Operating System Concepts

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

Trainer: Dr.Akshita Chanchlani
Email: akshita.chanchlani@sunbeaminfo.com

* **Introduction:**
  - Why there is need of an OS?
  - What is an OS?
  - Functions of an OS
  - System Calls

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

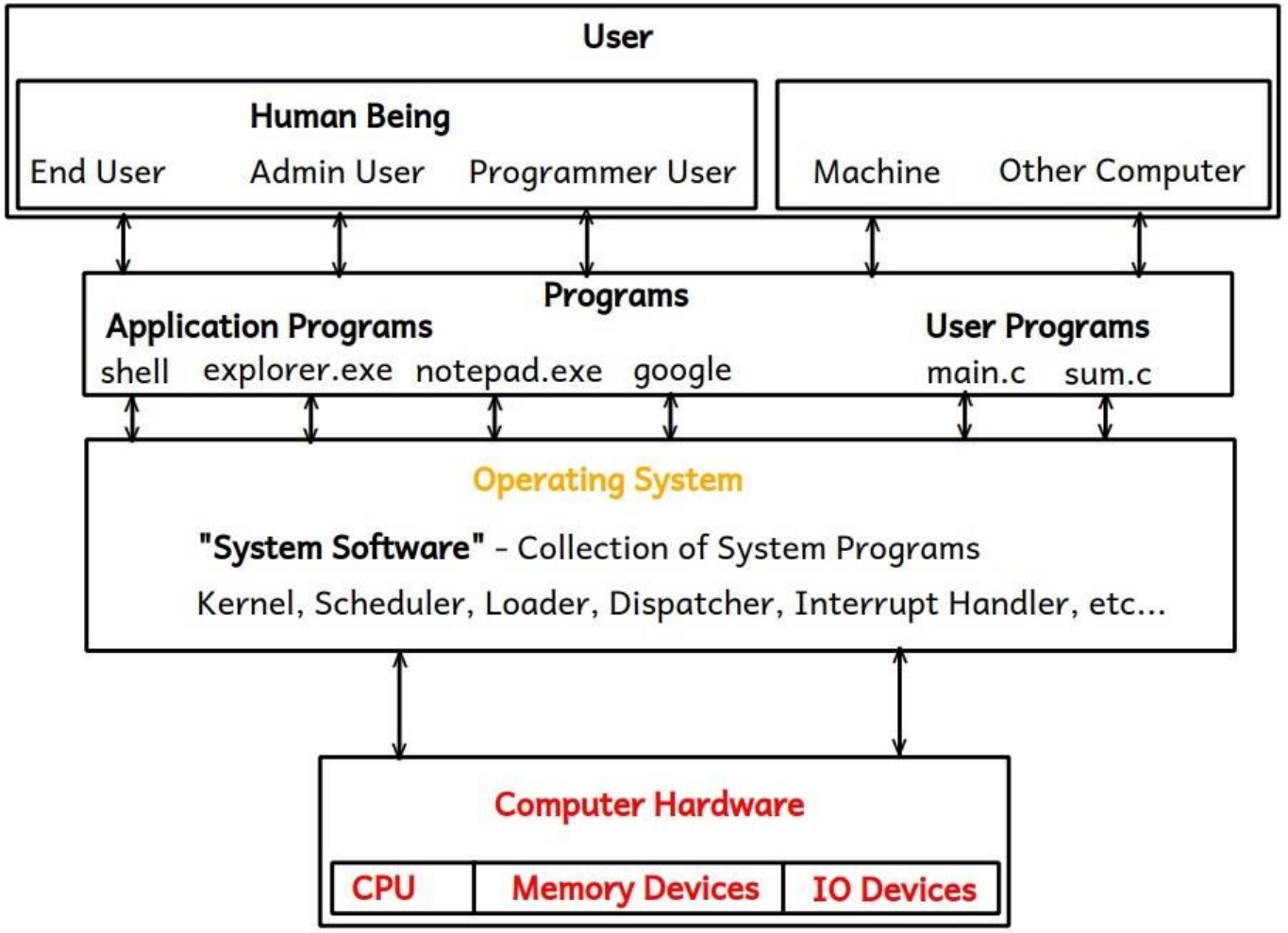
* **Memory Management**
  - Swapping
  - Memory Allocation Methods
  - Internal Fragmentation & External Fragmentation
  - Segmentation
  - Paging
  - Virtual Memory Management

* **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



Types of Programs

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...

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:

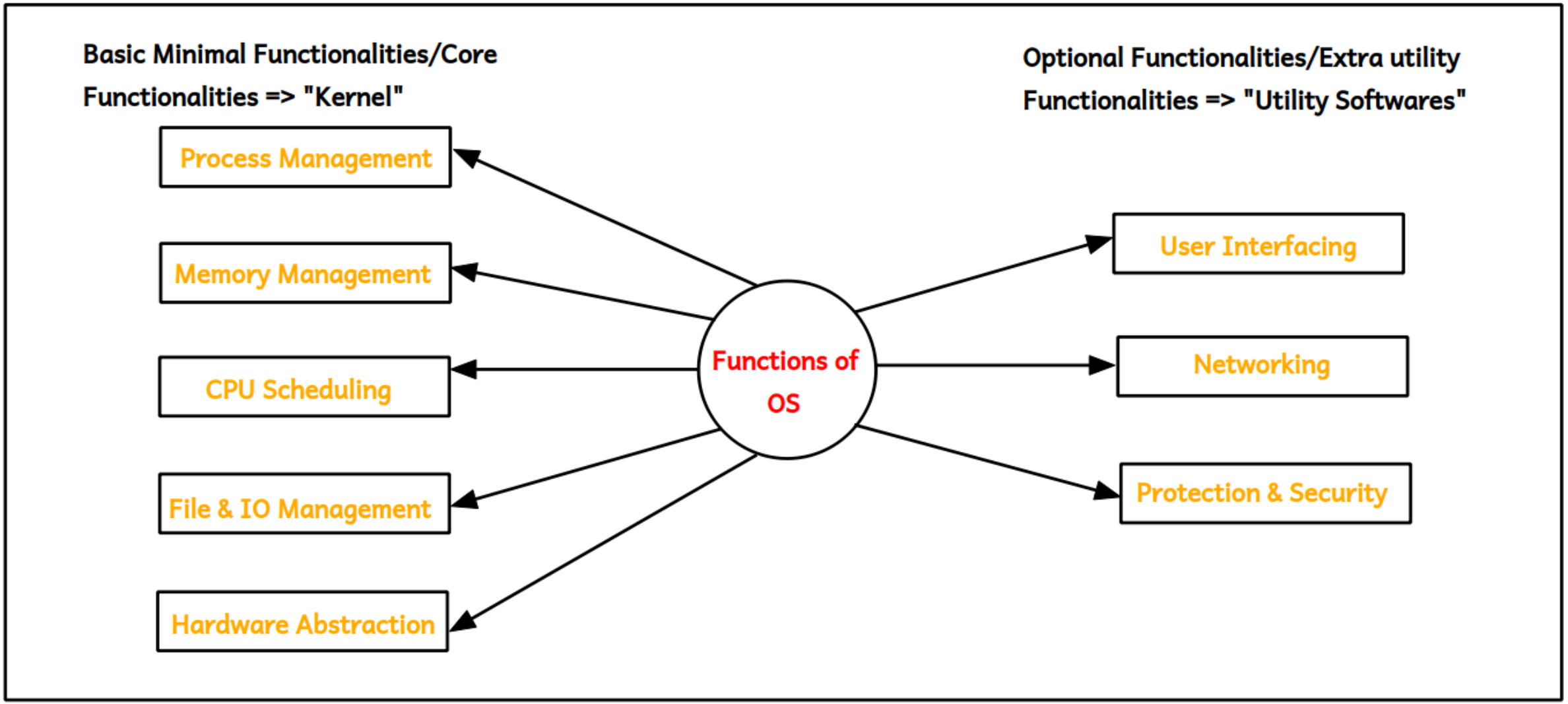
1. Kernel: It is a core program/part of an OS which runs continuosly into the main memory does basic minimal functionalities of it.

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 Operating System



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

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

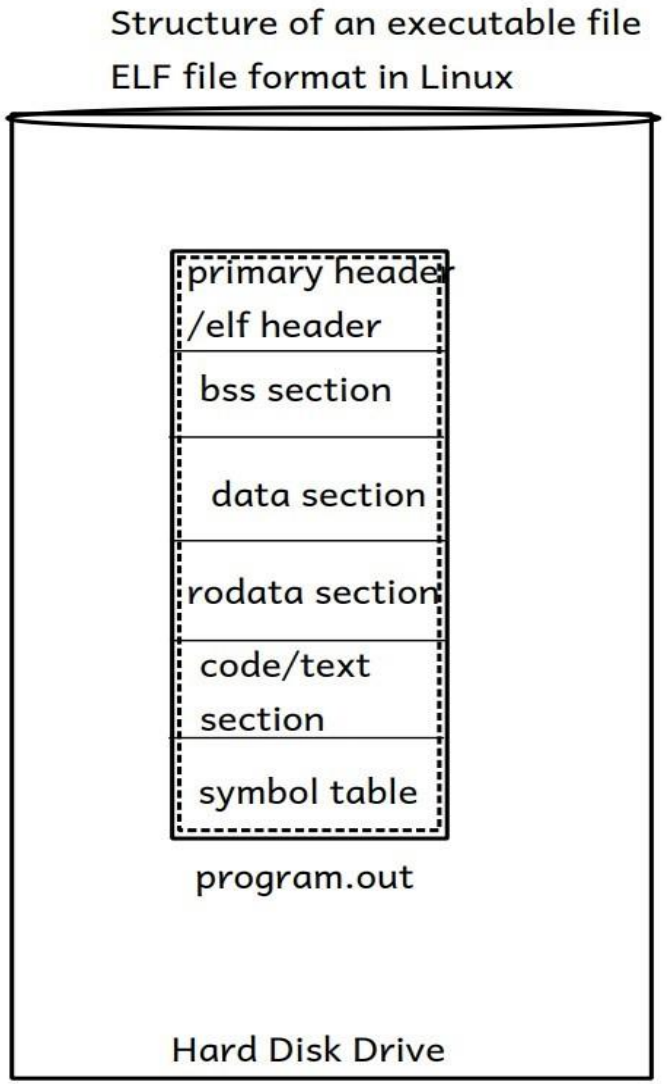
- by using this kind of interface user can interacts with an OS by means entering commands onto the terminal/command line in a text format.

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).



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.

File Format

- 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.
- in Linux file format of an executable file is ELF:
- 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

 # 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.

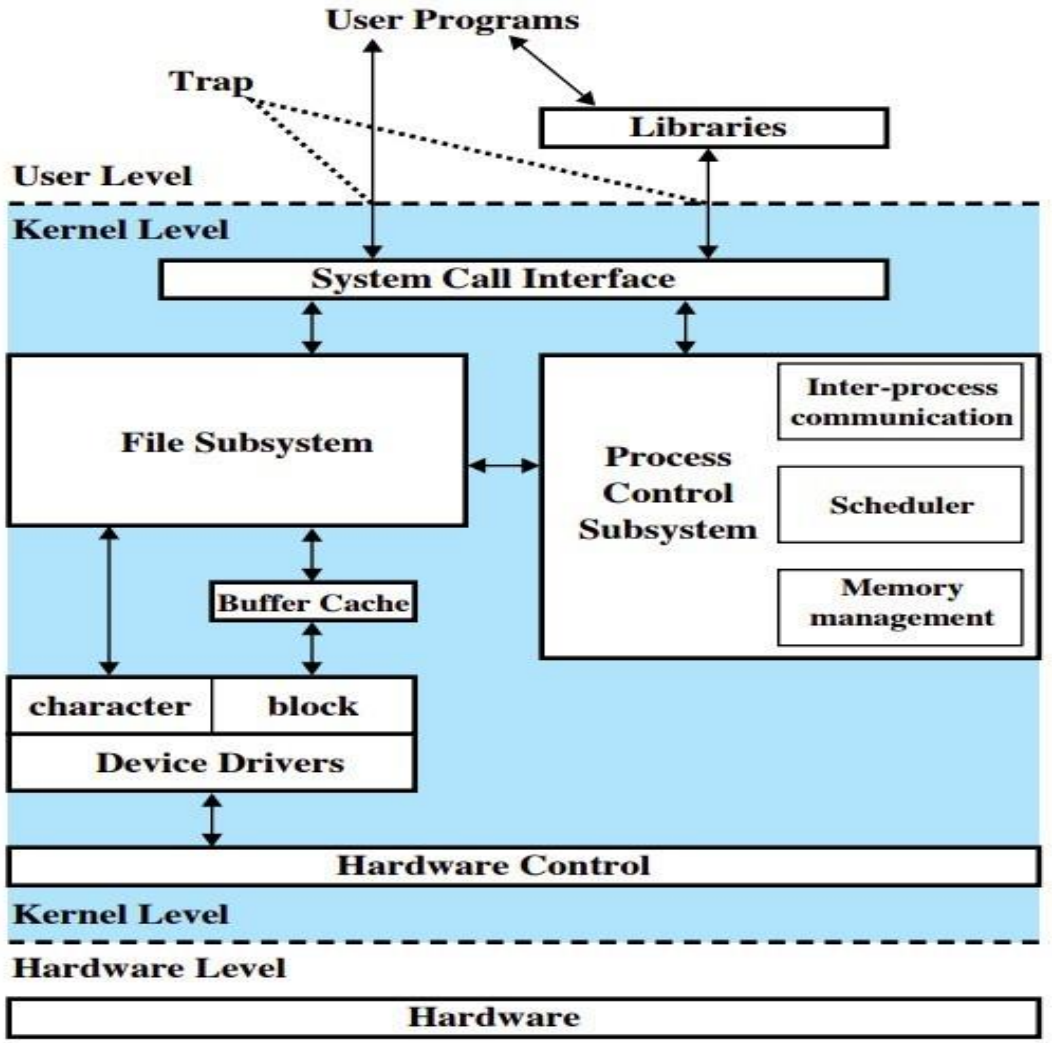
- It was first run on a machine **DEC-PDP-7** (Digital Equipment Corporation – Programmable Data Processing-7).

- 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.**

OS Architecture



Operating System Architecture

- 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 reffered as a process.

- File has space & Process has life.

Operating System Architechture

- 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

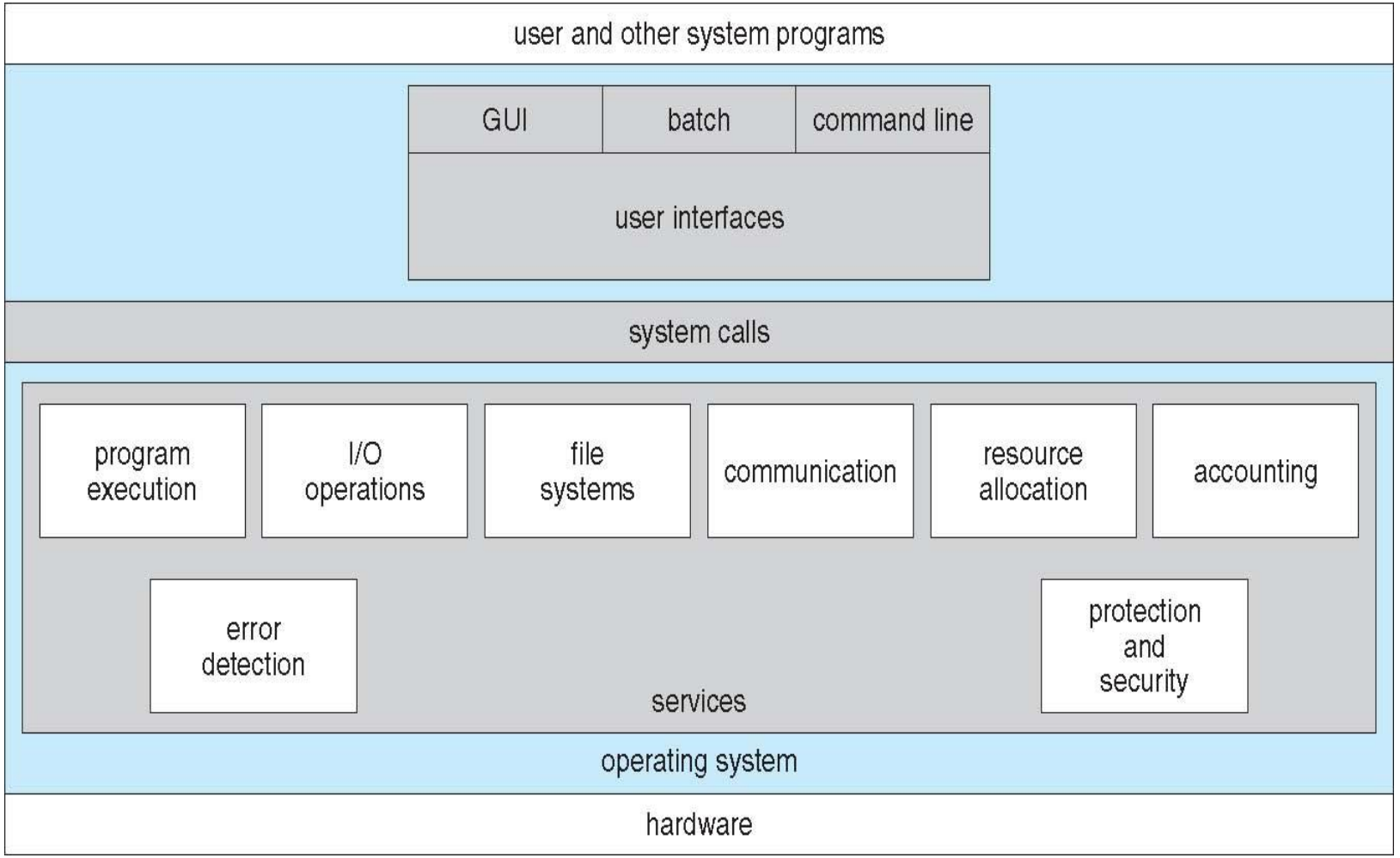
e.g. keyboard, mouse, printer, monitor etc...

2. Block Devices: Devices from which data gets transfered block by block --> block special device file

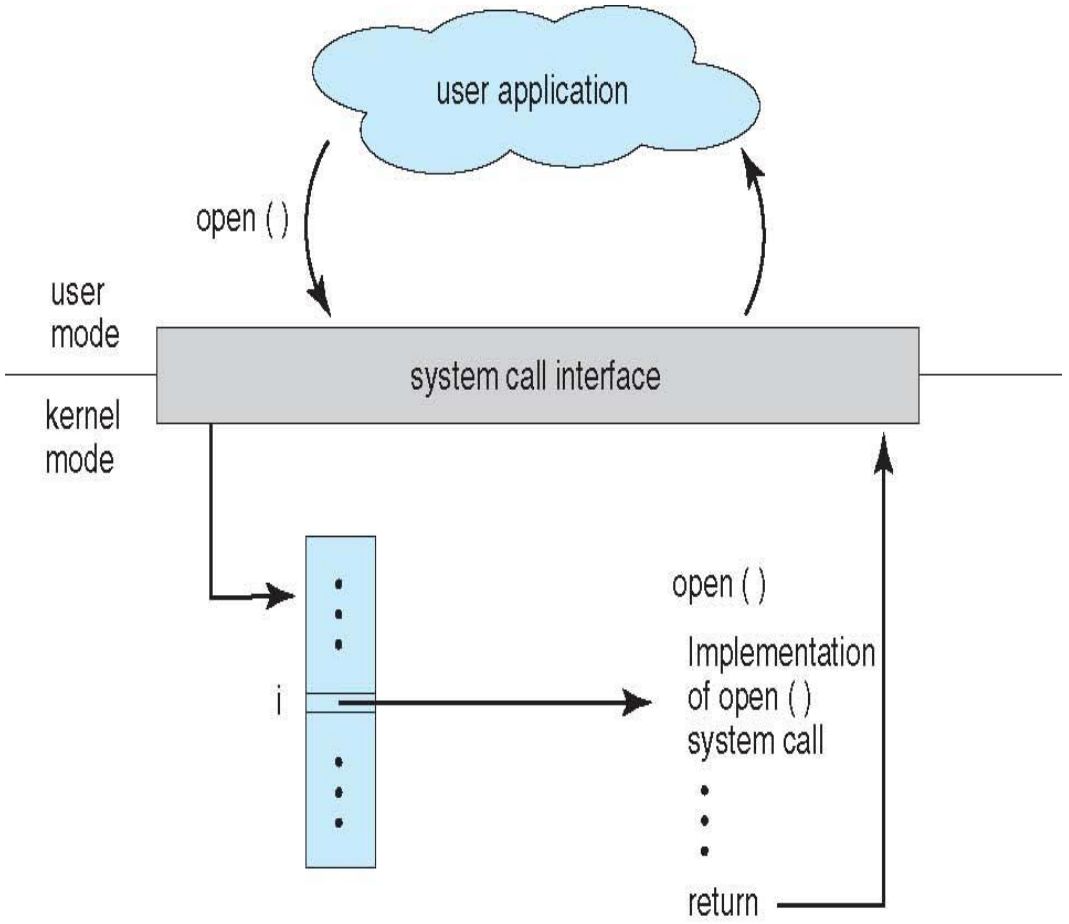
e.g. all storage devices.

- **Device Driver:**It is a program/set of programs enable one or more hardware devices to communicate with the computer's operating system.

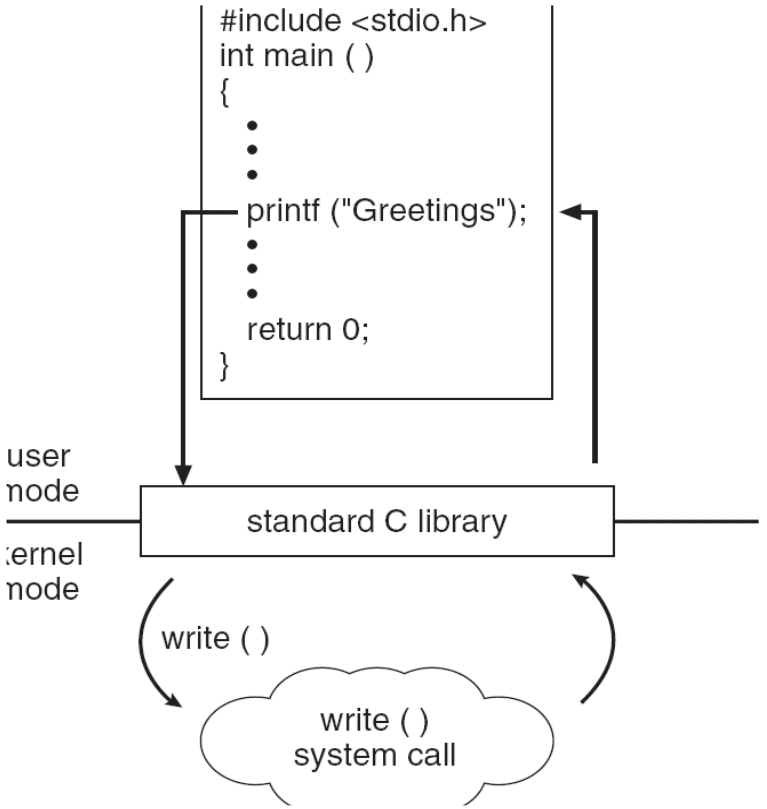
View of OS Services



System Call and OS Relationship



C Example



System Call Categories

Process control

(fork(),exit(),wait()

- load, execute, end, abort
- create process, terminate process
- get and set process
- allocate and free memory

File management

Read(),write()

- create file, delete file, open, close file
- read, write

Device management

Read(),write()

- request device, release device
- read, write
- get device attributes, set device attributes

Communications

Pipe(),shmget()

- send, receive messages
- transfer status information

Protection and Security

Chmod(),chown()

- Grant permissions
- Change ownership

Information maintenance

Getpid(),sleep()

- get time or date, set time or date
- get system data, set system data

System Calls

- 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...

System Calls

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 ModeOperation

 # Dual Mode Operation:

- System runs in two modes: System Mode and User Mode

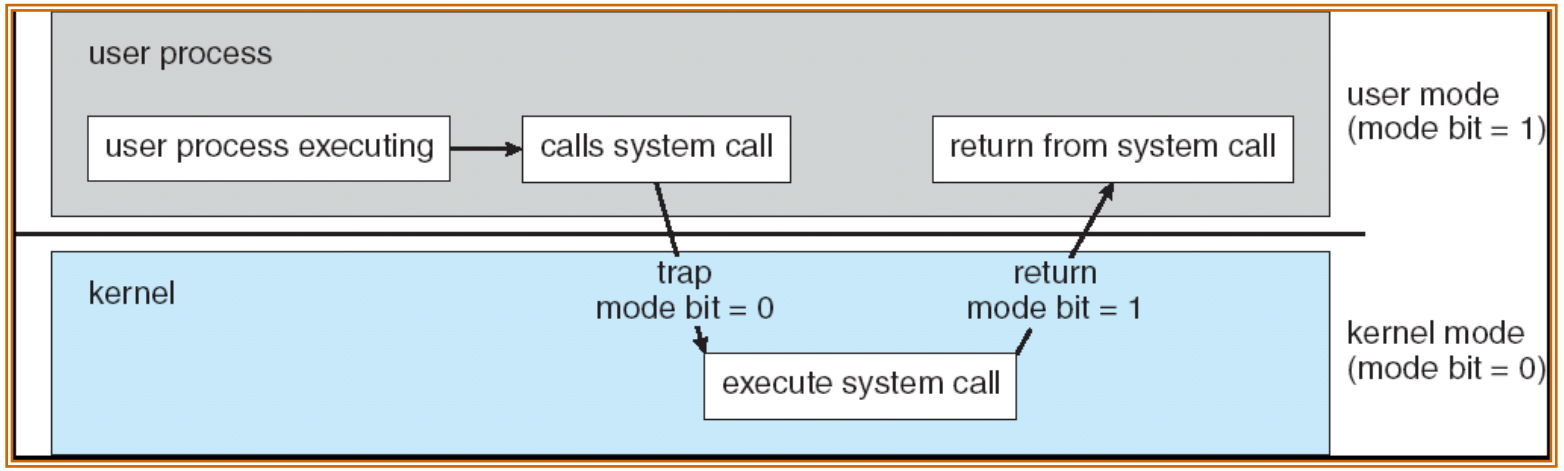
1. System Mode:

- 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/ privileged 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

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.**

- Todifferentiate 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.**

Kernel space vs User space

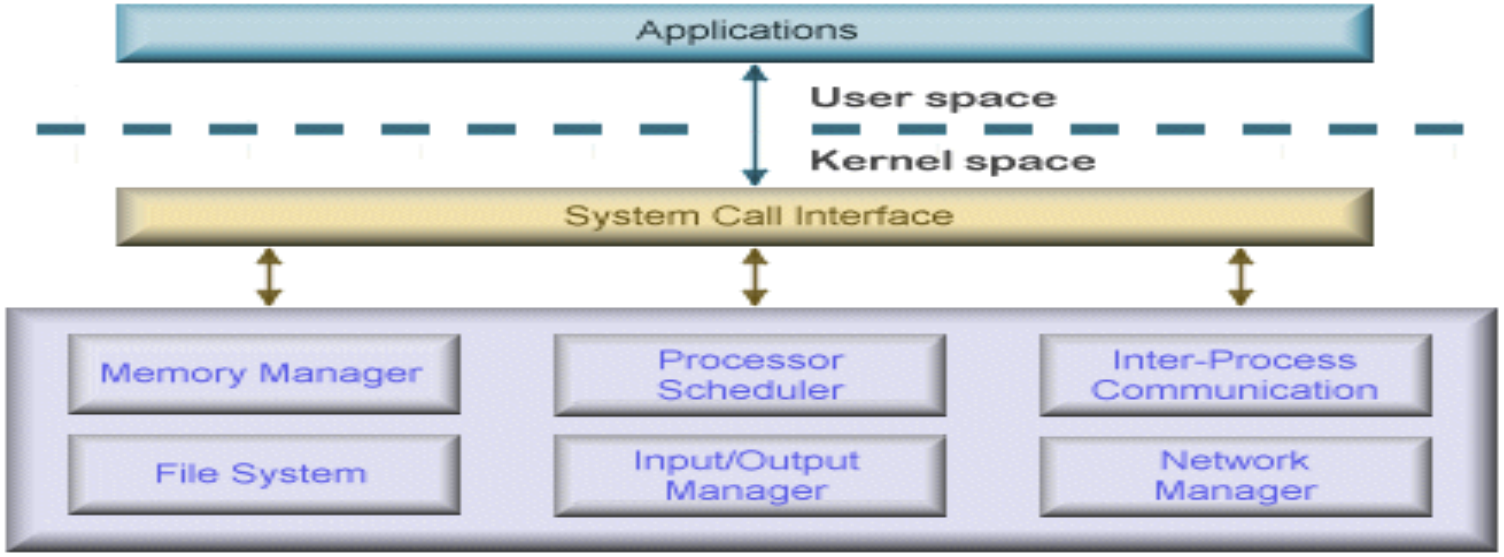
Part of the OS runs in the kernel model

known as the OS kernel

Other parts of the OS run in the user mode, including service programs , user applications, etc.

they run as processes

they form the user space (or the user land)



Thank you

Email: akshita.chanchlani@sunbeaminfo.com