Operating System Concepts

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

* Introduction:

- Why there is need of an OS?

- What is an OS?

- Booting process in brief

- Functions of an OS

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.

- 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
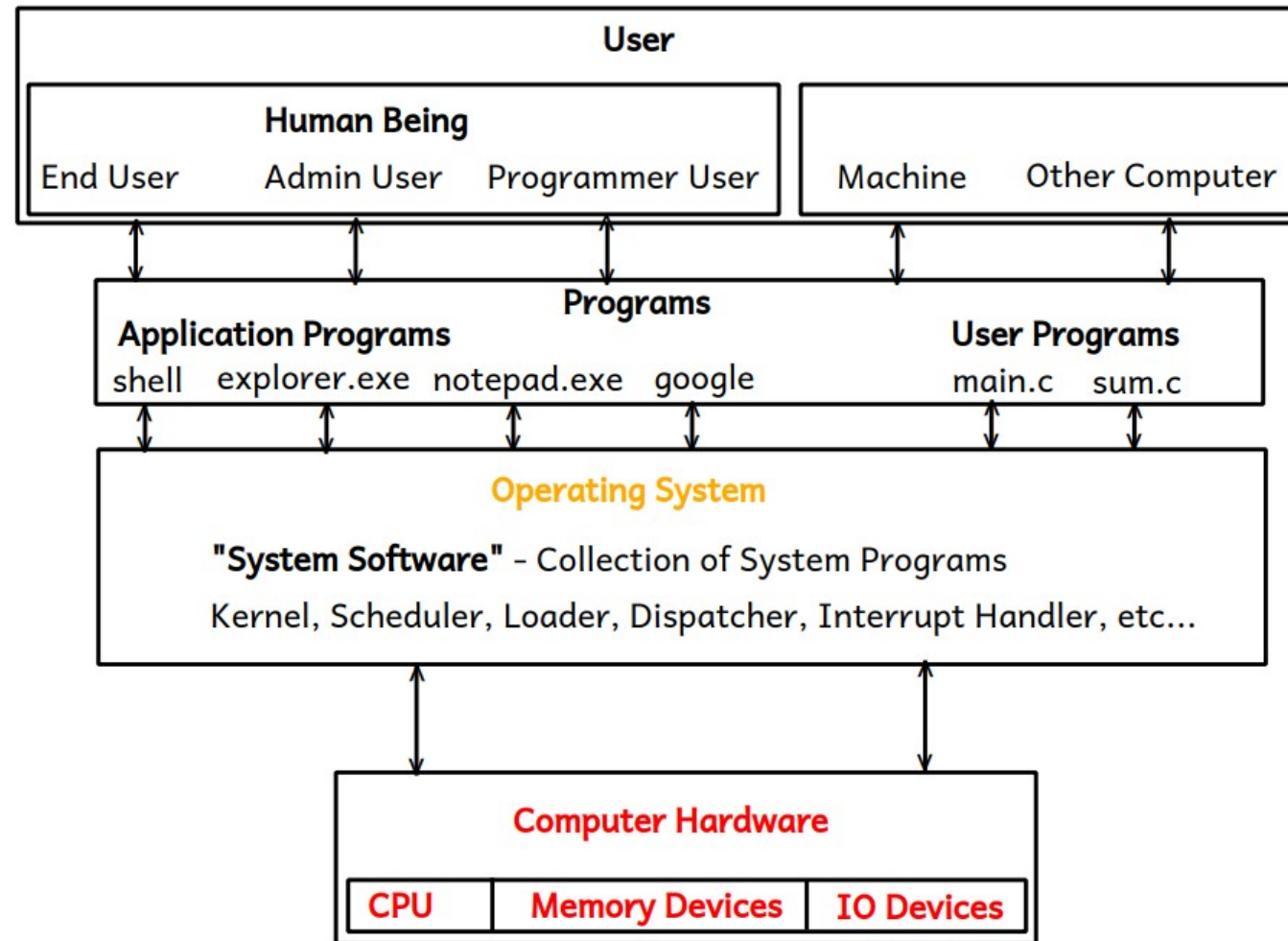
Operating System Concepts

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...

Operating System Concepts

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 used for to write a source code.

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

3. "Compiler": it is an application program which convert 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.
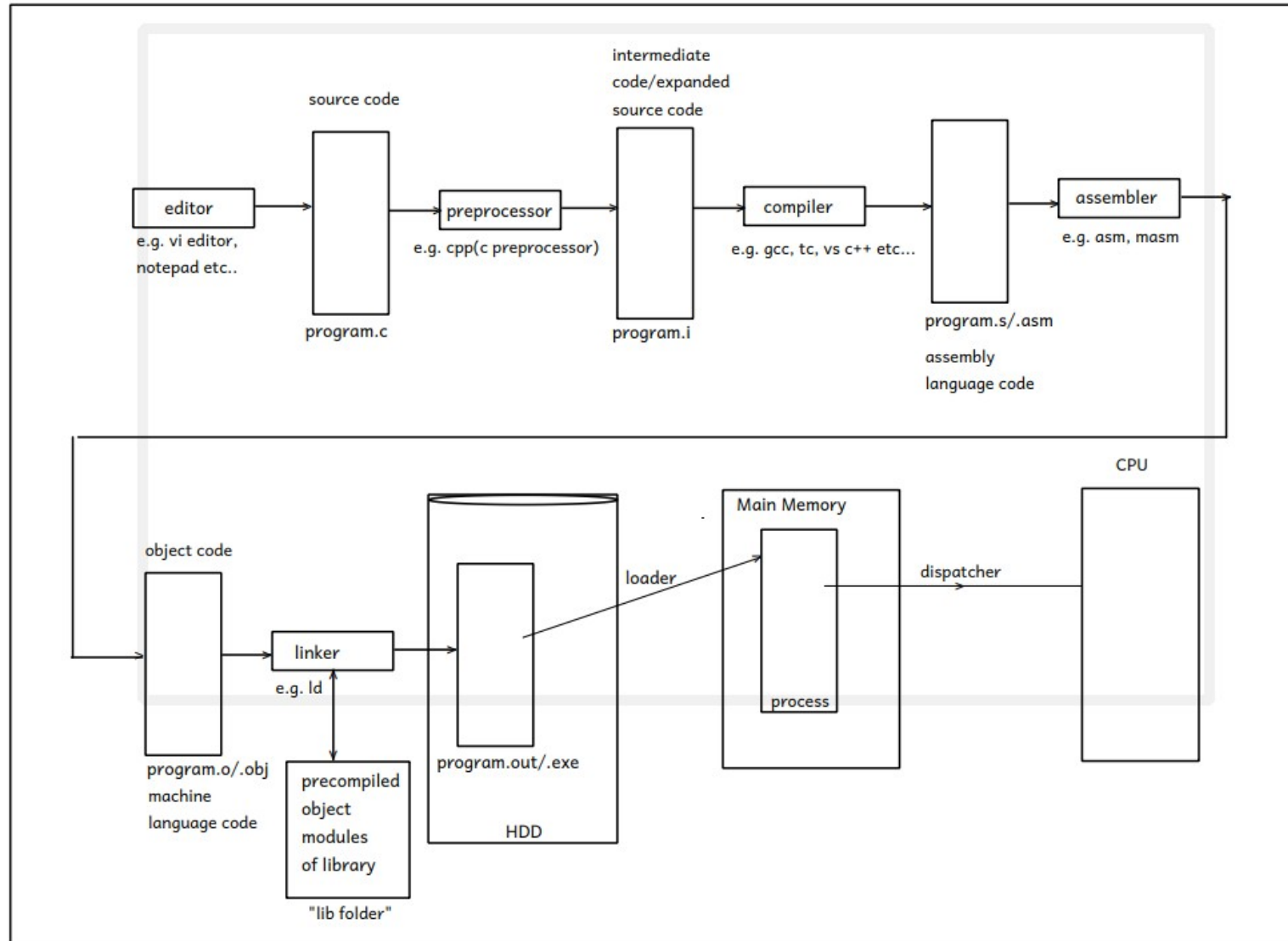
e.g. masm, tasm etc...

- Program written in any programming language is called as a **"source code".**

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.
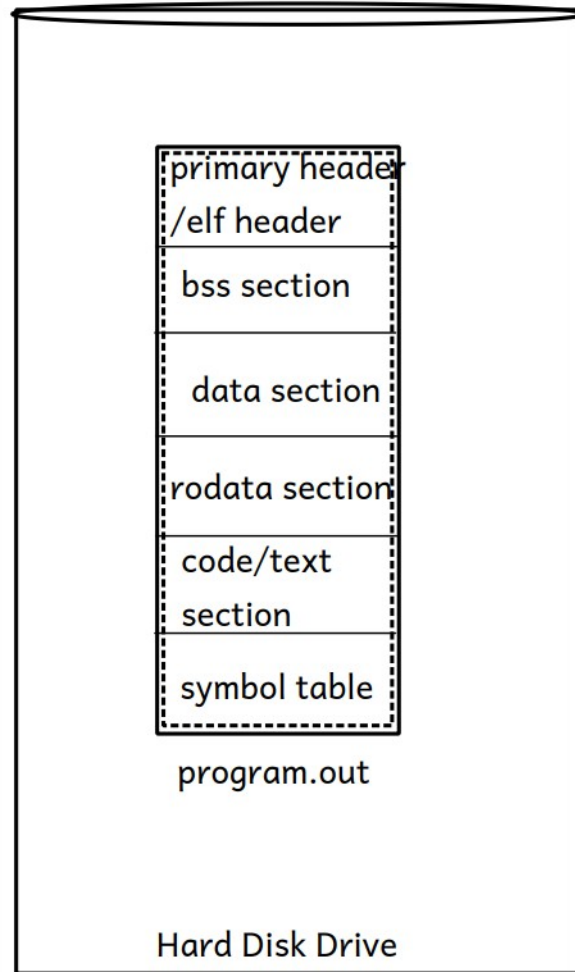
e.g. ld: link editor in Linux.

Operating System Concepts

Operating System Concepts

Structure of an executable file
ELF file format in Linux

| primary header /elf header |
| bss section |
| data section |
| rodata section |
| code/text section |
| symbol table |

program.out

Hard Disk Drive

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

Operating System Concepts

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:

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
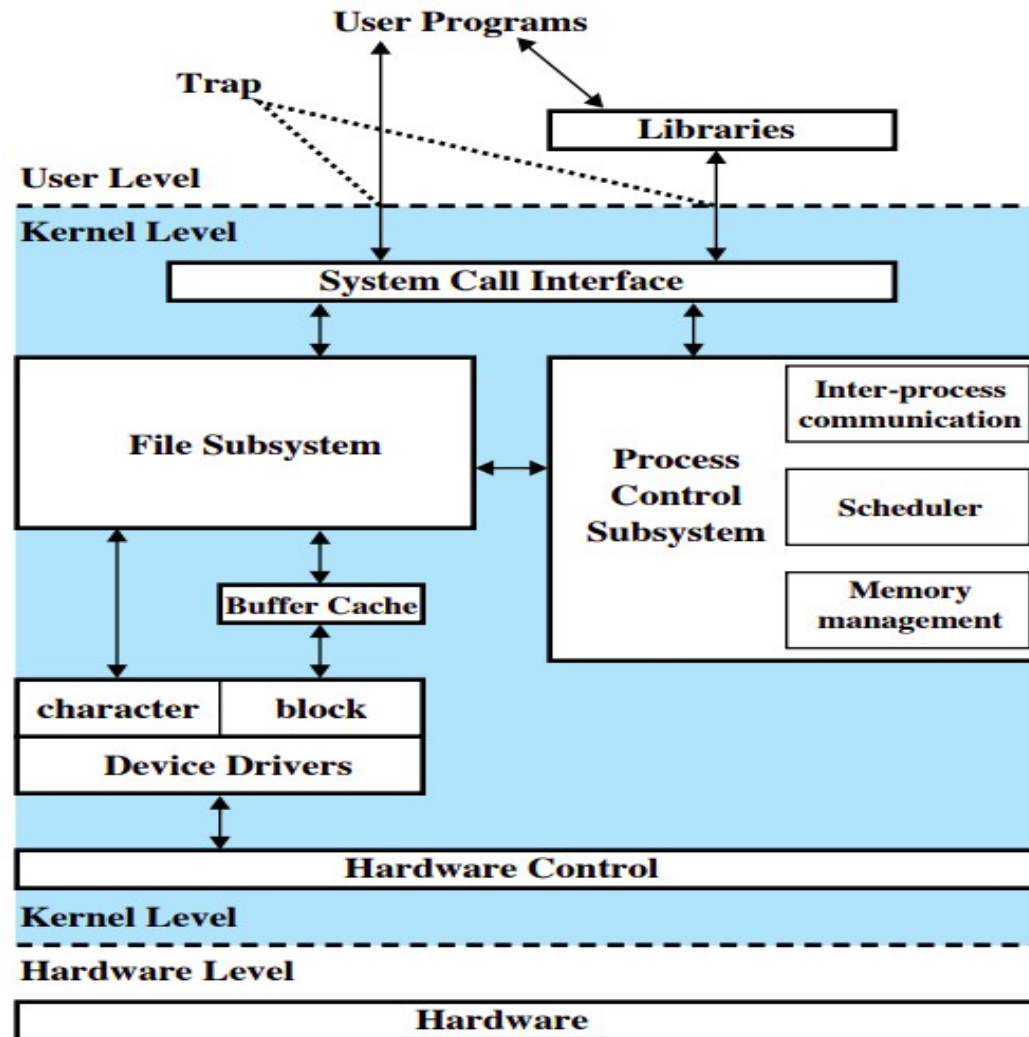
UNIX Operating System:

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

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

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

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

- UNIX was specially designed for developers by developers

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

Operating System Concepts

- Kernel acts as an interface between programs and hardware.

- Operating System has subsystems like **System Call Interface, 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.

Operating System Concepts

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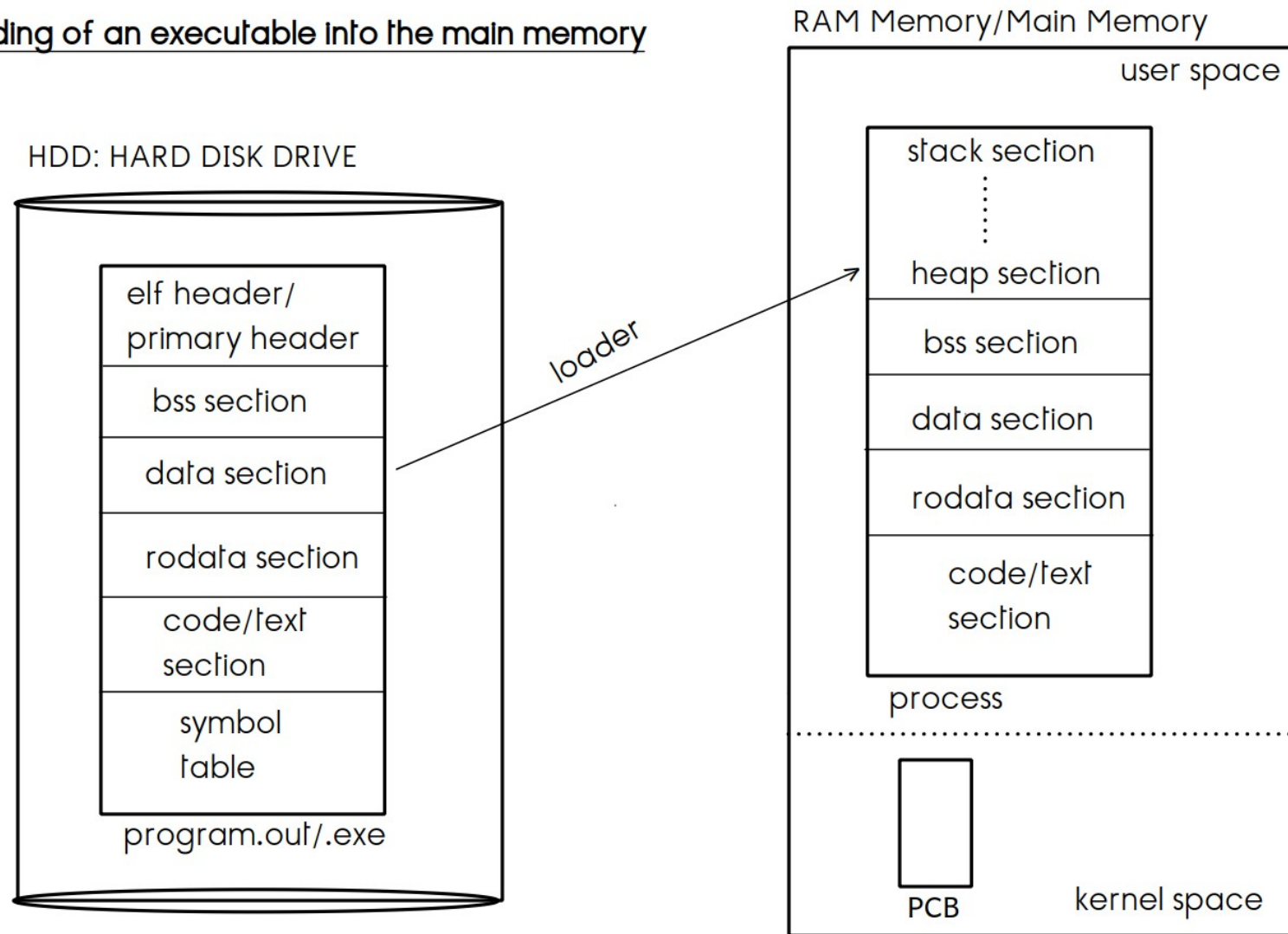
stack section: contains function activation records of called functions.

heap section: dynamically allocated memory

Operating System Concepts

Loading of an executable into the main memory

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 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.