Computer Fundamentals
And
Operating System Concepts

❑Module introduction

➢CCAT point of view:

- Operating Systems: 9 Questions
- Computer Fundamentals + Networking: 10 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 Management

- 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

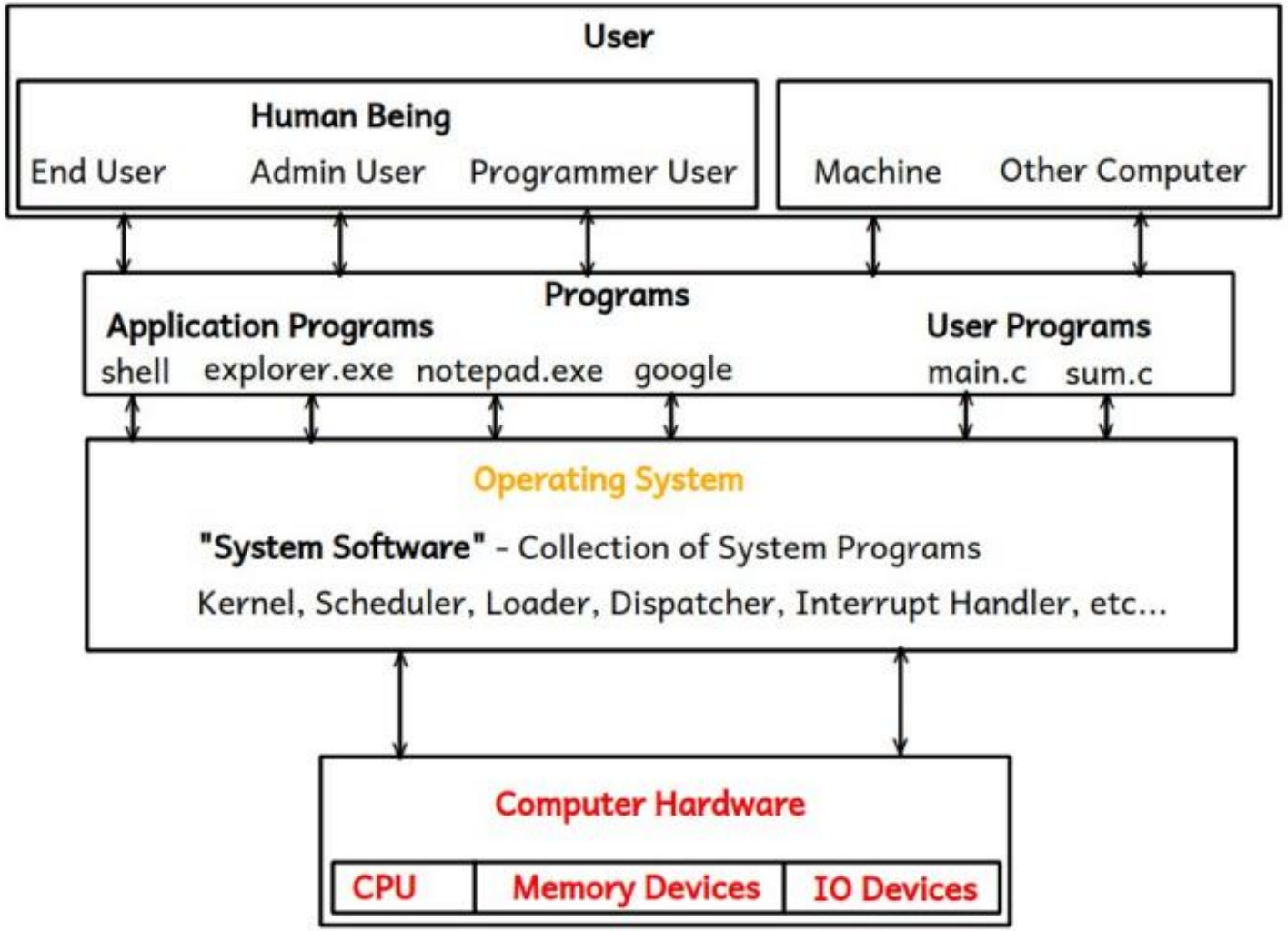
- What is file?
- What is file system & file system structure?
- Disk structure (CF)
- Disk space allocation methods
- Disk scheduling algorithms
- Computer structure (CF)
- Interrupts (CF)
- Direct Memory Access (CF)
- Input-Output (CF)
- System calls

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.

Diagram :OS


User
Human Being
End User
Admin User
Programmer User
Machine
Other Computer
Programs
Application Programs
shell
explorer.exe
notepad.exe
google
User Programs
main.c
sum.c
Operating System
"System Software" - Collection of System Programs
Kernel, Scheduler, Loader, Dispatcher, Interrupt Handler, etc...
Computer Hardware
CPU
Memory Devices
IO Devices

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:

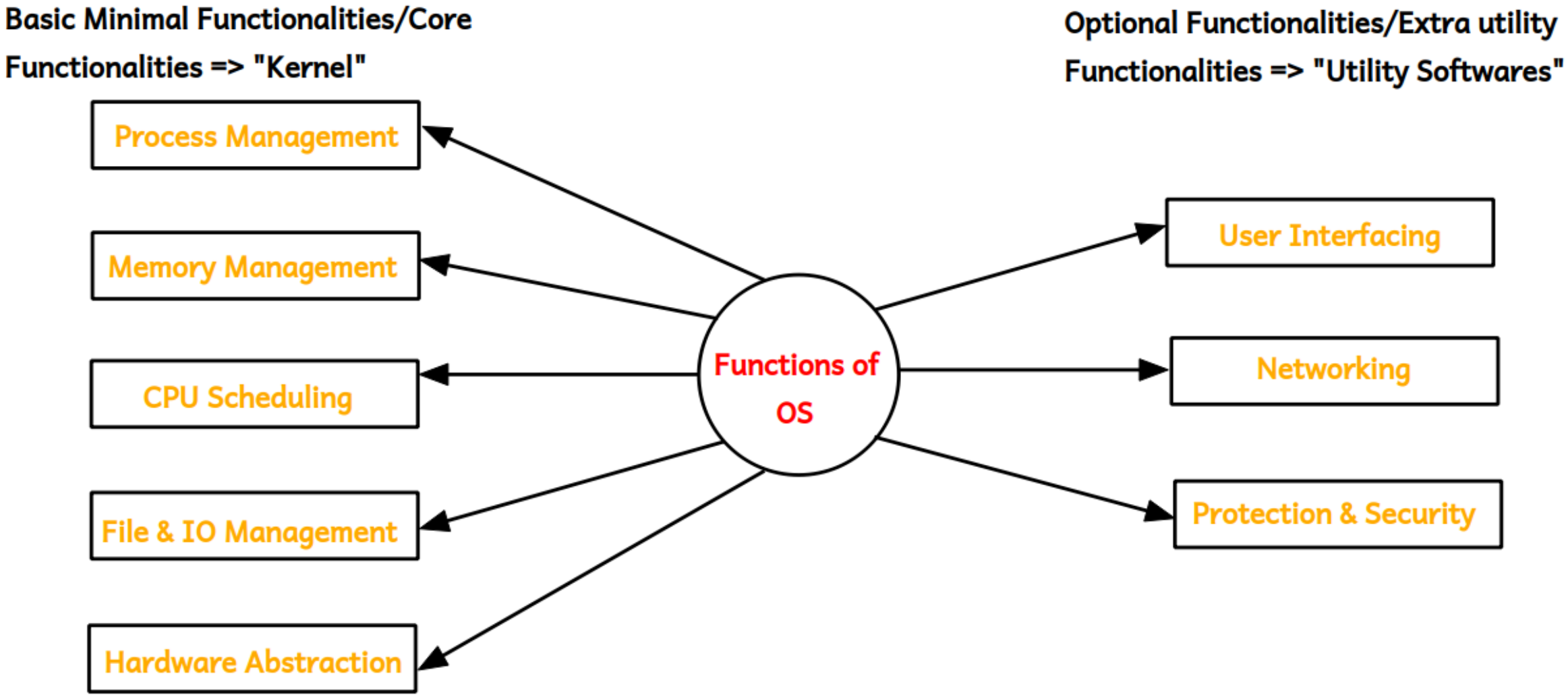
Basic minimal functionalities/Kernel functionalities:

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

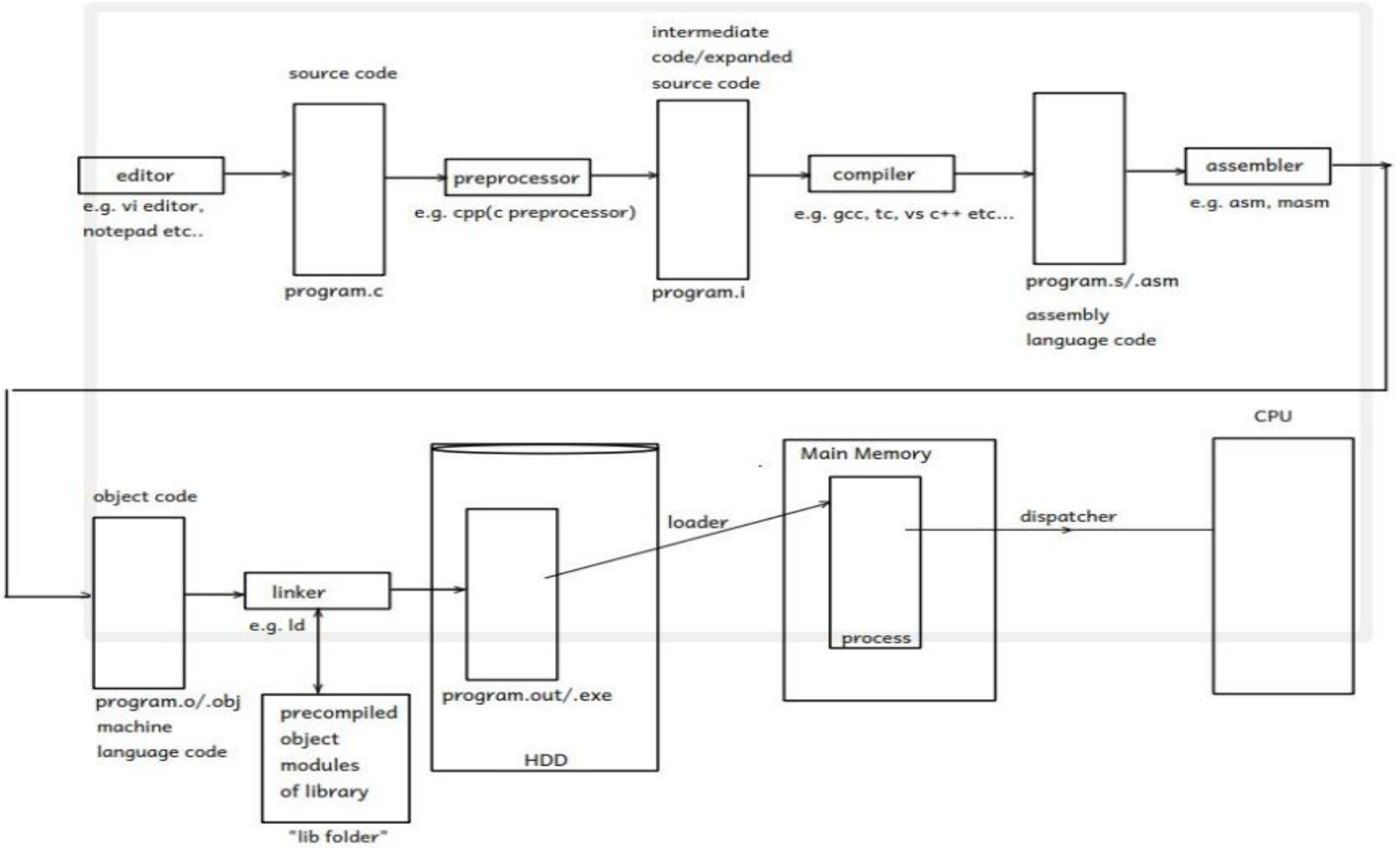
Extra utility functionalities/optional:

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

Operating Systems Concepts



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....

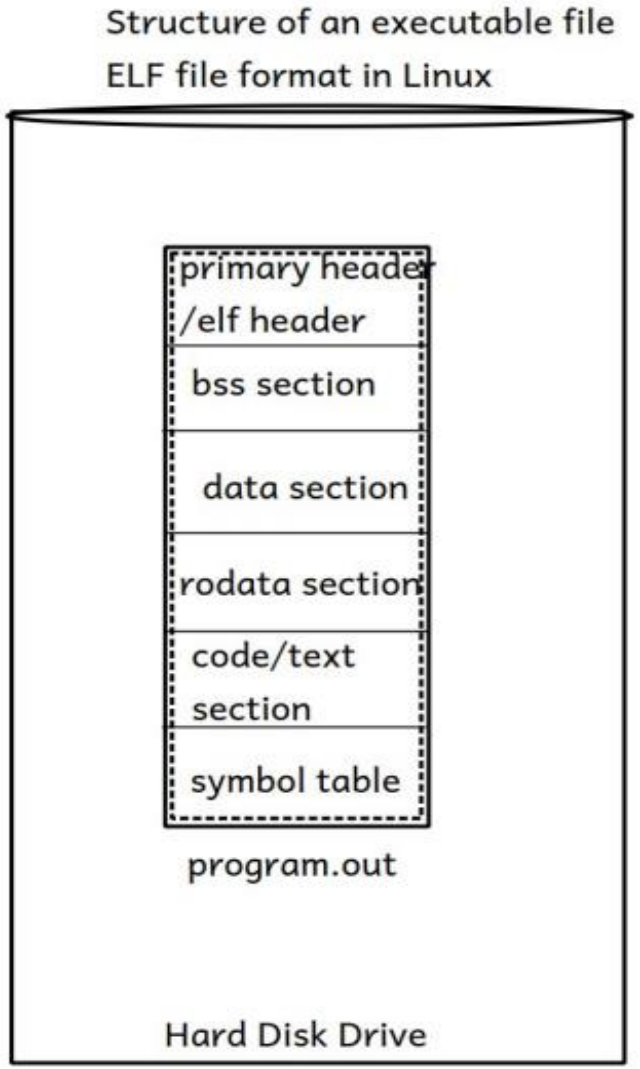
1. **"Editor":** it is an application program **used to write a source code.**
   Source Code – Program written in any programming language.
   e.g. notepad, vi editor, gedit etc...
2. **"Pre-processor":** it is an application program gets executes before compilation and does two jobs
   - it executes 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.



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.

Operating Systems Concepts

- **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)

Operating Systems Concepts

- **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.

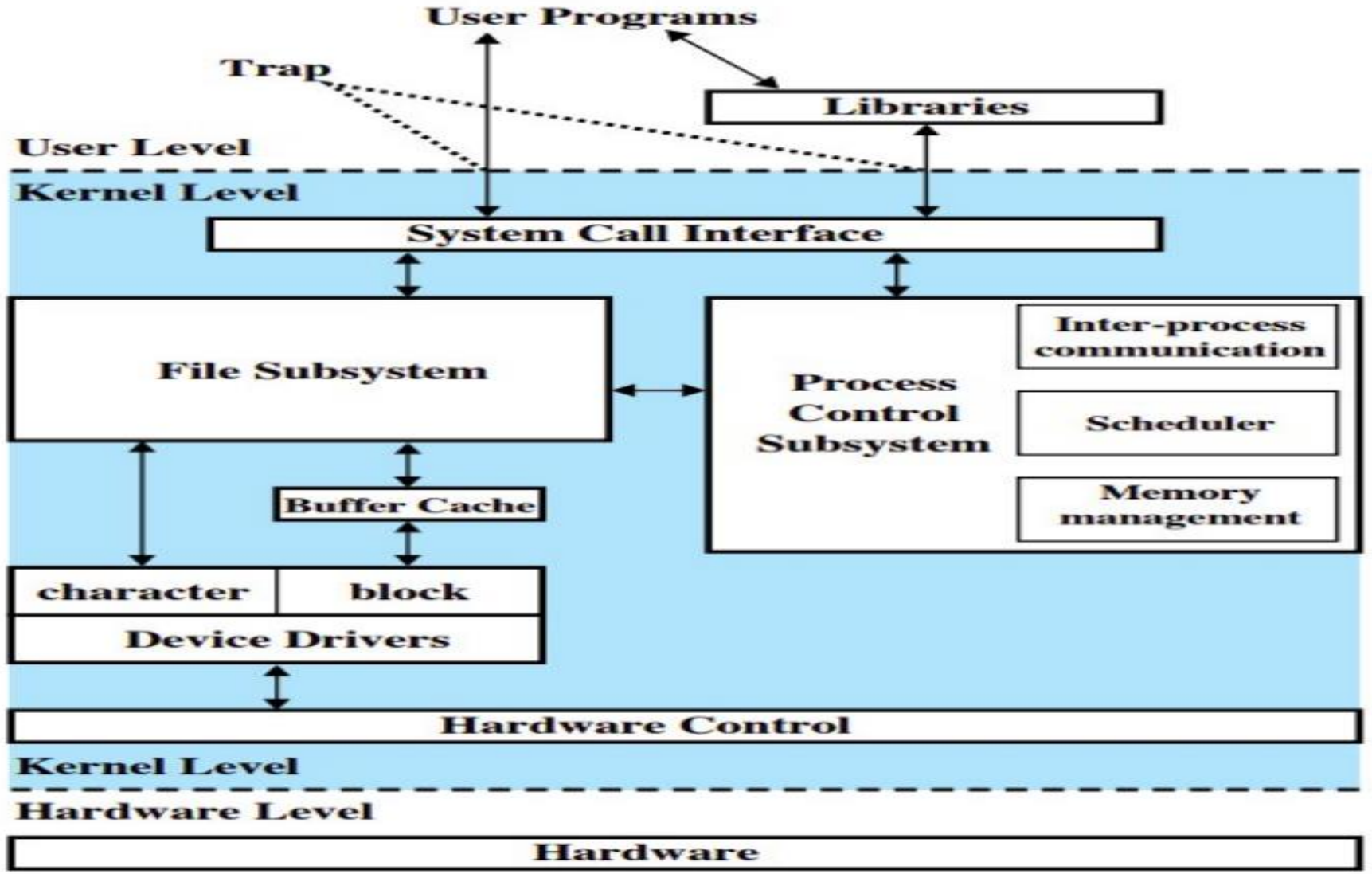
2. System Boot:

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 Systems Concepts



Operating Systems Concepts

- Kernel acts as an interface between programs and hardware.
- Operating System has subsystems like System **Call Interface Block, File Subsystem Block, Process Control Subsystem Block (which contains 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 referred as a process.**
- **File has space & Process has life.**

Operating Systems Concepts

- From **UNIX point of view all devices are considered as a file**
- In UNIX, devices are categorizes into two categories:

  1. Character Devices: Devices from which data gets transferred character by character

  →character special device file

  e.g. keyboard, mouse, printer, monitor etc...

  2. Block Devices: Devices from which data gets transferred 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.

➢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.**

- **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 categories of system calls:

1. Process Control System Calls: e.g. fork(), _exit(), wait() etc...
    1. fork() : To create new processes
    2. _exit() : To exit processes
    3. wait() : To hold/wait processes
2. File Operations System Calls: e.g. open(), read(), write(), close() etc...
    1. open() :
    2. read() :
    3. write() :
    4. close() :

3. Device Control System Calls: e.g. open(), read(), write(), ioctl() etc...
4. Accounting Information System Calls: e.g. getpid(), getppid(), stat() etc...
    1. getpid() : To get Process ID
    2. getppid() : To get Parent Process ID
    3. stat() : To get File Information
5. Protection & Security System Calls: e.g. chmod(), chown() etc..
    1. chmod() : Change user mode / permission
    2. chown() : get file owner info
6. Inter Process Communication System Calls: e.g. pipe(), signal(), msgget() etc...

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/right 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).

Thank you!

Kiran Jaybhave

email – kiran.jaybhave@sunbeaminfo.com