Operating Systems

Module Introduction

- CCAT Exam
  - Operating Systems: 9 Questions
  - Computer Fundamentals + Networking: 10 Questions
- Quiz
  - Operating Systems : 4 Quizzes (10 marks)
  - Computer Fundamentals: 1 Quiz (10 marks)
  - Operating Systems: Module end quiz (20 marks)

- 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

end
User

| Editor | IDE | Browser | Media player | . . . . | appln software |

Computer Hardware

| CPU | RAM | disk | KBD | Monitor |

① Interface
② Resource allocator
③ Control Prog
④ CD/DVD = Core OS + appln + ultility
⑤ Kernel = Core OS

OS → operating System

Program — set of Insth. (.exe)
'(·by the machine understandble) ,

.c → Src file ;

Software — set of Programs.

Utility →. Prog within the OS.

Operating System Concepts

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 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.**

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

Operating System Concepts

- From End <u>User:</u> An OS is a software (i.e. collection of programs) comes either in <u>CD/DVD</u>, has following main components:889IKM

1. <u>Kernel</u>: 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: e.g. disk <u>manager</u>, <u>Control panel</u>, <u>windows firewall</u>, <u>anti-virus software</u> etc...

3. Application Software: e.g. google chrome, shell, notepad, MS office etc.

IDE - Integrated Development Environment

C, C++ java

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 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...

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

Extra utility functionalities/optional:

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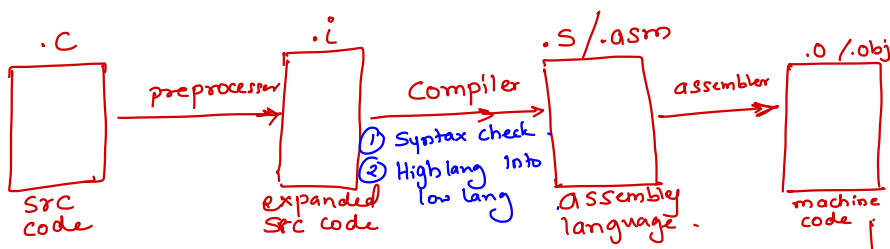
→ OS .

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 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...

.c
src code

→ preprocessor →

.i
expanded src code

① Syntax check
② High lang into low lang

→ Compiler →

.s/.asm
assembly language

→ assembler →

.o (.obj)
machine code

↓

Process → Program Under Execution

PE → Portable Executable

ELF → Excutable Linkable formate

.dll/.lib
Library
so /.a

→ Linker ←

↓

Identify file
  formfat 2 or 4byte ← magic number

.bmp = BM
.exe = MZ.
.out : ?ELF.

addr of entry point fn. ← main()

addr of each section.

.out /.exe.

→ harddisk

exe headerfile

ro data → "string constant"

bss. → Unintialiced global or static

Data. → Initialiced global or static variable.

text/code → machine code/ binary

Symbol. → addr, name, size

Program.

RAM
FAR/SF
↳ local var
↳ formal arg
↳ return ad
  variable

dynamic
malloc()

Stack
heap
ro data
Data / BSS
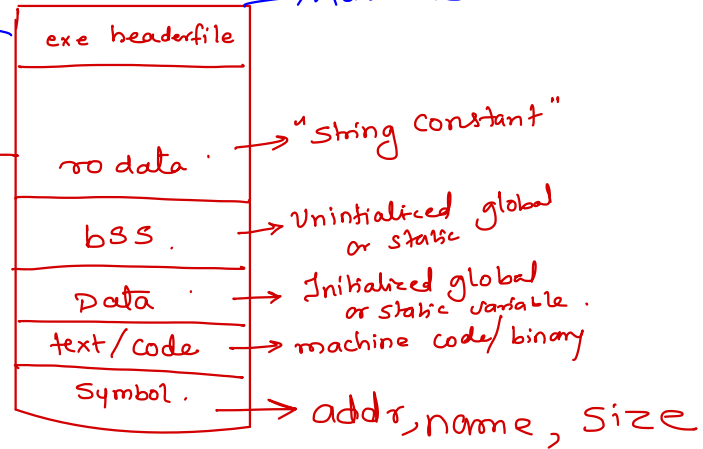text/code

← loader .
    ↑
  with OS

PCB → inform about the execution of process

OS

① Pid.
② Sched Info (state, priority, alog).
③ Mem Mang.
④ File I/o
⑤ Kernel Stack
⑥ Execution Unit.
⑦ Exit Status.
⑧ IPC.

```
main ( )
{ int a;

   ≡≡≡
 }  return 0;


main ( )
{

  }  exit (0);
```

success .
↑
0

−1 →

4. "Assembler": it is an application program which converts assembly language code into machine language code/object code.
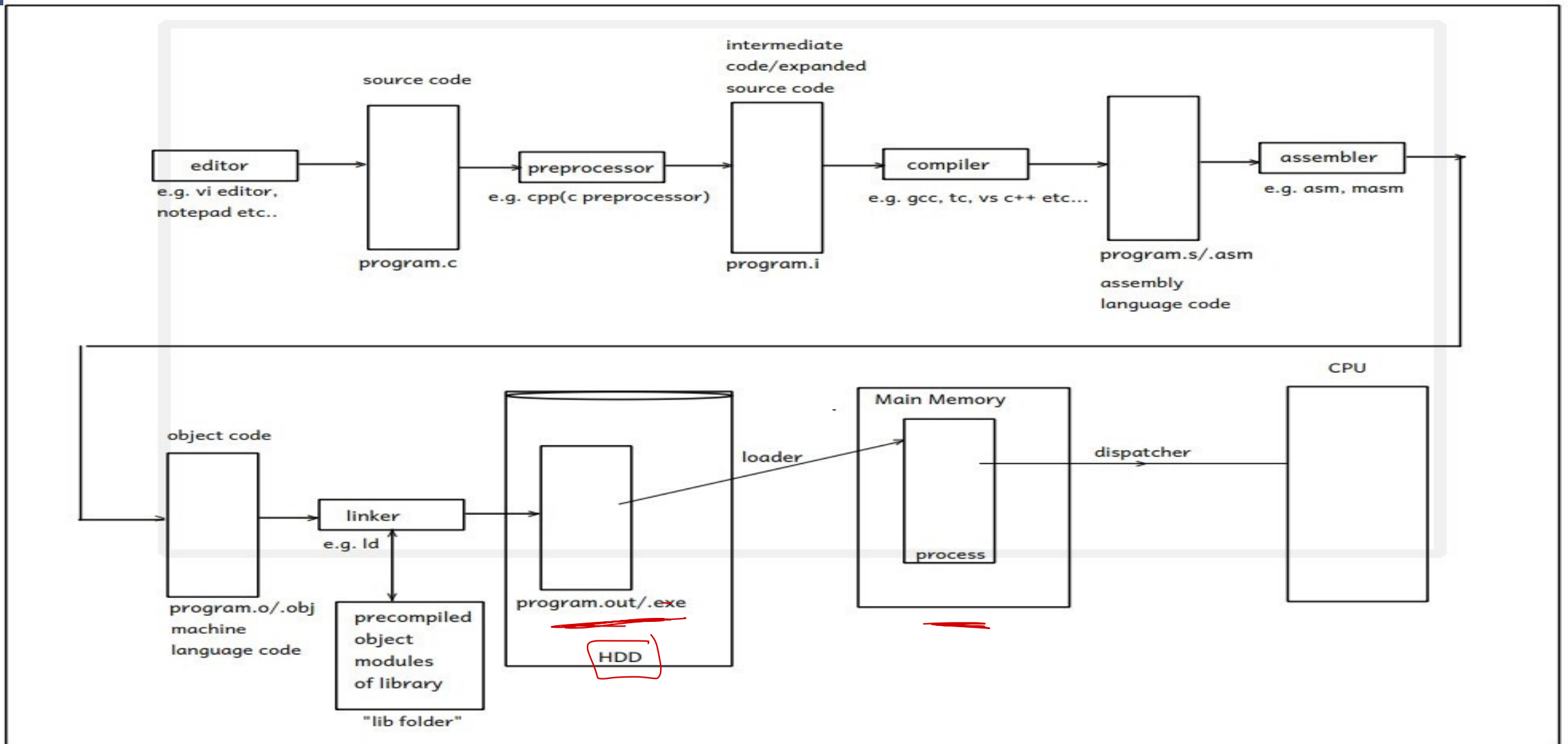
e.g. masm, tasm etc...

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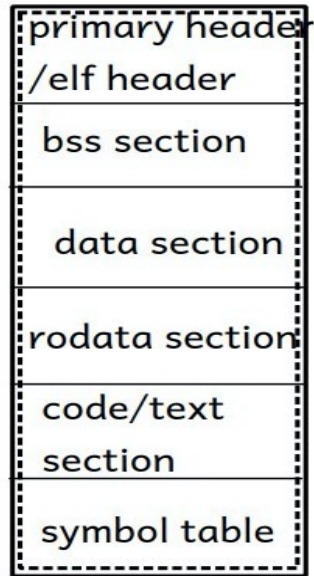
e.g. ld. link editor in Linux.

Operating System Concepts

Operating System Concepts

Structure of an executable file
ELF file format in Linux

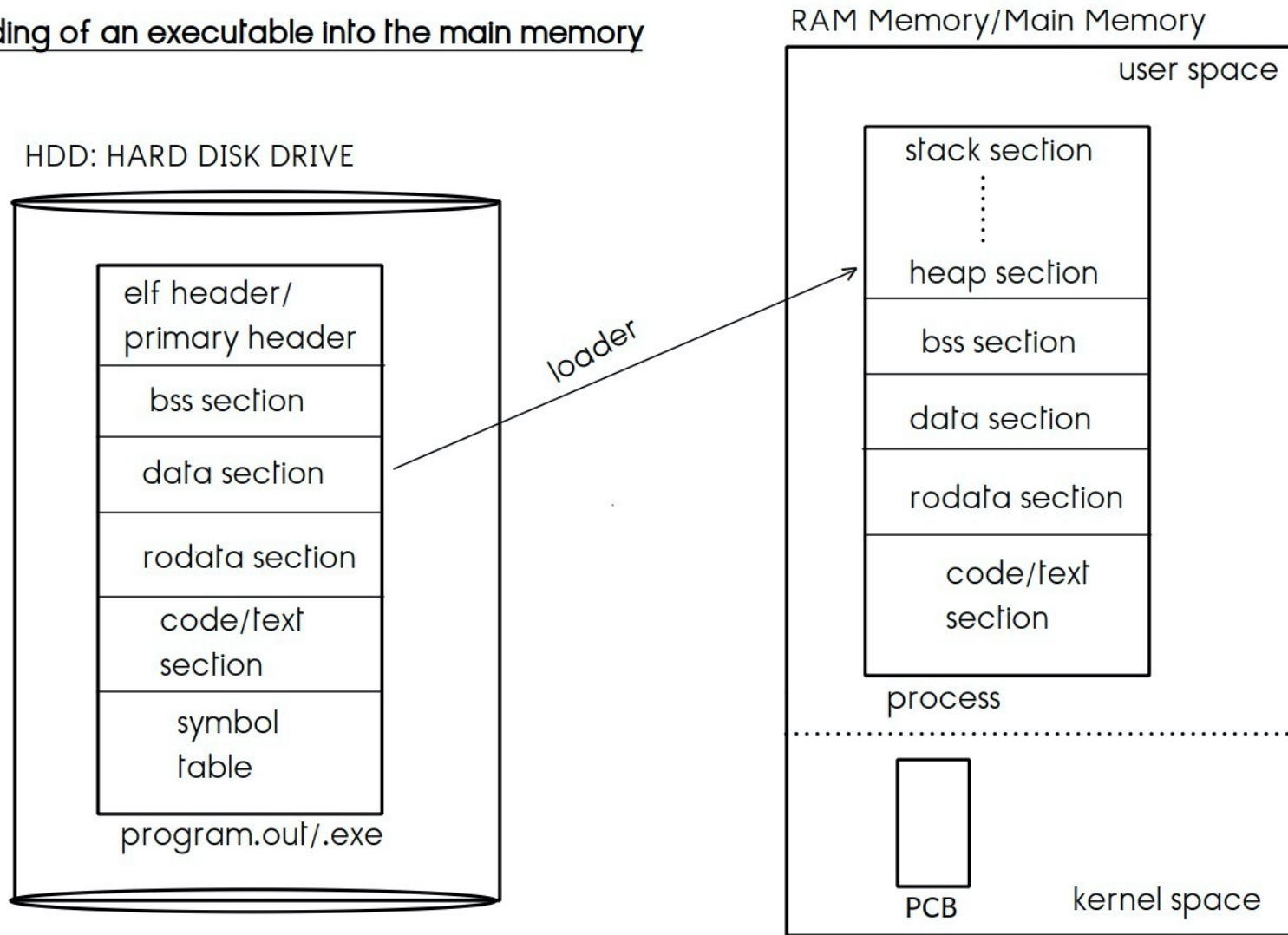| primary header /elf header |
| bss section |
| data section |
| rodata section |
| code/text section |
| symbol table |

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.

Operating System Concepts

Loading of an executable into the main memory

HDD: HARD DISK DRIVE

program.out/.exe

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

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

Process

- Program under execution.

- Process execute in RAM.

- Process control block contains information about the process (required for the execution of 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 as process descriptor (PD), uarea (UNIX), or task_struct (Linux).

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

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

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

- 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 program is also knows as **Process.**

- When a program gets loaded in to memory is also known as **Process.**

- A **Program** is a set of instructions given to the machine to do 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.....
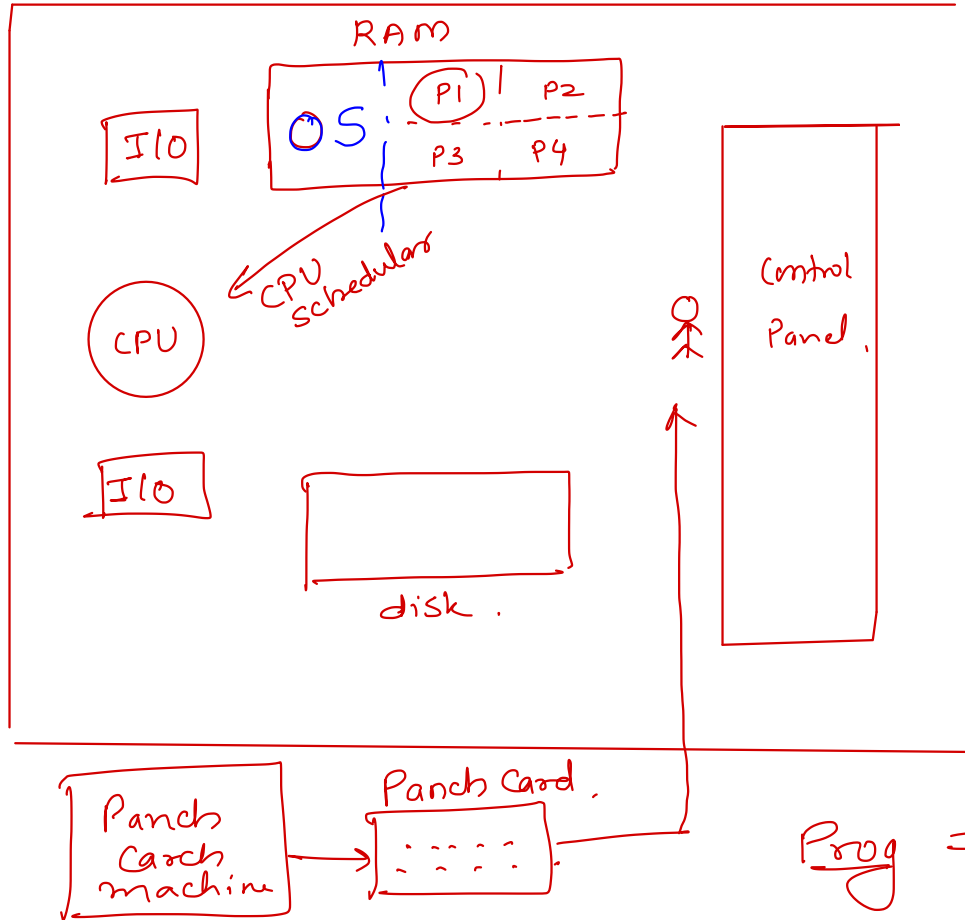
- Windows = User friendly GUI.

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).
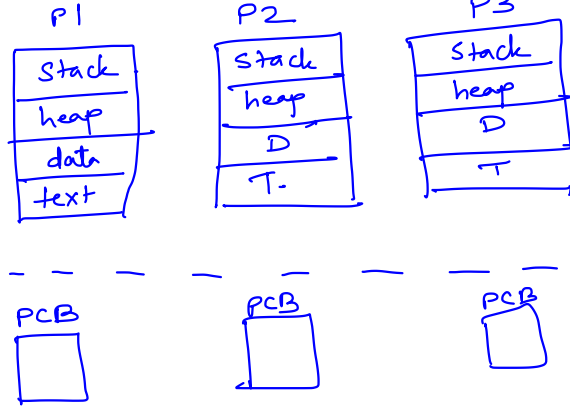
mainframe comp.

RAM

OS : P1 | P2
       P3 | P4

I/O

CPU Schedular

CPU

I/O

disk.

Control Panel.

Punch Card.
: - - - - :

Punch Carch machine

History of OS.

① Resident monitor
② Batch Prog.
③ Multi-programming
   ↳
④ time-sharing/multitasking

multi-processing   multi-thread.

$Prog = \underset{time}{\underset{Brust}{CPU}} + \underset{time}{\underset{Brust}{I/O}}$

Process-based

P1

Stack
heap
data
text

P2

Stack
heap
D
T.

P3

Stack
heap
D
T

PCB

PCB

PCB

Thread Based

Th1

Stack
Stack
Stack
heap
D
T

PCB

TCB

TCB
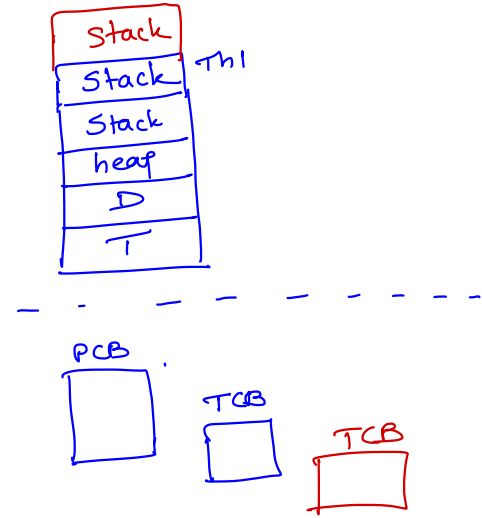
Operating System Concepts

- **History of Operating System**

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