# Operating System Concepts

## SunBeam Institute of Information & Technology, Hinjwadi, Pune & Karad.

**# PreCAT Course**

**- Subject Name : Operating System Concepts.**

**- CCAT Exam : Section-B -- 9 Questions.**

**- Mostly all the questions are concept oriented.**

- **Reference Book:**

  1. Operating System Concepts, By Galvin.

* **Introduction:**
  - Why there is need of an OS?
  - What is an OS?
  - Booting in brief
  - Functions of an OS

* **Computer Fundamentals:**
  - Major Components: Processor, Memory Devices & IO Devices.
  - Memory Technologies and its characteristics
  - IO Techniques

# Operating System Concepts

* **UNIX System Architecture Design**
  - Major subsystem of an UNIX system: File subsystem & Process Control subsystem.
  - System Calls & its catagories
  - Dual Mode Operation

* **Process Management**
  - What is Process & PCB?
  - States of the process
  - CPU scheduling & CPU scheduling algorithms
  - Inter Process Communication: Shared Memory Model & Message Passing Model

# Operating System Concepts

* **Process Management**
  - Process Synchronization/Co-ordination
  - Deadlocks & deadlock handling methods

* **Memory Management**
  - Swapping
  - Memory Allocation Methods
  - Internal Fragmentation & External Fragmentation
  - Segmentation
  - Paging
  - Virtual Memory Management

* **File Management**
  - What is file?
  - What is filesystem & filesystem structure?
  - Disk space allocation methods
  - Disk scheduling algorithms

**Q. Why there is a need of an OS?**

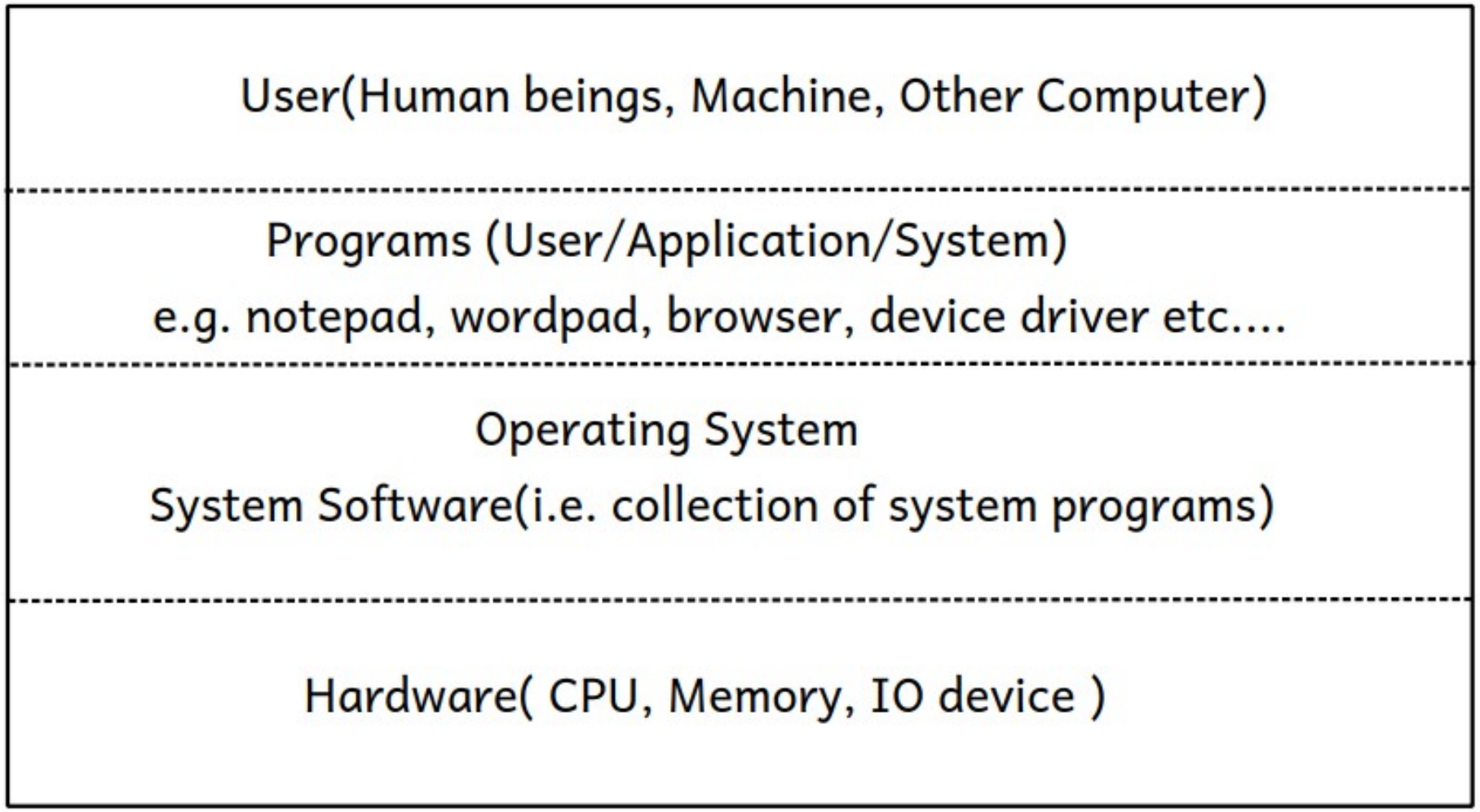
- Computer is a machine/hardware, does different tasks efficiently & accurately. Computer hardware mainly contains **Processor, Memory Devices & IO Devices.**

- Basic functions of computer:

1. **Data Storage**
2. **Data Processing**
3. **Data Movement**
4. **Control**

- As any user cannot communicates/interacts directly with computer hardware to do different tasks, and hence there is need of some interface between user and hardware.

# Operating System Concepts



## Q. What is an Operating System?

- An OS is a **system software** (i.e. collection of system programs) which acts as an interface between user and hardware.

- An OS also acts as an interface between programs and hardware.

- An OS allocates resources like main memory, CPU time, i/o devices access etc... to all running programs, hence it is also called as a **resource allocator.**

- An OS controls an execution of all programs and it also controls hardware devices which are connected to the computer system and hence it is also called as a **control program.**

**Q. What is an Operating System?**

- An OS manages limited available resources among all running programs, hence it is also called as a **resource manager.**

- From End User: An OS is a software (i.e. collection of programs) comes either in CD/DVD, has following main components:

**1. Kernel:** It is a core program/part of an OS which runs continuosly into the main memory does basic minimal functionalities of it.

e.g. Linux: vmlinuz, Windows: ntoskrnl.exe

**2. Utility Softwares:** e.g. disk manager, windows firewall, anti-virus software etc...

**3. Application Softwares:** e.g. google chrome, shell, notepad, msoffice etc...

**# Functions of an OS:**

**Basic minimal functionalities/Kernel functionalities:**

1. Process Management
2. Memory Management
3. Hardware Abstraction
4. CPU Scheduling
5. File & IO Management

**Extra utility functionalities/optional:**

6. Protection & Security
7. User Interfacing
8. Networking

**# Booting Process:**

- **Bootable Device/Bootable Partition:** if any storage device/partition contains a special program called as **bootstrap program** in its first sector i.e. in a boot sector then it is referred as a bootable device/partition.

**- Peripheral Devices:** devices which are connected to the motherboard externally referred as peripheral devices or **peripherals.**

**- There are two steps of booting:**

**1. Machine Boot**

**2. System Boot**

## 1. Machine Boot:

**Step1:** when we switch on the power supply current gets passed to the motherboard, on which ROM memory is there which contains one microprogram referred as **BIOS**, gets executes first.

First step of BIOS is **POST(Power On Self Test)**, under POST it checks wheather all peripherals are connected properly or not and their working status.

**Step2:** after POST, BIOS executes **bootstrap loader program**, which searches for an available bootable devices in a system and as per the priority decided in a BIOS settings it selects any one bootable device at a time.

## 2. System Boot:

**Step3:** upon selection of a bootable device, **bootloader program** (e.g. Linux - GRUB, LILO, Windows - NTLDR) in that bootable device gets executes and displays list of names of an operating systems installed on it, from which user need to select any one OS for booting.

**Step4:** upon selection of any one OS from the list, **bootstrap program** of that OS gets invoked, which first locates the kernel and load it into the main memory.

- Kernel of any OS resides inside the main memory till we do not shutdowns the machine.

**# Computer Fundamentals:**

- Computer is a hardware mainly contains:

**1. Processor**

**2. Memory Devices**

**3. IO Devices**

- There are two basic/fundamental **structural** and **functional units** of computer hardware system:

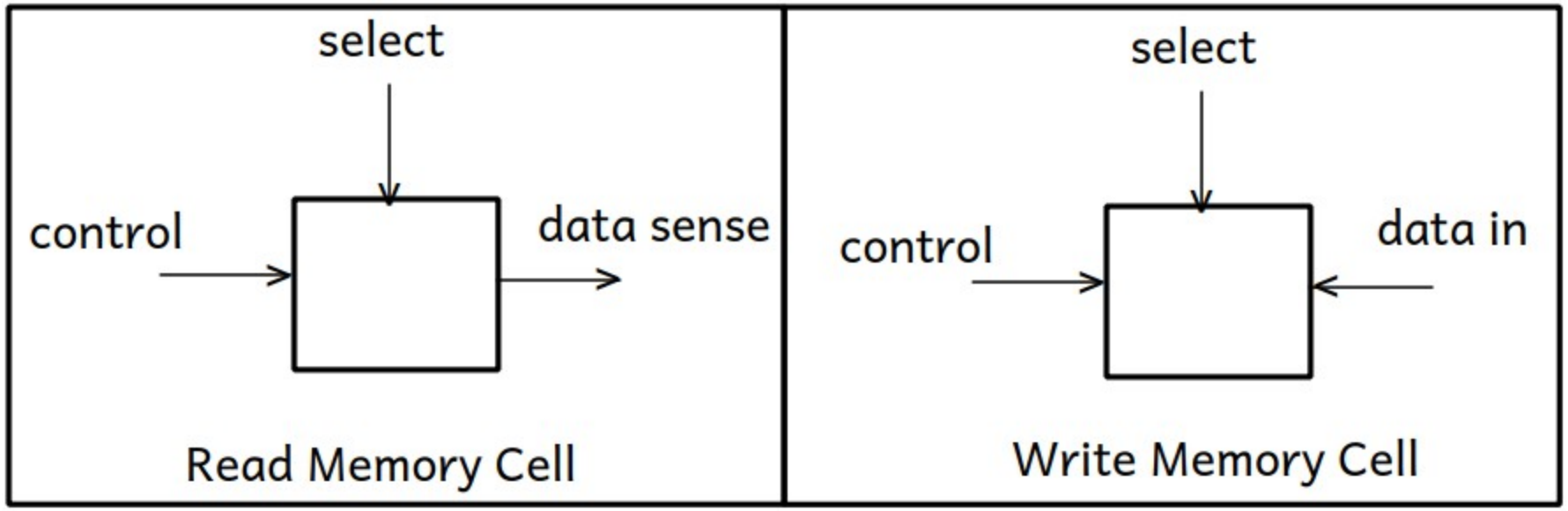
1. Memory cell

2. Gates

- As the data stoarage can be acheived by memory cell, data processing, data movement and control functions can be achieved by using gates, hence memory cell & gates are referred as functional units, wheras computer is a digital device made up of collection of millions of memory cells & gates, and hence memory cell & gates are referred as structural units as well.

## # Computer Fundamentals:

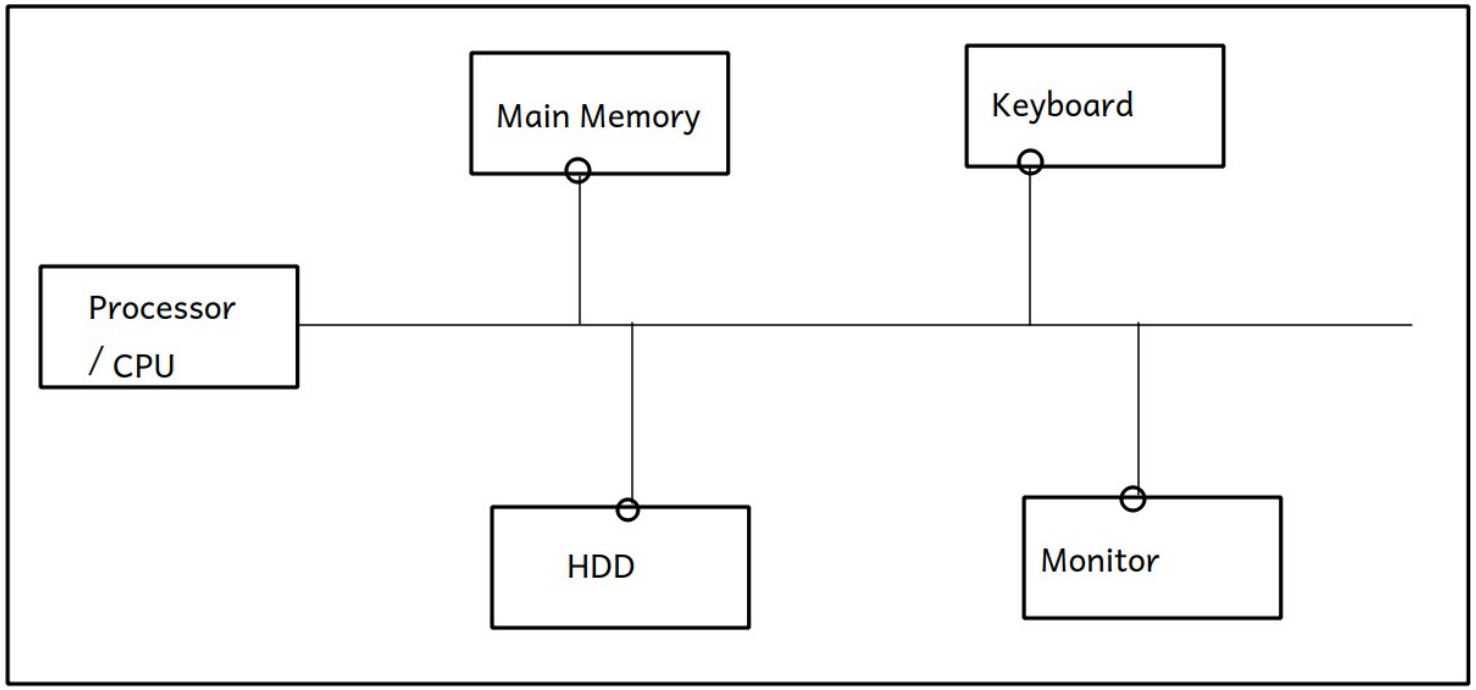
**1. Memory cell:** memory has the capacity to store one value at a time in it either 0 OR 1.




**2. Gates:** gate a basic digital device (flip-flop gates) can be used for performing arithmetic & logic operations as well as for control.

# Processor:

- Processor contains: ALU(Arithmetic Logic Unit), CU(Control Unit), Registers.
- Processor is also called as **CPU(Central Processing Unit).**



# Operating System Concepts

- Computer system mainly contains Processor, Main Memory, Hard Disk Drive, Keyboard & Monitor.

- In a Computer system each and every device has got its own dedicated processor which controls operations of that device specifically, and there exists one processor which controls all the operations and devices in a computer system centrally by co-ordinating with dedicated processors.

For Example: Hard Disk Drive has **disk controller** which controls all disk operations, whereas processor in a block diagram controls all operations

- All components in a computer are connected via conducting wires through which transfer of data, addresses and control signals takes place, these conducting wires are referred as **buses.**

# Operating System Concepts

- Three types of buses are there:

1. **Data Bus:** transfers data
2. **Address Bus:** transferes addresses
3. **Control Bus:** transferes control signals

- Major components of computer system ( i.e. components which are onto the motherboard like CPU, Cache Memory, Main Memory etc...) are connected via a bus referred as a **system bus**.

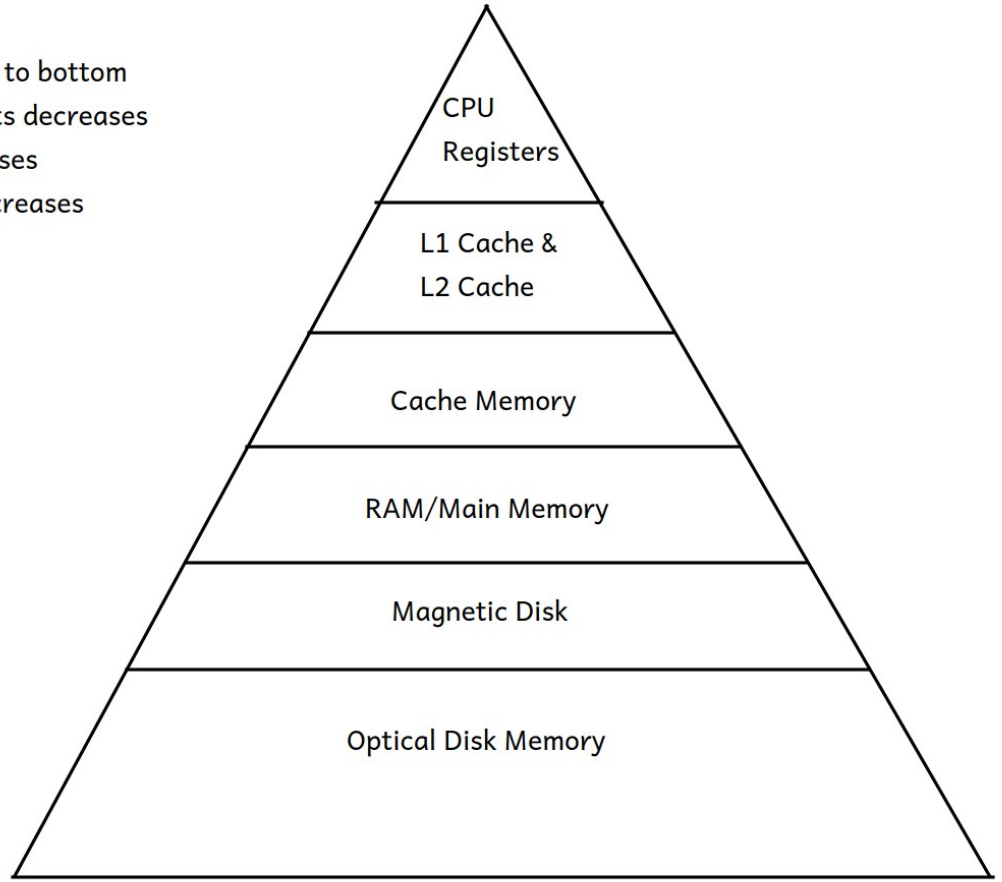
- As data which is sent by one component can be recieved by any other component connected to the bus, hence it is a **shared communication pathway.**

- Control signals sent by the CPU to other devices referred as commands.

1. **TEST:** to check the status of any device
2. **WRITE**
3. **READ**
4. **CONTROL**

# # Computer Memory Technologies:

as we go from top to bottom
- access speed gets decreases
- cost gets decreases
- capacity gets increases




Computer Memory Hierarchy

**# Computer Memory Technologies**:

- **CPU Registers:** memory which is very close to the CPU are registers which is at the top in a computer memory heirarchy.

- Instructions and data currently executing by the CPU can be kept temporarily into the CPU registers.

- MAR, MBR, IOAR, IOBR, PC, SP, Accumulator etc...

- Computer memory can be catagorised into two catagories as per its location: **Internal Memory & External Memory.**

**- Internal Memory:** memory which is internal to the motherboard is referred as an internal memory.

e.g. CPU registers, L1 & L2 cache Cache memory, RAM.

**- External Memory:** memory which is external to the motherboard  is referred as an external memory.

e.g. magnetic disk, optical disk, magetic tape etc...

**# Computer Memory Technologies**:

- Computer memory can also catagorised into two catagories: **Primary Memory & Secondary Memory.**

**- Primary Memory:** memory which can be accessible directly by the CPU is referred as primary memory, i.e. memory which can accessible by the CPU with the help of instruction set  having with the CPU.

e.g. CPU registers, L1 & L2 Cache, Cache Memory, RAM

**- Secondary Memory:** memory which cannot be accessed directly by the CPU is referred as secondary memory.

e.g. Magetic Disk, CD/DVD, PD etc..

- If the CPU want to access disk contents, first it gets fetched into the RAM and then it can be accessed by the CPU from RAM.

- As for an execution of every program RAM memory is must and hence RAM is also called as **Main memory**.

**Q. Why there is a need of cache memory?**

- As the rate at which the CPU can execute instructions is faster than the rate at which data can be accessed from the main memory, so even the CPU is very fast, with the same speed data do not gets fetched from the main memory for execution, hence due to this speed mismatch overall system performance gets down.

- To reduce speed match between the CPU and the main memory Cache memory (hardware) can be added between them and system performance can be increases by means reducing speed mismatch.

**Q. What is Cache Memory?**

- Cache memory is **faster** memory, which is a type RAM i.e. **SRAM**, in which **most recently accessed main memory contents** can be kept/stored in an **associative manner i.e. in a key-value pairs.**

**- There are two types of RAM:**

**1. DRAM (Dynamic RAM):** memory cells are made up of capacitors

- Main memory is as example of DRAM.

**2. SRAM (Static RAM):** memory cells are made up of flip-flop gates

- Cache Memory is an example of SRAM.

- Cache Memory has **C** no. of lines, whereas each line is divided into two parts, each line contains k words of data (rcently accessed main memory contents ) and its main memory addresses can be kept in few tag bits.

**1. First part** of a line : **few tag bits** contains main memory addresses of k words of data in that line

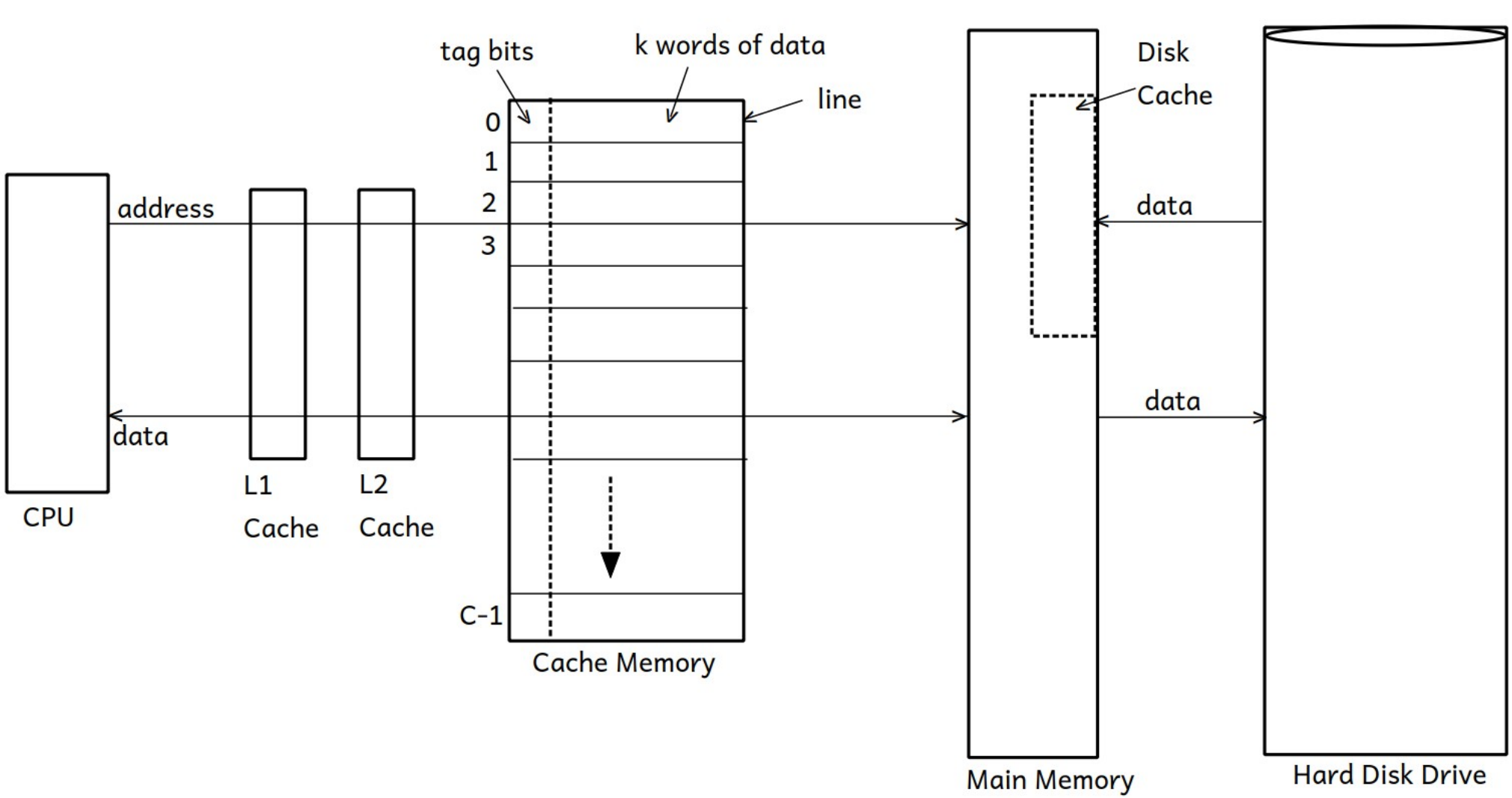
**2. Second part** of a line contains k words of data.

- When the CPU want to fetch data from the main memory it requests for its address, and this requested address gets searched into the cache memory first, if requested addr is found in the cache memory then data also found in a cache memory, it is referred as **cache hit**, whereas if the requested address and hence data is not found in a cache memory then it is referred as a **cache miss,** in that data gets fetched from main memory and gets transferred to the CPU via cache memory only.

- Even after adding cache memory between the CPU and main memory, the rate at which the CPU can execute instructions is faster than the rate at which at which data can be accessed from cache memory, and hence to reduce speed mismatch between the CPU and cache memory one or more levels of cache memory i.e. **L1 cache & L2 cache** can be added between them.

- **Disk Cache:** it is **purely a software technique** in which portion of the main memory can be used as a cache memory in which most recently accessed disk contents can be kept in an associative manner, so whenever the CPU want to access data from hard disk drive it first gets searched into the disk cache.

- Disk cache technique is used to reduce speed mismatch between the CPU and Secondary memory.


tag bits
k words of data
line
0
1
2
3
C-1
Cache Memory
address
data
CPU
L1
Cache
L2
Cache
Disk
Cache
data
data
Main Memory
Hard Disk Drive

# Computer Fundamentals: Memory Technologies

There are four methods by which data can be accessed from the computer memory:

1. Sequential Access: e.g. Magentic Tape
2. Direct Access: e.g. Magnetic Disk
3. Random Access: e.g. RAM Memory
4. Associative Access: e.g. Cache Memory

# Computer Fundamentals: IO Devices

Input Output Devices

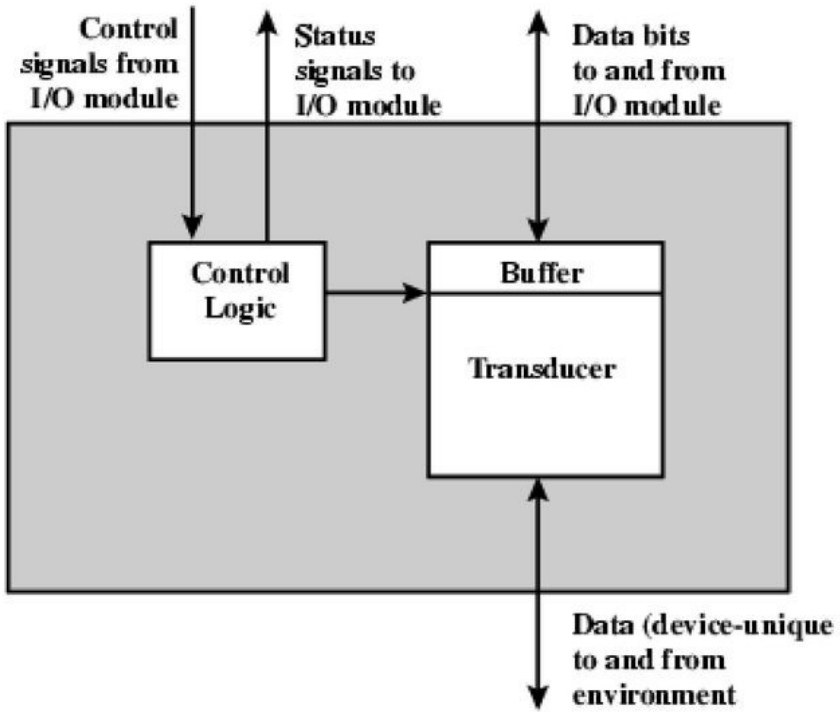
- Devices which are connected to the motherboard externally through ports reffered as **peripheral devices** or **peripherals**.

- An IO Devices are also reffered as an external devices.

**# Input Devices:** Keyboard, Mouse, Scanner, Bar Code Reader, Eye Recognition System, Voice Recognition System, Touch Pad, Touch Screen etc...

**# Output Device:** Monitor, Printer, Speakers, Projector etc...

## External Device Block Diagram



**Structure of an External Device:**

- External Device has three major blocks:

**1. Control Logic Block(Controller):** controlls all the operations of that device.

**2. Buffer:** each device has gots its own memory in which data can be stored temporarily reffered as a buffer.

**3. Transducer:** this component converts any other form of energy into an electrical enery and converts an electrical energy into another form, this block of an external device is used to do commuincation with the outside world.

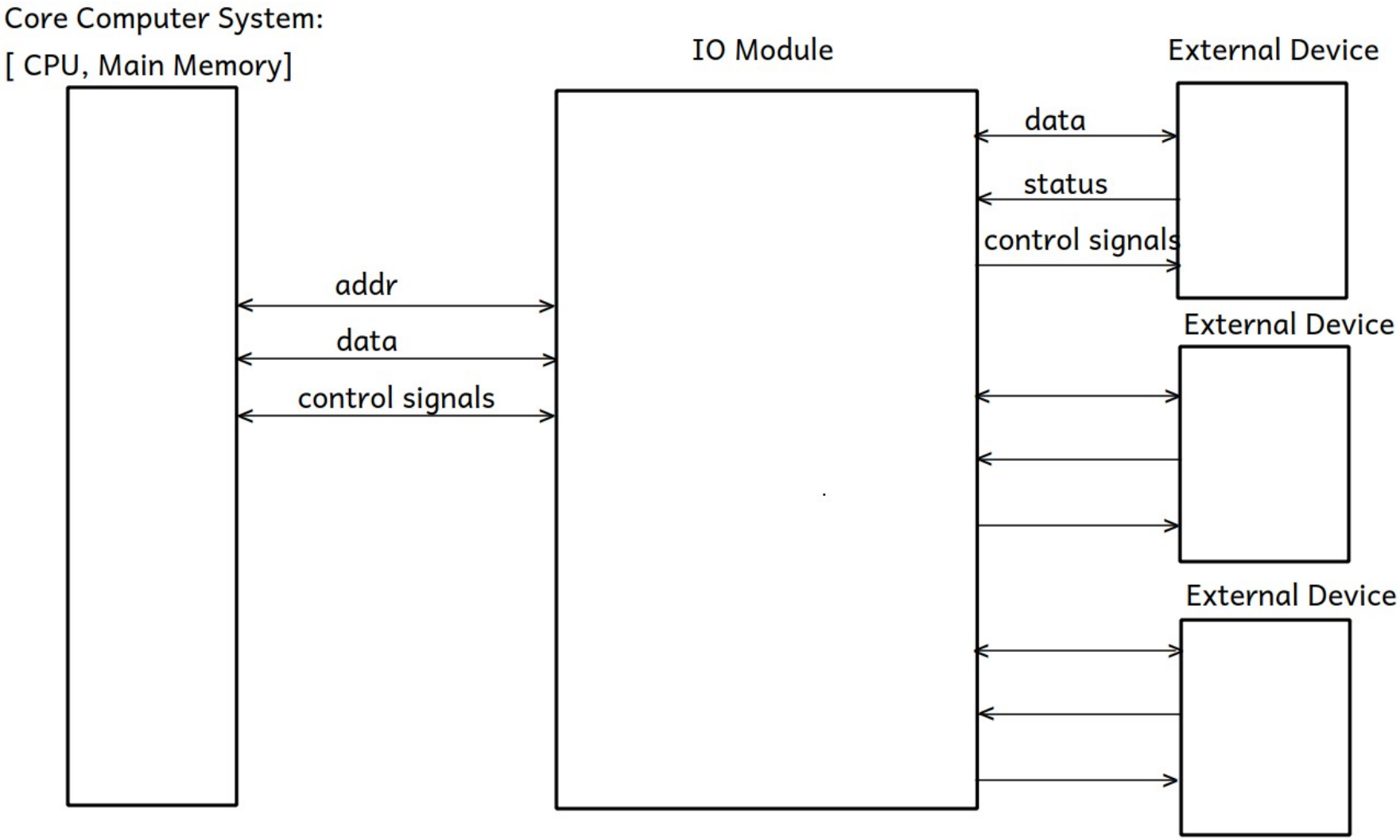
# IO Modules/IO Ports:

- Core Computer system is not able to communicates directly with any external device and hence IO modules acts as an interface between core computer system and an io devices.

- IO Modules contains all the logic to communicates with an io devices.

- Single IO module can be used for communication between one device or with more one devices as well.

# Computer Fundamentals: IO Devices



# Computer Fundamentals: IO Techniques

- Whenever there is transfer of data either from core computer system (i.e. Bus) to an IO devices or vice-versa, it is reffered as an IO.

**- There are three IO techniques:**

**1. Program driven IO**

**2. Interrupt IO**

**3. DMA i.e. Direct Memory Access**

## 1. Program driven IO:

- All the logic/steps required for an IO is there into one program, and by means of executing that program by the CPU io can be done.

**# Advantages:**

- Simple

**# Disadvantages:**

- As the CPU remains wholely involved in an IO, less CPU utilization, and hence system performance is low.

## 2. Interrupt IO:

### # What is an interrupt?

- An interrupt is a signal recieved by the CPU due to which it stops an execution of one job and starts an execution of another job.

### # Advantages:

- In this IO, the CPU remains involved in an IO whenever gets interrupted, and hence its utilization can be maximized.

### # Disadvantages:

- When there is a data transfer between main memory & secondary memory unneccessary involvement of the CPU is there.

## 3. DMA (Direct Memory Access):

- Whenever there is a transfer of data between core computer system and io devices (e.g. main memory and secondary memory), the CPU initiates an IO and gives control of an IO process to the DMA controller, and hence onwards that IO process is controlled by the DMC controller till the end i.e. the DMA controller will work on behalf of the CPU and after finishing an IO it sends acknowledgement to the CPU, and by the time the CPU can execute another jobs, and utilization of the CPU can be maximized further.

e.g. **8237 DMA controller.**

**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.

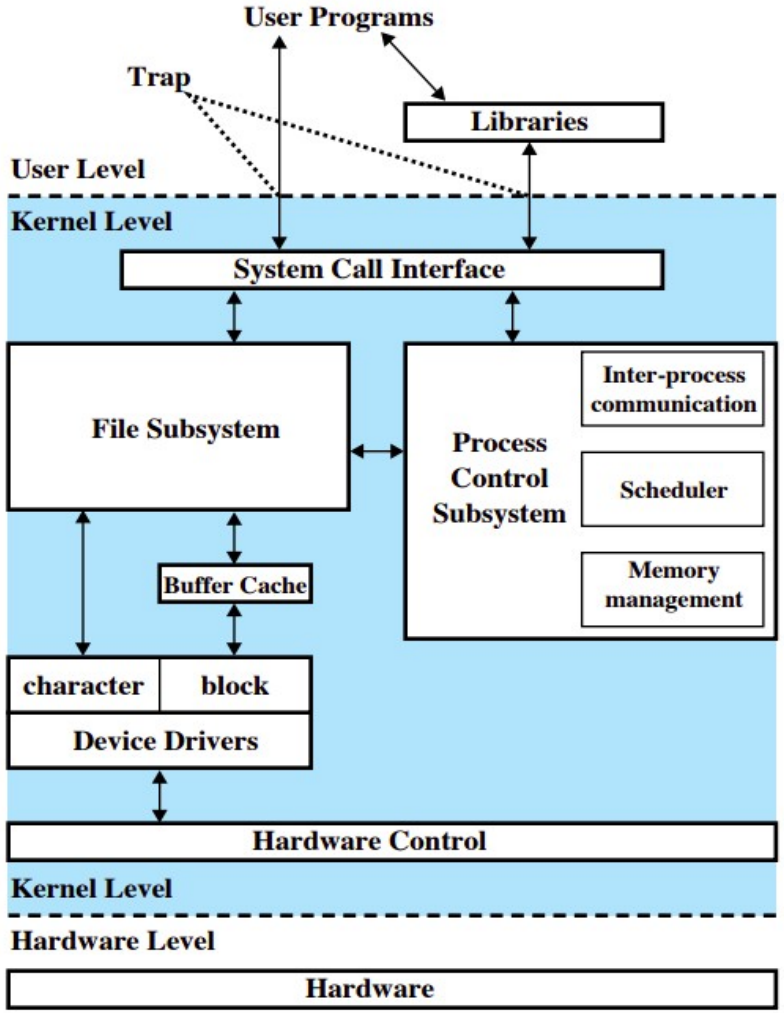
- It was first run on a machine **DEC-PDP-7** (Digital Equipment Corporation – Programmable Data Processing-7).

- 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



# Operating System Concepts

- Kernel acts as an interface between programs and hardware.

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

- There are two major subsystems:

1. Process Control Subsystem

2. File Subsystem

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

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

- From UNIX point of view all devices are considered as a file
- In UNIX, devices are catagorises into two catagories:

**1. Character Devices:** Devices from which data gets transfered character by character --> character special device file

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

**2. Block Devices:** Devices from which data gets transfered 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.

# Operating System Concepts

- Hardware Control Layer/Block does communication with control logic block i.e. controller of a hardware.

**# System Calls:** are the functions defined in a C, C++ & Assembly languages, which provides interface of services made available by the kernel for the user (programmer user).

- If programmers want to use kernel services in their programs, it can be called directly through system calls or indirectly through set of libary functions provided by that programming language.

- There are 6 catagories of system calls:

1. **Process Control System Calls:** e.g. fork(), _exit(), wait() etc...
2. **File Operations System Calls:** e.g. open(), read(), write(), close() etc...
3. **Device Control System Calls:** e.g. open(), read(), write(), ioctl() etc...

**4. Accounting Information System Calls:** e.g. getpid(), getppid(), stat() etc...

**5. Protection & Security System Calls:** e.g. chmod(), chown() etc...

**6. Inter Process Communication System Calls:** e.g. pipe(), signal(), msgget() etc...

- In UNIX 64 system calls are there.

- In Linux more than 300 system calls are there

- In Windows more than 3000 system calls are there

- When system call gets called the CPU switched from user defined code to system defined code, and hence system calls are also called as **software interrupts/trap.**

**4. Accounting Information System Calls:** e.g. getpid(), getppid(), stat() etc...

**5. Protection & Security System Calls:** e.g. chmod(), chown() etc...

**6. Inter Process Communication System Calls:** e.g. pipe(), signal(), msgget() etc...

- In UNIX 64 system calls are there.

- In Linux more than 300 system calls are there

- In Windows more than 3000 system calls are there

- When system call gets called the CPU switched from user defined code to system defined code, and hence system calls are also called as **software interrupts/trap.**

**# Dual Mode Operation:**

- System runs in two modes:

1. System Mode

2. User Mode

**1. System 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/priviledged 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 an execution of any program, the CPU keeps switcesh in between kernel mode and user mode and hence system runs in two modes, it is reffered as **dual mode operation.**

- To differentiate 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 wheather 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.**

## # Process Management

- When we say an OS does process management it means an OS is responsible for process creation, to provide environment for an execution of a process, resource allocation, scheduling, resources management, inter process comminication, process coordination, and terminate the process.

## Q. What is a Program?

- **User view:** Program is  a set of instructions given to the machine to do specific task.

- **System view:** Program is an executable file divided into sections like exe header, bss section, data section, rodata section, code section, symbol table.

**Q. What is a Process?**

**User view:**

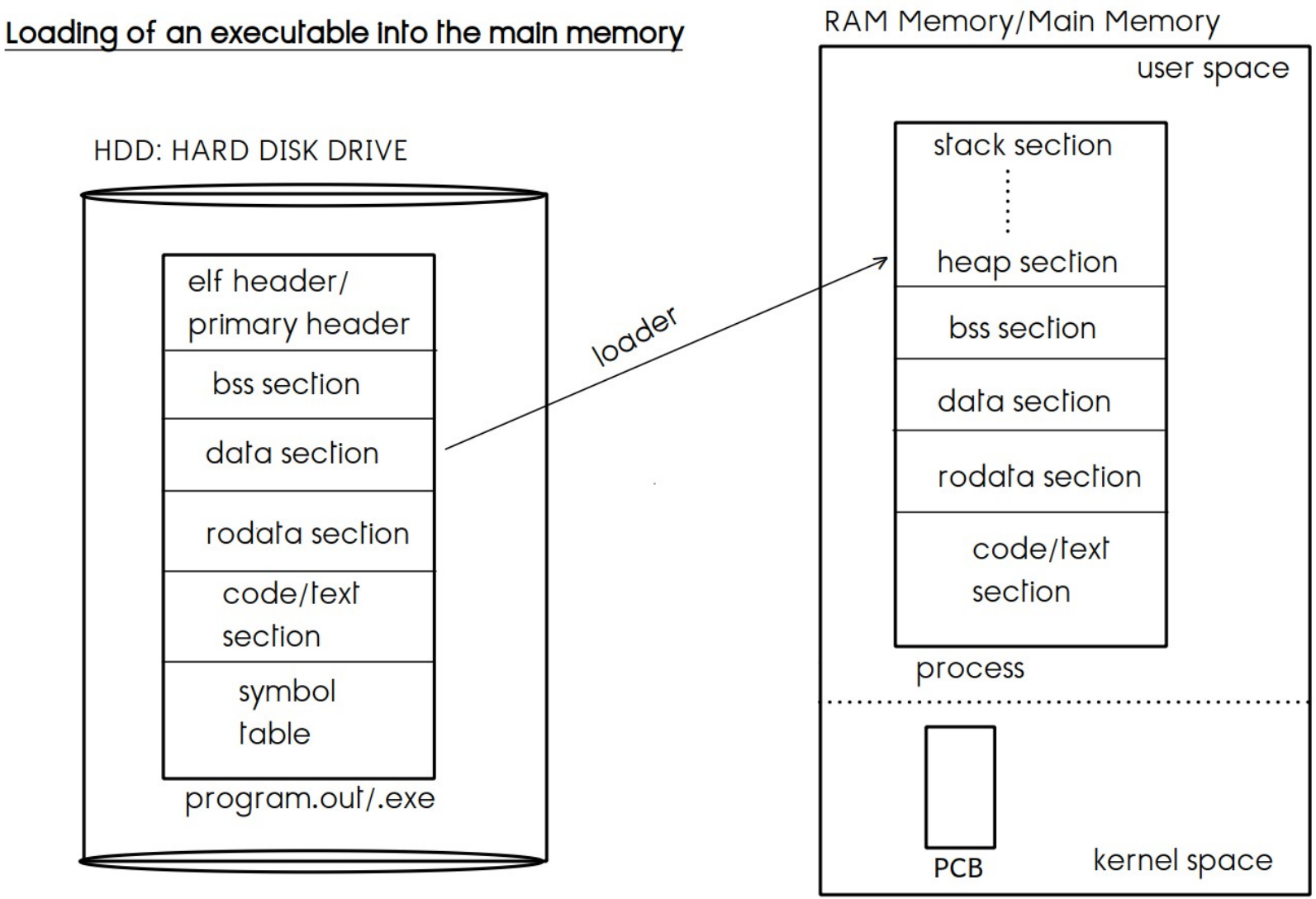
- Program in execution is called as a process.
- Running program is called as a process.
- When a program gets loaded into the main memory it is reffered as a process.
- Running instance of a program is reffered as a process.

**System view:**

- Process is a file loaded into the main memory which has got bss section, rodata section, code section, and two new sections gets added for the process:

**stack section:** contains function activation records of called functions.

**heap section:** dynamically allocated memory


Loading of an executable into the main memory
HDD: HARD DISK DRIVE
elf header/
primary header
bss section
data section
rodata section
code/text
section
symbol
table
program.out/.exe
loader
RAM Memory/Main Memory
user space
stack section
heap section
bss section
data section
rodata section
code/text
section
process
PCB
kernel space

# Operating System Concepts

- As a kernel, core program of an a OS runs continuosly into the main memory, part of the main memory which is occupied by the kernel reffered as kernel space and whichever part is left is reffered as an user space, so main memory is divided logically into two parts: kernel space & user space.

- User programs gets loaded into the user space only.

- When we execute a program or upon submission of a process very first one structure gets created into the main memory inside kernel space by an OS in which all the information which is required to control an execution of that process can be kept, this structure is reffered as a **PCB: Process Control Block**, is also called as a **Process Descriptor.**

- Per process one PCB gets created and PCB remains inside the main memory throughout an execution of a program, upon exit PCB gets destroyed from the main memory.

- PCB mainly contains: PID, PPID, PC, CPU sched information, memory management information, information about resources allocated for that process, execution context etc...

**# Process States:**

- Throughout execution, process goes through different states out of which at a time it can be only in a one state.

- States of the process:

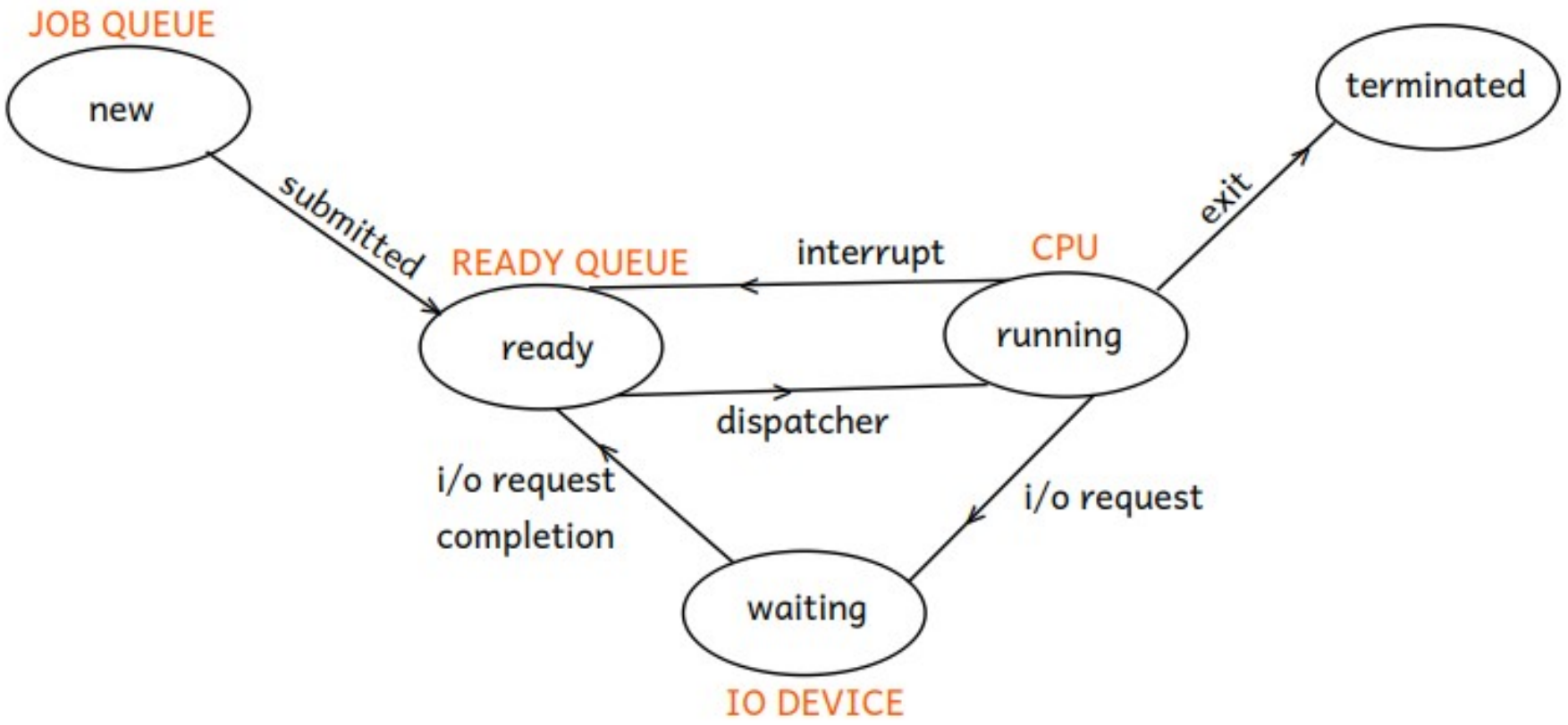
**1. New state:** upon submission or when a PCB for a process gets created into the main memory process is in a new state.

**2. Ready state:** after submission, if process is in a main memory and waiting for the CPU time.

**3. Running state:** if currently the CPU executing a process then it is reffered as in a running state.

**4. Waiting state:** if a process is requesting for i/o device then it is considered as that process is in a waiting state.

**5. Terminated state:** upon exit process goes into terminated state and its PCB gets destroyed from the main memory.




PROCESS STATE DIAGRAM

# Operating System Concepts

**# Features of an OS:**

**1. multi-programming:** system in which more than one processes can be submitted/an execution of more than one programs can be started at a time.

- **degree of multi-programming:** no. of programs that can be submitted into the system at a time.

**2. multi-tasking:** system in which it seems that the CPU can execute more than one programs simultaneously/concurrently.

- time-sharing: system in which CPU time gets shared among all running programs.

**3. multi-threading:** system in which it seems that the CPU can execute more than one threads which are of either same process or are of different processes simultaneously/concurrently.

**4. multi-processor:** system can run on a machine in which more than one CPU's are connected in a closed circuit.

**5. multi-user:** system in which multiple users can loggedin at a time.

**# Process States:**

- To keep track on all running programs an OS maintains few data structures reffered as kernel data structures:

**1. Job queue:** it contains list of PCB's of all submitted processes.

**2. Ready queue:** it contains list of PCB's of processes which are in a main memory and waiting for the CPU time.

**3. Waiting queue:** list of PCB's of processes which are requesting for that particular device.

**1. Job Scheduler/Long Term Scheduler:** it is a system program which selects/schedules jobs/processes from job queue to load them onto the ready queue.

**2. CPU Scheduler/Short Term Scheduler:** it is a system program which selects/schedules job/process from ready queue to load it onto the CPU.

**- Dispatcher:** it is a system program which loads a process onto the CPU which is scheduled by the CPU scheduler, and the time required for the dispatcher to stops an execution of one process and to starts an execution of another process is reffered as **dispatcher latency.**

**# Context Switch:**

- As during context-switch the CPU gets switched from an execution context of one process onto an execution context of another process, and hence it is reffered as "context-switch".

- context-switch = state-save + state-restore

- **state-save** of suspended process can be done i.e. an execution context of suspended process gets saved into its PCB.

**- state-restore** of a process which is scheduled by the CPU scheduler can be done by the dispacther, dispather copies an execution context of process scheduled by the cpu scheduler from its PCB and restore it onto the CPU registers.

- When a high priority process arrived into the ready queue, low priority process gets suspended by means of sending an interrupt, and control of the CPU gets allocated to the high priority process, and its execution gets completed first, then low priority process can be resumed back, i.e. the CPU starts executing suspended process from the point at which it was suspended and onwards.

- CPU Scheduler gets called in the following four cases:

1. Running -> Terminated
2. Running -> Waiting
3. Running -> Ready
4. Waiting -> Ready

- There are two types of CPU scheduling:

**1. Non-preemptive:** under non-preemptive cpu scheduling process releases the control of the CPU by its own i.e. voluntarily.

e.g. in above case 1 & case 2

**2. Preemptive:** under preemptive scheduling control of the CPU taken away forcefully from the process.

e.g. in above case 3 & 4.

**- Following algorithms used for CPU Scheduling:**

**1. FCFS (First Come First Served) CPU Scheduling**

**2. SJF (Shortest Job First) CPU Scheduling**

**3. Priority CPU Scheduling**

**4. Round Robin CPU Scheduling**

- As multiple algorithms are there for CPU scheduling and hence there are certain criterias which algorithm is best suited in which situation and which one efficient are reffered as CPU scheduling criterias.

**# CPU Scheduling Criterias:**

**1. CPU Utilization:** one need to select such an algorithm in which utilization of the CPU must be as maximum as a possible.

**2. Throughput:** total work done per unit time. One need to select such an a algorithm in which throughput must be as maximum possible.

**3. Waiting Time:** it is the toal amount of time spent by the process in a ready queue for waiting to get control of the CPU from its time of submission. One need to select such an algorithm in whih waiting time must be as minimum as possible.

**4. Response Time:** it is a time required for the process to get first response from the CPU from its time of submission.

**5. Turn-Around-Time:** it is the total amount of time required for the process to complete its execution from its time of submission.

- One need to select such an algorithm in which turn-around-time must be as minimum as possible.

- Turn-around-time is the sum of periods spent by the process on ready queue for waiting and onto the CPU for execution from its time of submission.

**Execution Time:** it is the total amount of time spent by the process onto the CPU to complete its execution.

**CPU Burst Time:** total no. of CPU cycles required for the process to complete its execution.

- **Turn-Around-Time = Waiting Time + Execution Time.**

## 1. FCFS: First Come First Served

- In this algorithm, process which arrived first into the ready queue gets the control of the CPU first i.e. control of the CPU gets allocated for processes as per their order of an arrival into the ready queue.

- This algorithm is simple to impelement and can be implemented by using fifo queue.

- It is a non-preemptive scheduling algorithm.

**# Convoy effect:** due to an arrival of longer processes before shorter processes, shorter processes has to wait for long time and their average waiting time gets increases, which results into an increase of an average turn-around-time and hence overall system performance gets down.

## 2. SJF(Shortest Job First):

- In this algorithm, process which is having minimum CPU burst time gets the control of the CPU first.
- SJF algorithm ensures minimum waiting time.
- Under non-preemptive SJF, algorithm fails as the submission time of all processes are not same, and hence it can be implemented as  preemptive as well.
- Non-preemptive SJF is also called as **SNTF(Shortest-Next-Time-First).**
- Preemptive SJF is also called as **SRTF(Shortest-Remaining-Time-First).**

## 3. Priority Scheduling

- In this algorithm, each process is having priority in its PCB and process which is having highest priority gets control of the CPU first.

- Minimum priority value indicates highest priority.

- This algorithm is purely preemptive.

- Due to the very low priority process may gets blocked into the ready queue and control of the CPU will never gets allocated for such a process, this situation is reffered as a **starvation** or **indifinite blocking.**

- **Ageing:** it is a technique in which, an OS gradually increments the priority of blocked process, i.e. priority of blocked process gets incremented after some fixed time interval by an OS, so that priority of blocked process becomes sufficient enough to get control of the CPU, and starvation can be avoided.

## 4. Round Robin Scheduling Algorithm

- In this algorithm, fixed **time slice** or **time quantum** gets decided in advanced, and at a time control of the CPU may remains allocated with any process maximum for decided time-slice, once the given time slice is finished process gets suspended and control of the CPU gets allocated to the next process for maximum the decided time slice and so on each process gets control of the CPU in a round robin manner.

- If any process completes its execution before allocated time slice then control of the CPU gets allocated for the next process as soon as it is completed for maximum utilization of the CPU.

- This algorithm is purely preemptive.

- This algorithm ensures minimum response time.

**Inter Process Communication**

- Processes running into the system can be divided into two catagories:

**1. Independent Processes:**

- Process which do not shares data with any other process reffered as an independent process.

- Process which  do not affects or not gets affected by any other process reffered as an independent process.

**2. Cooperative Processes:**

- Process which shares data with any other process reffered as cooperative process.

- Process which  affects or gets affected by any other process reffered as cooperative process.

## Inter Process Communication

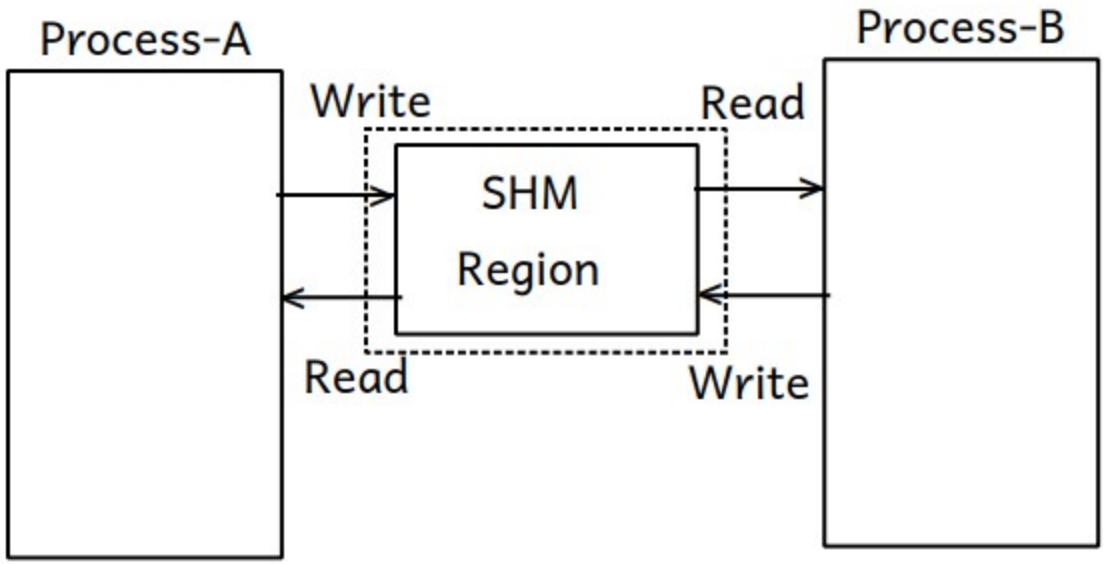
- Inter process communication takes place only between cooperative processes.

- There are two techniques by which IPC can be done/there are two IPC Models:

**1. Shared Memory Model:** under this technique processes can communicates with each other by means of reading and writing data into the shared memory region which is provided by an OS on request by a processes want to communicates.

**2. Message Passing Model:** under this technique, processes can communicates with each other by means of sending message.

- Shared Memory Model is faster than Message Passing Model.




SHARED MEMORY MODEL

**Inter Process Communication**

**2. Message Passing Model:** there are further different IPC techniues under message passing model.

**i. Pipe:**

- By using pipe mechanism one process can send message to another process, vice-versa is not possible and hence it is a unidirectional communication technique.

- By using Pipe only processes which are running on the same system can communicates.

## Inter Process Communication

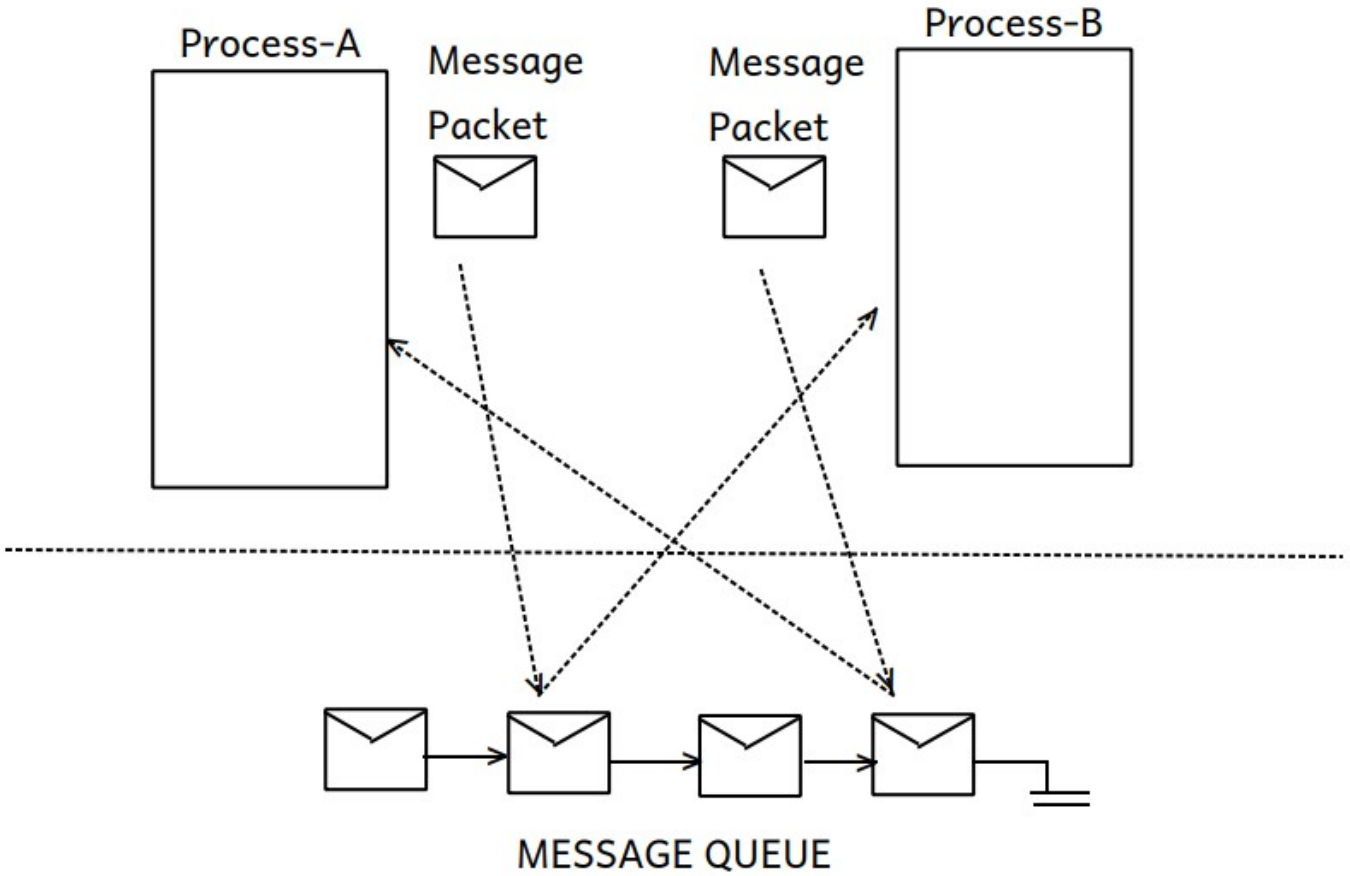
### ii. Message Queue:

- By using message queue technique, processes can communicates by means of sending as well as recieving message packets to each other, and hence it is a bidirectional communication.

- **Message Packet: Message Header(Information about the message) + Actual Message.**

- Internally an OS maintains message queue in which message packets sent by one process are submitted and can be sent to reciever process and vice-versa.

# Inter Process Communication



**Inter Process Communication**

**iii. Signals:**

- Processes communicates by means of passing signals as well.

- One process can send signal to another process.

- An OS send signal to any proces but any process cannot send signal to an OS.

- When we shutdown the machine an OS send SIGTERM signal to all processes due to which processes gets terminated normally, few processes can handle SIGTERM and even after recieving this signal from an OS continues execution, and hence to such processes an OS send SIGKILL signal due to which processes gets terminated forcefully.

e.g. SIGSTOP, SIGCONT, SIGSEGV etc...

## Inter Process Communication

### iv. Socket

- Limitation of above all IPC techniques is, only processes running on the same system can communicates, to overcome this limtation Socket IPC mechanism has been designed.

- By using socket IPC mechanism, process which is running on one machine can communicates with process running on another machine whereas machines are at remote distance from each other provided they connected in a network (either LAN/WAN/Internet).

**- Socket = IP Address + Port Number.**

e.g. chat application

# Process Coordination/Process Synchronization

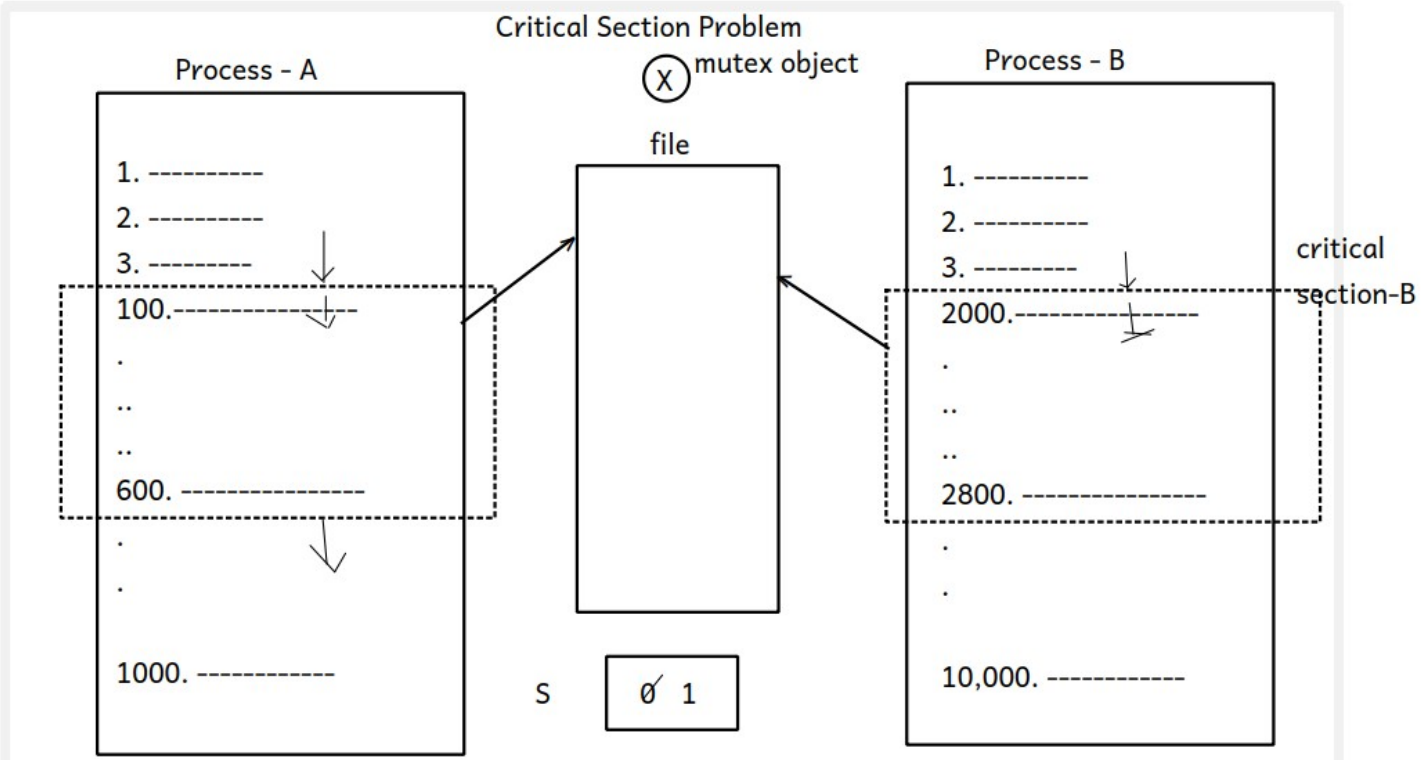
**Race Condition:** if one or more processes are trying to access a single resource at a time, race condition occures, and data inconsistency problem may takes place.

- Race condition can be avoided by an OS by

1. deciding their an order of allocation of resource for processes, and

2. whichever changes did by the last accessed process onto the resource remains final changes.

# Critical Section Problem:




"data incosistency" problem occure in above case only when both the sections of process A & B are runniing at a same time, and hence these sections are reffered as critical section, and hence data inconsistency problem may occure when two or more processes are running in their critical sections at a same time, and this problem is also reffered as "critical section problem".

**Synchronization Tools:**

**1. Semaphore:** there are two types of semaphore

**Binary semaphore:** it can be used when at a time resource can be acquired by only one process.

- It is an integer variable having either value is 0 or 1.

**2. Mutex Object:** it can be used when at a time resource can be acquired by only one process.

- Mutex object has two states: locked & unlocked, and at a time it can be only in a one state either locked or unlocked.

**Deadlock:**

- There are four neccessary and sufficient conditions to occure deadlock.

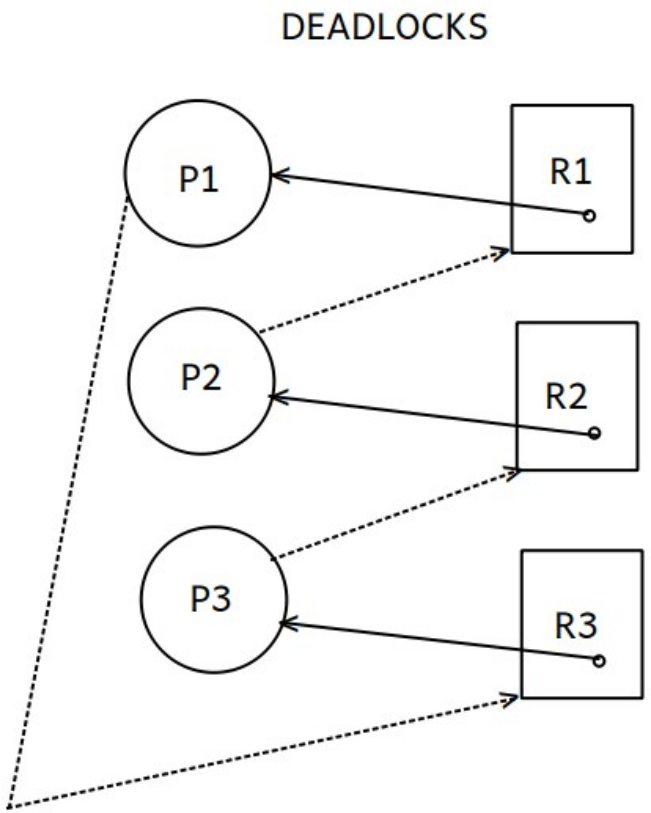
**1. Mutual Exclusion:** at a time resource can be acquired by only one process.

**2. No Preemption:** control of the resource cannot be taken away forcefully from any process.

**3. Hold & Wait:** every process is holding one resource and waiting for the resource which is held by another process.

**4. Circular Wait:** process P1 is holding resource and waiting for the resource held by another process P2, and process P2 is also holding resource and waiting for the resource held by process P1.

# Deadlock: Resource Allocation Graph



- Three deadlock handling methods are there:

**1. Deadlock Prevention:** deadlock can be prevented by discarding any one condition out of four neccessary and sufficient conditions.

**2. Deadlock Detection & Avoidance:** before allocating resources for processes all the input given to deadlock detection algorithm in advanced and if there are chances to occure deadlock then it can be avoided by doing neccessary changes.

- There are two deadlock detection algorithms:

1. Resource Allocation Graph Algorithm
2. Banker’s Algorithm

## 3. Deadlock Recovery:

- System can be recovered from the deadlock by two ways:

**1. Process termination:** in this method any one process gets selected and terminated to recover system from deadlock, process which gets terminated is reffered as a **victim process.**

**2. Resource preemption:** in this method control of the resource taken away forcefully from a process to recover system from deadlock.

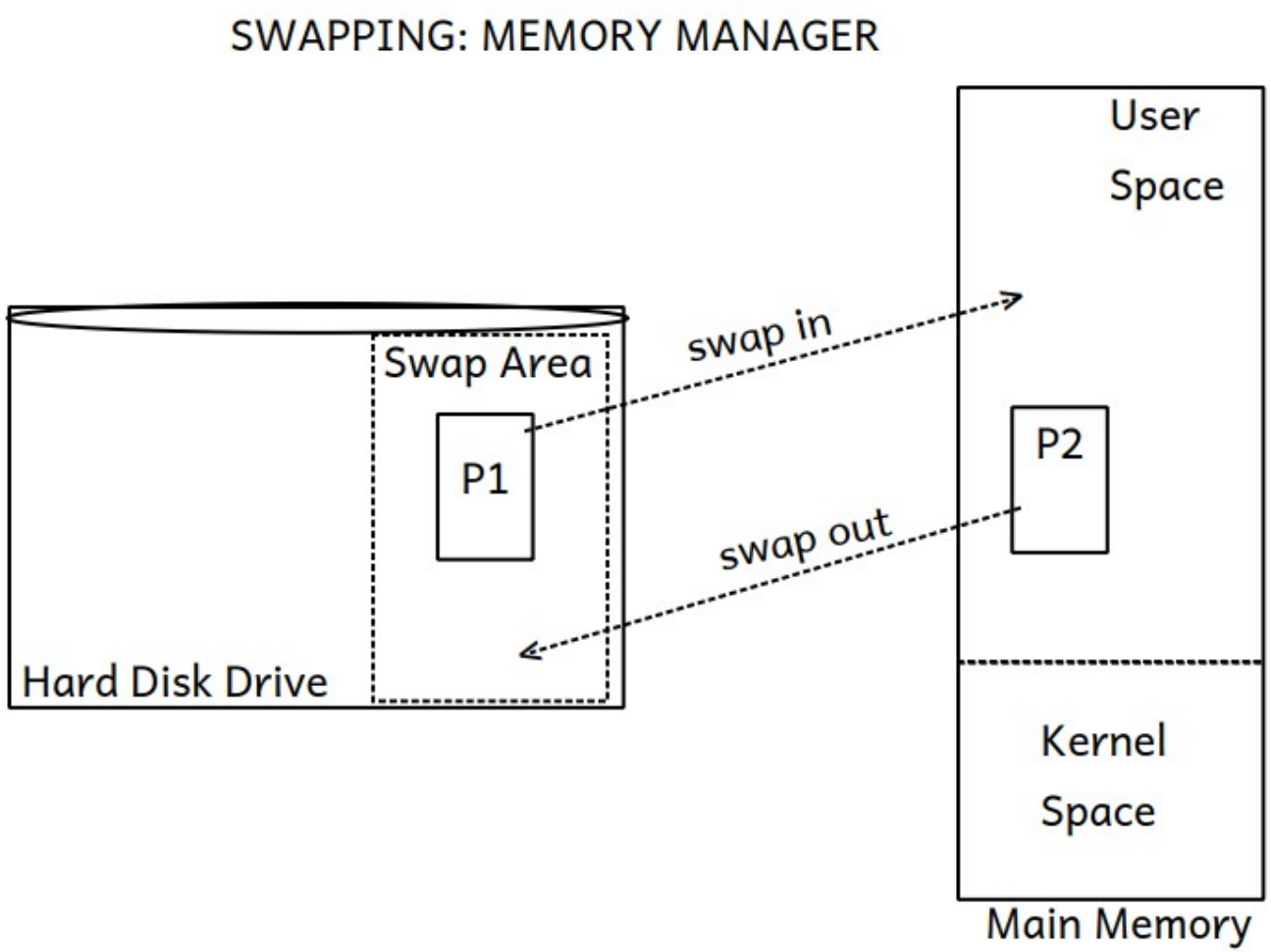
**# Memory Management**

**Q. Why there is a need of memory (main memory )management?**

- As main memory is must for an execution of any program and it is a limited memory, hence an OS manages main memory.

- To achieve maximum CPU utilization, an OS must support multi-tasking, and to support multi-tasking multiple porcesses must be submitted into the system at a time i.e. it must support multi-programming, but as main memory is limited to support multi-programming an OS has to do memory management to complete an execution of all submitted processes.

- Memory space of one process should gets protected from another process.


SWAPPING: MEMORY MANAGER
User
Space
Swap Area
P1
swap in
P2
swap out
Hard Disk Drive
Kernel
Space
Main Memory

**# Swapping:**

**- Swap area:** it is a portion of the hard disk drive ( keep reserved while installation of an OS) can be used by an OS as an extension of the main memory in which inactive running programs can be kept temporarily and as per request processes can be swapped in and swapped out between swap area and the main memory.

- In Linux swap area can be maintained in the form of swap partition, whereas in Windows swap arear can be maintained in the form of swap files.

**- Conventionally size of the swap area should be doubles the size of the main memory,** i.e. if the size of main memory is 2 GB then size of swap area should be 4 GB, if the size of main memory is 4 GB then size of swap area should be 8 GB and so on.

**# Swapping:**

- Swapping done by the program of an OS named as **Memory Manager**, it swapin active programs into the main memory from swap area and swapout inactive running programs from the main memory and keep them temporarily into the swap area.

- there are two variants of swapping: swapin & swapout.

**# Swapping:**

- Swapping done by the program of an OS named as **Memory Manager**, it swapin active programs into the main memory from swap area and swapout inactive running programs from the main memory and keep them temporarily into the swap area.

- there are two variants of swapping: swapin & swapout.

- Addresses generated by compiler (i.e. compiler + linker) are reffered as **logical addresses.**

- Addresses which can see process when it is in the main memory reffered as **physical addresses.**

**- MMU (Memory Management Unit):** wich is a hardware unit converts logical address into physical address.

- MMU is a hardware contains adder circuit, comparator circuit, base register and limit register, values of base register and limit registers gets change during context-switch, and memory space of one process can be protected from another process.

- CPU always executes program in its logical memory space.

**Memory Allocation:**

- When a process is requesting for the main memory, there are two methods by which memory gets allocated for any process

1. Contiguos Memory Allocation
2. Non-contiguos Memory Allocation

**1. Contiguos Memory Allocation:**

Under this method, process can complete its execution only if memory gets allocated for it in a contiguos manner.

- There are two methods by which memory gets allocated for process under contiguos memory allocation method.

1. Fixed Size Partitioning
2. Variable Size Partitioning

## 1. Fixed Size Partitioning:

- In this method, physical memory i.e. main memory (user space) is divided into fixed number of partitions and size of each partition is remains fixed.

- When a process is requesting for the memory it can be loaded into any free partition in which it can fit.

### # Advantages:

- This method is simple to implement

**# Disadvantages:**

**- Internal fragmentation:** memory remains unused  which is internal to the partition.

- Degree of multi-programming is limited to the number of partitions in the main memory.

- Maximum size of a process is limited to max size partition in the main memory.

- To overcome limitations/disadvantages of fixed size partitioning method variable/dynamic size partitioning method has been designed.

**2. Variable/Dynamic Size Partitioning:**

- In this method, initially whole user space i.e. physical memory is considered as a single free partition, and processes gets loaded into the main memory as they request for it.

- Size of partition and number of partitions are not fixed in advanced, it gets decided dynamically.

**# Advantages:**

- Degree of multi-programming is not limited/fixed

- Size of the process is not also limited, any size process may get load into the main memory.

**# Disadvantages:**

**- External fragmentation:** due to loading and removing of processes into and from the main memory, main memory is fragmented i.e. gets devided into used partitions and free partitions.

## 2. Variable/Dynamic Size Partitioning:

**- External fragmentation:** due to loading and removing of processes into and from the main memory, main memory is fragmented i.e. it gets divided into used partitions and free partitions.

In such case, if any new process is requesting for the memory and even if the requested size of memory is available but due to unavailability of memory in a contiguos manner process cannot be loaded into the main memory, this problem is reffered as an external fragmentation.

- External fragmentation is the biggest problem under contiguos memory allocation, and hence there are two solutions on this problem:

**1. Compaction**

**2. Non-contiguos Memory Allocation**

**1. Compaction:** shuffling of main memory can be done in such a way that all used partitions can be shifted to one side and all free partitions can be shifted to other side and contiguos large free partition will be made available for the new processes.

- Compaction is practically not feasible as there is need to do recalculations of addresses every time.

**2. Non-contiguos Memory Allocation:**

- Under this method, process can complete its execution even if memory gets allocated for it in a non-contiguos manner, and it can be achieved by two memory management techniques:

**1. Segmentation**

**2. Paging**

- So by using segmentation & paging techniques, process can complete its execution even after memory gets allocated for it in a non-contiguos manner.

## 1. Segmentation

- In this technique, process in its logical memory is divided into small size segments like **stack segment, heap segment, data segment, bss segment, rodata segment, code segment etc**..., and when process is requesting for memory it is not requesting memory contiguosly for whole process, memory gets allocated contiguosly only for small size segments, and segments of one process may get load into the memory randomly at any locations, i.e. for a process memory gets allocated in a non-contiguos manner, and then only an execution of a process can be completed.

- As segments of a one process gets loaded randomly, and in a system thousands processes are running at a time, hence to keep track on all the segments of each process, an OS maintains one table per process reffered as a **segment table** in which information about all the segments of that process can be kept.

- Using segmentation an external  fragmentation can be reduced but cannot be completely avoided.

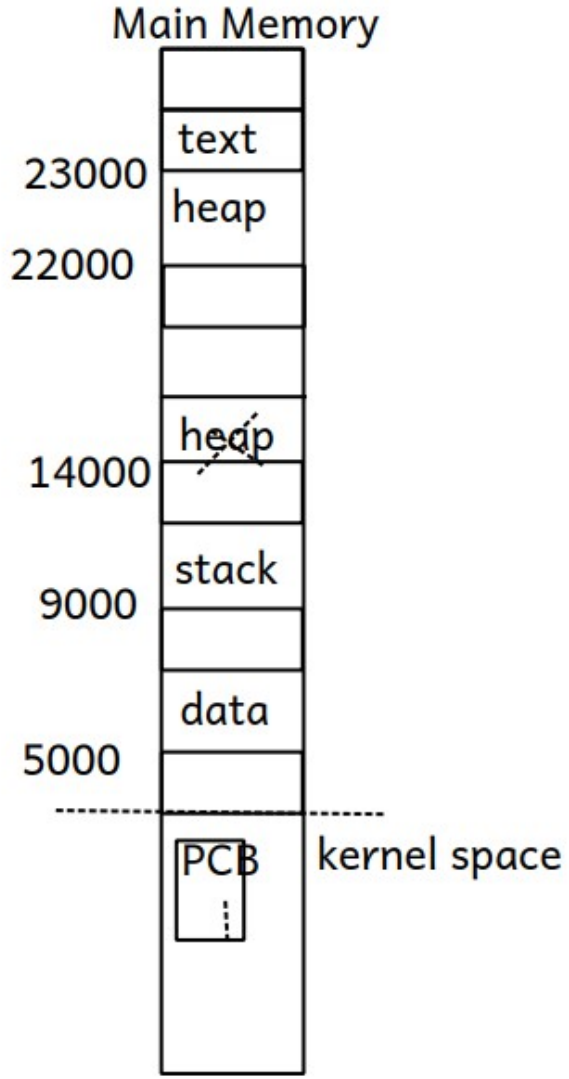
Segmentation

process: (6000)

| |
|---|
| stack - 1000 |
| heap - 1000 |
| data - 2000 |
| text - 2000 |

segment table:

| | limit | base |
|---|---|---|
| 0 | 1000 | 9000 |
| 1 | 1000 | 14000 |
| 2 | 2000 | 5000 |
| 3 | 2000 | 23000 |
| 4 | null | null |



**2. Paging**

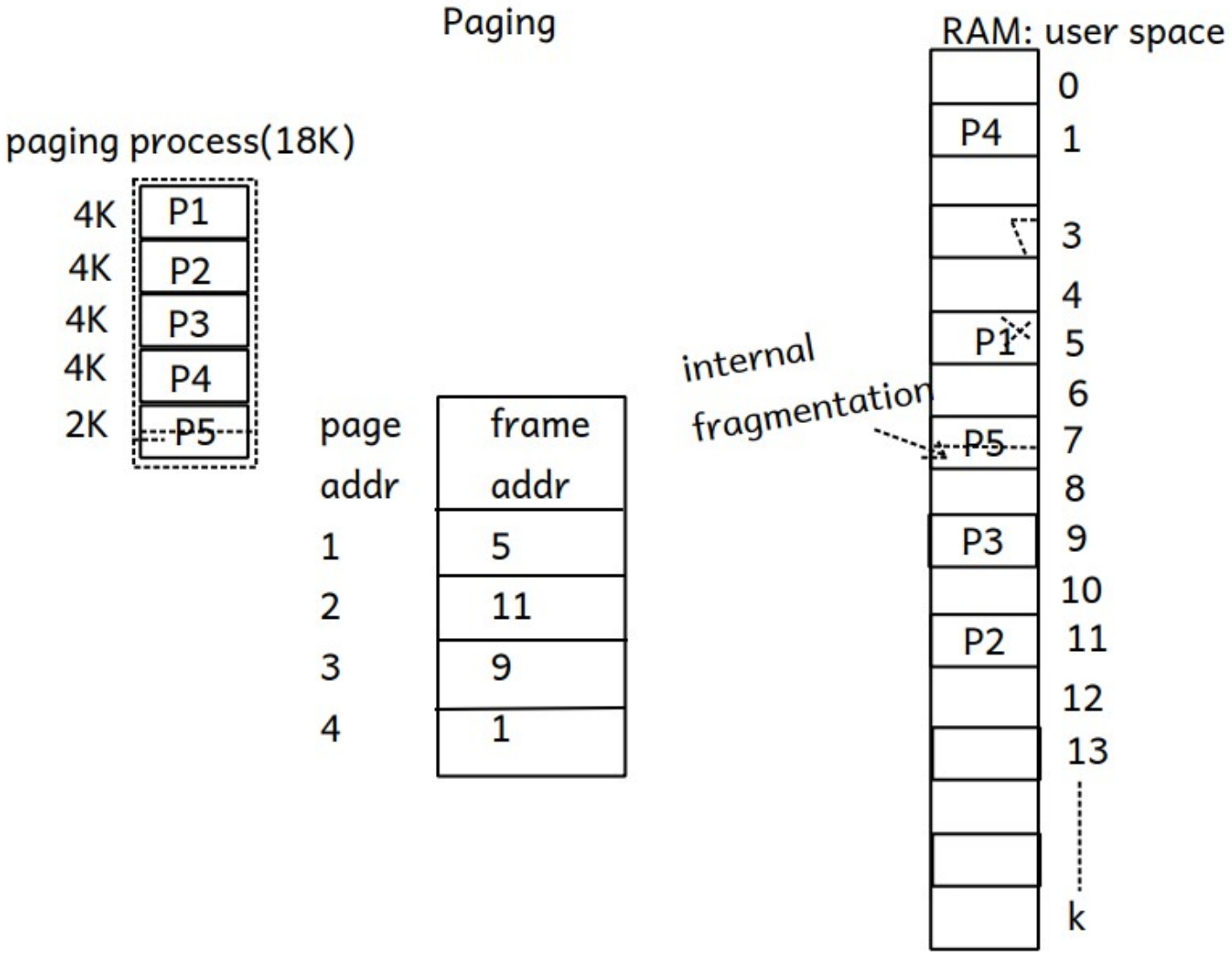
- In this technique, **physical memory** (i.e. user space of a main memory) is divided into fixed size of blocks reffered as **frames**, and process's **logical memory** space is divided into same size of blocks reffered as **pages**, whereas maximum size of page must be equal to size of frame, i.e. if e.g. size of frame = 4K, then maximum size of each page must be 4K, size of page may be less than 4K.

- As process is divided into pages, so when it is requesting for memory, pages of one process may gets loaded into the main memory at any free frames, and for a process memory gets allocated in a non-contiguos manner.

- As pages of a one process gets loaded randomly into the main memory, and in a system thousands processes are running at a time, to keep track on all the pages of each process, an OS maintains one table per process reffered as a **page table** in which information about all the pages of that process can be kept.

- There is no external fragmentation in paging.

- Internal fragmentation may exists in paging when the size of page is less than size of frame.


Paging
paging process(18K)
4K P1
4K P2
4K P3
4K P4
2K P5
page addr
frame addr
1 5
2 11
3 9
4 1
RAM: user space
internal fragmentation
0
P4 1
3
4
P1 5
6
P5 7
8
P3 9
10
P2 11
12
13
k

**Virtual Memory Management:**

- As we seen an OS does memory management for completing an execution of multiple submitted processes at once.

- An OS also able to complete an execution of such a process having size bigger than size of main memory itself, and to achieve this an OS manages swap area memory as well with main memory and hence it is reffered as **virtual memory management.**

- As an OS manages such a memory which is physically not a main memory and hence it is reffered as virtual memory management.

- Virtual memory management can be implemented by using **Paging + Swapping.**

- In this technique for a process to complete its execution it is not mandatory it must exists wholely in a main memory, even its part is their into the main memory then execution of such process can be completed part by part.

- Big size process is divided into pages and when a process is requesting for memory few pages gets loaded into the main memory and few pages can be kept into the swap area, and as per the request pages of that process can swappedin and swapped out between main memory and swap area.

**Virtual Memory Management:**

**- Demand Paging:** any page of a process gets loaded into the main memory only after requesting by that process i.e. on demand and hence reffered as **demand paging**, page which is never requested never gets loaded into the main memory and hence it also called as **pure demand paging**.

- If a process is requesting for any page and if that page is not exists in the main memory, then it is referred as **page fault**.

- As the size of process may be bigger than size of main memory itself, and in a system multiple processes are running at a time, hence no. of pages are more than no. of frames, so there are quite good chances that all frames becomes full, and in this case if any process is requesting for a page which does not exists in the main memory at that time there is need to remove any one page from the main memory so that into that free frame requested page will get loaded.

- So there is need to decide which page should get removed and requested page gets replaced in that frame, to do this there are certain algorithms referred as **page replacement algorithms.**

**Page Replacement Algorithms:**

**1. FIFO Page Replacement:** page which was inserted first gets replaced by the requested page.

**2. Optimal Page Replacement:** page which will not get used in a near future gets replaced by the requested page

**3. LRU(Least Recently Used) Page Replacement:** least recently used page gets replaced by the requested page.

**4. LFU(Least Frequently Used) Page Replacement:** least frequetly used page gets replaced by the requested page.

**5. MFU(Most Frequently Used) Page Replacement:** most frequetly used page gets replaced by the requested page.

- Algorithms 1, 2 & 3 uses **stack based** approach, whereas algorithms 4 & 5 uses counting based approach.

- Conceptually Optimal Page Replacement is the most efficient page replacement algorithm, as no. of page faults in this algorithm are very less, but as its practical implementation is not feasible hence LRU is the most efficient algorithm (implementationwise).

- **Thrashing:** If any process spends more time on paging rather than execution, then this high paging activity is referred as thrashing.

**# File Management**

**Q. What is file?**

**User view:**

- File is a named collection of logically related information/data.
- File is a container which contains logically related information/data.
- File is a collection of characters/records/lines.
- File is a basic storage unit

**System view:**

- File is a stream of bits/bytes.
- **File = data + metadata**

  data = actual file contents

  metadata = information about the file.

- Data of the file is exists inside the file whereas information about the file gets stored inside one structure referred as **FCB (File Control Block).**
- In UNIX environment FCB is also called as an iNode.
- FCB/iNode contains information about the file like:
    1. iNode number: unique identifier of a file
    2. Name of the file
    3. Type of the file
    4. Size of the file
    5. Parent folder location
    6. Access permissions
    7. Time stamps

    etc...

- Per file one iNode/FCB gets created by the system and hence no. of iNodes = no. of files onto the disk.

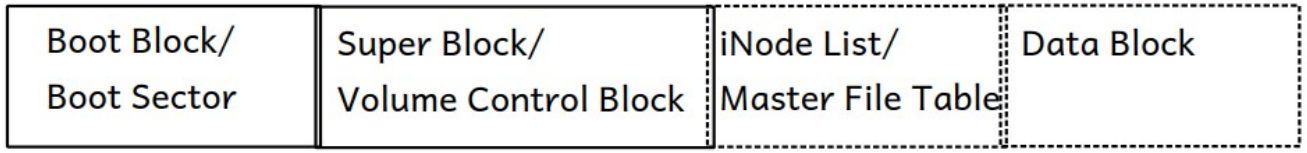
- data + metadata of all the files are kept onto the disk, as disk may contents thousands of files, so thousands of iNodes and millions of bytes of data gets stored onto the disk, and hence all this data + metadata of all files need be keep onto the disk in an organized manner so that it can be accessed efficiently.

- **Filesystem:** filesystem is a way to store data onto the disk in an organized manner so that it can be accessed efficiently and conveniently.

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...

**# Filesystem Structure:** Filesystem devides disk/partition logically into sectors/blocks, like **boot sector/boot block, volume control block/super block, master file table/iNode list block and data block.**

FILESYSTEM STRUCTURE




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.

**# Disk space allocation methods:**

- When a file is requesting for free data blocks, then in which manner free data blocks gets allocated for that file and how its information can be kept inside inode of that file is referred as **disk space allocation method.**

- Three disk space allocation methods are there:

**1. Contiguos Allocation:** free data blocks gets allocated for a file in a contiguos manner.

**2. Linked Allocation:** any free data blocks gets allocated for a file in a linked list manner.

**3. Indexed Allocation:** any free data blocks gets allocated for a file, as by maintaining an index data block information about allocated data blocks can be kept inside it.

**# Disk Scheduling Algorithms:**

- When system want to access data from a disk, request can sent to disk controller and disk controller accepts one request at a time and complete it.

- There are chances that at a time more than one requests for accessing data from the disk can be made by the processes running in a system, in that case all the requests can be kept in a waiting queue of the disk maintained by an OS, and there is need to schedule/select only one request at a time and sent it to the disk controller, to do this there are certain algorithms referred as disk scheduling algorithms.

**1. FCFS (First Come First Served):** request which is arrived first gets accepted and completed.

**2. SSTF (Shortest Seek Time First):** request which is closed to the current position of the head gets accepted and completed.

**3. SCAN:** head keeps scanning the disk from starting cylinder to end cylinder and whichever request came across gets accepted and completed.

**4. C-SCAN (Circular SCAN):** head scans the disk only in a one direction

**5. LOOK:** this policy can be used either with SCAN/C-SCAN, in this, if there is no request in a waiting queue then movement of the head gets stopped.