Operating Systems

Module Introduction

Networking → 5Q. → Section B.

- CCAT Exam
  - Operating Systems: 5 Questions → Section B
  - Computer Fundamentals + ~~Networking~~: 10 Questions → Section A.
- Quiz
  - Operating Systems : 4 Quizzes (10 marks)
  - Computer Fundamentals: 1 Quiz (10 marks)
  - Operating Systems: Module end quiz (20 marks)

Concepts of Programming.

- GitLab Repository
  - OS Galvin Slides
  - Practice MCQ
  - Notes/Diagrams (Daily)
  - Reference Book:
    Operating System Concepts - Galvin

Operating System Concepts

- **Introduction**
  - Introduction to Operating System, What is OS, Booting the System

- **System Architecture Design of OS**
  - System Calls, Dual Mode Operation: System mode and Kernel mode

- **Process Management**
  - What is Process & PCB?
  - States of the process
  - CPU scheduling & CPU scheduling algorithms
  - Inter Process Communication: Shared Memory Model & Message Passing Model
  - Process Synchronization/Co-ordination
  - Deadlocks & deadlock handling methods

Operating System Concepts

* **Memory Management**
  - Swapping
  - Memory Allocation Methods
  - Segmentation
  - Paging
  - Virtual Memory Management

Operating System Concepts

File Management

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

Operating System Concepts

Introduction:

- Why there is need of an OS?
- What is an OS?
- History of OS- Multi-threading, multiprocessing etc
- Functions of an OS
- What is Process?
- States of Process

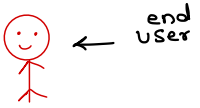
Program :- Set of Instn.
(executable file) → .exe

Software :- Set of Program.

Utility :- Prog associate with OS.

IDE → Integrated Development Environment

· C → Src Code ·



Editor | IDE | Web Browser | Media player ..... ← application Software.

C
C++
Java
python.

① Interface.
② Resource Mgn.
③ Control Prog.
④ CD/DVD → Core OS + appln software + Utility.
⑤ Kernel. → Core OS

Operating System / Kernel.

CPU | RAM | HDD | KBD | Monitor → Computer Hardware.

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

Q. What is an Operating System?

- An OS is a **system software** (i.e. collection of system programs) which acts as an interface between the 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 a **resource allocator.**
- An OS controls the execution of all programs and it also controls hardware devices which are connected to the computer system hence it is also called a **control program.**
- An OS manages limited available resources among all running programs, hence it is also called a **resource manager.**

Operating System Concepts

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

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

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

2. Utility Software: e.g. disk manager, Control panel, windows firewall, anti-virus software etc...

3. Application Software: e.g. google chrome, Shell, notepad, MS Office etc.

Operating System Concepts

Q. What is a Software?

-Software is a collection of programs.

Q. What is a Program?

- Program is a set of instructions written in any programming language (either low-level or high-level programming language) given to the machine to do a 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 come 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 into an OS/part of an OS.
   e.g. Kernel, Loader, Scheduler, Memory Manager etc...

 # Functions of an OS:

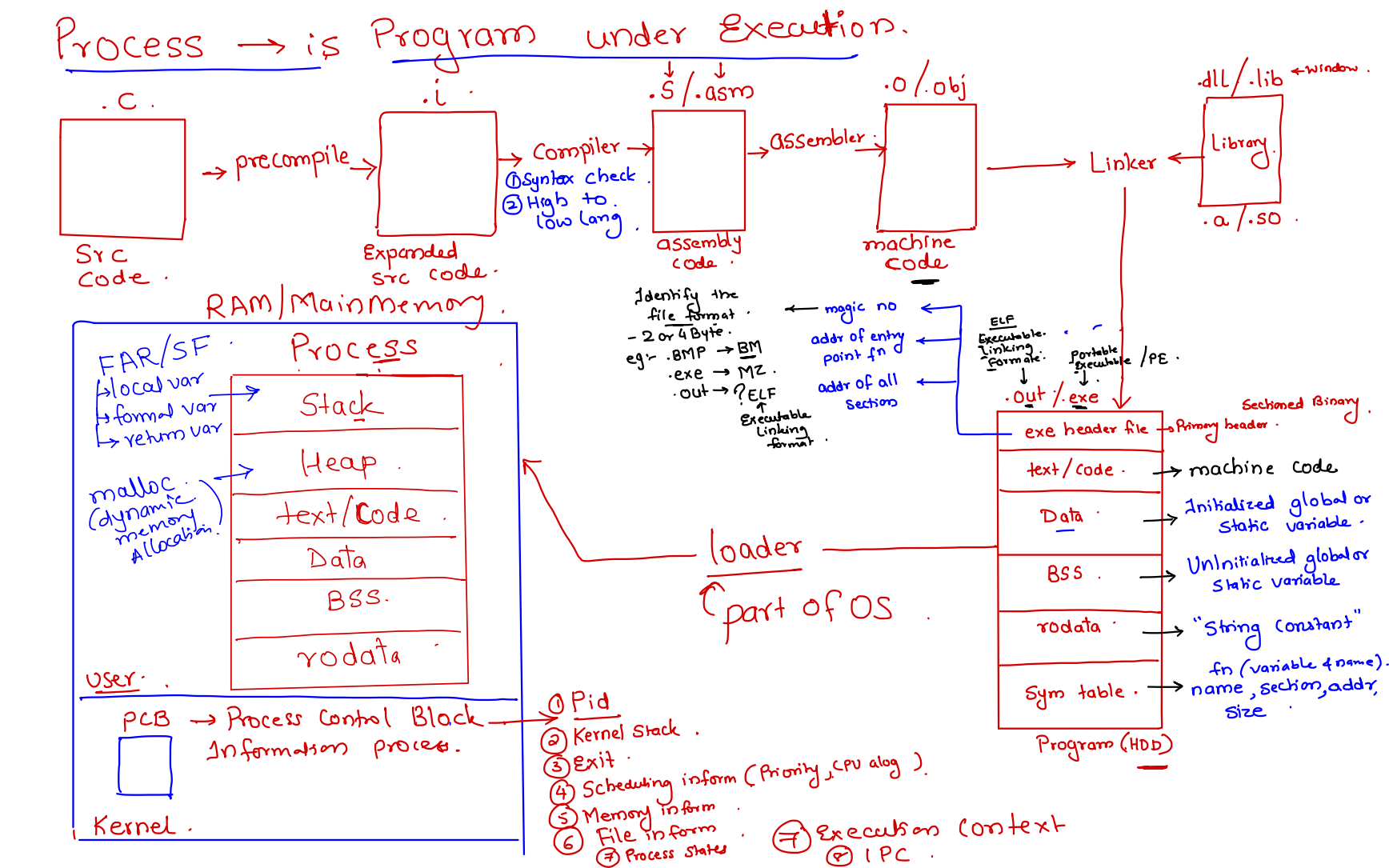
Basic minimal functionalities/Kernel functionalities:

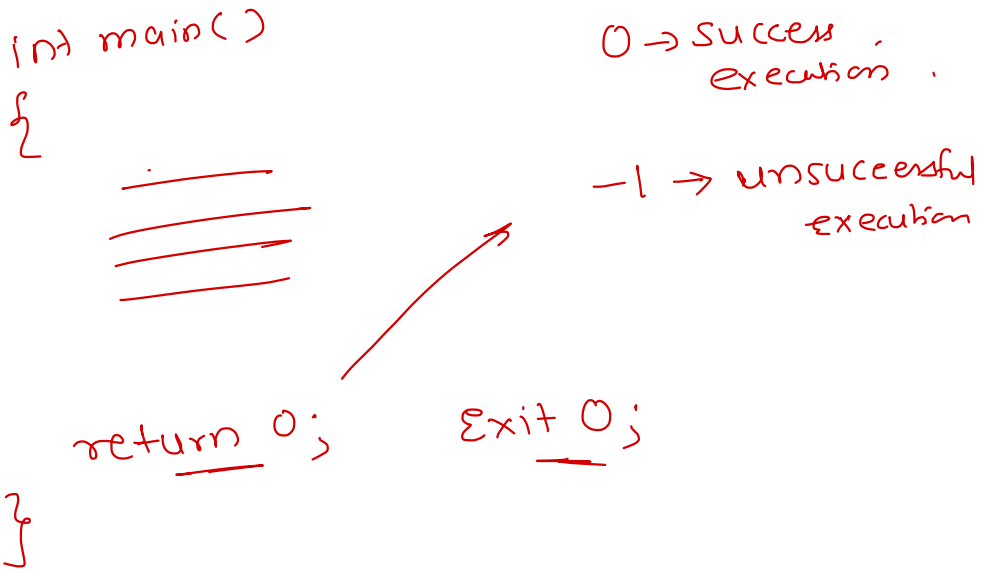
1. Process Management
2. Memory Management
3. Hardware Abstraction
4. CPU Scheduling
5. File & IO Management

Kernel fn.

Extra utility functionalities/optional:

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


Process → is Program under Execution.
.c
Src code
precompile
.i
Expanded src code
Compiler
① Syntax check
② High to low lang
.s / .asm
assembly code
assembler
.o / .obj
machine code
Linker
.dll / .lib ← window
Library
.a / .so
RAM / Main Memory
Process
Stack
Heap
text / Code
Data
BSS
rodata
FAR / SF
→ local var
→ formal var
→ return var
malloc (dynamic memory Allocation)
User
PCB → Process Control Block
Information process
Kernel
Identify the file format
- 2 or 4 Byte
eg:- .BMP → BM
.exe → MZ
.out → ?
ELF Executable Linking format
magic no
addr of entry point fn
addr of all Sections
ELF Executable Linking format
Portable Executable / PE
.out / .exe
Sectioned Binary
exe header file
Primary header
text / code
machine code
Data
Initialized global or Static variable
BSS
Uninitialized global or Static variable
rodata
"String Constant"
Sym table
fn (variable & name) name, section, addr, size
Program (HDD)
loader
part of OS
① Pid
② Kernel Stack
③ Exit
④ Scheduling inform (Priority, CPU alog)
⑤ Memory inform
⑥ File inform
⑦ Process States
⑦ Execution context
⑧ IPC


int main()
{
return 0;
Exit 0;
}
0 → success execution.
-1 → unsuccessful execution

Q. What is an IDE (Integrated Development Environment) ?

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

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.

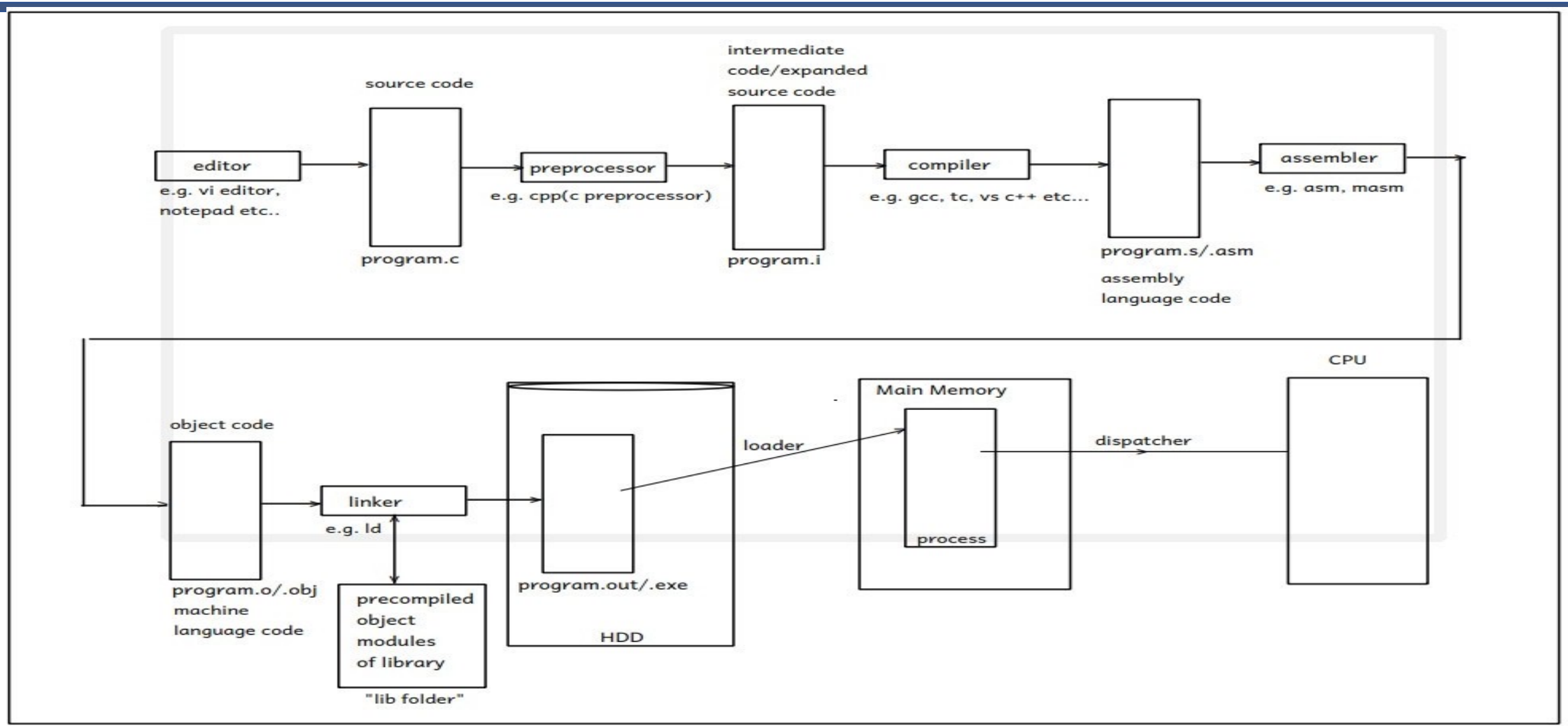
e.g. masm, tasm etc...

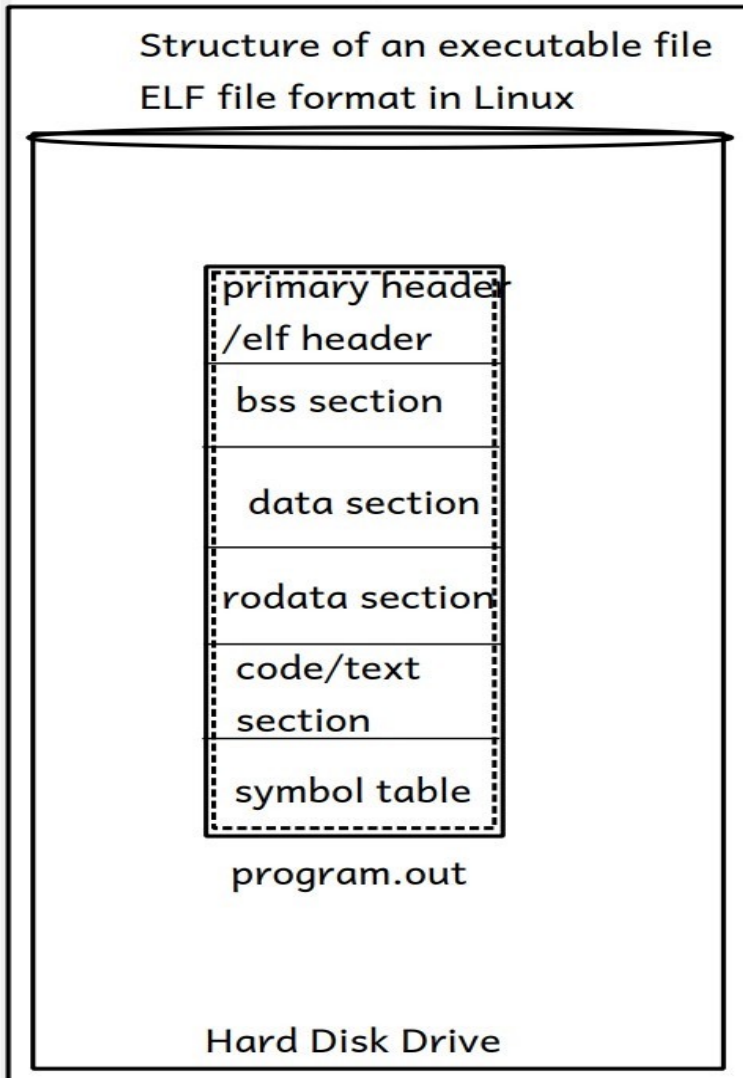
- Program written in any programming language is called as a **"source code".**

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.

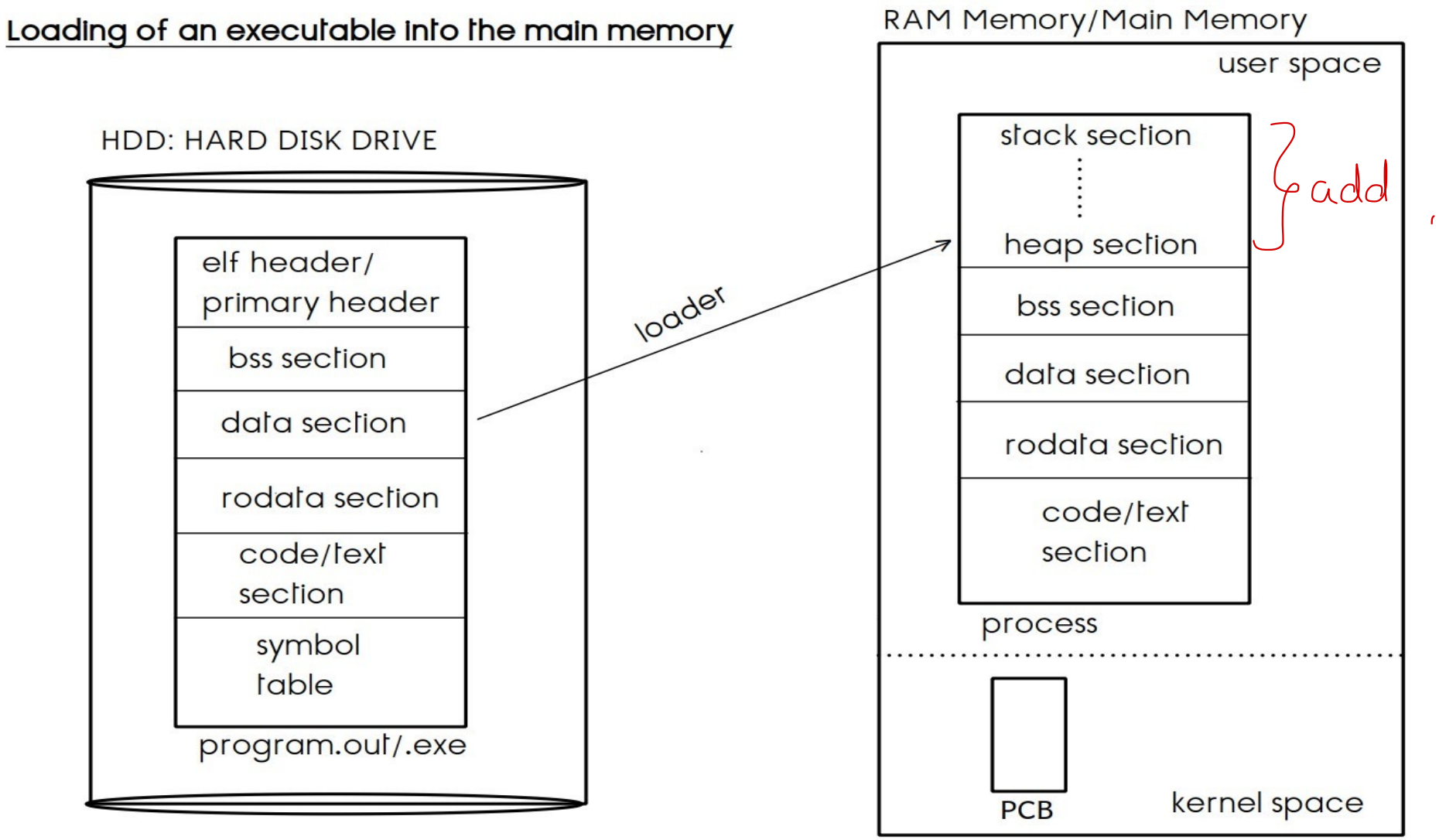
Operating System Concepts





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.

Operating System Concepts



Process

- Program under execution.
- Process execute in RAM.
- The process control block contains information about the process (required for the execution of the process).
  - Process id
  - Exit status
  - Scheduling information (State, Priority, Sched algorithm, Time, ...)
  - Memory information (Base & Limit, Segment table, or Page table)
  - File information (Open files, Current directory, ...)
  - IPC information (Signals, ...)
  - Execution context
  - Kernel stack
- PCB is also called process descriptor (PD), uarea (UNIX), or task_struct (Linux).

Q. What is a Process? User view:

- A program in execution is called a process.
- Running a program is called a process.
- When a program gets loaded into the main memory it is referred to as a process.
- Running an instance of a program is referred to as a process.

System view:

- The 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

- file format of an executable file in Windows is **PE** (Portable Executable),

whereas the file format of an executable file in Linux is **ELF (Executable & Linkable Format).**

- The file format is a specific way to store data & instructions of a program inside an executable file, and it is different in diff OS.

- ELF file format divides an executable file logically into sections and inside each section specific contents can be kept in an organized manner:

1. elf header
2. bss section (block started by symbol)
3. data section
4. rodata (read-only data)section
5. code/text section
6. symbol table

Memory Layout of Program and Process

Program Consist of
exe header/primary header
Block started by symbol (bss) section (un initialized static / global variables)
Data section (initialized static / global variables)
Rodata Section(Constant/literals)
code/text section (contains executable instructions)
Symbol Table

Process Consist of
Skipped
Block started by symbol (bss) section (un initialized static / global variables)
Data section (initialized static / global variables)
Rodata Section(Constant/literals)
code/text section (contains executable instructions)
Skipped
Stack Section
Heap Section

Process and Program

- A **process** is an instance of a program in execution.
- Running a program is also known as **Process.**
- When a program gets loaded into memory is also known as **Process.**
- A **Program** is a set of instructions given to the machine to do a specific task.

Interaction with an OS : Two Types of Interface (CUI and GUI)

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

- By using this kind of interface user can interacts with an OS by means entering commands onto the terminal/command line in a text format.

e.g. In Windows name of the program which provide CUI => cmd.exe command prompt

In Linux name of an application program which provides CUI => shell/terminal

In MSDOS name of the program which provides CUI => command.com (MicroSoft Disk Operating System).

2. GUI : Graphical User Interface

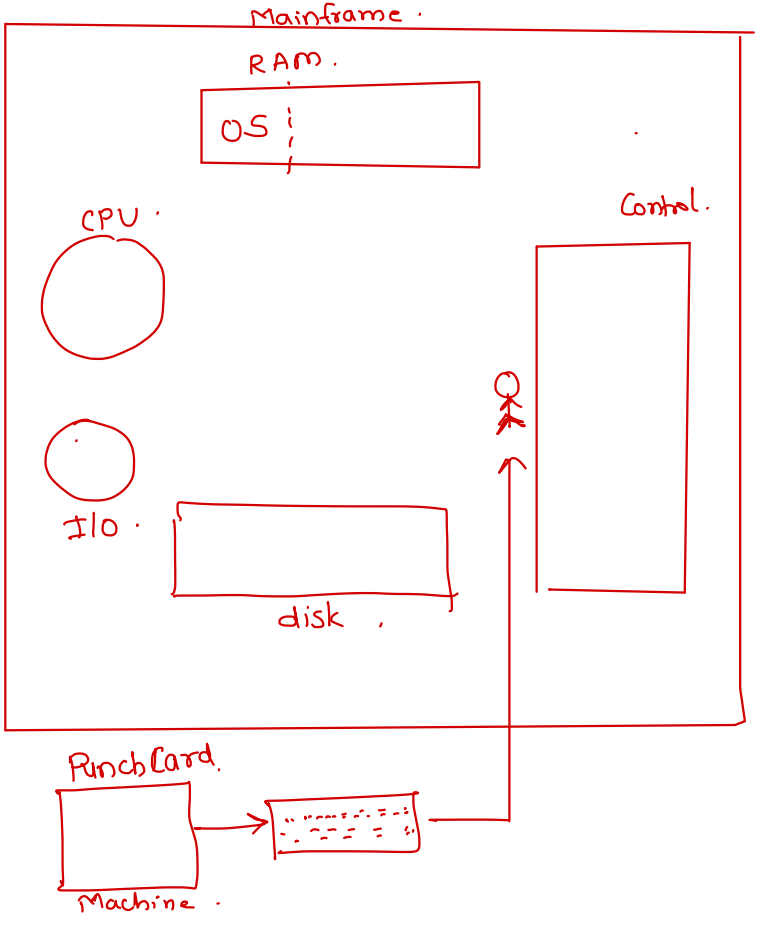
- by using this kind of interface user can interacts with an OS by means making an events like click on buttons, left click/rigyht click/double click, menu bar, menu list etc.....

- Windows = User friendly GUI.

e.g. In Windows name of an application program which provides GUI => explorer.exe

In Linux name of an application program which provides GUI => GNOME/KDE (GNU Network Object Model Environment / Common Desktop Environment).

History of OS.



Program / Process ⇒ CPU Burst time + I/O Burst time.

(1) Resident Monitor.