Operating System Concepts

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

Q. Why there is a need of an OS?

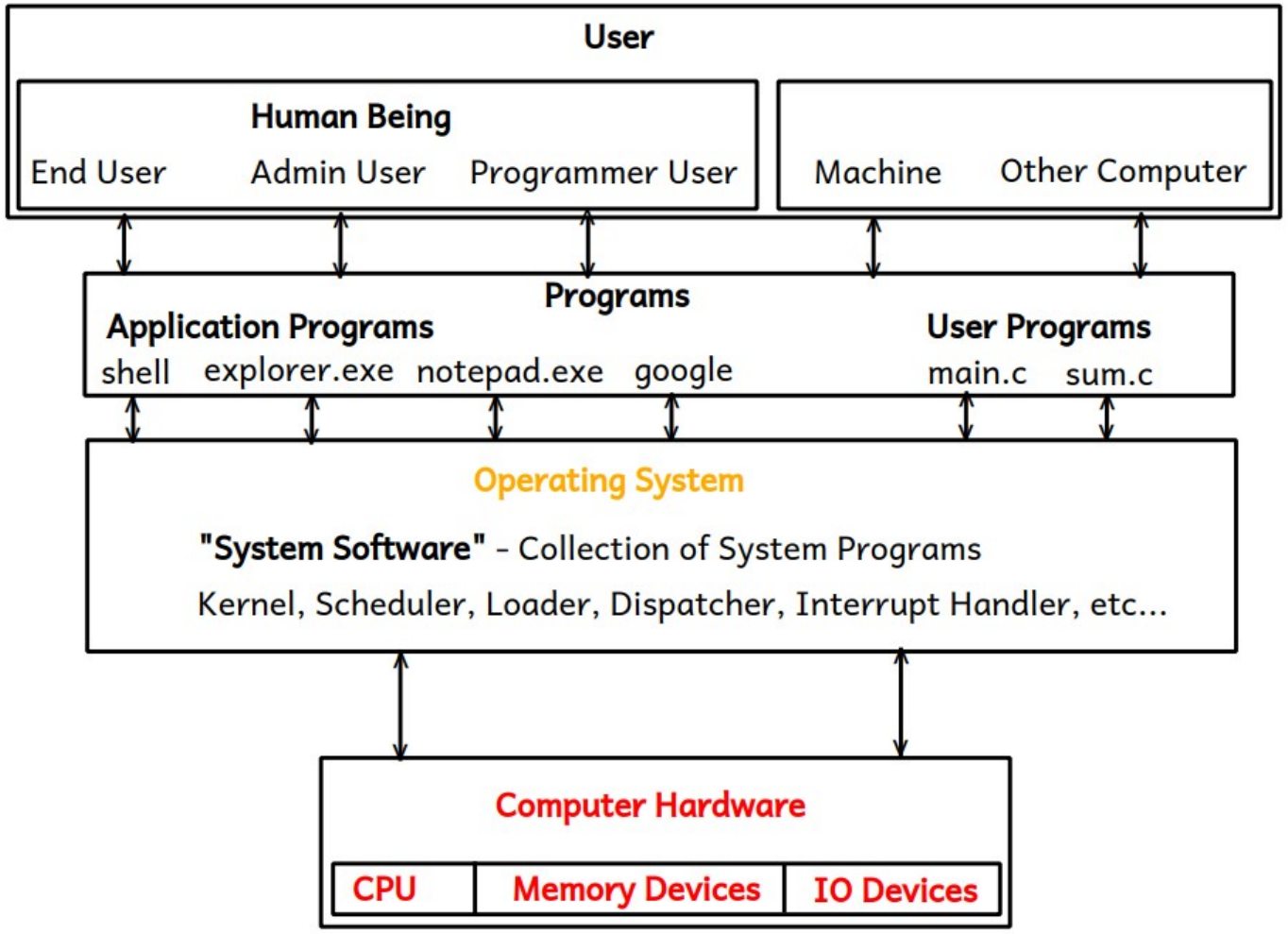
- Computer is a machine/hardware does different tasks efficiently & accurately.

- Basic functions of computer:

    1. Data Storage
    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



Operating System Concepts

- As any user cannot directly interacts with an OS, hence an OS gives two types of interfaces for the user in the form of programs:

1. CUI/CLI : Command User Interface/Command Line Interface

- in this type of interface, user can interacts with an OS by means of entering commands in a text format throgh command line.

e.g. cd, cp, gcc, ls, mv etc....

- In Linux name of the program which provides CUI is **shell/terminal**

- In Windows name of the program which provides CUI is **cmd.exe/command prompt**

2. GUI: Graphical User Interface

- in this type of interface, user can interacts with an OS by means of making an events like left click, right click, click on buttons, menu bar, menu list etc....

- In Linux name of the program which provides GUI is **GNOME/KDE**

- In Windows name of the program which provides GUI is **explorer.exe**

Operating System Concepts

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.

- Three 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...

Q. What is an IDE (Integrated Software Developement) ?

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

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

1. "Editor": it is an application program also used to write a source code in any programming lanaguage.

e.g. notepad, vi editor, gedit etc...

2. "Preprocessor": it is an application program gets executes before compilation and does two jobs - it executes all preprocessor directives and removes all comments from the source code.

e.g. cpp(c preprocessor)

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...

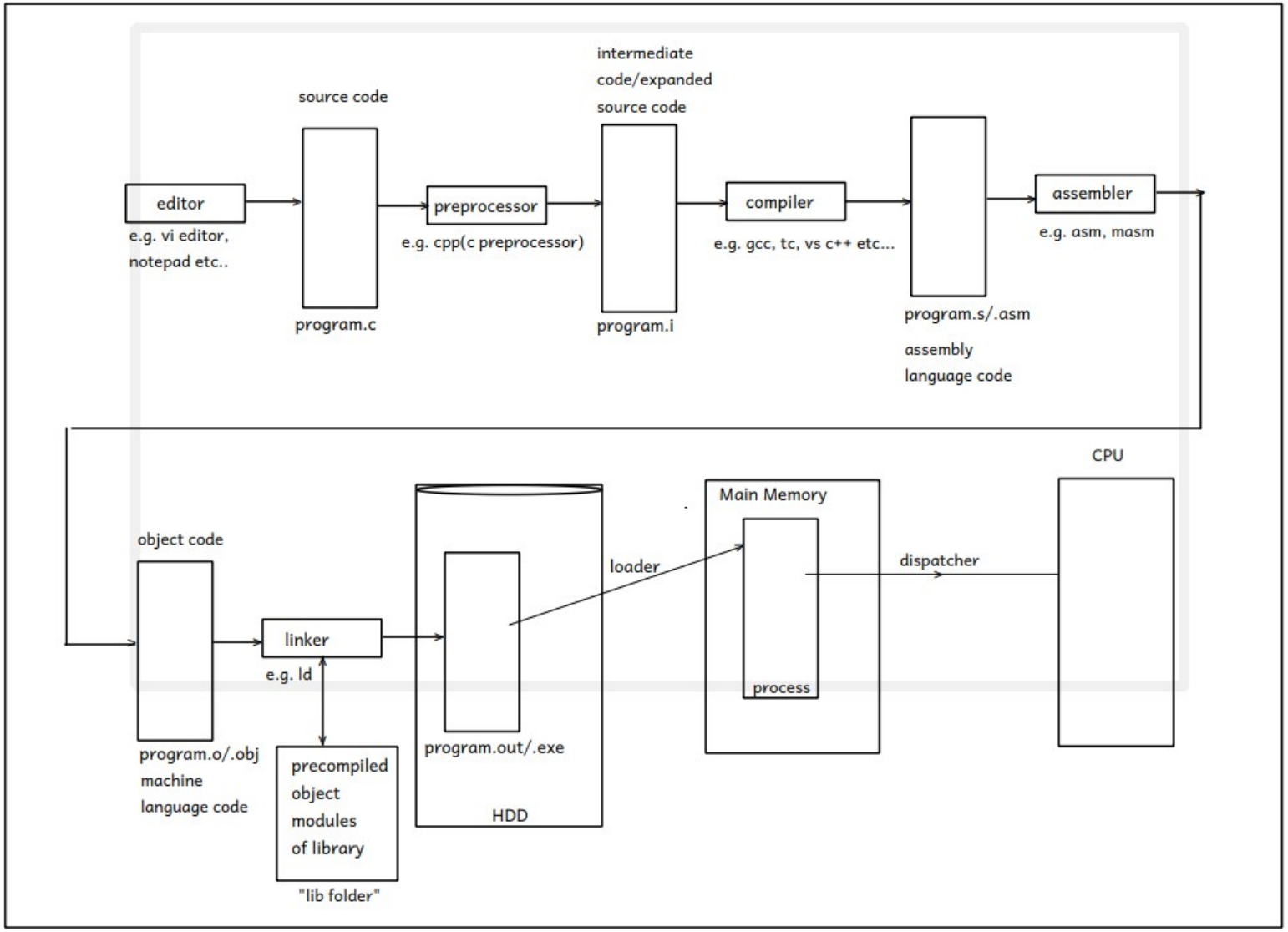
4. "Assembler": it is an application program which converts assembly language code into machine language code/object code.

e.g. masm, tasm etc...

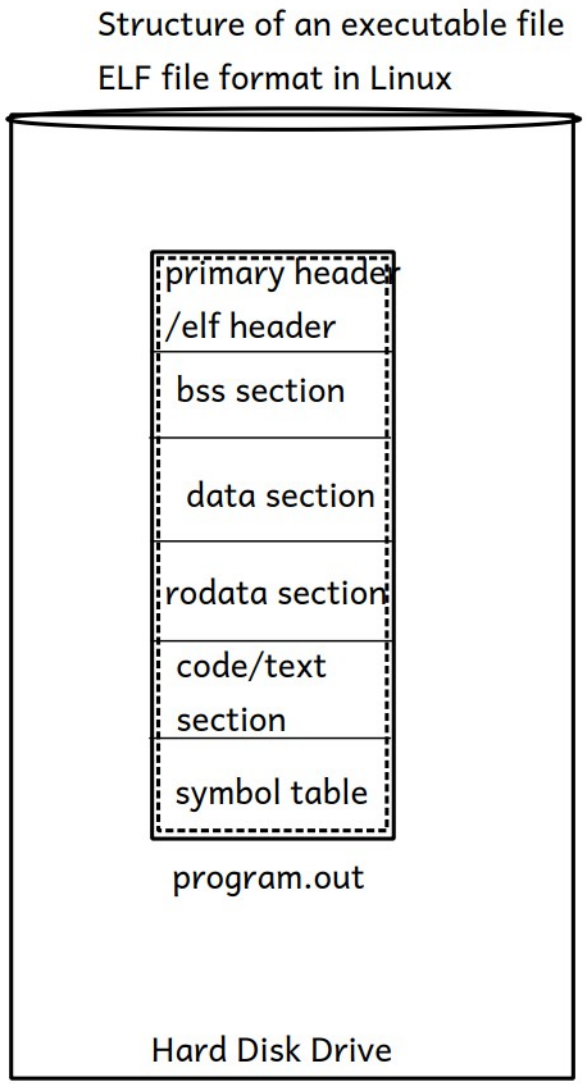
5. "Linker": it is an application program which links object file/s in a program with precompiled object modules of library functions exists in a lib folder and creates final single executable file.

e.g. ld: link editor in Linux.

Operating System Concepts



Operating System 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.

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

OS installation: to install any OS onto the machine, is noting but to store **OS software** (i.e. collection of thousands of system programs and application programs which are in a binary format) onto the HDD.

Booting:

- If an OS wants to becomes active, then first atleast its core program i.e. kernel must be laoded into the main memory.

- Loading of kernel program of an OS from HDD into the main memory is called as booting.

- **Bootstrap program** is responsible to load kernel from the HDD into the main memory.

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.

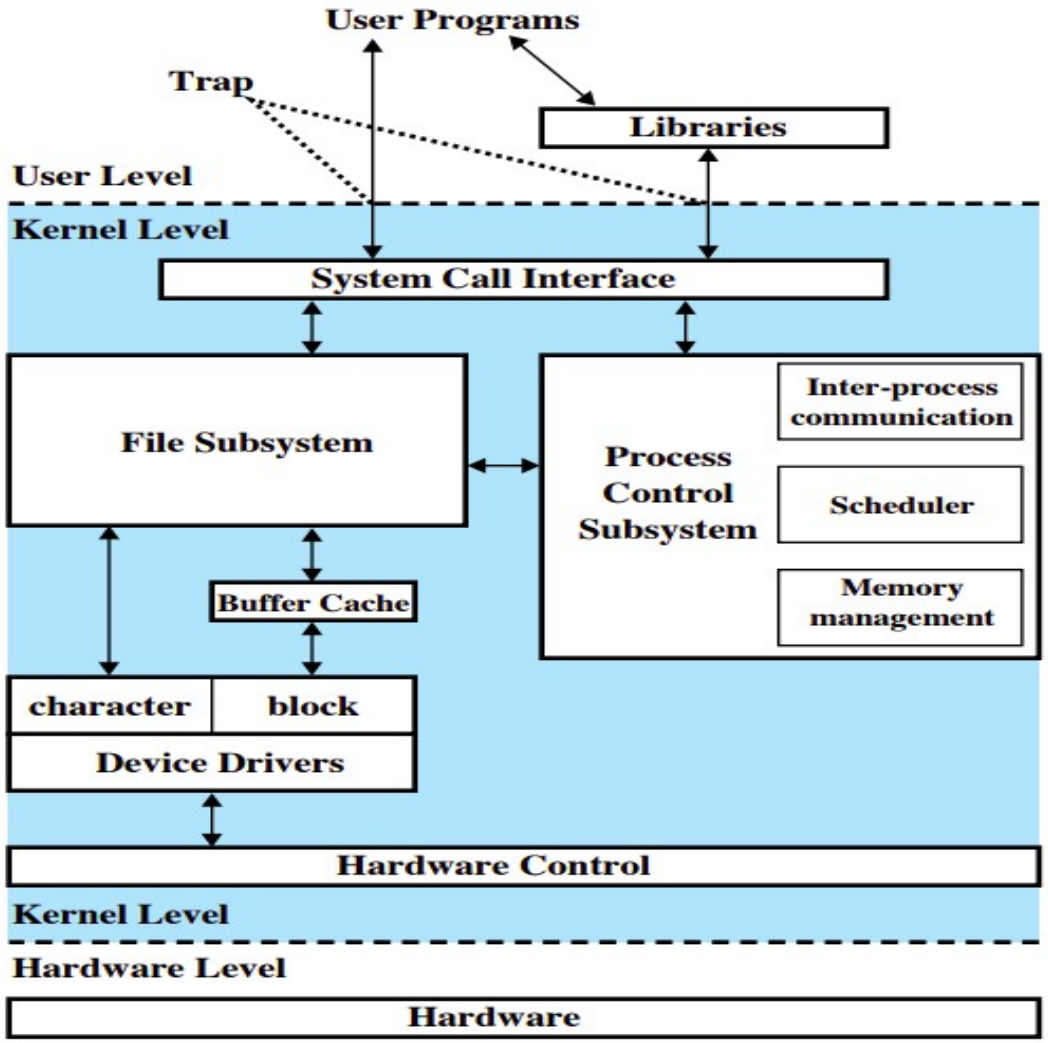
- It was first run on a machine **DEC-PDP-7** (Digital Equipment Corporation – Programmable Data Processing-7).

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

- UNIX was basically 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, 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 that can be stored is considered as a **file** and whatever is in a 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.

- 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.**

 # 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 which is divided logically into sections like exe header, bss section, data section, rodata section, code section, symbol table etc...

Q. What is a Process?

User view:

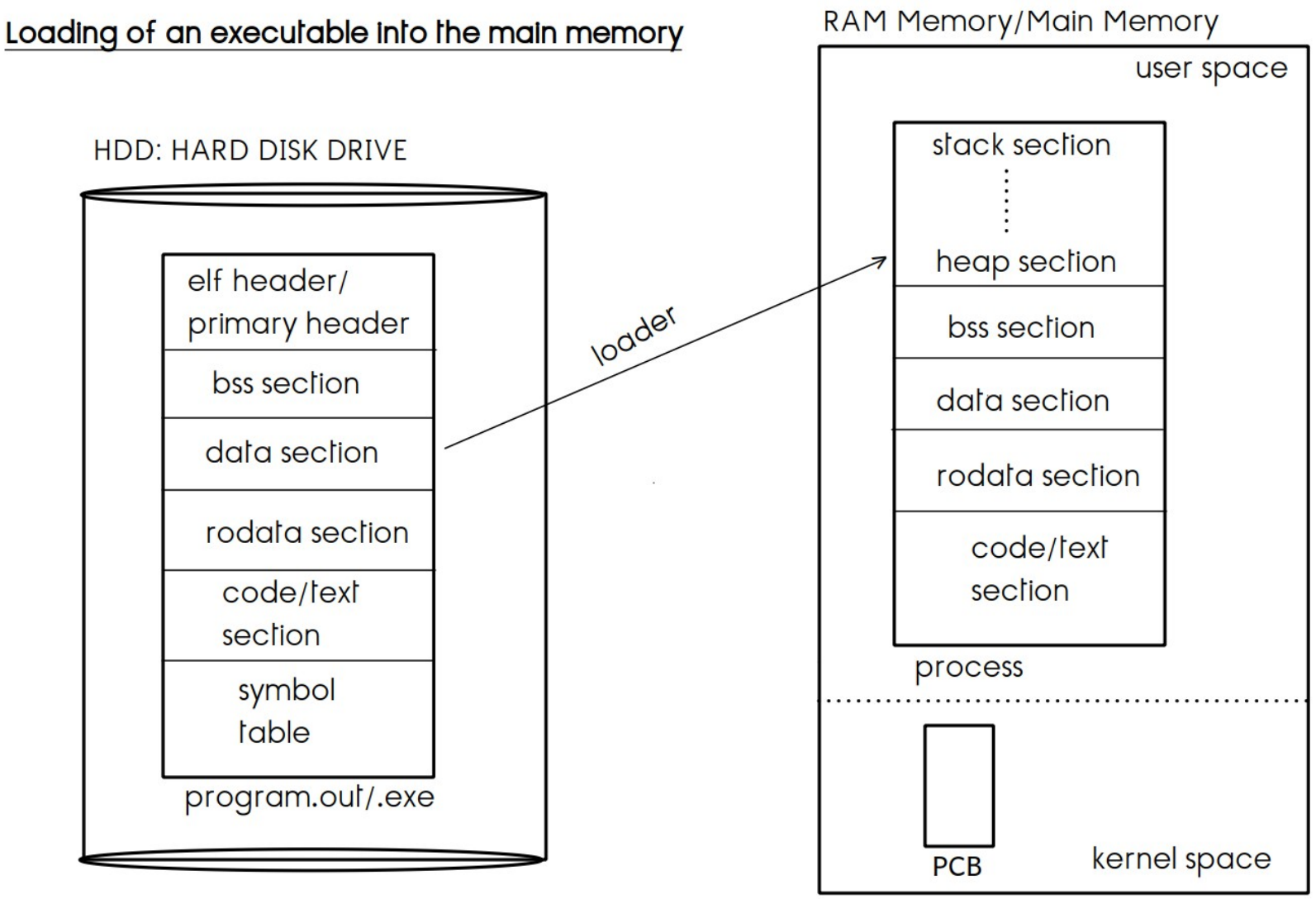
- Program in execution is called as a process.
- Running program is called as a process.
- When a program gets loaded into the main memory it is reffered as a process.
- Running instance of a program is reffered as a process.

System view:

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

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 for that process 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.**

- Upon submission of a process i.e. per process one PCB gets created and it 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:

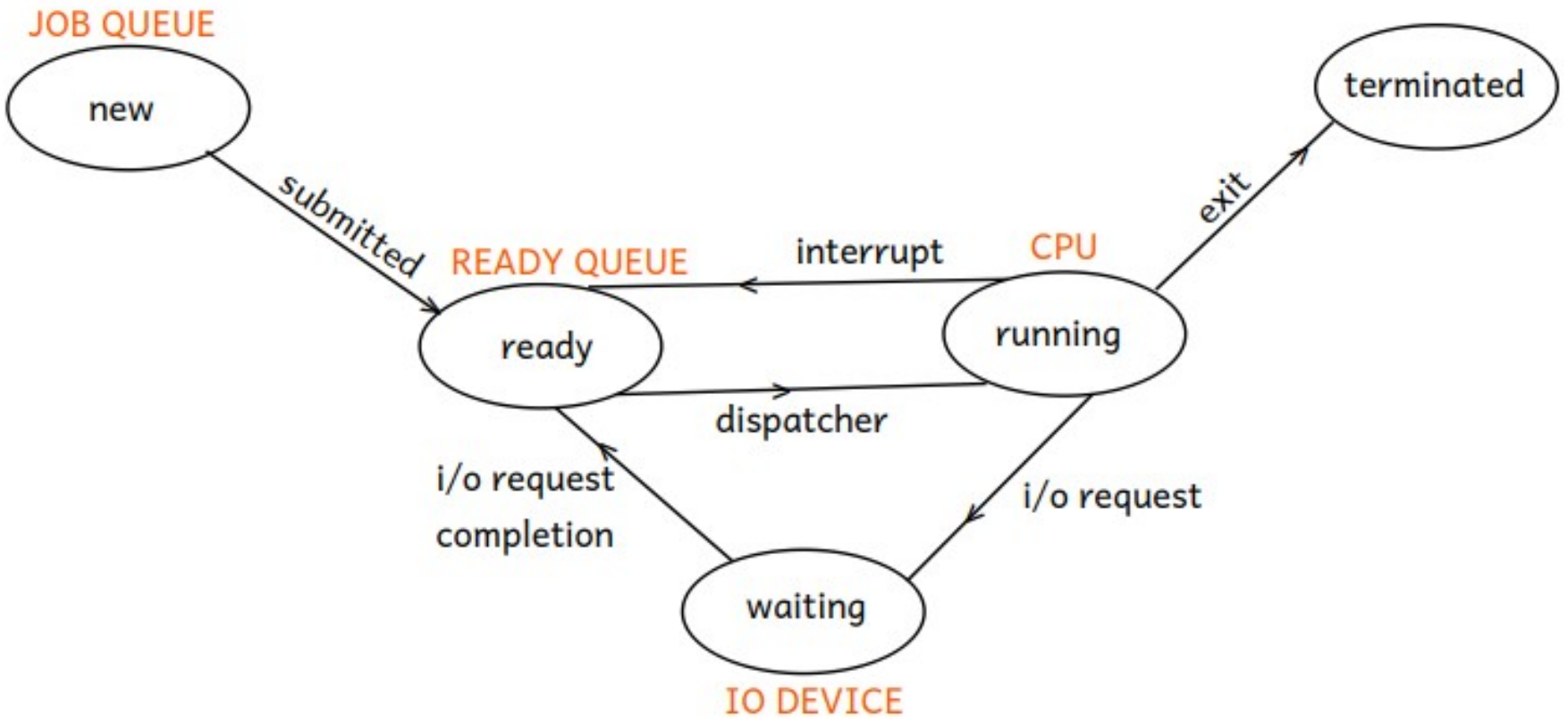
1. New state: upon process submission or when a PCB for a process gets created into the main memory process is in a new state.

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



PROCESS STATE DIAGRAM

Operating System Concepts

1. multi-programming: system in which more than one processes can be submitted/system in which 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, the CPU can execute more than one programs simultaneously/concurrently (i.e. one after another).

- the speed at which the CPU executes multiple programs simultaneosuly, it seems that it executes multiple programs at a time.

At a time the CPU can execute only one program.

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

3. multi-threading: system in which the CPU executes more than one threads which are of either same process or are of different processes simultaneously/concurrently, the speed at which it executes multiple threads simultaneosuly, it seems that it executes multiple threads at a time.

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.

Operating System Concepts

- 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 the main memory and waiting for the CPU time.

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 or 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 or 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.

- 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:

- When the CPU switched from one process to another process, it switches from an execution context of one process into 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, i.e. dispather copies an execution context of process which is 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 - due to an exirt

Case-2:: Running -> Waiting – due an io request

Case-3:: Running -> Ready – due to an interrupt

Case-4:: Waiting -> Ready – due to an io request completion

- 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 non-preemptive cpu scheduling takes place.

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

e.g. in above case-3 & case-4 preemptive cpu scheduling takes place.

- 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/solutions 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 criterias called as **scheduling criterias**: **cpu utilization, throughput, waiting time, response time and turn-around-time.**

 # 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 algorithm in which throughput must be as **maximum** possible.

3. Waiting Time: it is the toal 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 the 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.

- 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.

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.**