Concepts of Operating Systems And Linux Operating Systems

Trainer: Kiran Jaybhave

➢Contents

- **Operating System Introduction**
- **Linux Introduction**
- **Linux Architecture and Structure**
- **Basic commands / Advanced commands**
- **vi / vim editor**
- **Linux shell programming (shell scripts)**
- **Process Management**
- **Memory Management**

➢Evaluations

- **Theory** - 15 marks (CCEE)
- **Lab** - 20* marks (Linux commands and shell script)
- **Internal** - 10 marks

Learning OS

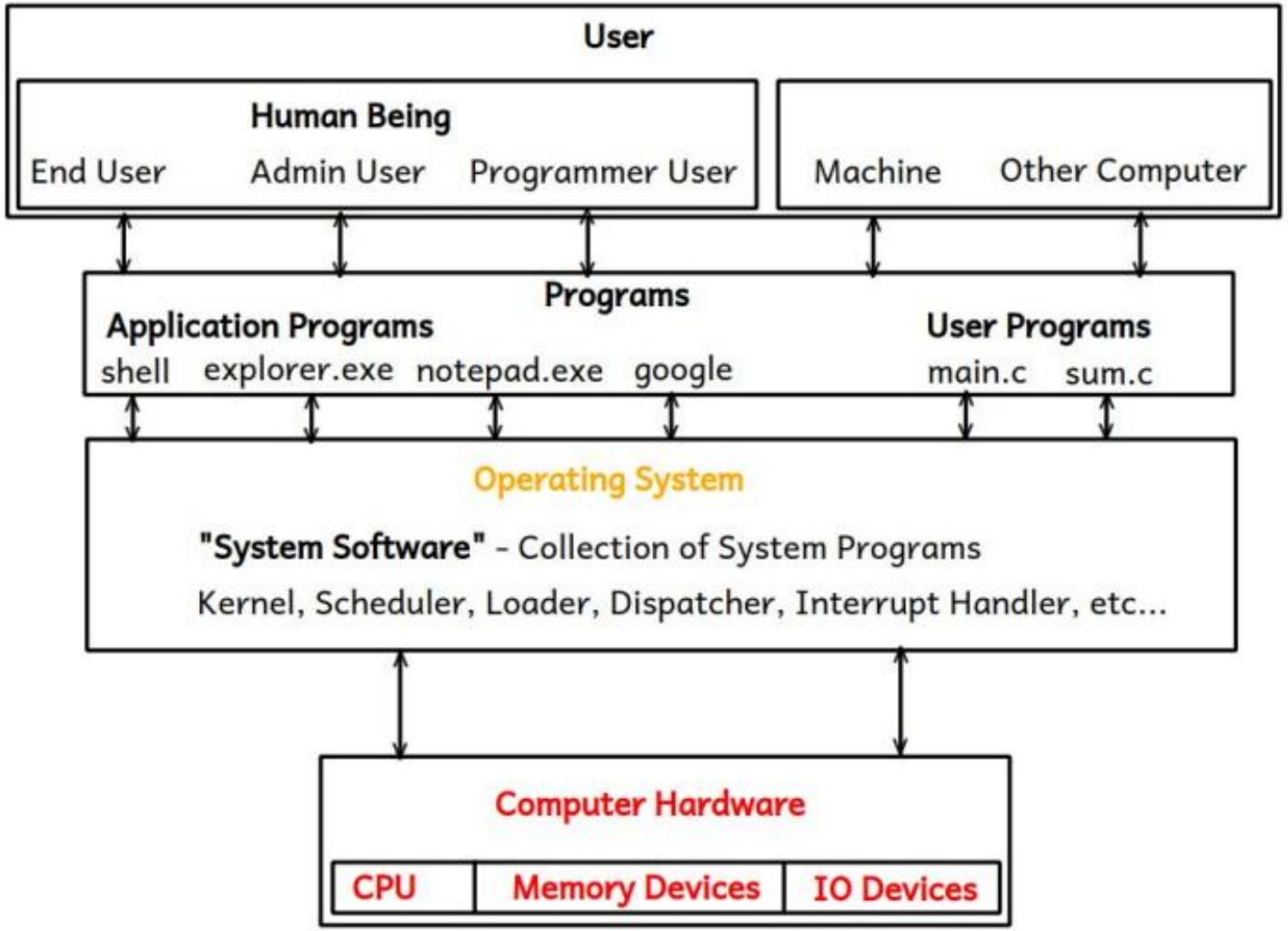
- **Step 1: End user**
  - Linux commands
- **Step 2: Administrator**
  - Install OS (Linux)
  - Configuration - Users, Networking, Storage, ...
  - Shell scripts
- **Step 3: Programmer**
  - Linux System call programming
- **Step 4: Designer/Internals**
  - UNIX & Linux internals

Q. Why there is a need of an OS?

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

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

What is an Operating System?

- OS also acts as an **interface between programs and hardware**.
- OS is a **system softwar**e (i.e. collection of system programs) .
- 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.**
- 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.**
- OS manages limited available resources among all running programs, hence it is also called as a **resource manager**.

Operating Systems Concepts

- **From End User:** OS is a software (i.e. collection of programs) comes either in Bootable CD/DVD, has following main components
  - **Bootable CD/DVD** = Core OS + Applications + Utilities
  - **Core OS** = Kernel -- Performs all basic functions of OS.

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:

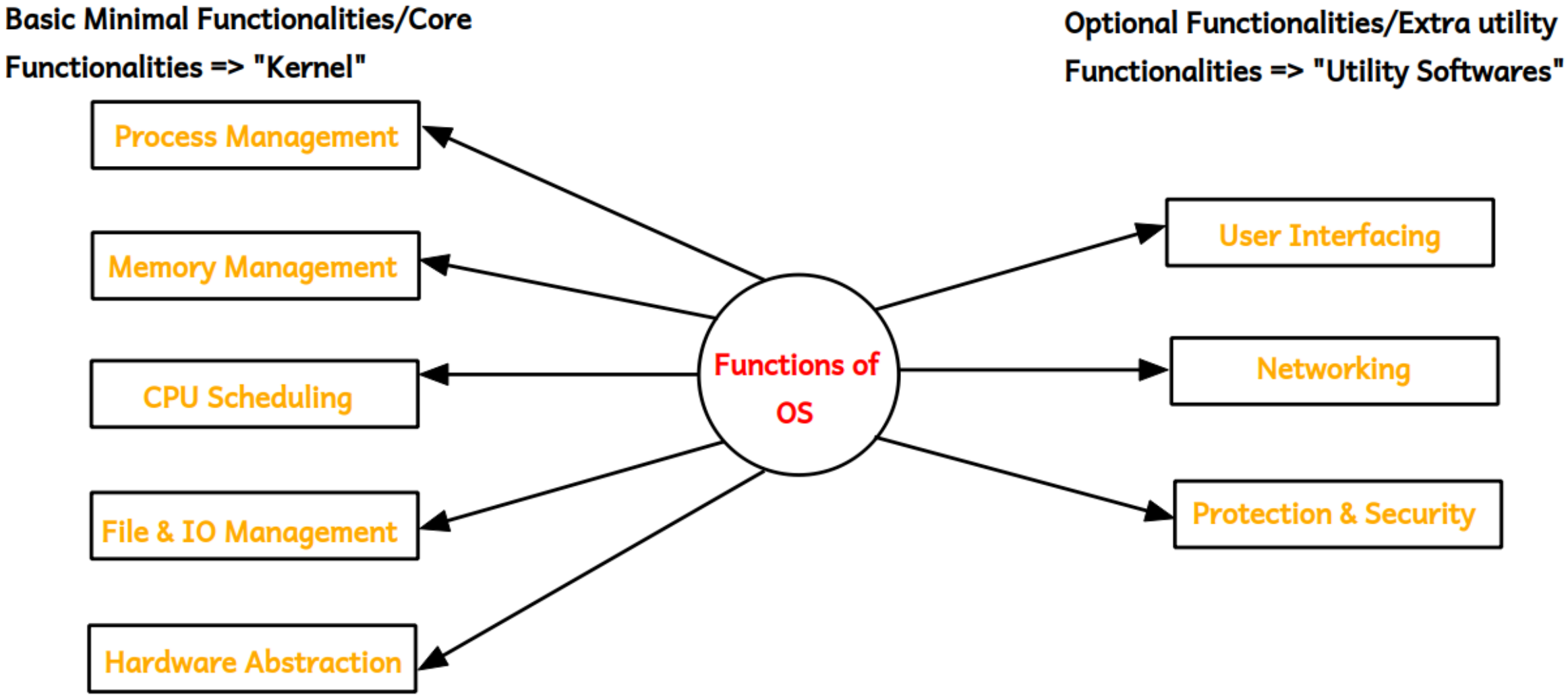
Basic minimal functionalities/Kernel functionalities:

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

Extra utility functionalities/optional:

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

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.

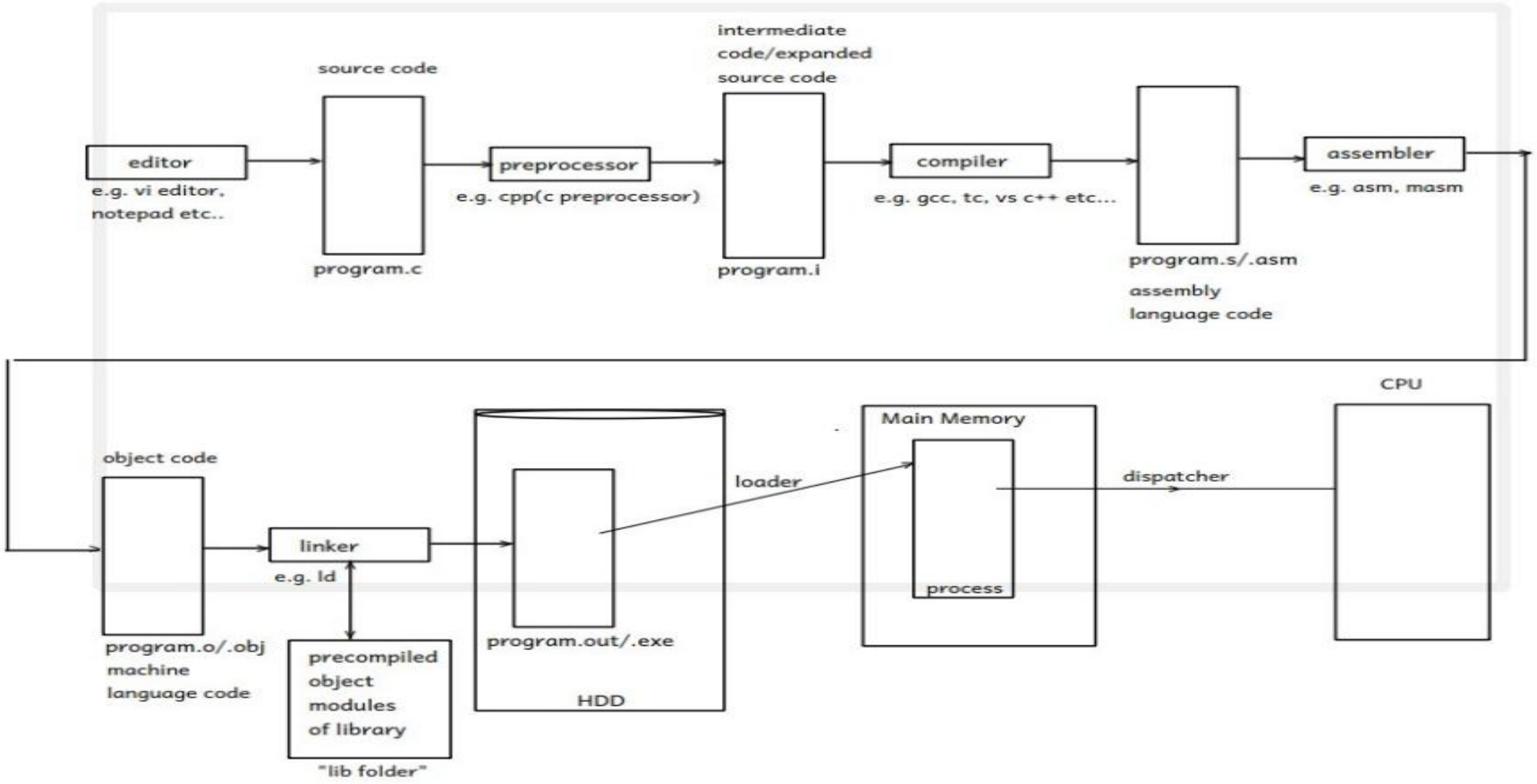
   e.g. masm, tasm etc...

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.

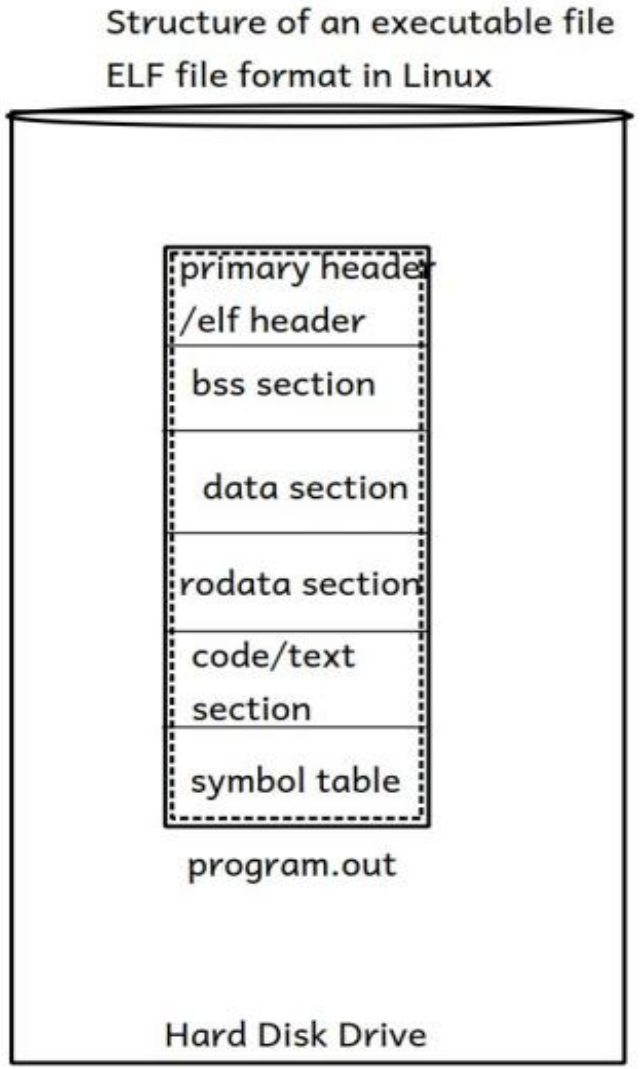
- **Loader :** It is a system program (i.e. inbuilt program of an OS) which loads an executable file from HDD into the main memory.
- **Dispatcher:** It is a system program (i.e. inbuilt program of an OS) which loads data & instructions of a program which is in the main memory onto the CPU (i.e. into the CPU registers).

Operating Systems Concepts



- **File format of an executable file is different in different OS .**
  - In **Linux** file format of an executable file is : **ELF**(Executable & Linkable File Format) .
  - In **Windows** file format of an executable file is : **PE**(Portable Executable)

- **What is a file format**?
  - File format of an executable file is a specific way to keep data & instructions in it in an organized manner.
  - Way to keep data & instructions inside an executable file is vary from OS to OS



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.

➢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 communication, process coordination, and terminate the process.**

Q. What is a Program?

❑User view:

- Program is a finite set of instructions written in any programming language given to the machine to do specific task.

❑System view:

- Program is an executable file in HDD which divided logically into sections like exe header, bss section, data section, rodata section, code section, symbol table.

Q. What is a Process?

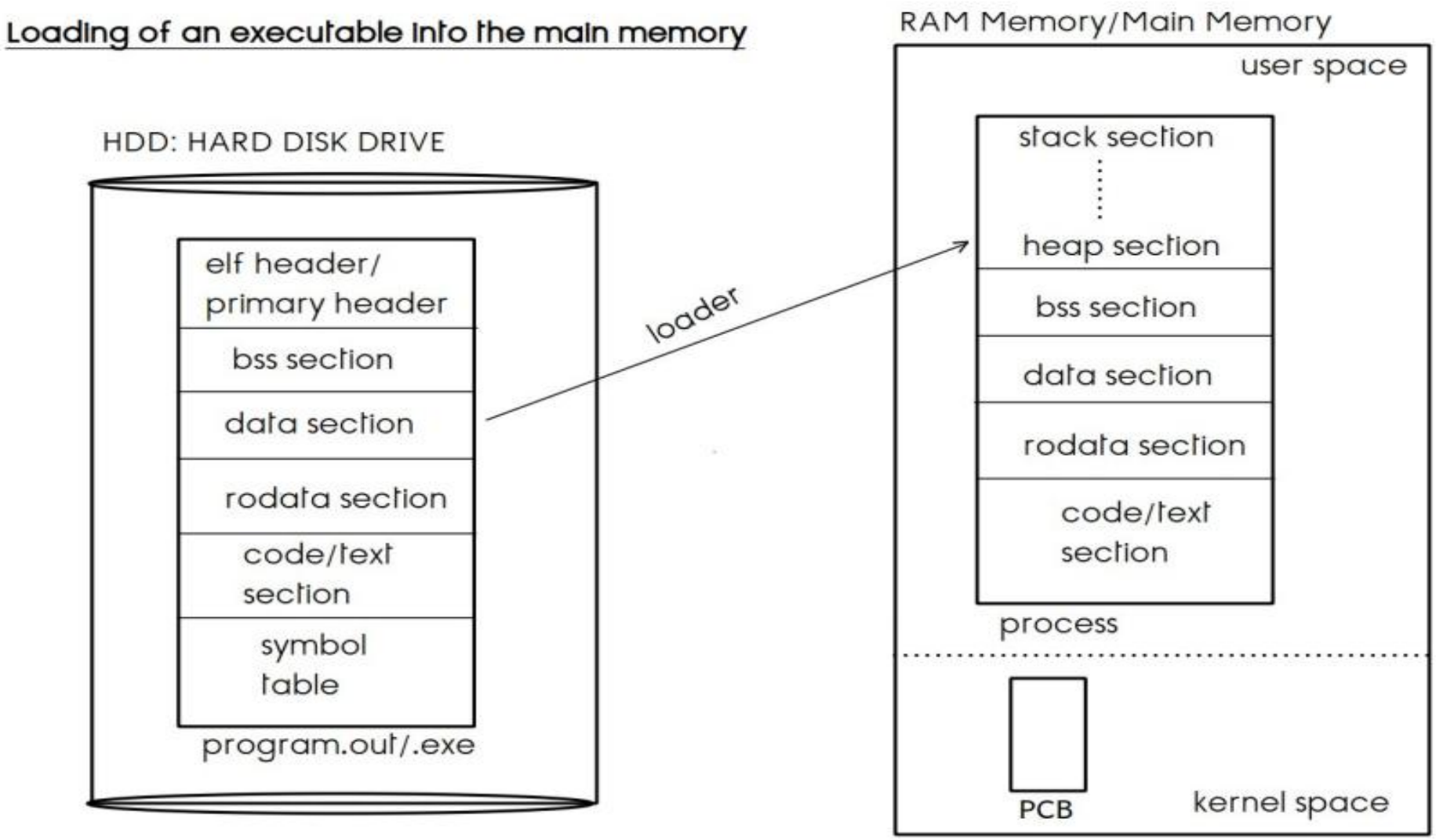
❑ **User view:**

- 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 referred as **a process.**
- Running instance of a program is referred as **a process**.

❑ **System view:**

- Process is a program loaded into the main memory which has got PCB into the main memory inside kernel space and program itself into the main memory inside user space 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


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

- As a kernel, core program of an a OS runs continuously into the main memory, part of the main memory which is occupied by the kernel **referred as kernel space** and whichever part is left **is referred 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 referred as a **PCB.**
- **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**.

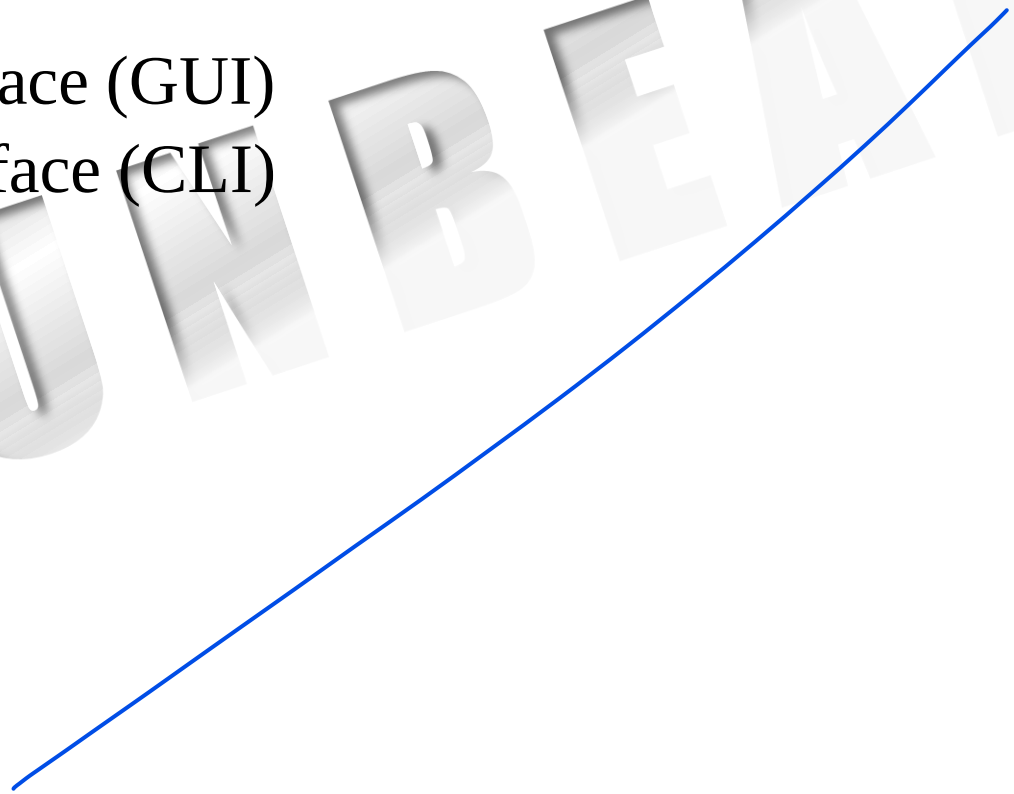
❖**Process control block contains information about the process (required for the execution of process):**

- Process id
- Exit status
  - 0 - Indicate successful execution
  - Non-zero - Indicate failure
- 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 (Values of CPU registers)
- Kernel stack
- PCB is also called as **process descriptor (PD**), **uarea (UNIX),** or **task_struct (Linux**).
- In Linux size of **task_struct** is approx 4KB

USER INTERFACE

❖**UI of OS is a program (Shell) that interface between End user and Kernel.**

- Shell : Command interpreter
- End user ➡ Command ➡ Shell ➡ Kernel
- **User interfacing (Shell)**
  - Graphical User Interface (GUI)
  - Command Line Interface (CLI)



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 Through CUI program referred **as Shell/Terminal**.
- Shell (also referred terminal) is an application program which accepts command from user in a text format interpret it and pass it to the kernel (to an OS) for execution.
- e.g. In **Windows** name of the program which provide CUI => **cmd.exe** (command prompt),**powershell.exe**
- e.g. In **Linux** name of an application program which provides CUI => **bsh(sh), bash , csh , ksh , zsh**.
- In MSDOS name of the program which provides CUI => command.com (Microsoft Disk Operating System).

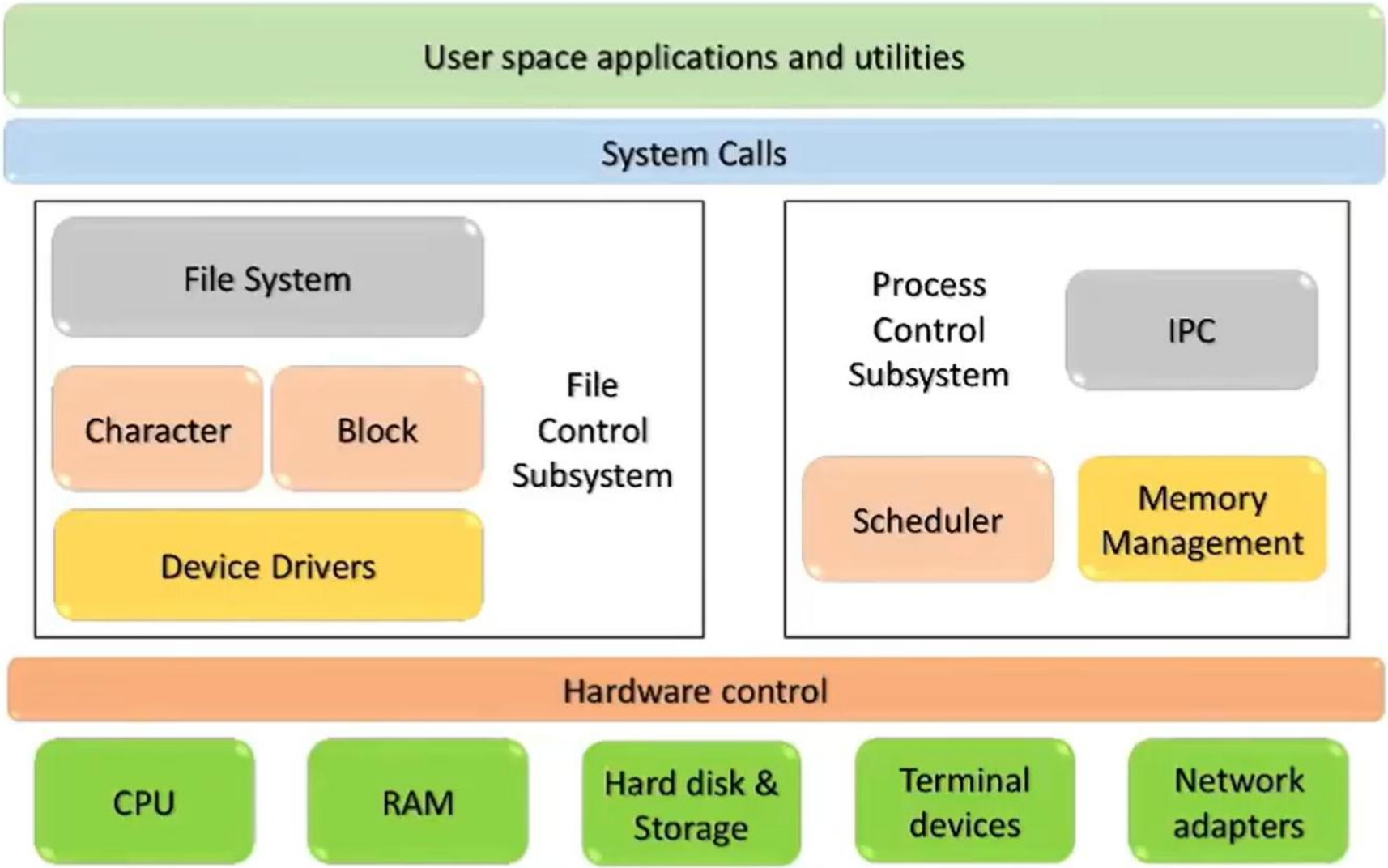
2. GUI : Graphical User Interface

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

UNIX Operating Systems

- **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.**
- 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.**

UNIX Kernel Architecture

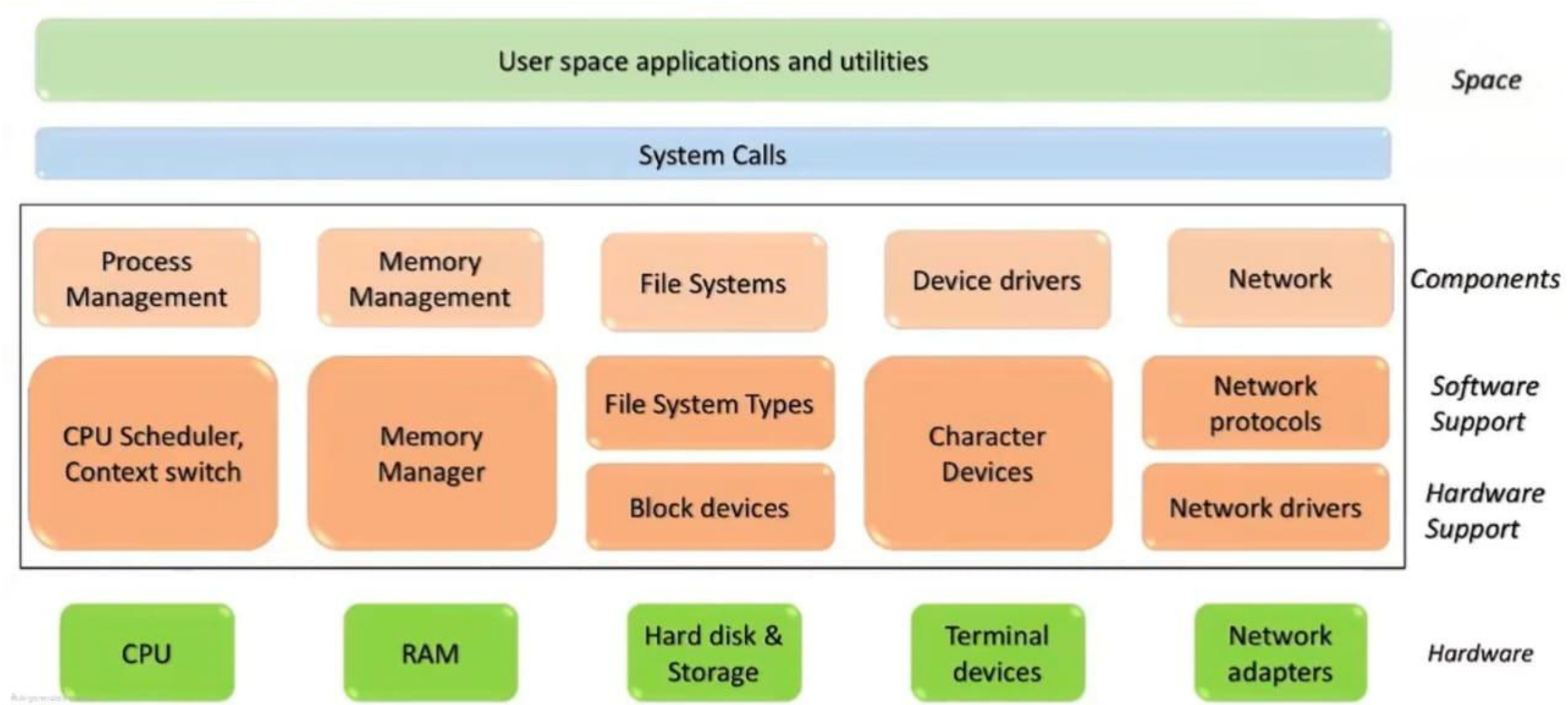


Introduction to Linux-

- **Linux** is a family of open-source, **UNIX-like operating systems** based on the **Linux kernel**.
- The **Linux kernel (vmlinuz)** was first released on **September 17, 1991**, by **Linus Torvalds**, a Finnish-American software engineer. He is the creator and historically the main developer of the **Linux kernel**, which powers various Linux distributions and operating systems like **Android**.

- There are two main types of **Linux distributions**:
- **Commercially-backed distributions**:
  - **Fedora** (sponsored by Red Hat)
  - **openSUSE** (supported by SUSE)
  - **Ubuntu** (developed by Canonical Ltd.)
- **Community-driven distributions**:
  - **Debian** – One of the oldest and most stable distributions.
  - **Slackware** – A minimalist and UNIX-like Linux distro.
  - **Gentoo** – A highly customizable source-based distribution.
  - **Arch Linux** – A lightweight, rolling-release distribution aimed at advanced users.

Linux Kernel Architecture



Linux File system Structure:-

- **While an installation of Linux system we need to create two partitions:**
  - **swap partition (will be use by Linux as a swap area)**
  - **"/" (root) partition also called as data partition**
- Linux follows FHS i.e. **File System Hierarchy Standards**
- Data can be store into the "**data partition**" in an organized manner, so linux filesystem structure is having **hierarchical structure**, which is like an **inverted tree.**
- When any user logged in into the system or **opens a terminal, system takes the user by default into the user's home dir.**
- **Linux file system structure starts with root dir ("/"),** and all the data can be stored or kept inside it **under sub-dir's in an organized manner.**
- **root** dir i.e. "/" dir contains sub-dir's like: "/**boot**", "/**bin**", "/**sbin**", "/**etc**","/**lib**", "/**usr**", "/**home**", "/**root**" "/**mnt**", "/**dev**" etc....

Linux File system Structure:

- **“/boot” :**
  - It contains information about booting the system
  - It contains Linux kernel by the name **vmlinuz.**
  - **grub** - boot loader
  - **config** - kernel configuration
  - **initrd/initramfs** - initail root file system
- **"/bin” :** contains user commands in a binary format, e.g. commands like ls, cat, cp, mv ,etc...
- **"/sbin” :** contains admin/system commands in a binary format, e.g. lscpu, adduser, deluser, mkfs etc...
- **"/home” :** contains home dir's of all users for multi-user system.
  - For any user on its creation of an account by default sub directory by the name of the user got created by the operating system, in which that user can store data, user can have read, write as well execute perms in that directory .
- **"/root“ :** root is a home directory of root user.
- **"/etc“ :** contains host specific system configuration files.
  - In computing configuration files (or config files) are files used to configure the parameters and initial settings for some computer programs.

Linux File system Structure

7. **/dev** – Contains device files for all hardware devices (e.g. char and block devices).
8. **/lib** – Contains essential shared libraries (.so) and kernel modules (.ko) required for system boot and basic programs.
9. **/mnt** – Temporary mount point for mounting other file systems manually.
10. **/opt** – Contains optional or third-party software packages.
11. **/proc** – It providing information about system processes and kernel.
12. **/sys** – It used for device management and kernel settings.
13. **/tmp** – Temporary files; contents are usually cleared on reboot.
14. **/usr** – Contains user-installed software and shared read-only data (includes binaries, libraries, docs, etc.).

Path

- It is an **unique location of any file in a file system structure that can be represented in a string format** .
- It is sequence of chars format separated filenames /sub-dir names by **delimiter char or delimiter.**
- There **are three delimiter chars**:
  1. **slash (/) -- In Linux**
  2. **backslash (\ ) -- In Windows**
  3. **colon (:) -- In Linux as well as in Windows**
- **In Linux there are two ways by which we can mention path of any file/dir:**
  1. **Absolute path:** It is a full path of any file/dir which starts with root dir ("/").
     - Path which starts with “/” is called as absolute path.
     - E.g. /home/sunbeam/MyData/Demos/demo01.sh
  2. **Relative path:** Path with respect to current directory is called as relative path
     - E.g. MyData/Assignments/assign02.pdf .

File Management

Q. What is file?

- **User view:**
  - File is a named collection of logically related information/data.
  - File is a container which contains logically related information/data.
  - File is a collection of characters/records/lines.
  - File is a basic storage unit
- **System view:**
  - File is a **stream of bits/bytes.**
    - **File = data + metadata**
      - data = actual file contents
      - metadata = information about the file.

- Data of the file is exists inside the file whereas information about the file gets stored inside one structure referred as **FCB (File Control Block).**
- In UNIX environment FCB is also called as an **iNode**.
- **FCB / iNode contains information about the file like:**
  1. types of files
  2. size: number of bytes
  3. links: number of hard links
  4. mode (permissions): (u) rwx, (g) rwx, (o) rwx
  5. user & group
  6. time-stamps: modification, creation, access.
  7. info about data blocks
- terminal> ls -l
  - type, mode, links, user, group, size, timestamp, name.
- terminal> stat filepath

❖**In Linux / UNIX based OS's there are 7 types of files:**

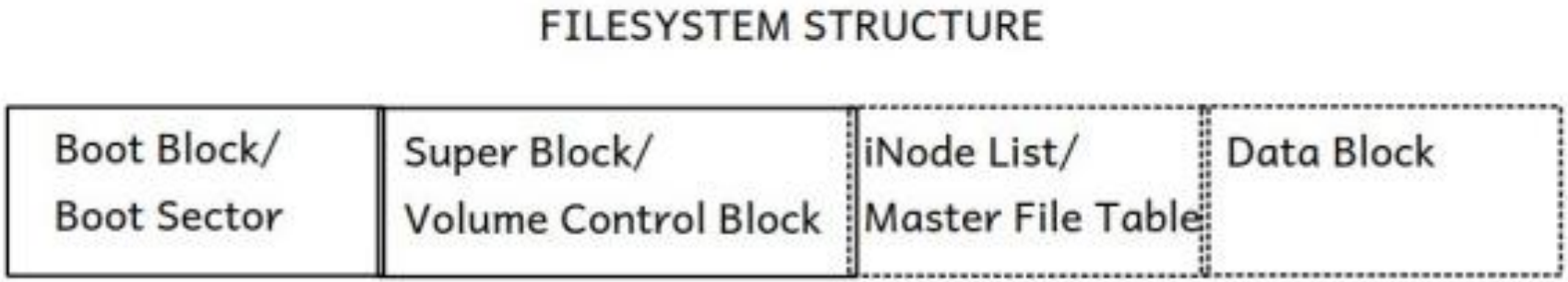
1. **regular file (-)** : all text files, source files, image files, audio files, video files, executable files etc..
2. **directory file (d) :** it is a special type of file in UNIX which is a like a container whose contents are names of files and sub directory in it.
3. **character special device file (c) :**
4. **block special devices file (b) :**
5. **linkable file (l)** : special type of file in Linux which is same like short cut files in Windows, which contains link of an original file and by using this linkable file contents of original file also can be accessed.
   - To access deeply located file more frequently we should create its linkable file.
6. **pipe file (p) :**
7. **socket file (s) :**

- **Pipe file and socket file use in IPC.**

Operating Systems Concepts

- Per file one iNode/FCB gets created by the system and **i.e**. **hence no. of iNodes = no. of files onto the disk.**
- **data + metadata** of all the files are kept onto the disk, as disk may contents thousands of files, so thousands of iNodes and millions of bytes of data gets stored onto the disk, and hence all this **data + metadata** of all files need be keep onto the disk in an organized manner so that it can be accessed efficiently.
- **File system :** file system is a way to store data onto the disk in an organized manner so that it can be accessed efficiently and conveniently.
  - e.g. Each OS has its own file system like,
  - UNIX: UFS(UNIX File system),
  - Linux: Extended File system ext2, ext3, ext4,
  - Windows: FAT, NTFS etc...,
  - MAC OSX: ReiserFS, XFS, HFS(Hierarchical File system) etc...

- **Files system Structure:** File system divides disk/partition logically into sectors/blocks, like boot sector/boot block, volume control block/super block, master file table/iNode list block and Data Block.



1. Boot Block: It contains information about booting the system like bootstrap program, bootloader etc...
2. Super Block: It contains information about remaining sections, like total no. of data blocks, no. of free data blocks, no. of allocated data blocks etc....
3. iNode List: It contains linked list of iNode's of all files exists on a disk.
4. Data Block: It contains actual data.

- **File System logically divide partition into 4 sections.**
  - **Boot block/Boot sector**
    - Contains programs/info required for booting of OS
    - Typically contains bootstrap program and bootloader program
  - **Super block/Volume control block**
    - Contains information of whole partition.
    - Capacity, Label.
    - terminal> df -h
      - Total number of data blocks/inodes.
      - Number of used/free data blocks/inodes.
      - Information of free data blocks/inodes.
  - **Inode List/Master file table**
    - Inodes (FCB) for each file
  - **Data blocks**
    - Stores data of the file.
    - Each file have zero or more data blocks.
    - Size of data blocks can be configured while creating file system
    - File system is created by the format utility while formatting the partition.

Creating File Systems in Windows and Linux

- **Windows:**
  - format.exe - : utility used to format a partition.
- **Linux:**
  - mkfs - : make file system command.
    - **Example:**
    - $ sudo mkfs -t ext3 /dev/sdb1
    - $ sudo mkfs -t vfat /dev/sdb1
      - -t <type> : specifies the file system type.
      - Common types : ext3, ext4, vfat, ntfs, xfs, etc.
      - Example partition name : /dev/sdb1

Link

- **Links are shortcuts to access deeply located files quickly.**
- **Hard Link**
  - A **hard link** is an **additional name** for an existing file. It points directly to the **same inode** (data on disk), so multiple filenames can reference the exact same data.
  - This way the hard link gets all the attributes of the original file and points to the same data block as the original file.
  - **Terminal : $ln [source_file] [link_name]**

  - **A new directory entry is created, which has a new name and same inode number.**
  - **No new file (inode and data blocks) is created.**
  - **Link count in the inode of the file is incremented.**
  - **If directory entry of target file is deleted (rm command), file can be still accessed by link directory entry.**
  - **Cannot create hard link for directories, because it may lead to infinite recursion (while traversing directories recursively e.g. ls -R)**

Link File

- **Symbolic Link**
  - **A symbolic link, also known as a symlink or soft link, is a special type of file that points to another file or directory.**
  - **Terminal : $ln -s [target_file_or_dir] [symlink_name]**
- **symlink()**
  - **syscall A new link file is created (new inode and new data block is allocated), which contains info about the target file (absolute or relative path).**
  - **Link count is not incremented.**
  - **If target file is deleted, the link becomes useless.**
  - **Can create symlinks for directories as well as file also.**

Thank you!

Kiran Jaybhave

email – kiran.jaybhave@sunbeaminfo.com