# Computer Fundamentals
# And
# Operating System Concepts

# ❑Module introduction

➢CCAT point of view:

- Operating Systems: 5 Questions
- Computer Fundamentals : 5 Questions

➢Reference Book:

- Operating System Concepts - Galvin

- Practice Exam :

  - Quiz : 4 quiz exam **OS** ( **10** Question )
  - Quiz : 1 quiz exam **Computer Fundamentals** ( **10** Question )
  - Module End Quiz Exam ( **20** Question )

## ➢ Introduction: -

- Why there is need of an OS?
- What is an OS?
- Booting process in brief
- Functions of an OS

## ➢Computer Fundamentals:

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

## ➢UNIX System Architecture Design

- Major subsystem of an UNIX system: File subsystem & Process Control subsystem.
- System Calls & its categories
- Dual Mode Operation

## ➢Process Management

- What is Process & PCB?
- States of the process
- Process life cycle
- CPU scheduling & CPU scheduling algorithms
- Inter Process Communication: Shared Memory Model & Message Passing Model
- Processor architecture (CF)

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

## ➢Memory Management

- Memory types (CF)
- Swapping
- Memory Allocation Methods
- Internal Fragmentation & External Fragmentation  Segmentation
- Paging
- Virtual Memory Management

## ➢File Management

- What is file?
- What is file system & file system structure?
- Disk structure (CF)
- Disk space allocation methods
- Disk scheduling algorithms
- Computer structure (CF)
- Interrupts (CF)
- Direct Memory Access (CF)
- Input-Output (CF)
- System calls

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

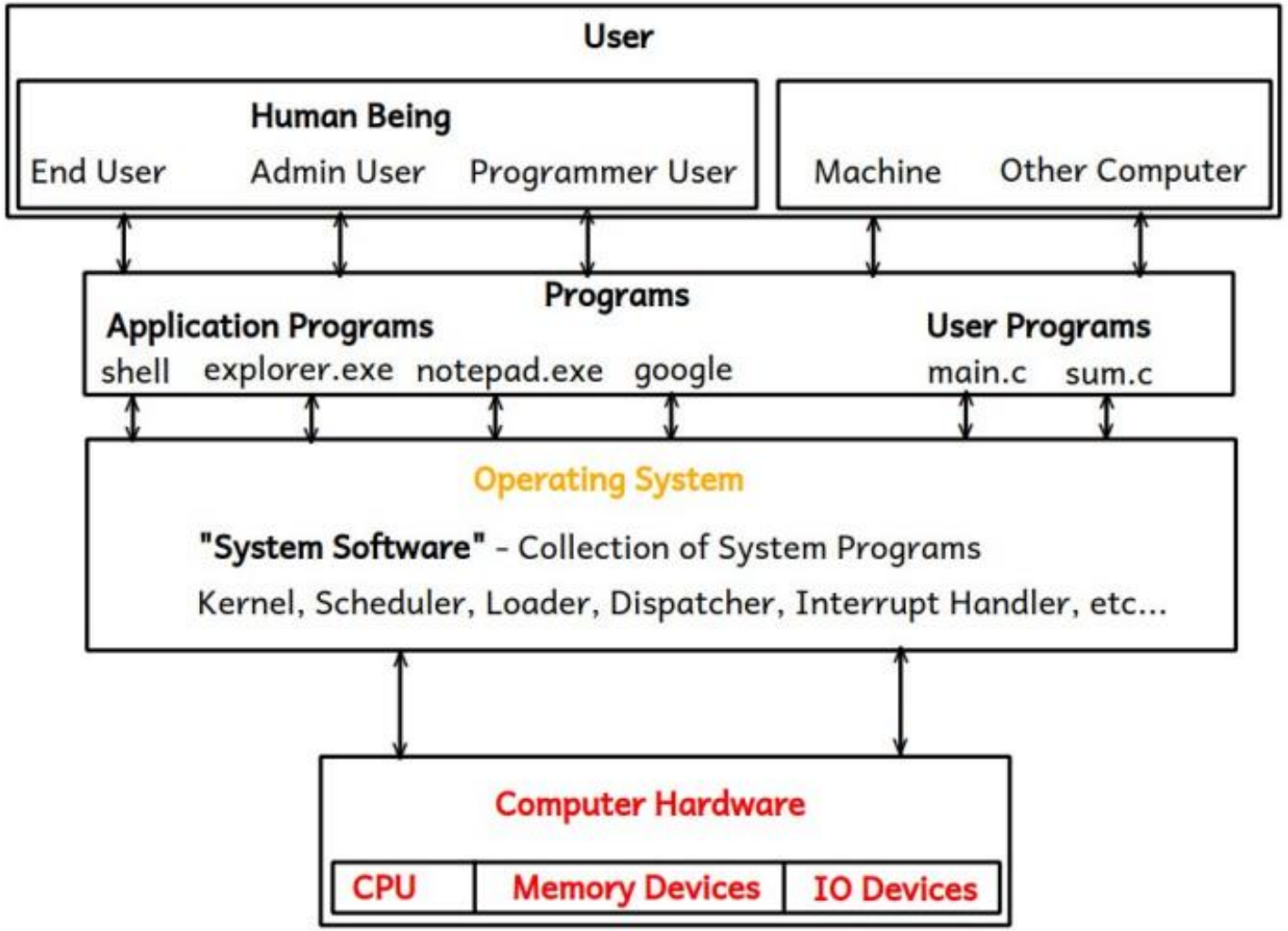
- Computer is a machine/hardware does different tasks efficiently & accurately.
- Basic functions of computer :
    1. Data Storage : Memory Devices
    2. Data Processing : CPU/Processor
    3. Data Movement : I/O Devices
    4. Control



- 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 Systems Concepts

**Diagram :OS**



## Q. What is an Operating System?

- An OS is a **system softwar**e (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.**

# Operating Systems Concepts

- 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 continuously into the main memory does basic minimal functionalities of it. e.g. Linux: vmlinuz, Windows: ntoskrnl.exe
2. **Utility Software's:** e.g. disk manager, windows firewall, anti-virus software etc...
3. **Application Software's**: e.g. Google chrome, shell, notepad, MS office etc...

## Q. What is a Software?

- Software is a collection of programs.

## Q. What is a Program?

- Program is a finite set of instructions written in any programming language (either low level or high level programming language) given to the machine to do specific task.

### ❖ 3 types of programs are there:

1. **"user programs":** programs defined by the programmer user/developers

   e.g. main.c, hello.java, addition.cpp etc....

2. **"application programs":** programs which comes with an OS/can be installed later

   e.g. MS Office, Notepad, Compiler, IDE's, Google Chrome, Mozilla Firefox, Calculator, Games etc....

3. **"System Programs":** programs which are inbuilt in an OS/part of an OS.

   e.g. Kernel, Loader, Scheduler, Memory Manager etc...

## ➢Functions of an OS:

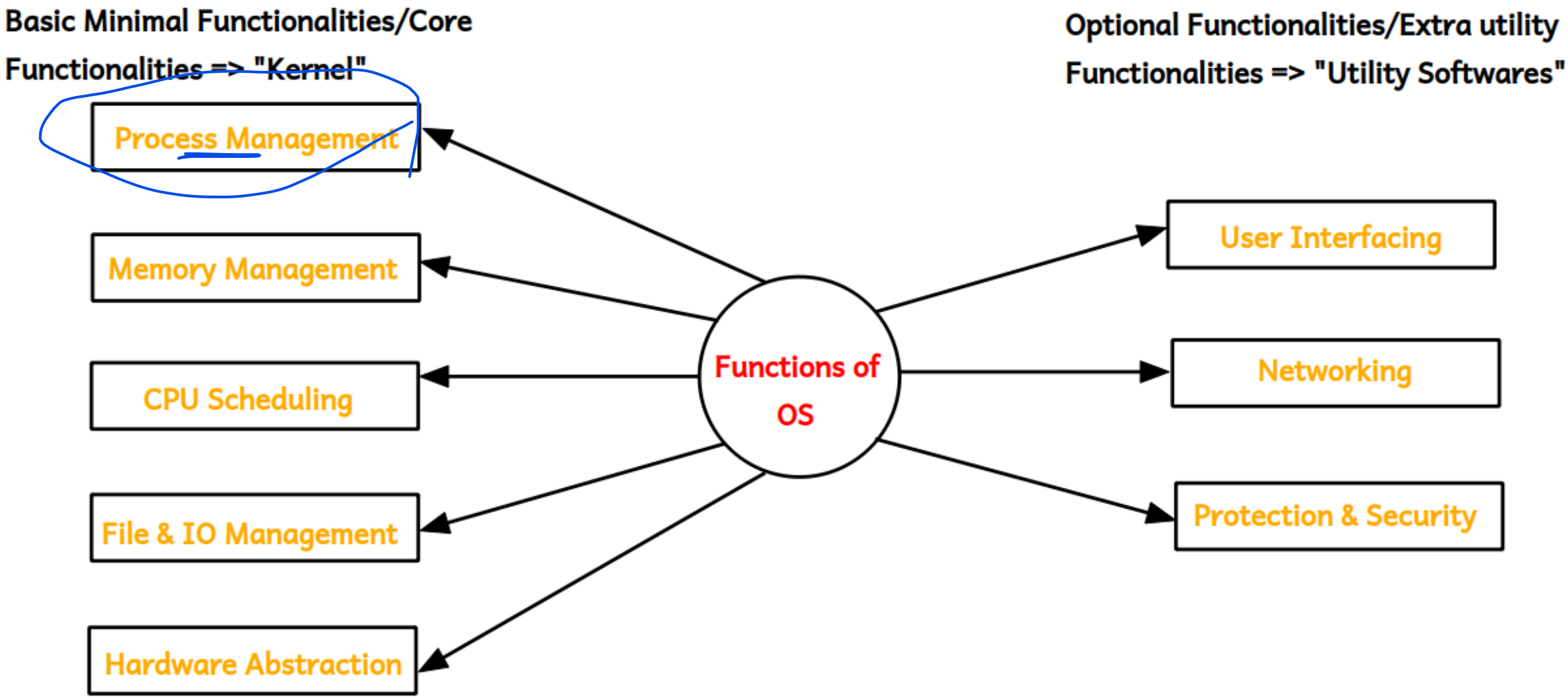
### Basic minimal functionalities/Kernel functionalities:

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

# Operating Systems Concepts



## ➢What is an IDE (Integrated Software Development) ?

- It is **an application software i.e. collection of tools/application programs like source code editor, pre-processor, compiler, linker, debugger etc... required for faster software development**.

e.g. VS code editor, MS Visual Studio, Net beans, Android Studio, Turbo C etc....

1. **"Editor":** it is an application program **used to write a source code.**
   Source Code – Program written in any programming language.
   e.g. notepad, vi editor, gedit etc...
2. **"Pre-processor":** it is an application program gets executes before compilation and does two jobs
   **- it executes all pre-processor directives and removes all comments from the source code.**
   e.g. cpp
3. **"Compiler":** it is an application program which converts high level programming language code into low level programming language **code i.e. human understandable language code into the machine understandable language code.**
   e.g. gcc, tc, visual c etc...

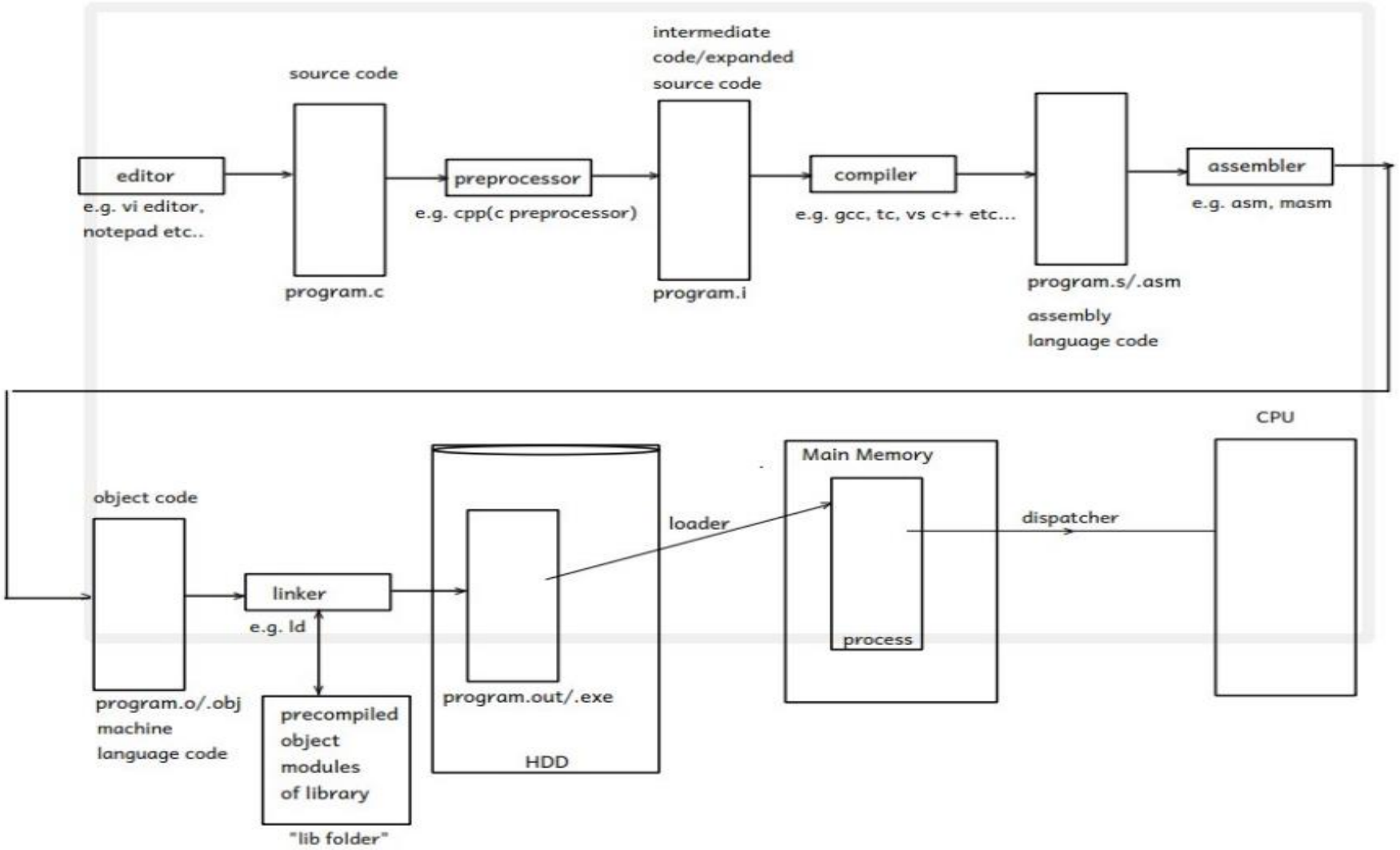
4. **"Assembler":** it is an application program which converts assembly language code into machine language code/object code.

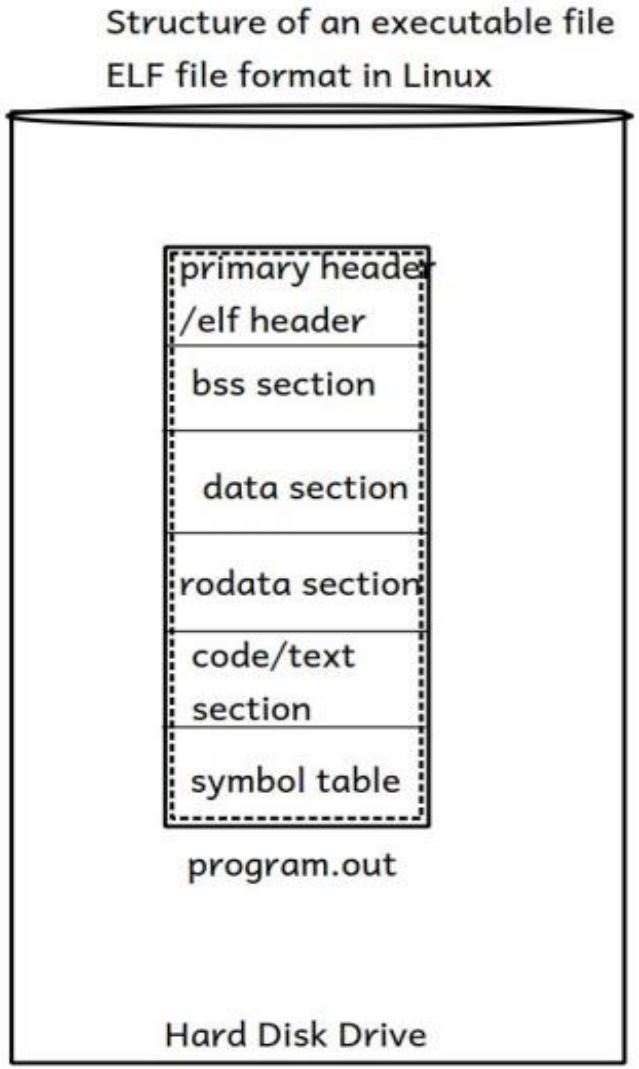
   e.g. masm, tasm etc...

5. **"Linker":** it is an application program which links object file/s in a program with precompiled object modules of library functions exists in a lib folder and creates final single executable file.

   e.g. ld: link editor in Linux.

# Operating Systems Concepts






1. **primary header/exe header:** it contains information which is required to starts an execution of the program.
   e.g. - addr of an entry point function --> main() function
   - **magic number:** it is constant number generated by the compiler which is file format specific.
     - magic number in Linux starts with ELF in its eq **hexadecimal format**.
     - info about remaining sections.
2. **bss(block started by symbol) section:** it contains uninitialized global & static vars
3. **data section:** it contains initialized global & static vars
4. **rodata (readonly data ) section:** it contains string literals and constants.
5. **code/text section:** it contains an executable instructions
6. **symbol table:** it contains info about functions and its vars in a tabular format.

# ➢Booting Process

- Terminologies :

- **Bootstrap Program**
  - It loads OS kernel into main memory
  - Each OS has Its own Bootstrap program.
  - Located in first sector of bootable storage device.
- **Bootable Device**
  - Storage device whose first sector (512 bytes) contains a special program “Bootstap Program “.
  - Usually it is device that stores OS installation setup;
  - E.g. Bootable CD/DVD ,Bootable Pendrive.
- **Bootloader**
  - When multiple OS are installed on single computer, at the beginning one program asks end user about which OS boot.
  - This  task perform by bootloader program.
  - Bootloader program bootstrap program of selected OS.
  - Located in second sector of bootable storage device.

- **BIOS/Firmwaer**
  - Firmware is a program/set of a program loaded into base ROM of motherboard.
  - It is developed by Manufacturer of motherboard.
  - The firmware developed for PC (by IBM) is named as BIOS : Basic Input Output System
  - BIOS Contain
    - POST/BIST
    - Bootstrap loader
    - Information utility
    - Bootable Device preference/settings
    - Basic/Minimal device driver.

- **POST/BIST**
  - **POST** - Power ON Self Test
  - **BIST** - Built In Self Test
  - Send signal to all peripheral(e.g. keyboard , mouse, monitor…)and test if they are functioning well.
  - Located in Base ROM(Part of Firmware)

- **Bootstrap Loader :**
  - Finds the bootable device in the computer and start it's Bootloader.
  - It check all devices in a order given in BIOS and start the first found bootable device.
  - Located in Base ROM (Part of Firmware).

**# Booting:**

- There are two steps of booting:

**1. Machine Boot:**

**Step-1:** when we switched on the power supply current gets passed to the motherboard on which from ROM memory one micro-program gets executes first called as **BIOS(Basic Input Output System).**

**Step-2:** first step of BIOS is **POST(Power On Self Test),** under POST it checks wheather all peripheral devices are connected properly or not and their working status.

**Step-3:** After POST it invokes **Bootstrap Loader** programs, which searches for available **bootable devices** presents in the system, and it selects only one bootable device at a time as per the priority decided in BIOS settings.

**2. System Boot:** HDD

**Step-4:** After selection of a bootable device (budefault HDD), **Bootloader Program** in it gets invokes which displays list of names operating systems installed on the disk, from which user need to select any one OS.

**Step-5:** Upon selection of an OS, **Bootstrap Program** of that OS gets invokes, which locates the kernel and load into the main memory

## ➢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 referred **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 referred **as non - privileged 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 switches in between **kernel mode and user mode** and hence system runs in two modes, it is referred **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 weather 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 value of 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 communication, process coordination, and terminate the process.**

**Q. What is a Program?**

**❑User view:**

- Program is a finite set of instructions written in any programming language given to the machine to do specific task.

**❑System view:**

- Program is an executable file in HDD which divided logically into sections like exe header, bss section, data section, rodata section, code section, symbol table.

## Q. What is a Process?

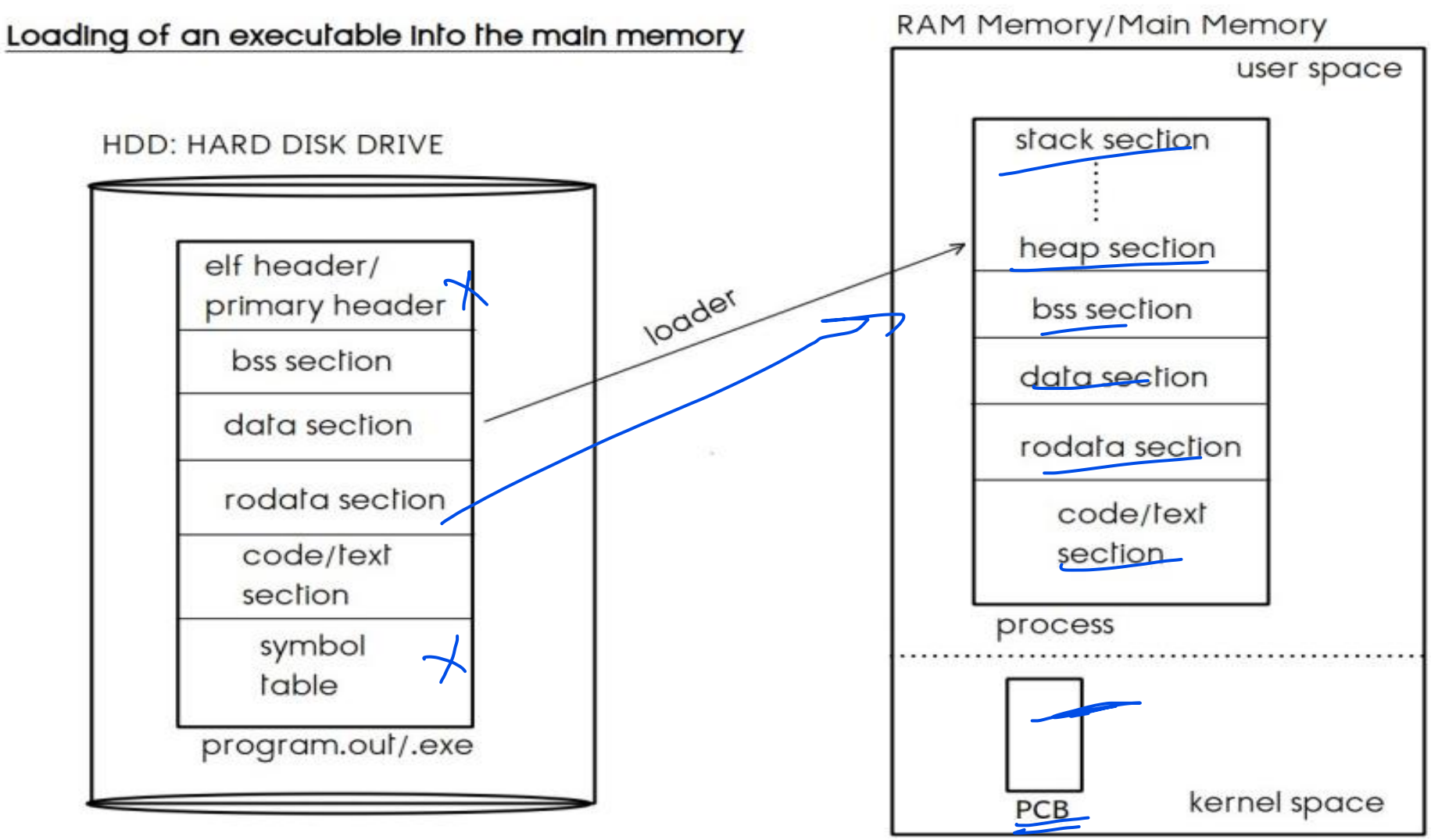
❑ **User view:**

- 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 referred as **a process.**
- Running instance of a program is referred as **a process**.

❑ **System view:**

- Process is a program loaded into the main memory which has got PCB into the main memory inside kernel space and program itself into the main memory inside user space has **got bss section, rodata section, code section, and two new sections gets added for the process .**

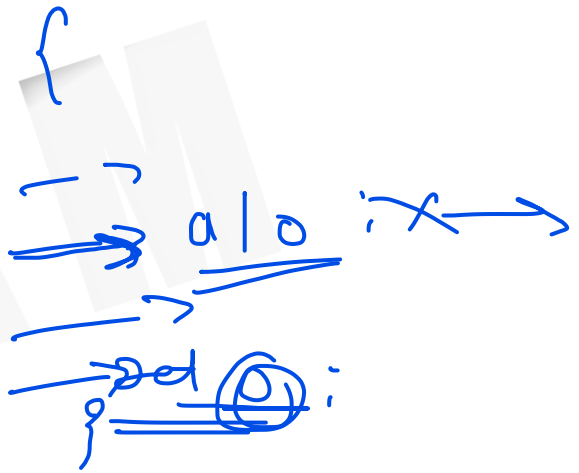
- **stack section:** contains function activation records of called functions.
- **heap section:** dynamically allocated memory


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

- As a kernel, core program of an a OS runs continuously into the main memory, part of the main memory which is occupied by the kernel **referred as kernel space** and whichever part is left **is referred 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 referred as a **PCB.**
- **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 : Process ID
- PPID : Parent Processes ID
- PC : Program Counter
- Execution context
- Kernel stack
- Exit status



- CPU sched information, memory management information, information about resources allocated for that process, execution context etc...

1. **Resident monitor**
2. **Batch System**

   -The batch/group of similar programs is loaded in the computer, from which OS loads one program in the memory and execute it. The programs are executed one after another.

   - In this case, if any process is performing IO, CPU will wait for that process and hence not utilized efficiently.

3. **Multi-programming**

   - Better utilization of CPU
   - Loading multiple Programs in memory
   - Mixed program(CPU bound + IO bound)

## 4. Time-sharing/Multitasking

- Sharing CPU time among multiple process/task present in main memory and ready for execution
- Any process should have response time should be less then 1sec
- Multi-tasking is divided into two types
  - Process based multitasking
  - Thread based multitasking
    - Thread is a light weight process .
    - When new thread is created a new stack and new TCB is created.
    - Thread Share text, data, heap sections with the parent process

- Process and thread
  - -Process is a container for resources.
  - -Thread is unit of execution/ scheduler.
  - - For each process one thread is created by default it is called as main thread

## 5. Multi-user system

- Multiple users runs multiple programs concurrently.

## 6. Multi-processor/ Mutli-core system

- System can run on a machine in which more than one CPU's are connected in a closed circuit.
- Multiprocessing Advantage is it increased throughput (amount of work done in unit time)
- There are two types of multiprocessor systems:

❖ **Asymmetric Multi-processing**

❖ **Symmetric Multi-processing**

- **Asymmetric Multi-processing**

- OS treats one of the processor as master processor and schedule task for it. The task is in turn divided into smaller tasks and get them done from other processors.

- **Symmetric Multi-processing**

- -OS considers all processors at same level and schedule tasks on each processor individually. All modern desktop systems are SMP.

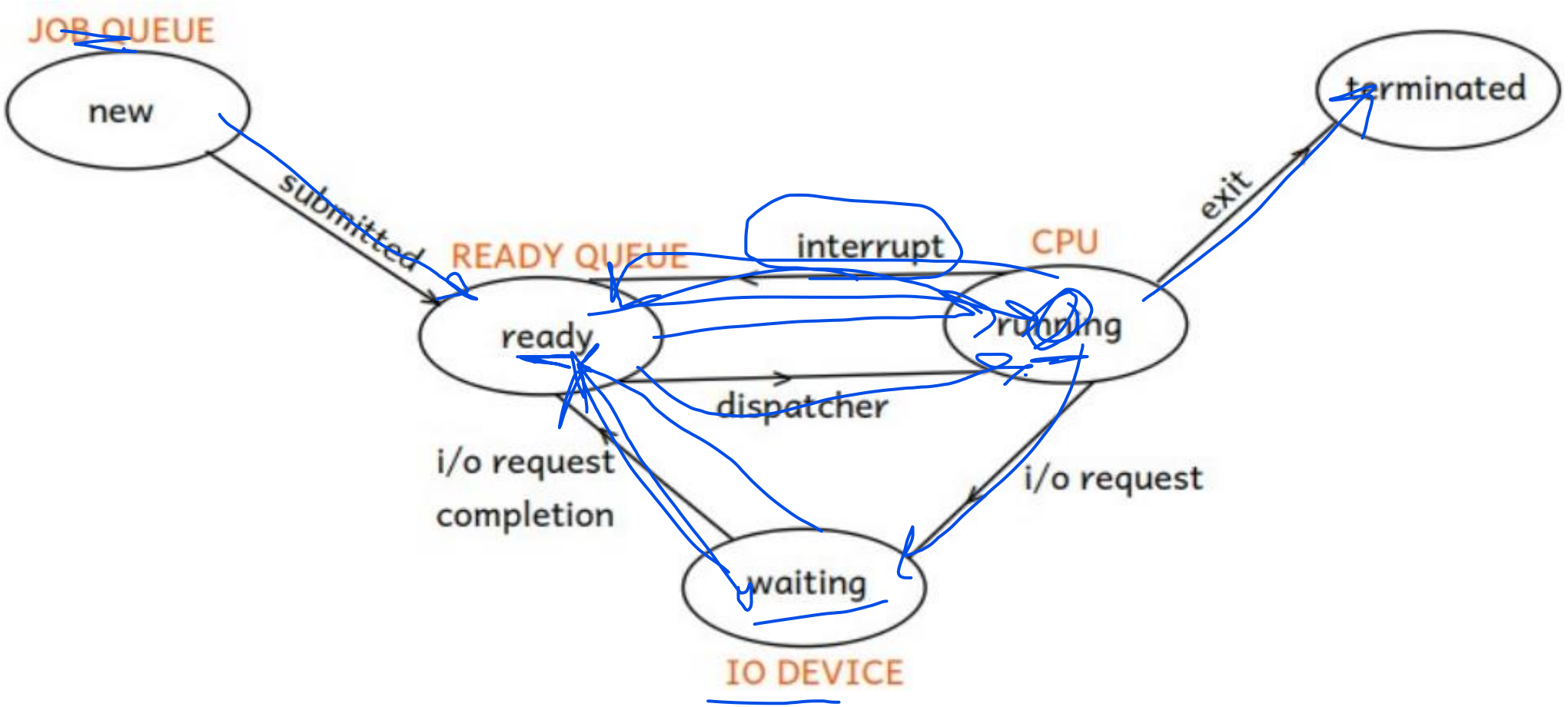
## ➢Process States:

- 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 the main memory and waiting for the CPU time, it is in **a ready state**.
3. **Running state:** if currently the CPU is executing any process then state of that process is considered **as a running state**.
4. **Waiting state:** if a process is requesting for any i/o device then state of that process is considered as **a waiting state**.
5. **Terminated state:** upon exit, process goes into terminated state and its PCB gets destroyed from the main memory.

# Operating Systems and Computer Fundamentals




PROCESS STATE DIAGRAM

## ➢Process States:

- To keep track on all running programs, an OS maintains few data structures referred **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 the main memory and waiting for the CPU time.
3. **3. Waiting queue:** it contains 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.**
3. **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 referred 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 referred 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 dispatcher, dispatcher 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.

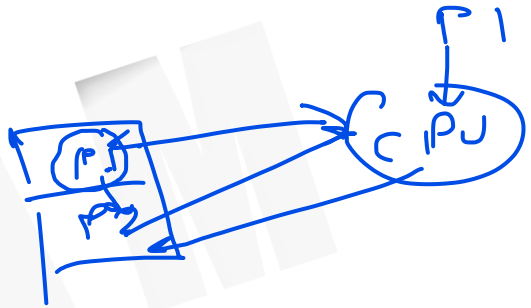
- CPU Scheduler gets called in the following four cases:

**Case-1: Running -> Terminated**

**Case-2: Running -> Waiting**

**Case 3: Running -> Ready**

**Case-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 cpu 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. **Round Robin CPU Scheduling**
4. **Priority CPU Scheduling**

- Multiple algorithms are there for CPU scheduling, so there is need to decide which algorithm is best suited at specific situation and which algorithm is an efficient one, to decide this there are certain criteria's **called as scheduling criteria's: cpu utilization, throughput, waiting time, response time and turn-around-time.**

## ➢CPU Scheduling Criteria's:

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

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.**
- **Execution Time:** it is the **total amount of time spent by the process onto the CPU to complete its execution**.
  - **OR CPU Burst Time:** total no. of CPU cycles required for the process to complete its execution.
- **Turn-Around-Time = Waiting Time + Execution Time.**
- **Turn-around-time is the sum of periods spent by the process into ready queue for waiting and onto the CPU for execution from its time of submission.**

## ➢CPU Scheduling

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

- In this algorithm, process which is 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 impalement and can be implemented by using fifo queue.
- It is a non-preemptive scheduling algorithm.

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

## ➢FCFS Scheduling

| Process | Arrival Time | CPU Burst | Wait Time | Turn Around Time |
|---|---|---|---|---|
| P1 | 0 | 24 | | |
| P2 | 0 | 3 | | |
| P3 | 0 | 3 | | |

## 2. SJF(Shortest Job First) CPU Scheduling:

- In this algorithm, process which is having minimum CPU burst time gets the control of the CPU first, and whenever tie is there it can be resolved by using fcfs. - SJF algorithm ensures minimum waiting time.
- Under non-preemptive SJF, algorithm fails if the submission time of 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).

❑ **Starvation:** in this algorithm, as shorter processes has got higher priority, process which is having larger CPU burst time may gets blocked i.e. control of the CPU will never gets allocated for it, such situation is called as starvation/indefinite blocking.

# ➢ SJF/SNTF Scheduling

| Process | Arrival Time | CPU Burst | Wait Time | Turn Around Time |
|---|---|---|---|---|
| P1 | 0 | 7 | | |
| P2 | 2 | 4 | | |
| P3 | 4 | 1 | | |
| P4 | 5 | 4 | | |

# ➢ SRTF Scheduling

| Process | Arrival Time | CPU Burst | Remaining Time | Wait Time | Turn Around Time |
|---|---|---|---|---|---|
| P1 | 0 | 7 | | | |
| P2 | 2 | 4 | | | |
| P3 | 4 | 1 | | | |
| P4 | 5 | 4 | | | |

## 3. Round Robin Scheduling Algorithm

- In this algorithm, before allocating the CPU for processes, **some fixed time slice or time quantum** gets decided in advanced, and at any given time control of the CPU may remains allocated with any process maximum for that decided time-slice, once the given time slice is finished of that process, it gets suspended and control of the CPU will be allocated to the next process again for maximum that decided time slice and so on..., each process gets control of the CPU in a round robin manner i.e. cpu gets shared among processes equally.
- If any process completes its execution before allocated time slice then control of the CPU will be released by that process and CPU gets allocated to the next process as soon as it is completed for effective utilization of the CPU.
- There **is no starvation in RR Scheduling algorithm**.
- This algorithm is **purely preemptive.**
- This algorithm **ensures minimum response time**.
- **If time slice is minimum then there will be extra overhead onto the CPU due to frequent context-switch.**

# ➢ Round Robin Scheduling

| Process | CPU Burst | Remaining Time | Wait Time | Response Time |
|---|---|---|---|---|
| P1 | 53 | | | |
| P2 | 17 | | | |
| P3 | 68 | | | |
| P4 | 24 | | | |

## 4. Priority Scheduling

- In this algorithm, process which is having highest priority gets control of the CPU first**, each process is having priority in its PCB**.
- priority for a process can be decided by two ways:
    1. **internally** – priority for process can be decided by an OS depends on no. of resources required for it.
    2. **externally** – priority for process can be decided by the user depends on requirement.
- **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 referred as a starvation or indefinite blocking**.
- **Ageing:** it is a technique in which, an OS gradually increments 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.

# ➢ Priority Scheduling

| Process | Arrival Time | CPU Burst | Priority | Wait Time |
|---|---|---|---|---|
| P1 | 0 | 10 | 3 | |
| P2 | 0 | 1 | 1 | |
| P3 | 0 | 2 | 4 | |
| P4 | 0 | 5 | 2 | |


# Thank you!

Kiran Jaybhave

email – kiran.jaybhave@sunbeaminfo.com