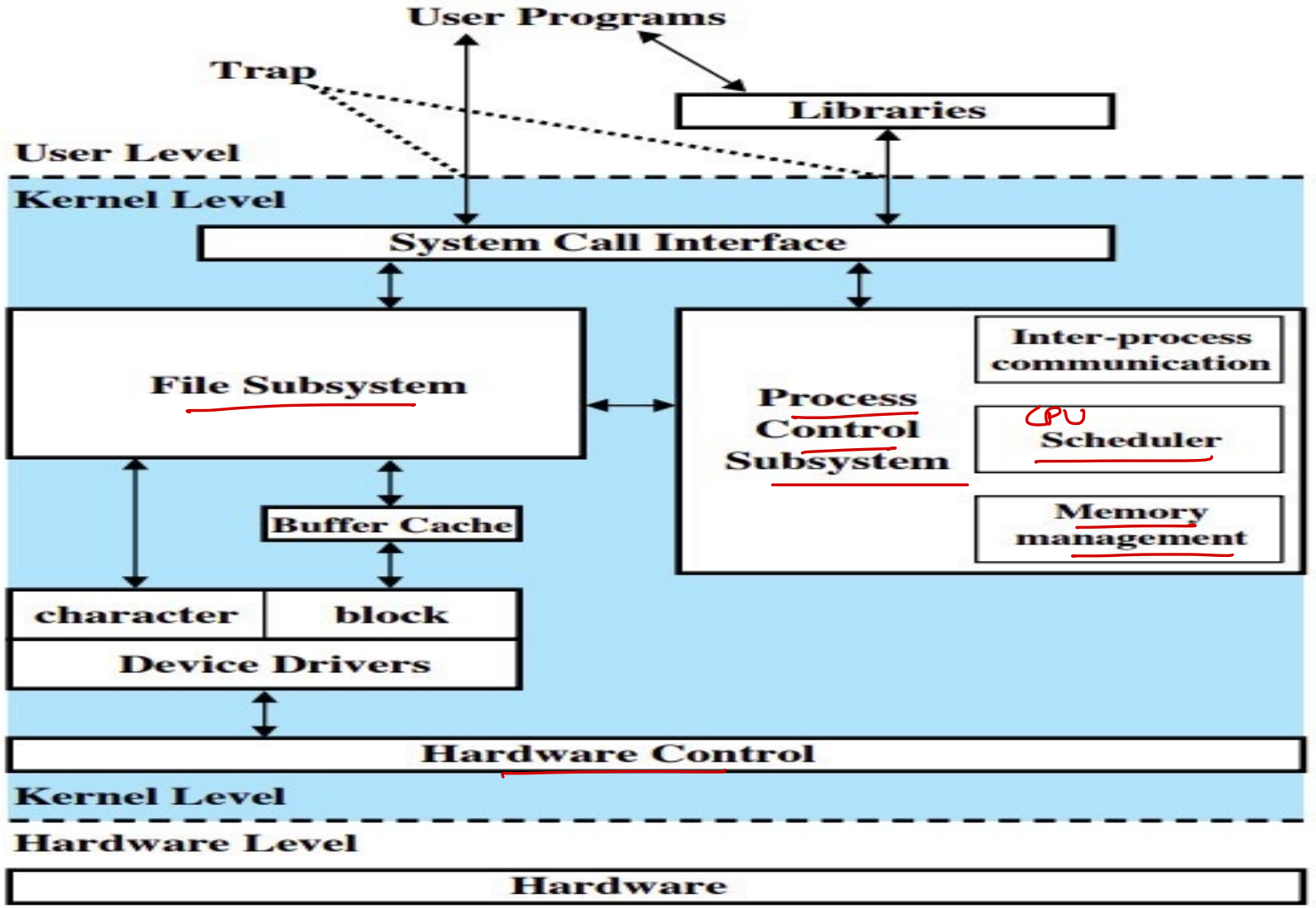
- It was first run on a machine **DEC-PDP-7** (Digital Equipment Corporation – Programmable Data Processing-7).

- UNIX is the first **multi-user, multi-programming** & **multi-tasking** operating system.

- UNIX was specially designed for developers by developers

- System architecture design of UNIX is followed by all modern OS's like Windows, Linux, MAC OS X, Android etc..., and hence UNIX is referred as **mother of all modern operating systems.**

Operating System Concepts



- Kernel acts as an interface between programs and hardware.

- Operating System has subsystems like **System Call Interface Block, File Subsystem Block, Process Control Subsystem Block (which contains IPC, Memory Management & CPU Scheduling), Device Driver, and Hardware Control/Hardware Abstraction Layer.**

- There are two major subsystems:

1. Process Control Subsystem
2. File Subsystem

- In UNIX, whatever can be stored is considered a file and whatever is active is referred to as a process.

- **File has space & Process has life.**

Operating System Concepts

- From the UNIX point of view all devices are considered a file
- In UNIX, devices are categorized into two categories:

1. Character Devices: Devices from which data gets transferred character by character → character special device file

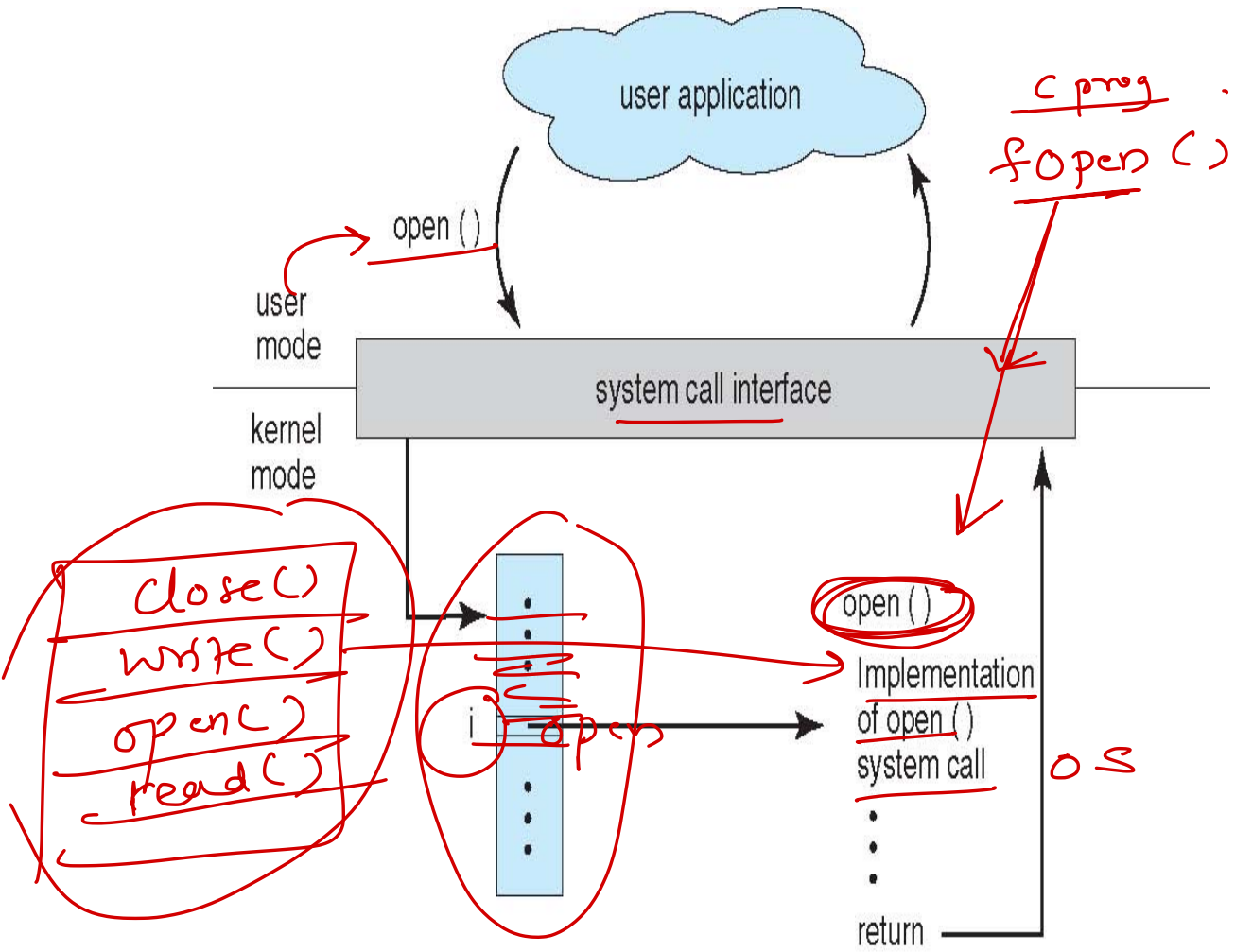
e.g. keyboard, mouse, printer, monitor etc...

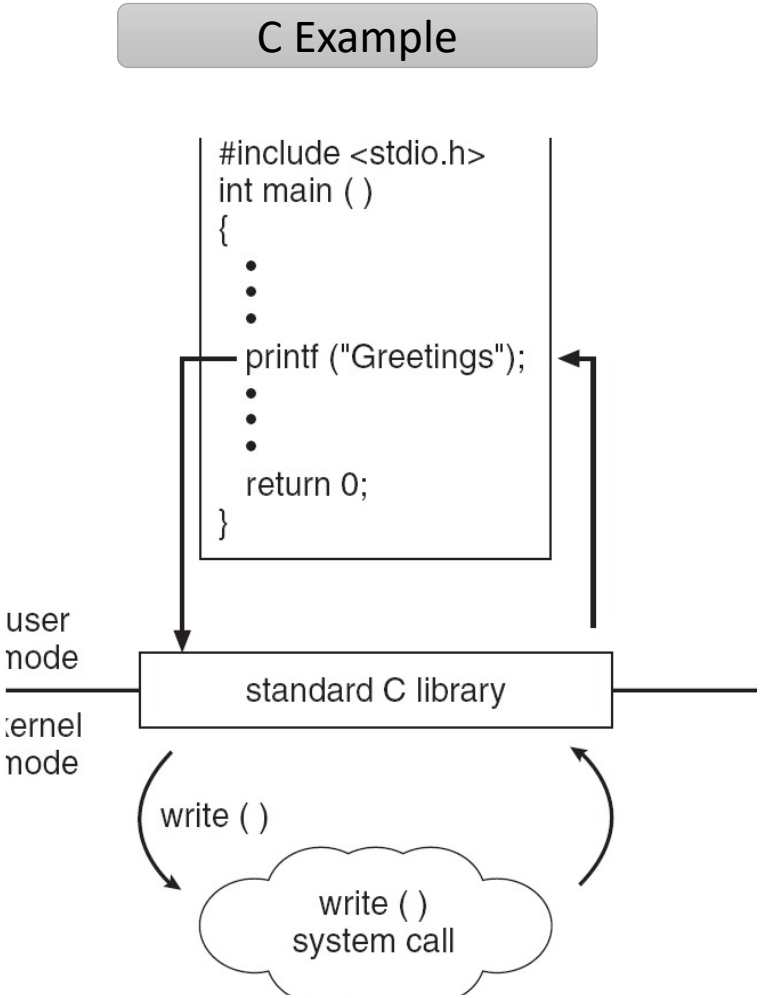
2. Block Devices: Devices from which data gets transferred block by block → block special device file

e.g. all storage devices.

- **Device Driver:** It is a program/set of programs enable one or more hardware devices to communicate with the computer's operating system.

System Call and OS Relationship





System Call Categories

Process control

(fork(),exit(),wait()

- load, execute, end, abort
- create process, terminate process
- get and set process
- allocate and free memory

File management

Read(),write()

- create file, delete file, open, close file
- read, write

Device management

Read(),write()

- request device, release device
- read, write
- get device attributes, set device attributes

Communications

Pipe(),shmget()

- send, receive messages
- transfer status information

Protection and Security

Chmod(),chown()

- Grant permissions
- Change ownership

Information maintenance

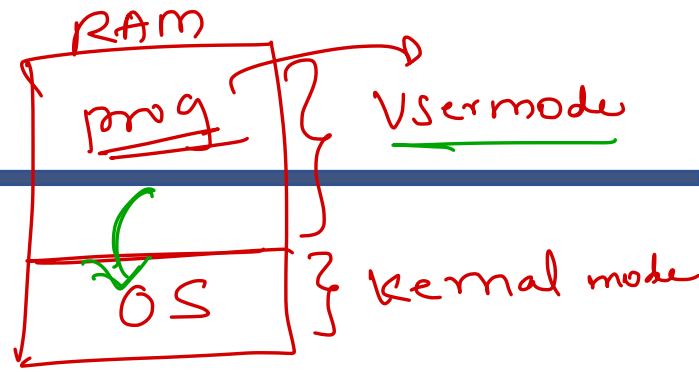
Getpid(),sleep()

- get time or date, set time or date
- get system data, set system data

System Calls

- File management
  - Open()→ called by fopen() in C
  - Close()→ called by fclose() in C
  - Read()→ called by fread(), fscanff(), fgets(), fgetc() in C
  - Write()→ called by fwrite(), fprintf(), fputs(), fputs() in C
  - Lseek()→ called by fseek(), ftell()
  - Chmod()→ to change the file permissions(rwx)
  - Chown()→ to change the file owner(~~rwx~~) name owner.

- Memory management
  - brk() → called by malloc() in C
  - nmap()



trap / interrupt

Dual Mode Operation:

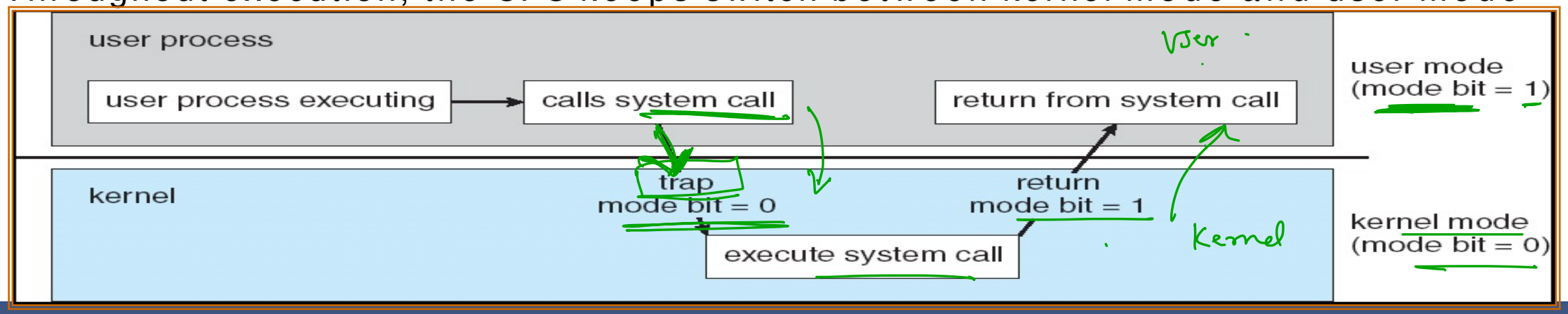
- System runs in two modes: System Mode and User Mode

1. System Mode: / Kernel Mode

- When the CPU executes system defined code instructions, system runs in a system mode.
- System mode is also reffered as kernel mode/monitor mode/supervisor mode/ privileged mode.

2. User Mode:

- When the CPU executes user-defined code instructions, system runs in a user mode.
- User mode is also reffered as non-priviledged mode.
- Throughout execution, the CPU keeps switch between kernel mode and user mode



Dual Mode Operation:

- Throughout the execution of any program, the CPU keeps switcesh in between kernel mode and user mode and hence system runs in two modes, it is referred as **dual-mode operation.**

- Todifferentiate between user mode and kernel mode one bit is there onto the CPU which is maintained by an OS, called as **mode bit**, by which the CPU identifies whether the currently executing instruction is of either system-defined code instruction/s or user-defined code instruction/s.

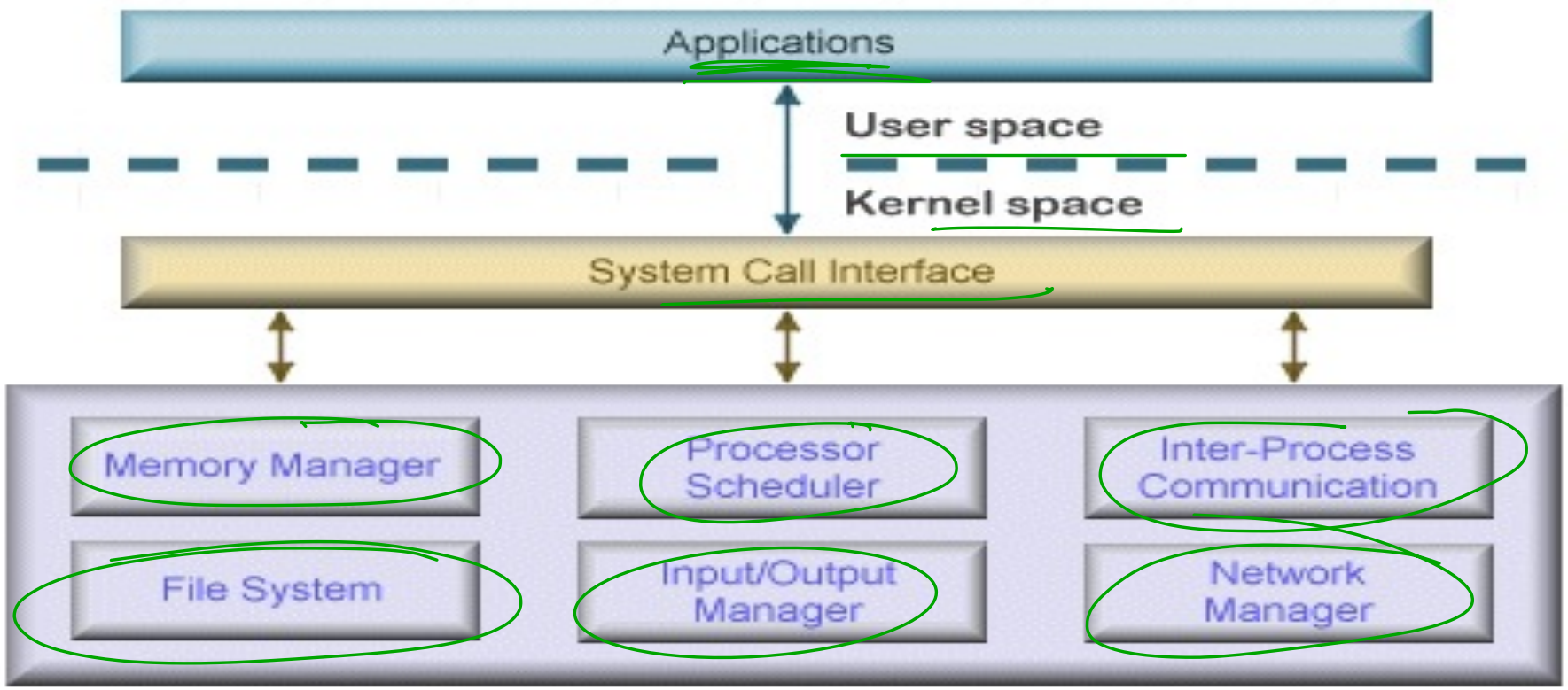
- In Kernel mode value of **mode bit = 0**, whereas

- In User mode **mode bit = 1.**

Kernel space vs User space

Part of the OS runs in the kernel model known as the OS kernel / Kernel space

Other parts of the OS run in the user mode, including service programs , user applications, etc.they run as processes they form the user space (or the user land)

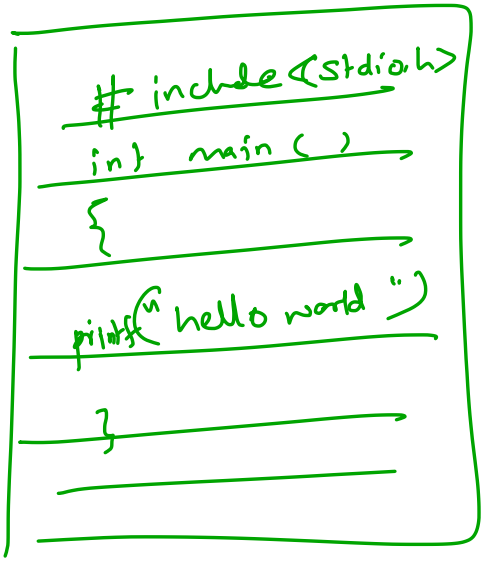


Save → hello.c → created owner →

Time :

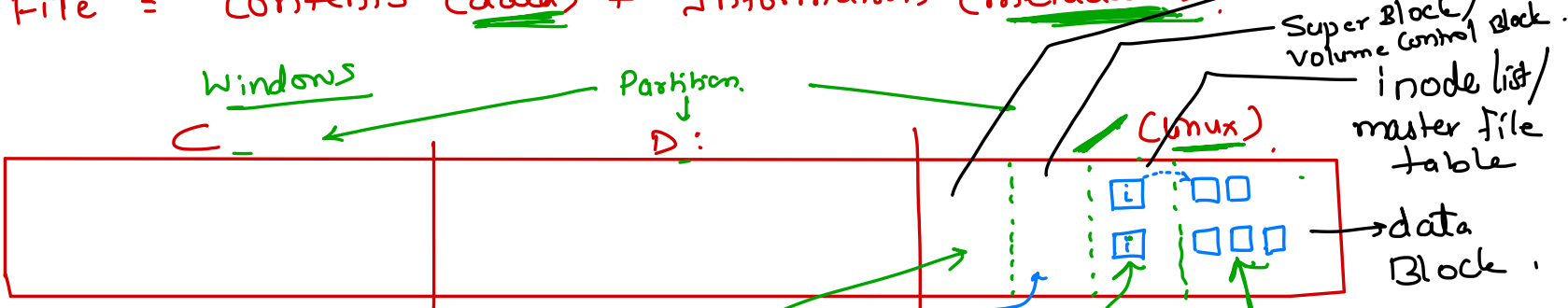
created →

modify →.



File = Collection of data & info on storage device.

File = Contents (data) + Information (metadata)



Booting info

1. Bootstrap prog.
2. Bootloader

Partition Information.
1. label
2. Capacity
3. free Block

Metadata/info
File Control Block
FCB/inode
1. Type
2. Permission/mode
3. User and group
4. size
5. name
6. timestamp
   - create
   - modify

data blocks → data
Size can be 1KB to 2MB. (depends on FS)

Way of organizing file on disk = File System

File

- file is a collection of logically related data or information
- file is a stream of bytes/bits
- file is a basic storage unit
  - File = data + metadata
  - Data: Actual File Contents
  - Metadata: Information about file.
- File Attributes:
  - Name, type, location, size etc..
- File Operations
  - Create, delete, write, read

- **"file system" is a way to store data onto the disk in an organized manner so that it can accessed efficiently**

- e.g. Each OS has its own filesystem like UNIX: UFS(UNIX Filesystem), Linux: Extended filesystem ext2, ext3, ext4, Windows: FAT, NTFS etc..., MAC OSX: HFS(Hierarchical Filesystem) etc...

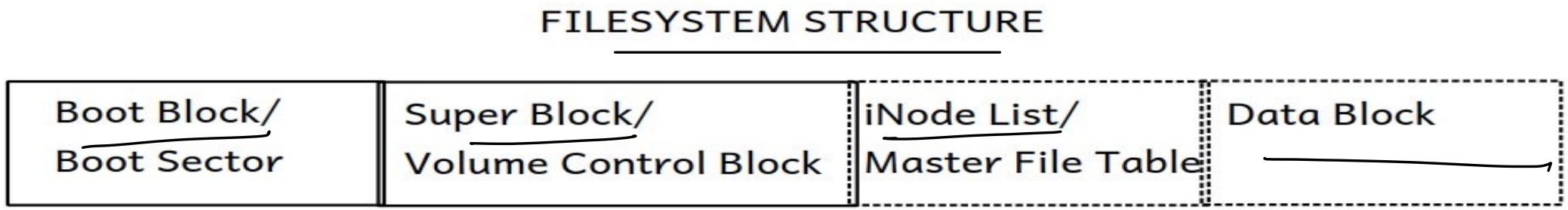
What is an inode / FCB

- An inode (index node) is a control structure that contains key information needed by the OS to access a particular file. Several file names may be associated with a single inode, but each file is controlled by exactly ONE inode.

- On the disk, there is an inode table that contains the inodes of all the files in the filesystem. When a file is opened, its inode is brought into main memory and stored in a memory-resident inode table.
- Information about the file can be kept in one structure referred as "FCB" i.e. File Control Block/iNode
  - inode/FCB contains info about the file like:
    - name of the file
    - type of the file
    - size of the file
    - parent folder location
    - access perms for user/owner, grp member and others etc

File System Structure

File system divides disk/partition logically into sectors/blocks, like **boot sector/boot block, volume control block/super block, master file table/iNode list block and data**



1. Boot Block: It contains information about booting the system like bootstrap program, bootloader etc...
2. Super Block: It contains information about remaining sections, like total no. of data blocks, no. of free data blocks, no. of allocated data blocks etc....
3. iNode List: It contains linked list of iNode's of all files exists on a disk.
4. Data Block: It contains actual data.

File Allocation on Disk

- Low level access methods for a file depend upon the disk allocation scheme used to store file data
  - Contiguous
  - Linked
  - Block or indexed

Contiguous Allocation.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

	Start	Size
f1.txt	12	5
f2.txt	38	3
f3.txt	64	10

Stored in inode or dir entry of file

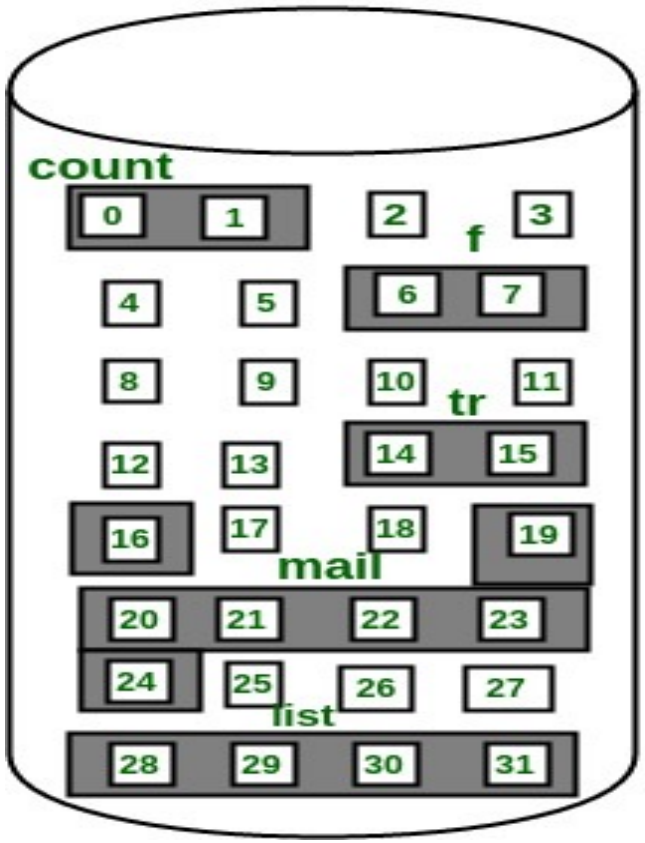
× File cannot grow ×

✓ Sequential

✓ Random access.

× External fragmentation

Contiguous Allocation



Directory

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

- File is allocated large contiguous chunks
- Expanding the file requires copying
- Dynamic storage allocation - first fit, best fit
- **External fragmentation occurs on disk**

Linked Allocation

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

	Start	End	→ inode
f1.txt	24	71	
f2.txt	77	96	

eg :- FAT

adv

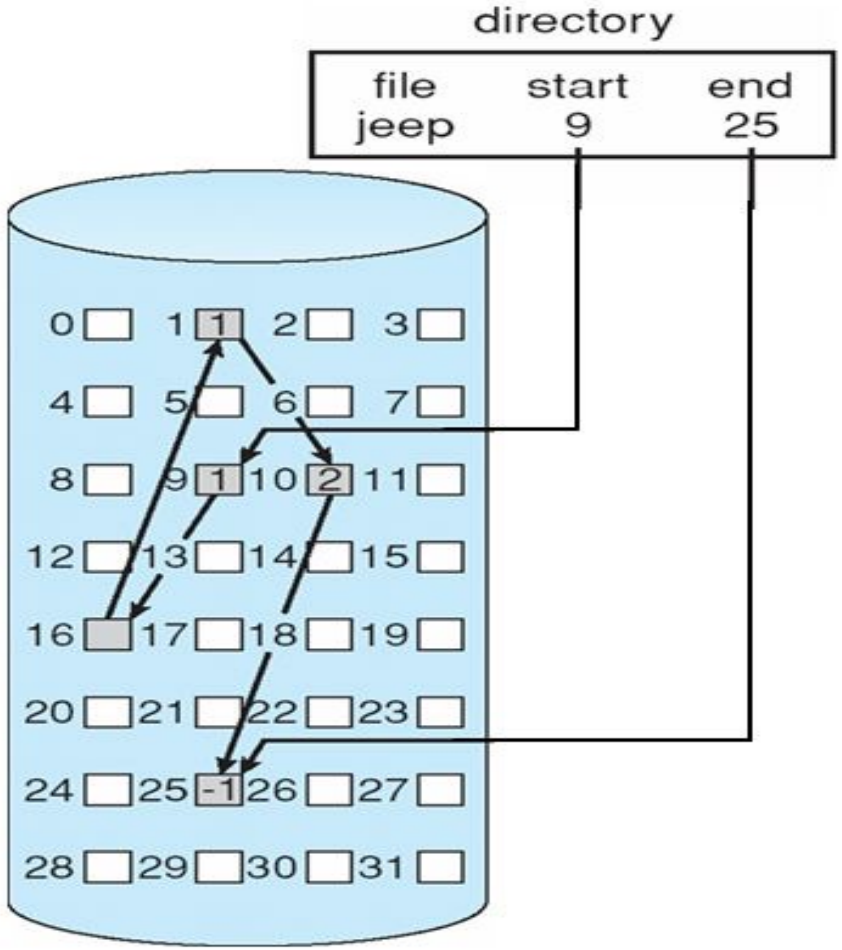
- ✓ External Frag
- ✓ Sequential
- ✓ File grown

disadv

- ✗ Random access.

Linked Allocation

- Each file is a linked list of disk blocks, which may be scattered on the disk
- Directory contains a pointer to the first and last blocks, and each block contains a pointer to the next block
- **Advantages**:
  - No external fragmentation
  - Easy to expand the size of a file
- **Disadvantages**:
  - Not suitable for random access within a file
  - Pointers take up some disk space
  - Difficult to recover a file if a pointer is lost or damaged
- Blocks may be collected into **clusters** of several blocks
  - Fewer pointers are necessary
  - Fewer disk seeks to read an entire file
  - Greater internal fragmentation



Indexed Allocation

Index → stored in inode

48

f1.txt

eg:- UFS, EXT2/3

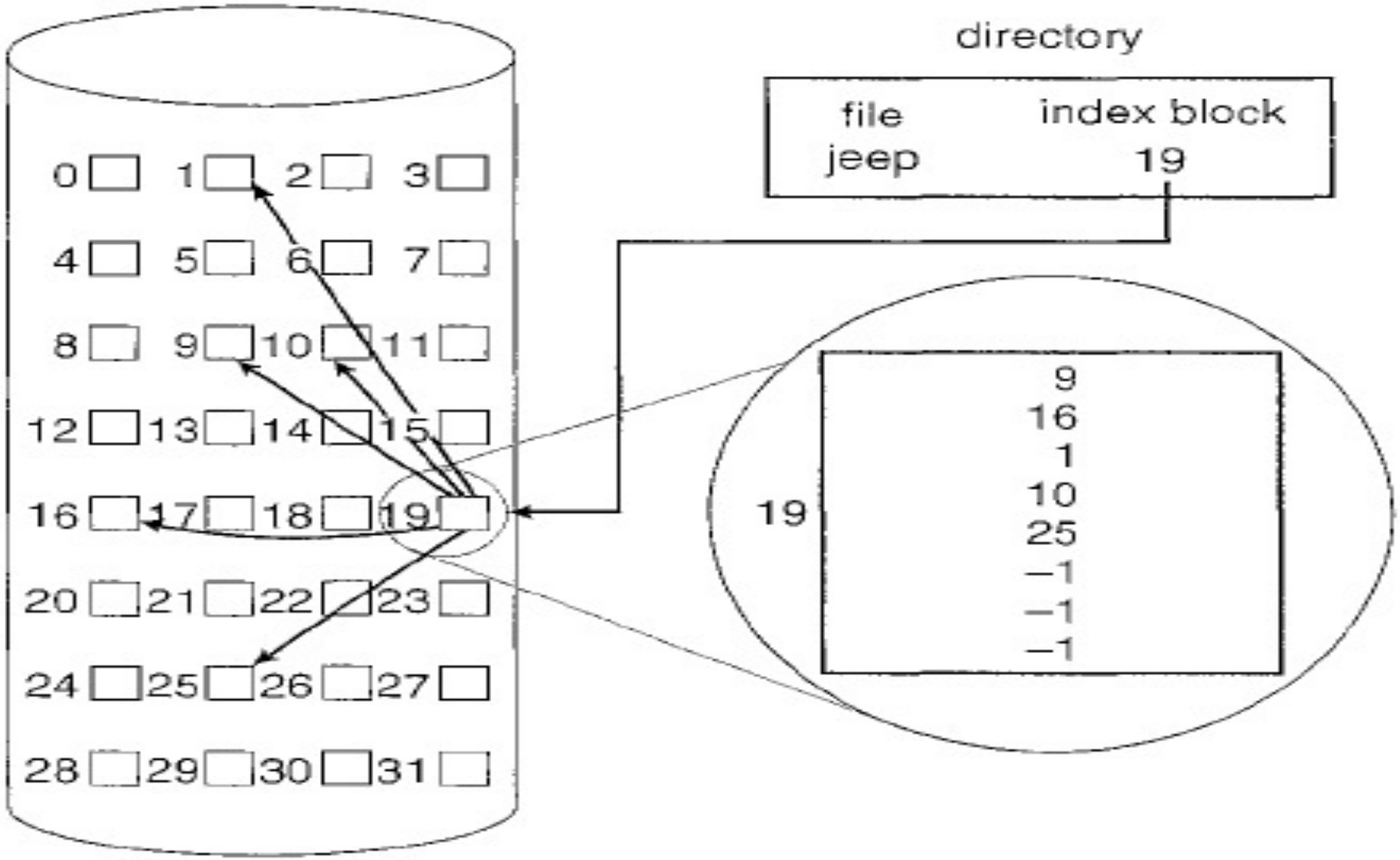
1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

26
44
64
94
77
19
-1
-1

Block / Indexed

- A special block known as the **Index block** contains the pointers to all the blocks occupied by a file.
- The ith entry in the index block contains the disk address of the ith file block.
- The directory entry contains the address of the index block.
- When the file is created, all pointers in the index block are set to *nil.*
- *When* the i^{th} block is first written, a block is obtained from the free-space manager and its address is put in the i^{th} index-block entry.

Indexed Allocation



Disk Scheduling

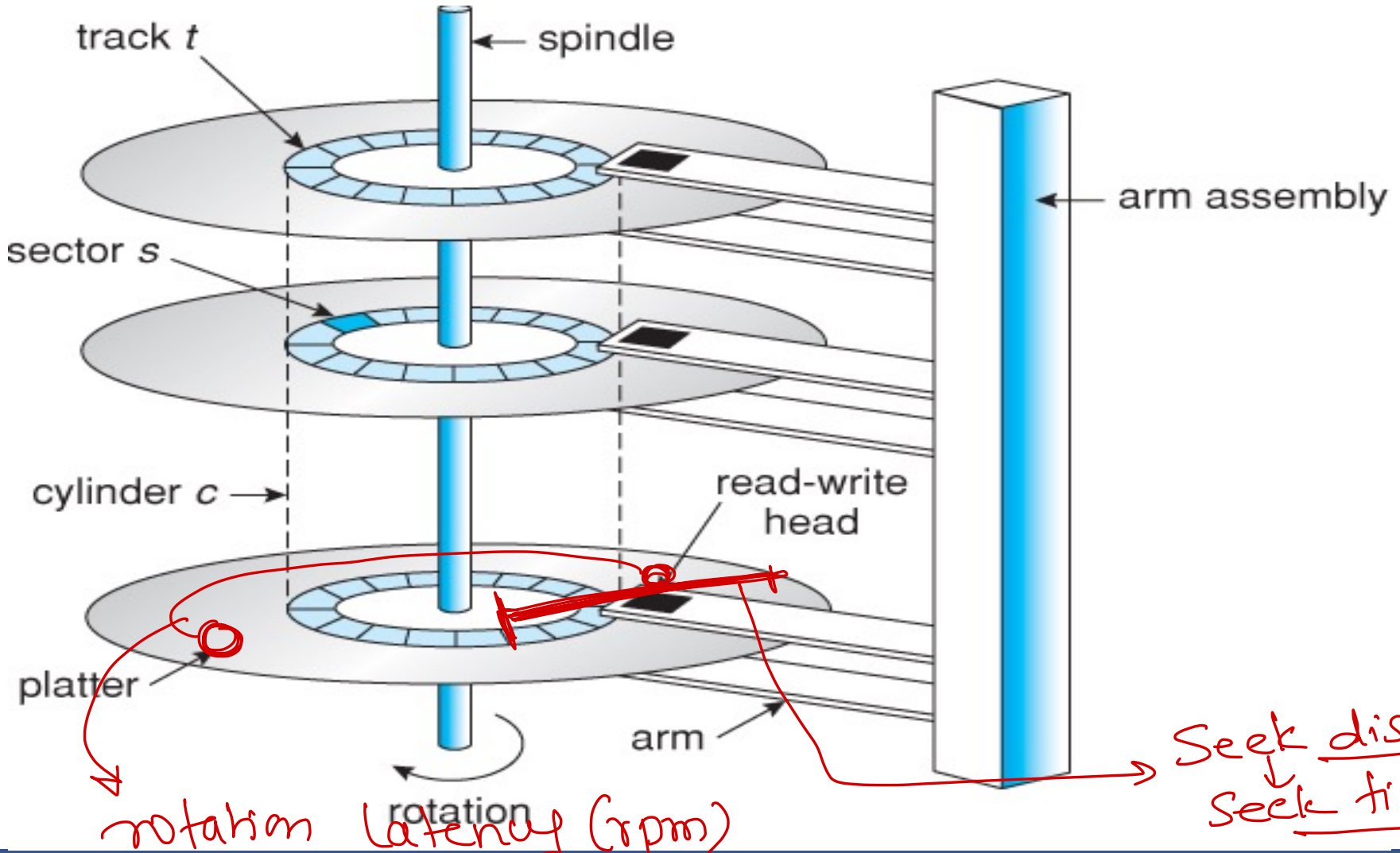
- **Disk scheduling** is done by operating systems to schedule I/O requests arriving for the disk.
- Disk scheduling is important because:
  - Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.
  - Two or more request may be far from each other so can result in greater disk arm movement.
  - Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.

Hard Disk

magnetic disk access = rotation latency + seek time + transfer data.

Response time

N → S = 0
S → N = 1



Disk Scheduling Criteria

Seek Time

- Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write.
- minimum average seek time is better.

Rotational Latency

- The time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads.
- minimum rotational latency is better.

Transfer Time

- The time to transfer the data.
- It depends on the rotating speed of the disk and number of bytes to be transferred.

Disk Access Time

- **SeekTime+Rotational Latency +Transfer Time**

Disk Response Time

- The average of time spent by a request waiting to perform its I/O operation.
- minimum variance response time is better.

Disk Scheduling Algorithms

First Come First Serve

- FCFS, the requests are addressed in the order they arrive in the disk queue.

Shortest Seek Time First

- SSTF (Shortest Seek Time First), requests having shortest seek time are executed first.
- it decreases the average response time and increases the throughput of system.

Scan/Elevator

- SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of disk, it reverses its direction and again services the request arriving in its path.

CSCAN

- disk arm moves in a circular fashion and this algorithm is also similar to SCAN algorithm and hence it is known as C-SCAN (Circular SCAN).

Look

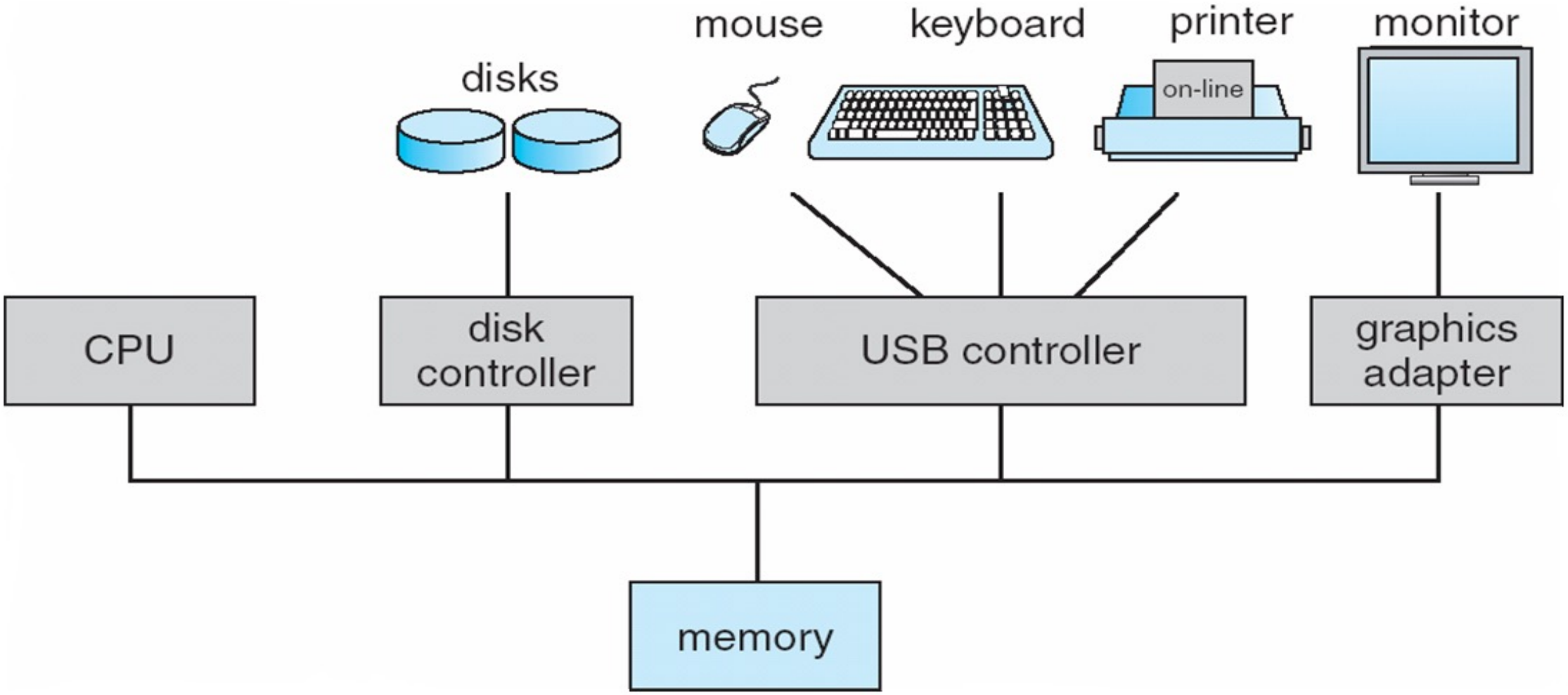
- similar to the SCAN disk scheduling algorithm except for the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only.

Clook

- CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request.

Computer Fundamentals

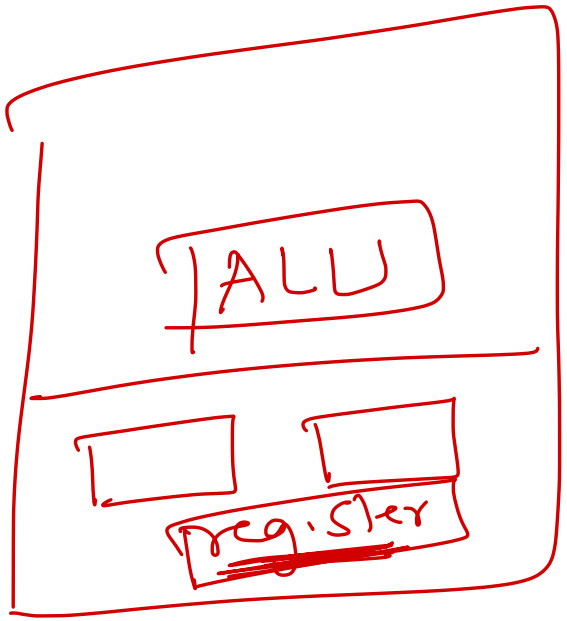
A Computer System



Computer Fundaments

Computer Structure

* CPU: Central Processing Unit
* RAM: Main memory
* Disk: Secondary storage
* IO Devices: Keyboard, Monitor, ...
* Bus: Set of wires connecting CPU to other peripherals
    * Address bus
    * Data bus
    * Control bus



Computer Fundaments

IO Device

* The Input unit allows programs and data to be entered into the computer.
    * e.g. Keyboard (primary), Mouse, Joystick, Touchpad, Touch pen, Scanner, Microphone, Webcam, Punch card, Bar code scanner, MICR scanner, Fingerprint, ...
* The Output unit allows the results of processing to be exported to the outside world or other devices or saved to be used later.
    - e.g. Monitor (primary), printer, plotter, Speakers, projector, ...

Computer System Components

1. **Hardware** – provides basic computing resources (CPU, Memory, I/O devices, Communication).
2. **Operating System** – controls and coordinates use of the hardware among various application programs for various users.
3. **System & Application Programs** – ways in which the system resources are used to solve computing problems of the users (Word processors, Compilers, Web browsers, Database systems, Video games).
4. **Users** – (People, Machines, other computers).

What happens when we start a computer?? (Booting Process)

- Hardware doesn't know where the operating system resides and how to load it.
- **Bootstrap Program :**
  - Initial program to run a system
  - Locating and Loading OS Kernel in main memory
- Where it is stored ??? ROM

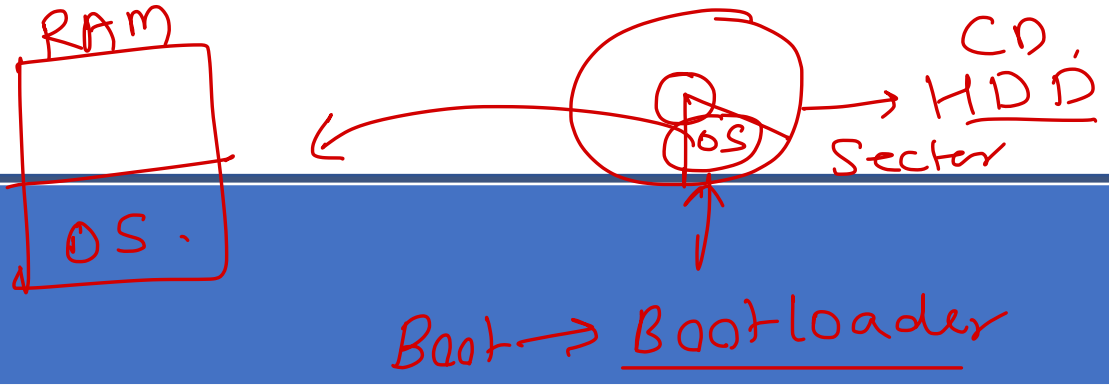
•If any storage device/partition contains one special program called as **"bootstrap program"** in its first sector i.e. in a boot sector then such a device/partition is referred as **bootable device/partition.**

•e.g. hard disk drive, pen drive, CD/DVD

Steps of Booting

1. Machine Boot

- When we switch on the power current gets passed to the motherboard and one program gets invoked named as **"BIOS"** which is exists in the ROM memory on motherboard.
- BIOS -- Basic Input Output System -- which is **a micro-program.**
- A micro-program is a program which is smaller in size and can be stored into the memory with its all possible set of input values.
- first step of BIOS is **"POST" - Power On Self Test**, under POST BIOS checks whether all peripherals are connected properly or not and their working status.
- **"peripherals or peripheral devices" --** devices which are connected to the motherboard externally are called as peripherals.
- after POST BIOS executes **"bootstrap loader",** bootstrap loader searches for available bootable devices and selects any one out of it as per the defined priorities.



2.System boot:

- if hard disk drive got selected as a bootable device and if it contains multiple OS's have installed on it, then "**bootloader**" program gets executes.
- **Boot loader program** displays list of operating system installed onto the machine, so that user can select any one at a time from and it invokes bootstrap program of selected operating system.
- Bootstrap program locates the kernel and load it into the main memory.