# Operating System

"A program that acts as an intermediary between a user of a computer and computer hardware"

"Operating system controls and coordinates the use of the hardware among various applications programs for various users"

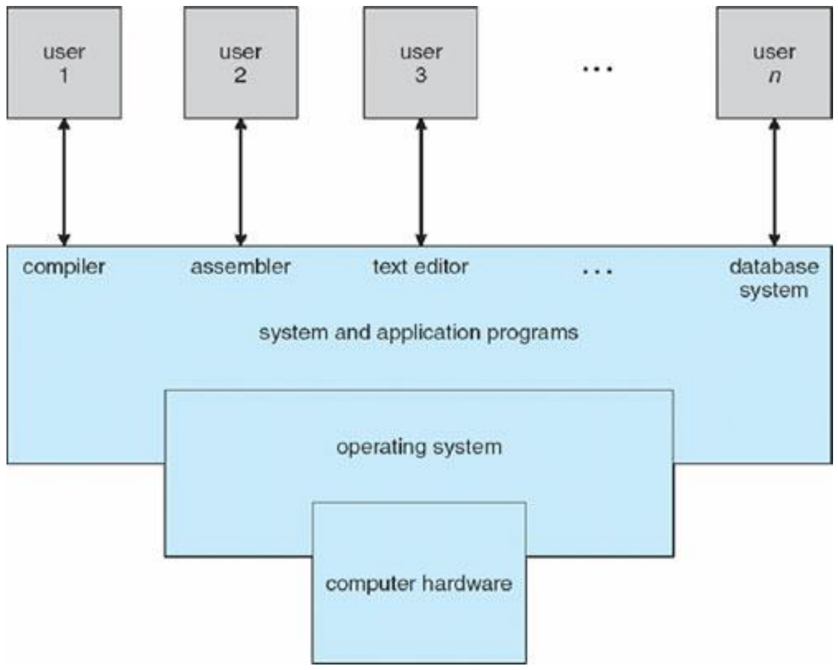
"The operating system is the one program that is running all times on the computer (usually called the Kernel), with all else being application program.

**Primary Goal:** Convenience of use (For small PCs)

**Secondary Goal:** Efficient utilization of hardware (for large, shared and multiuser systems).

**Components of Computer System**

i. Hardware : CPU, Memory, I/O Services
ii. Operating System:
iii. Applications program
iv. Users



**NOTE: Operating System and Computer Architecture have had a great deal of influence on each other. To facilitate the use of the hardware, operating system was developed. As operating system were designed and used, it become obvious that changes in the designing of hardware could simplify them.**

## Bootstrapping:

Booting is the process of loading operating system into main memory. For a computer to start running, it needs to have an initial program to load and execute the boot program, which in turn loads the operating system.

During bootstrapping, the kernel is loaded into memory and begins to execute. A variety of initialization tasks are performed and the system is made available for the users. The primitive loader program that can load and execute the Boot program is called Bootstrap Program. Boot strap program is generally stored in ROM. On- Start up, the computer automatically reads the **bootstrap program**. This primitive loader is then executed and loads the boot program in predefined memory location.

The boot program is generally stored on the disk with predefined address, called boot sector. The boot program then loads the operating system into memory. This arrangement is known as **bootstrapping.**

## Uses of Operating System:

- **As a Government:** Like a government, the OS perform no useful function for itself. It simply provides an environment within which other program can do useful work.
- **As a resource Manager:** The OS acts a manager of the computer system resources (like software and hardware and more specifically CPU time, memory space, file storage space, I/O devices etc) and allot them to specific program and uses as necessary for their tasks.

  Since there may be many possibly conflicting requests for recourses, the operating system must decide which resource to operate the computer system efficiently and fairly.

- **As a Control Program:**
  The Operating system controls the execution of user programs to prevent error and improper use of the computer. It is especially concerned with the operation and control of I/O devices.

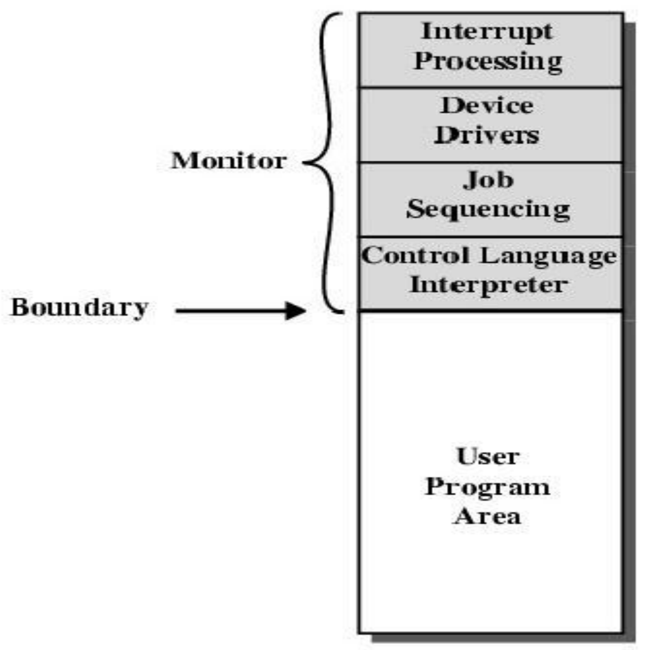
## Evolution of Operating System:

1. **Early Computers :**
   - Large Machine Run from the Console.

- The programmer would write the program and then operate the program directly from operator’s console.
- Program would be loaded manually into memory from the front panel switches (One instruction at a time) from the paper tape or from the punch card.
- Appropriate button would be pushed to set the starting address and to start the execution of the program.
- The programmer / operator could monitor its execution by the display lights on the Panel.
- If the error were discovered the programmer would have the program, examine the contents of the memory and registers and debug the program directly from the console.
- Output was printed or was punched onto paper tape or card for later processing.

## 2. Simple Batch Systems ( Resident Monitor)

- The emphasis was on high utilization of computers to get as much as they could from their investments.
- Professional computer operators were hired which has reduced the job set up time.
- Jobs with similar needs were batched together and run through the computer as group.
- Automatic job sequencing was developed which was the first rudimentary operating system. This was procedure for automatically transferring control from one job to the next.
- A small program called a resident monitor was created for this purpose. It was always resident in the memory.



**(Resident Monitor)**

When the computer was turned on, the resident monitor was invoked, and it would transfer control to the program terminated, it would return control to the resident monitor, which would then go to the next program. Thus the resident monitor would automatically sequence from one program to another and from job to another.

- ➢ Control card were introduced to provide a short description of what program were to be run on what data.
- ➢ To distinguish control cards from simple jobs cards dollar sign ( $) in the first column was used.
- ➢ IBM' Job Control Language ( JCL ) used slash marks ( //) in the first two column.
- ➢ Control card interpreter was responsible for reading and carrying out the instruction on the cards at the point of execution.
- ➢ The control card interpreter at intervals invokes a loader to load the systems programs and application programs into memory. Thus loader is a part of resident monitor.
- ➢ Both Control card interpreter and the loader need to perform I/O, so the resident monitor has set of device drivers for the system's I/O devices.
- ➢ A Batch Operating System, thus normally reads a stream of separate jobs, each with its own control cards that predefined what the job does. When the job is compltes it output is printed.
- ➢ Lack of interaction between user and the job while the job is executing.

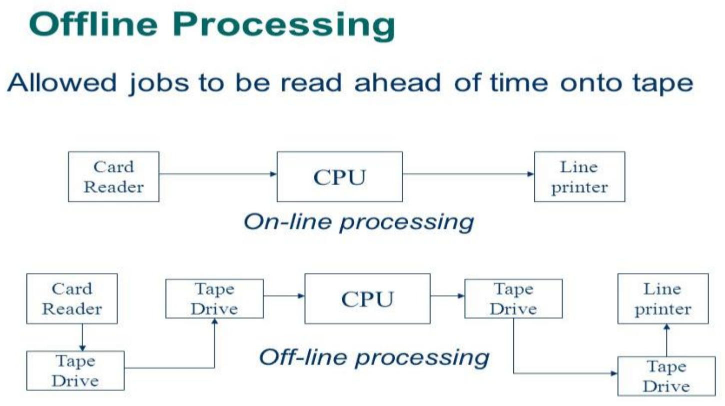
- The delay between job submission and job completion is called TURNAROUND time, may result the amount of computing needed or from delays before the OS starts to process the job.

## OVERLAPPED CPU and I/O: --

Even with automatic job sequencing, the CPU is often idle. The problem is the speed of the mechanical I/O devices which are intrinsically slower than electronic devices. For this problem the following solution are recommended:

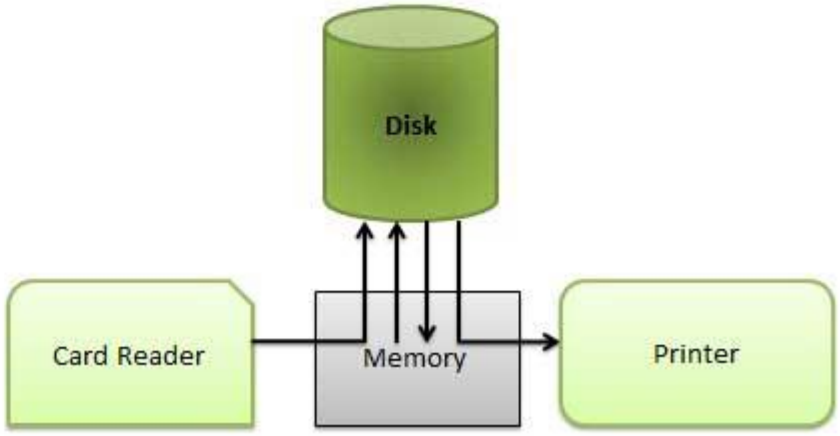
### A. Off-Line Processing :

- Replacement of the very slow card readers (input devices) and line printers (output device) with magnetic tape unit.
- Rather than the CPU read directly from cards, the cards were first copied onto a magnetic tape via a separate device.
- When the tape was sufficiently full, it was taken down and carried over to the computer.
- For input, the equivalent record was read from the tape. Similar for output, it was written to the tape and the contents of the tape would be printed later.
- The card readers and line printers were operated OFFLINE rather than by the main computer.
- Main advantage of OFF-LINE operations was that the main computer was no longer constrained by the speed of the card readers and line printers but was limited by only the speed of the much faster magnetic tape units.
- Now longer delays in getting a job run. It must first be read onto tape. Then, there is delay until enough other jobs are read onto the tape to fill it.



## B. Spooling ( Simultaneous Peripheral Operation On-Line)

- Random-Access Device like disk system was introduction.
- In disk system, cards are read directly from the card reader onto the disk.
- Location of card images is recorded in a table kept by the operating system.
- When a job is executed, the OS satisfies its requests for card reader input by reading from the disk.
- Similar, when the job requests the printer to output the line, tht line is copied into a system buffer and is written to the disk.
- When the job is completed, the output is actually printed.
- This form of processing is called Spooling.
- It is also used for processing data at remote sites. The remote processing is done at its own speed with no CPU intervention. The CPU just needs to be notified when the processing is completed, so that it can spooled the next batch of data.



**(SPOOLING)**

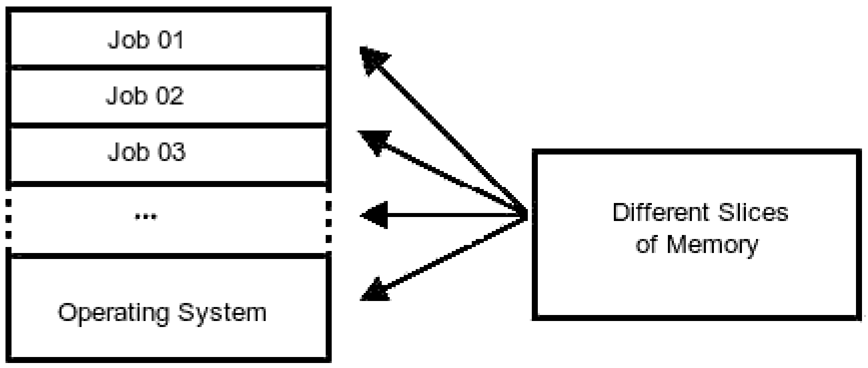
- Spooling overlaps the I/O of one job with the computation of other jobs.
- Spooling has a direct beneficial effect on the performance of the system
- Spooling can keep the CPU and the I/O devices working at same time at much higher rates.

## 3. MULTIPROGRAMMED SYSTEMS :

- Spooling provides a job pool (i.e. several jobs read into memory waiting on disk and ready to run.)
- Job Scheduling becomes possible which facilitate ability to multiprogramming.

- Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has something to execute.
- The OS keeps several jobs in memory at a time which is subset of the jobs kept in the job pool.
- The OS picks and begins to execute one of the jobs in the memory.
- The OS switches to and executes another job, when one job needs to wait for some I/O operation.
- As long as there is always some jobs to execute, the CPU utilization increases.
- If several jobs are ready to be brought into memory and there is not enough room for all of them, then the system must choose among them. This decision is **job scheduling**.
- When several programs are in memory at the same time, some kind of memory management is required.

Multiprogramming



- In addition if several jobs are ready to run at the same time, the system must choose among them. This decision is **CPU Scheduling**.
- Multiple jobs running concurrently requires that their abilities to affect one another be limited in all phases of the OS, including process scheduling , disk storage and memory management.

## 4. TIME SHARING SYSTEMS (Multitasking)

- In Batch system user cannot interact with the jobs when it is executing, the user must set up the control cards to handle all possible outcomes.
- A long turnaround time inhibits experimentation with a program.
- Time sharing (Multitasking) is a logical extension of multiprogramming. Multiprogramming jobs are executed by the CPU Switching between them, but the switched occurs so frequently that the users may interact with each program while it is running.
- An interactive or hands on computer system provides On-line communication between the user and the system.

- The user gives instruction through a keyboard and receive an immediate response on a display screen (on CRT or Monitor).
- When the OS finishes the execution of one command it seeks the next not from the card reader but from user's keyboard.
- Most system have an interactive text editor for entering programs and an interactive debugger for assisting in debugging programs.
- An on-line file system was available; the file may be free form such as text files or rigidly formatted.
- An interactive system is used when a short response is required.
- A time shared system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time shared computer.
- Each user has at least one separate program in memory. A program that is loaded into memory and is executing is commonly referred to as a process.
- It allows many users to share the computer simultaneously.
- It provides mechanism for concurrently execution, which requires sophisticated CPU scheduling scheme.
- To ensure orderly execution, the system must provide mechanism for jobs synchronization and communication and must ensure that jobs do not get stuck in an deadlock forever waiting for each other.

## 5. PERSONAL COMPUTER SYSTEM (PCs) [ Appeared in 1970's]

- These systems were dedicated to single user.
- I/O devices have changed with panels of switches and card reader replaced with keyboard and mice.
- Line printer and card punches have succumbed to display screen and small fast printers.
- They are micro computers, considerably smaller and less expensive than main frame systems.
- Lacking the features needed to protect an operating system from user program.
- Neither multi user nor multi tasking.
- Goal of these OS have changed with time, instead of trying to maximize CPU and peripheral utilization, the system opt for user convience and responsiveness.
- IBM PC Family of computers running MS DOS Operating System.

**NOTE: MS-DOS has been extended by Microsoft to include a Window System and IBM has upgraded MS DOS with the OS/2 multitasking system.**

- The personal workstation is a large personal computer such as SUN, HP/Apollo or IBM RS/6000 computers.

## 6. PARALLEL SYSTEMS: (Multi-Processing)

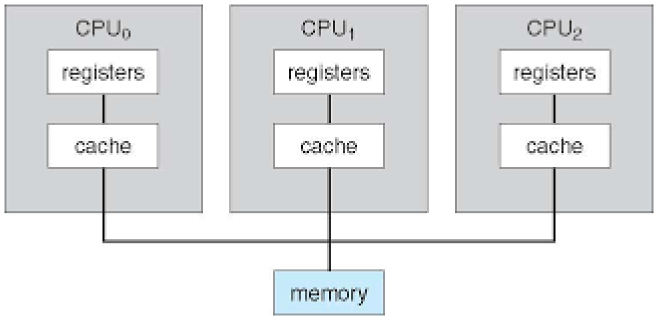
Having more than one processor in close communication sharing the computer bus, the clock and sometimes memory and peripheral devices. These systems are referred as tightly coupled systems.

- One Advantage is increased throughput.
- Save money compared to multiple single systems because of sharing of peripheral, cabinets and power supply.
- Increase reliability and provide **“graceful degradation”** (ability to continue providing services proportional to the level of non-failed hardware).
- System designed for graceful degradation is called **“Fail-Soft”.**
- **“Tandem System”** uses both Hardware and Software duplication to ensure continued operation despite faults. The system consists of two identical processors, each with its own local memory and connected by a bus.
- One processor is called **Primary** and other is called **Backup or Secondary** processor.
- Two copies are kept of each process; One for primary and other for Secondary.
- If Failure is detected, the backup copy is activated and is restarted from the most recent checkpoint.

**Types:**

**i. Symmetric Multiprocessing :**

- Each processor runs an identical copy of the OS and copies communicate with one another as needed.
- Each processor performs all tasks within OS.
- All Processor are Peer and no Master Slave relationship exists.
- In Symmetric Multiprocessing treats all processors as equals and no I/O can be processed on any CPU.



**(Symmetric Multiprocessing)**

### ii. Asymmetric Multiprocessing

- Each processor is assigned a specific task.
- A master processor controls the system. The other processors look to the master for instructions or predefined task. Thus, this defines master-slave relationship.
- Master Processor schedules and allocates the work to the slave processor.
- Each processor has its own address space.

**Difference** between symmetric multiprocessing and asymmetric multiprocessor operating system is that: **Symmetric Multiprocessing** treats all processors as equals and no I/O can be processed on any CPU while **Asymmetric Multiprocessing** has one master CPU and the remainder CPUs are slave. The master distribute tasks among the slave processors and I/O usually done be master only.

## 8. Distributed System:

- Distribute computation among several processors.
- The processor does not share memory or a clock. Instead each processor has its own local memory
- The Processor communicates with one another through various communications line such as high speed buses and telephone lines.
- These systems are usually referred as **Loosely Coupled Systems.**
- The processors may vary in size and functionality. These processors are refered as sites, nodes and computers.

**Advantages:**

i. **Resources Sharing:**
   - Sharing and printing files at remote sites
   - Processing information in a distributed database – using remote specialized hardware devices

ii. **Computational Speed up:**
   - Computation can be partitioned into number of subcomputations that can run concurrently.
   - Distributed system allow to distribute the computation among the various sites to run that computation concurrently.

iii. **Reliability :**
   - via redundancy either in both hardware or data
   - The failure of a site must be detected and recovered by the system

iv. **Communication:**
   - Numbers of sites are connected to each other via communication network.

- Processors at different sites have the opportunity to exchange information through file transfers, message passing or e-mails.

## 9. REAL TIME SYSTEMS

- It is used when there are rigid time requirement on the operation of a processor or the flow of data and the thus is often used as a control device in a dedicated application.
- They tend to have very specific tasks. They have no user interface.
- Sensors bring data to the computer.
- The computer must analyze the data and possibly adjust controls to modify the sensor input.
- These systems are embedded system where the tasks are already integrated in circuits which executed without an OS.
- **Example:** System that control scientific experiments, medical imaging system, industrial control system and some display systems are some real time systems. Some automobile engine, fuel injection systems, Home appliance controllers and weapon systems are also example of real time system.
- It has well defined fixed time constraints. Processing must be done within the defined constraints or the system will fail.

There are **two** types of Real Time Systems:

### i. Hard Real Time System:

- It guarantees that critical tasks complete on time. This goal requires that all delays in the systems be bounded from the retrieval of stored data.
- Secondary Storage of any sort is limited or missing with data instead being stored in short term memory or in Read Only Memory (ROM)
- Virtual Memory is almost never found on real time system.
- Since, none of the existing general purpose OS support hard real time functionality.

### ii. Soft Real Time System:

- Less restrictive than hard real time system.
- In this a critical task gets priority over other tasks and retain that priority until it completes.
- As in Hard real time system, kernel delays still need to be bounded.
- A real task cannot be kept waiting for indefinitely for the kernel to run it.

- Soft real time is an achievable goal that is amenable to mixing with other type of system
- It has more limited utility than Hard Real time system.
- Given their lack of deadline support, they cannot be used for industrial control and robotics.
- These systems need advanced operating System features that cannot be supported by hard real time systems. Because of the expanded uses for the Soft real time functionality, it is finding its way into most recent OS.
- It is less stringent timing constraints and donot support deadline scheduling.

## System Components of Operating System:

The components of an operating system all exist in order to make the different parts of a computer work together. Each component has its pre-defined functions to handle a specific task for operating system. These components include:

- Process Manager
- Main Memory Manager
- File Manager
- Secondary Storage Manager
- I/O Manager
- Networking Manager
- Protection System Manager
- Command Interpreter

### 1. Process Manager

A process is a program in execution. A process needs certain resources, including CPU time, memory, files, and I/O devices, to accomplish its task.
Process Manager performs following activities for an Operating System:

- The creation and deletion of both user and system processes
- The suspension and resumption of processes.
- The provision of mechanisms for process synchronization
- The provision of mechanisms for deadlock handling.

### 2. Main Memory Manager

Memory is a large array of words or bytes, each with its own address. It is storage of quickly accessible data shared by the CPU and I/O devices. Main memory is a volatile storage device. It loses its contents in the case of system failure or system power off.
Main Memory Manager performs following activities for Operating System:

- Keep track of which parts of memory are currently being used and by whom.
- Decide which processes are to be loaded into memory when memory space becomes available.

- Allocate and de allocates memory space as needed.

## 3. File Manager

A file is a collection of related information defined by its creator. Commonly, files represent programs (both source and object forms) and data or any form of audio or video
File Manager performs following tasks for an Operating System

- The creation and deletion of files.
- The creation and deletion of directory.
- The support for manipulating files and directories.
- The mapping of files onto disk storage.
- Backup of files on stable (nonvolatile) storage.
- Protection and security of the files.

## 4. Secondary Storage Manager

Since main memory (primary storage) is volatile and too small to accommodate all data and programs permanently, the computer system must provide secondary storage to back up main memory. Secondary storage includes CD-ROM, hard disks, pen drives etc.
Secondary Storage Manager performs following tasks for an Operating System

- Free space management
- Storage allocation
- Disk Scheduling

## 5. I/O Manager

The I/O system consists of a buffer-caching system, a general device-driver interface or the drivers for specific hardware devices.
I/O Manager performs following tasks for an Operating System

- Managing buffer caching system
- To activate a general device driver code
- To run the driver software for specific hardware devices as and when required.

## 6. Networking Manager

A distributed system is a collection of processors that do not share memory or a clock. Each processor has its own local memory and clock. The processors in the system are connected through a communication network.
Networking Manager performs following tasks for an Operating System

- Setup of a networking connection (Wireless & LAN)
- Mechanism for maintaining IP Address (Manual & by using DHCP)
- Managing networking protocol
- Security of network connection
- Firewall management for blocking unauthorized access of computer

## 7. Protection System Manager

Protection refers to a mechanism for controlling access by programs, processes, or users to both system and user resources.
Protection System Manager performs following tasks for an Operating System

❖ Provides a mechanism for controlling the access of programs, processes, or users to the resources defined by a computer controls.

❖ Improve reliability by detecting latent errors at the interfaces between component subsystems.

❖ Early detection of interface errors to prevent healthy subsystem by a subsystem that is malfunctioning.

❖ Provide a mean of authentication and authorization

## 8. Command Interpreter

Command-Interpreter system is a system program, which is the interface between the user and the operating system. This interface can be character based like DOS or it can be Graphical User Interface like Mac and Windows.
Command Interpreter performs following tasks for an Operating System

❖ Provides a mechanism to read command from user or process.

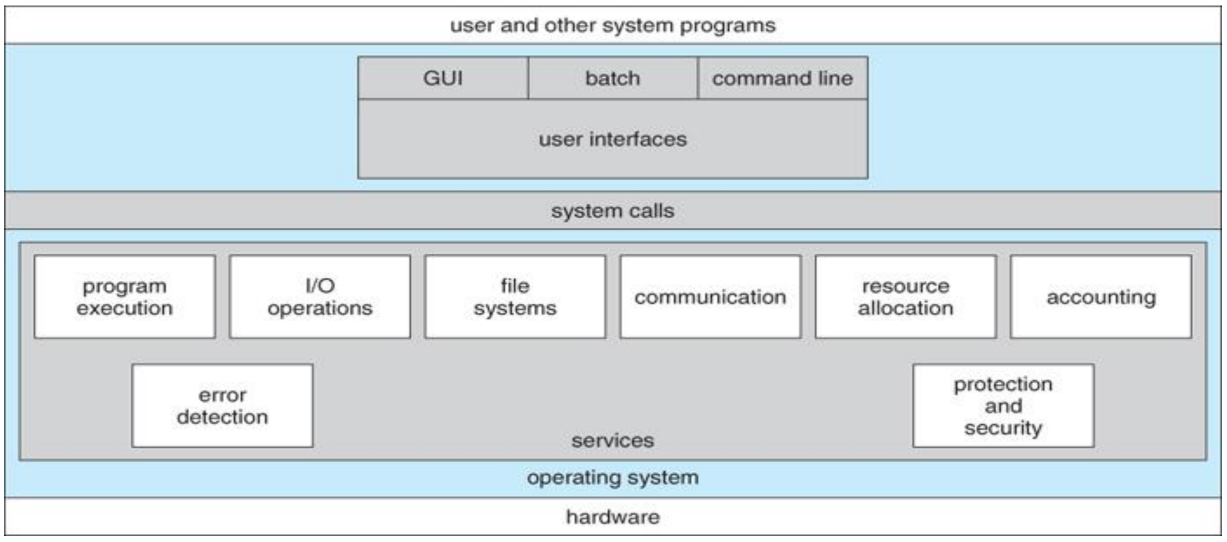
❖ Interpret command into low level language

❖ Interact with required hardware, resource or file.

❖ Generate output defined in the command.

## System Structure

One set of operating system services provide functions helpful to the user.



**(View of operating system services)**

## Operating System Services

- **User Interface:**
    - **Command Line Interface (CLI):** It uses text command and method for entering them.
    - **Batch Interface:** Commands and directives to control those commands are entered into files and those files are executed.
    - **Graphical User Interface (GUI):** Interface with pointing device to direct I/O.
- **Program execution** – system capability to load a program into memory and to run it. Program must be able to ends its execution either normally and abnormally.

  **Program Loading →Program Execution → Program Termination**
- **I/O operations** – since user programs cannot execute I/O operations directly, the operating System must provide some means to perform I/O operation either from file or I/O devices.
- **File-system manipulation** – Many OS provides variety of file systems by which program is able to read, write, create, and delete files.
- **Communications** – Exchange of information between processes executing either on the same computer or on different systems tied together by a network. Implemented via shared memory or message passing.
- **Error detection** – Ensure correct computing by detecting errors in the CPU (such as power failure) and memory hardware, in I/O devices (such as connection failure), or in user programs.
- **Resource Allocation** –
    - Many types of resources are managed by the operating system.
    - CPU cycle, main memory and file storage may have special allocation code.
    - I/O device may have much more general request and release code.
    - For example CPU scheduling routines that takes into account the speed of CPU.

- There may also be routines to allot printers, modem, USB storage device and other peripheral device.

➢ **Protection and Security:**

- Protection involves ensuring that all access to system resource is controlled.
- Security of the system from outside is also important.
- Security starts with each user having to authenticate himself to the system usually by means of password to allow the access to the resources.

➢ **Accounting:-**

- To keep track of which users and how much and what kind of computer resources.
- Used for accumulating usage statics.
- Usage statics may be valuable tool for the who wish to reconfigure the system to improve computing services.
- This record keeping may be for accounting (so that users be billed or simply for accumulating usage statistics.

## Operating System Design and Implementation

➢ Design and Implementation of an OS not **"Solvable"** but some approach have proven successful.

➢ Internal structure of different OS can vary widely.

➢ Start by defining goals and specifications.

➢ Affected by the choice of hardware and type of system.

➢ User goal and System goal :

- **User goal :** OS should be convenient to use , easy to learn , reliable , state and fast.
- **System Goal:** OS should be easy to design , implement and maintain as well as flexible , reliable , error free and efficient.

➢ **Important principle to separate**

- **Policy:** What to be done.
- **Mechanism:** How to do it.

➢ Mechanism determine how to do something, policies decide what to be done.

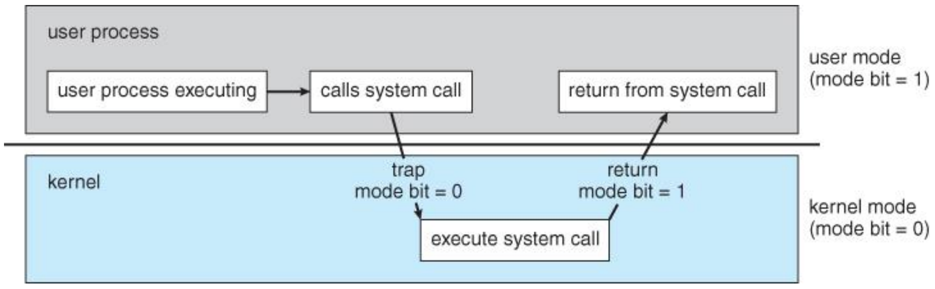
➢ Separation of policy from mechanism is very important principle, it allows maximum flexibility if policy decision are be changed later.

## System Protection

- As the operating Systems developed, the control of the system is given to the system is given to OS. In addition to improve system utilization, OS began to share system resources among several programs simultaneously.
- The sharing of resources among various program create both improved utilization and increased problems too. With sharing, many processes could be adversely affected by a bug in one program.
- A properly designed OS must ensure that an incorrect or malicious program cannot cause other programs to execute incorrectly.
- Many errors are detected by the H/W and handled by OS.
- If a user program fails in some way, the H/W will trap to the OS. The trap transfers control through the interrupt vector to the operating system just like an interrupt. Whenever a program occurs, the OS must abnormally terminate the program. An appropriate error message is given and the memory of the program is dumped.

### Dual Mode Operation of an Operating System:

- To ensure proper operation, we must protect the OS and all other programs and their data from any malfunctioning program.
- Protection is needed for the shared resources.
- Hardware support is provided to allow us to differentiate among various modes of executions.
- At least two separate of mode of operation: **User Mode** and **Monitor Mode** (Also Called Kernel mode or Supervisor Mode) are needed.
- **User Mode:** Executing code has no ability to directly access H/W or reference memory**.**
- **Monitor Mode/Kernel mode/Supervisor Mode:** Executing code has complete and unrestricted access to the underlying Hardware.
- A bit called the mode bit is added to the H/W of the computer to indicate the current mode. User Mode : **User(1)** and Monitor Mode : **Monitor(0)**
- Mode bit allow us to distinguish b/w an execution that is done on the behalf of the operating system and one that is done on behalf of the user.
- At system Boot Time, the H/W starts in monitor mode. The OS is then loaded and starts user processes in user mode.



- Whenever a trap or interrupt occurs, the h/W switches from user mode to monitor mode/kernel mode.
- The dual mode of operation provides us with the means for protecting the OS from errant users and the errant's users from one another.
- Privileged instructions (machine instructions which may cause harm) are executed only in monitor mode.
- MS-DOS written in 8088 Intel architecture has no mode bit and hence no dual mode.
- More advanced version of Intel CPU like 80486 provides dual mode operation. As a result, more recent OS like WIN/NT and IBM OS/2 take advantages of this feature and provide greater protection for the operating system.
- The lack of H/W supported dual mode can cause serious shortcomings in the operating system.

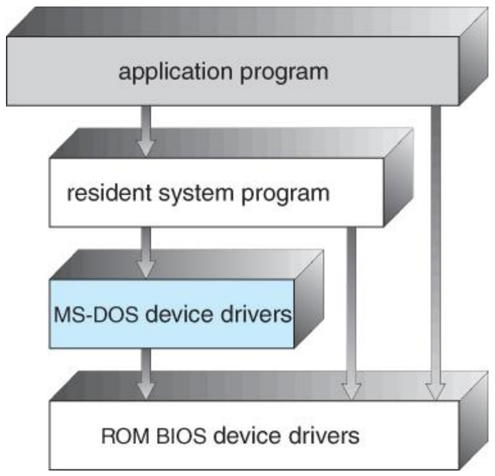
## Operating System Structure:

System as large and complex, modern operating system must be engineered or designed carefully if it is to function properly and to be modified easily.

### --- Simple Structure of OS:

- It does not have well defined structure.
- Such OS started as small, simple and limited system and grew beyond their original scope.

**MS-DOS:--** written to provide the most functionality in the least space because the limited hardware on which it ran, so it was not divided into modules carefully.



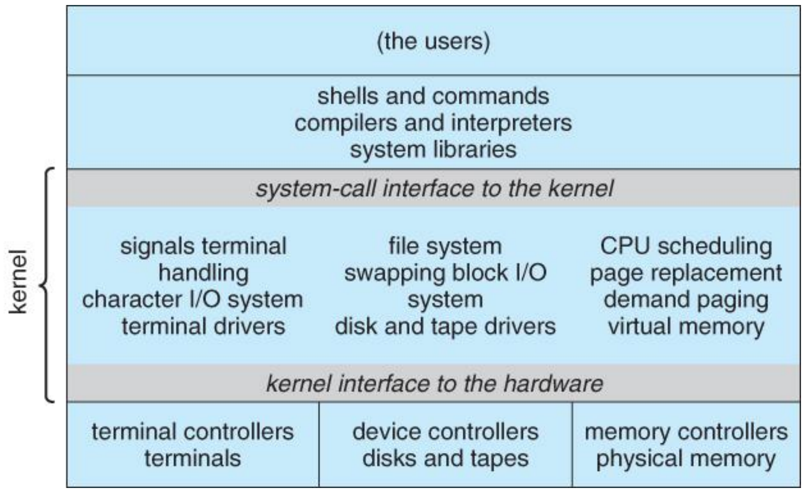
(**MS-DOS Simple Structure**)

MS DOS have some structure; its interface and level of functionality are not well separated. For instance application program are able to access the basic I/O routine to write directly to the display and disk drive which leaves the MS-DOS vulnerable to errant programs causing the entire system crashes or disk erases when user program fails.

Applications Programs are able to access directly ROM BIOS and hardware as there is only single mode. (User and kernel mode is absent)

**Basic Input Output System (BIOS):** BIOS is the program of personal computer's micro processor uses to get the PC started after you can turn it on. It also manages control between Operating System and attached devices.

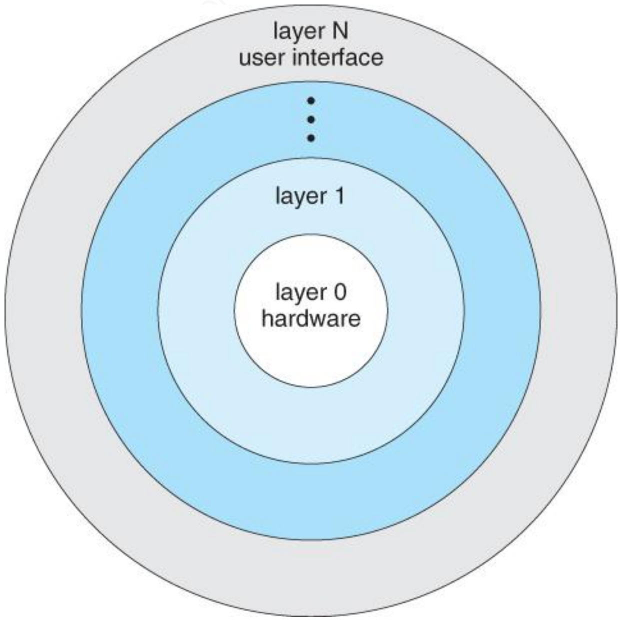
Another example of the simple structure is the original UNIX operating system. It was initially limited by hardware functionality.



- It consist of two separable parts: **Kernel and System Program**
- Kernel is further separated into serious of interfaces and device drivers which have been added and expanded over the years as UNIX has evolved.
- The Kernel provides the file system, CPU scheduling, memory management and operating systems functions through system calls.
- System programs use the kernel supported system calls to provide useful functions like compilation and file manipulations.
- System call defines the program interface to UNIX.

## Layered Approach:

- To provide proper H/W support, OS may be broken into smaller, more appropriate piece than those allowed by the original UNIX or MS-DOS.
- The OS can then retain much greater control over the computer and the applications that make use of the computer.
- Implementations have more freedom to change the inner working of the system.
- Under the top-down approach, the overall functionality and features can be determined and separated into components.
- Information hiding is also important, leaving programmer free to implement low level routines as they see fit, provides that external interface of the routines remains unchanged.



**(Layered Approach of the Operating System)**

- OS is divided into number of layers (Levels) each build on the top of the lower level.

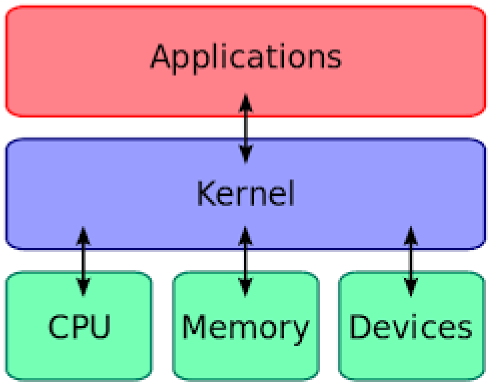
- The Bottom Layer (Layer-0) is the Hardware Layer and highest layer (Layer-N) is the interface layer.
- An OS layer is implementations of an abstract object that is the encapsulation of data and operation that can manipulate those data.
- Upper layer can invoke operation on lower level layer.
- The main advantage is **MODULARITY.**
- The layers are selected such that each uses functionality of only lower level layers. This approach simplifies the debugging and system verification.

Operating System basically contains **TWO** components: **Shell and Kernel**

**Shell** provides the interface to the user and command interpretation task**.**

**Kernel** is the central module of an OS. It is responsible for memory management/Process management/ Input Output management, resource management, Protection and Security etc**.**

Kernel is the central module of an OS. It is the part of OS that loads first and remains in the main memory. As it stays in the main memory, it is important for the kernel to be as small as possible. Kernel is the main module which provides services to the other part of OS and application software. The Kernel code is usually stored in the protected area of the memory to prevent it from being overwritten by program and other part of OS.

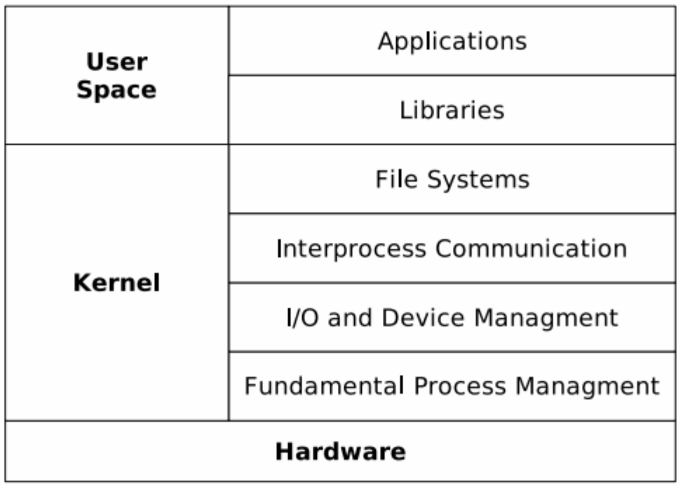


( Kernel of the Operating System)

Kernels are of following type:

1. **Monolithic Kernel**
   - ➢ Traditional UNIX OS uses monolithic Architecture.
   - ➢ The Entire OS runs as single program in Kernel Mode. Programs contains OS core functions and devices drivers.

- Most of the operations are performed by kernel via system call.
- Windows-95/98, Linux and FreeBSD OS use monolithic Kernel.



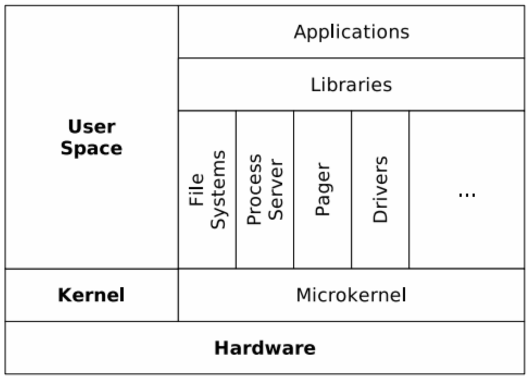
**(Monolithic kernel)**

- The monolithic is an older kernel used in UNIX, MS-DOS and early MAC-OS.
- In the monolithic kernel, every basic service like the process and memory management, interrupt handling and I/O communication; file system, etc. run in kernel space. It is constructed in a layered fashion, application at the top, then the libraries in user-space.

To overcome the drawbacks of the monolithic kernel, microkernel came into existence

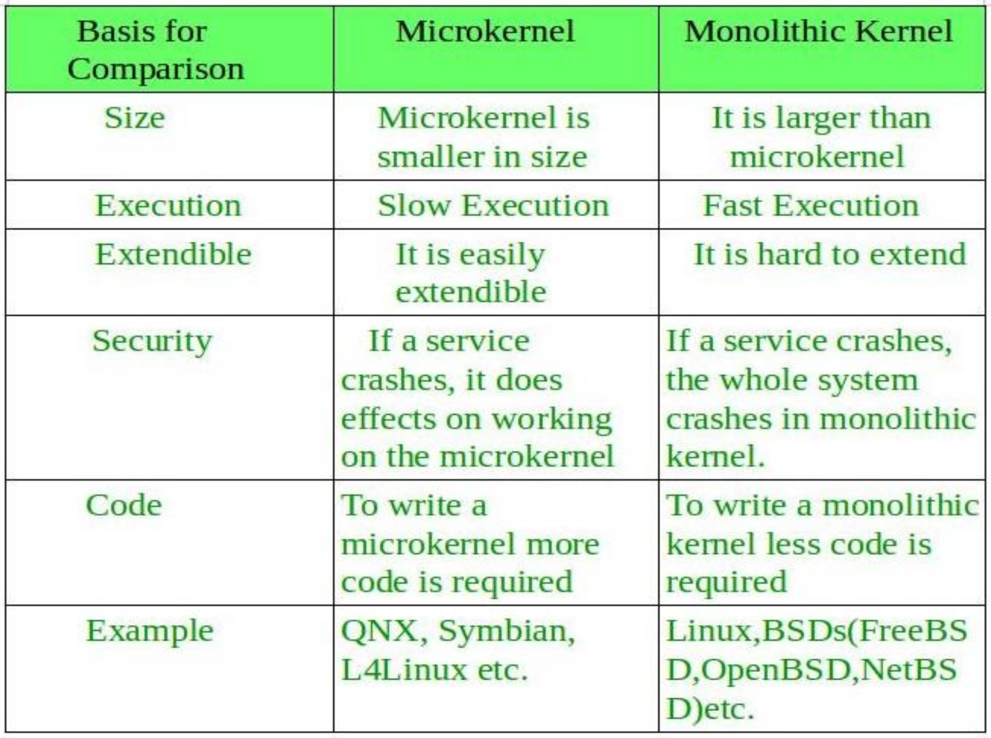
**2. Micro Kernel**

- Microkernel provides minimal services like defining memory address space, IPC and process management.
- In the micro-kernel, process resides in user-space in the form of servers.
- It has small operating core.



- Communication between kernel space and user space is done via **message parsing** which allows independent communication

**Difference between microkernel and monolithic kernel:**



| Basis for Comparison | Microkernel | Monolithic Kernel |
|---|---|---|
| Size | Microkernel is smaller in size | It is larger than microkernel |
| Execution | Slow Execution | Fast Execution |
| Extendible | It is easily extendible | It is hard to extend |
| Security | If a service crashes, it does effects on working on the microkernel | If a service crashes, the whole system crashes in monolithic kernel. |
| Code | To write a microkernel more code is required | To write a monolithic kernel less code is required |
| Example | QNX, Symbian, L4Linux etc. | Linux,BSDs(FreeBSD,OpenBSD,NetBSD)etc. |

## 3. Reentrant Kernel:

- Several processes may be in kernel mode at the same time. A reentrant kernel is able to suspend the current running process even if process is in the kernel mode.
- A reentrant kernel enables processes to give away the CPU while in kernel mode.
- They do not hinder other processes from also entering in kernel mode.
- UNIX kernel is reentrant kernel as several processes can be executed at the same time.
- A kernel that implement reentrant routines are called Reentrant Kernal.
- When two tasks executes the function at the same time without interfering each other, then the function is Reentrant.
- In Non Reentrant kernel mode, every single process acts on its own set of memory locations and thus cannot interfere each other.

- Reentrant functions only modify local variable but do not alter global data structure. To implement Reentrant Kernel, it is implemented as set of different Reentrant functions.

*********************************************************