# UNIT-1 (Introduction)

## Operating System

- Operating system is a system program that act as an interface between hardware and user.
- It manages system resources.
- It provides a platform on which other application programs are installed.

- Core Part of Operating System is called kernel.

**Goals of operating System**

**Primary Goal:**To make system user friendly (convenient).

**Secondary Goal:**To make system efficient.

**Role of Operating System:** We can explore operating Systems from two viewpoints that of the user and that of the system.

**User's View:** In user's view operating system is designed to maximize the work (CPU utilization) that the user is performing. In this context OS is designed mostly for **ease of use** with some attention paid to performance and none paid to resource utilization.

**System's View:** In system point of view operating system is a program that works as a **resource allocator**. Every computer system has many resources like CPU Time, Memory Space, File Storage Space, I/O Device and so on. The operating system act as manager of these resources.

## Functions of Operating System

Operating System is responsible for following activities.

**Process Management: Operating** System is responsible for following activities in connection with Process Management.

- ✓ Scheduling Processes and threads on the CPUs.
- ✓ Creating and deleting both user and system processes.
- ✓ Suspending and resuming processes.
- ✓ Providing mechanisms for process synchronization.
- ✓ Providing mechanism for process communication.

**Memory Management:** Operating System is responsible for following activities in connection with Memory Management.

- ✓ Keeping track of which parts of memory are currently being used and by whom.
- ✓ Deciding which processes and data to move into and out of memory.
- ✓ Allocating and de-allocating memory space as needed.

**Disk Management:** Operating System is responsible for following activities in connection with Disk Management.

- ✓ Free space management in disk.
- ✓ Storage Allocation in disk.
- ✓ Disk Scheduling.

**File Management:** Operating System is responsible for following activities in connection with File Management.

- ✓ Creating and deleting files.
- ✓ Creating and deleting directories to organize files.
- ✓ Supporting primitives for manipulating files and directories.
- ✓ Mapping files onto secondary storage.
- ✓ Backing up files on stable storage media.

**I/O Device Management:** Operating System is responsible for I/O Device management needed by various processes.

**Network Management** Operating System is responsible for network management.

**Security and Protection:**Operating System also provides security and protection to computer system.

# Services of Operating System

An operating system provides an environment for execution of programs. It provides mainly following services to programs and to the users.

**User Interface:** Almost all operating systems provide user interface to the users.

**Program Execution:** Operating system provides the service of program execution. Program is loaded into the memory and CPU is assigned to it for its execution.

**Access I/O Devices:** Operating system provides I/O devices to running program.

**File System Access:** Operating system provides services like read and write files and directories, search for a given file and directory, access and deny to files and directories etc to the running program.

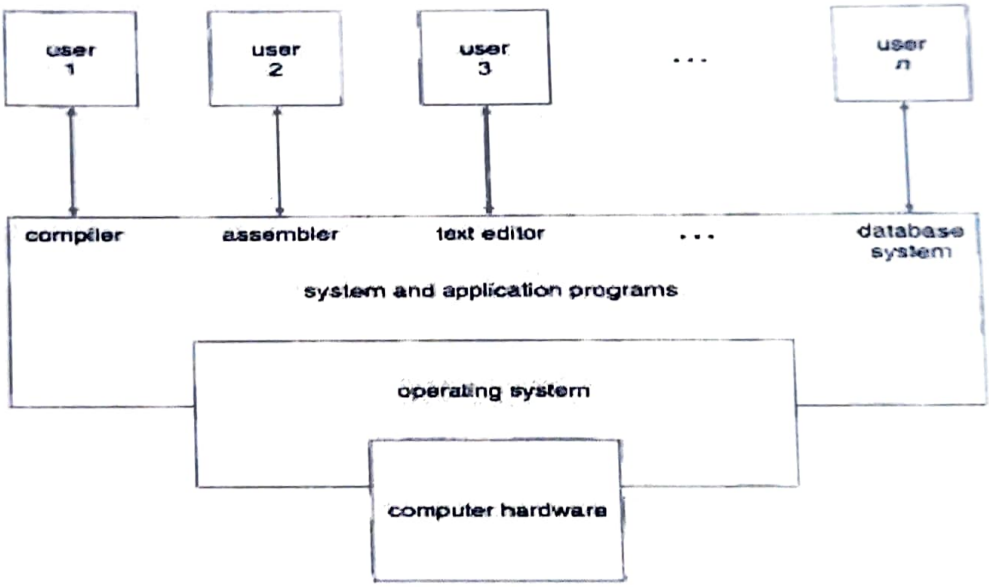
**Error Detection and Response:** Errors may be generated by the user programs or hardware. Operating system should take the appropriate action to ensure correct and consistent computing.

**Communication:** There are many circumstances in which one process needs to exchange information with another process. Such communication may occur between processes that are executing on same computer or between processes that are executing on different computer systems tied together by a computer network communication may be implemented via shared memory or through message passing by the operating system.

# Components of Computer System

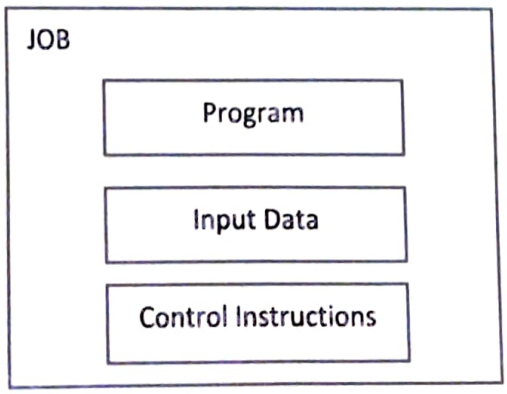
A computer system canbe roughly divided into following four components.

1. **Hardware:** Basic computing resources like CPU, Memory, I/O Devices etc.
2. **Application Programs:** Programs that are used to solve users' computing problems e.g. web browsers, compilers, word processors etc.
3. **Operating System:** It controls the hardware and coordinates its use among various application programs for different users. Operating system is similar to government; just like government it provides an environment within which other programs can do useful work.
4. **Users:** Users perform the computation with the help of application program.



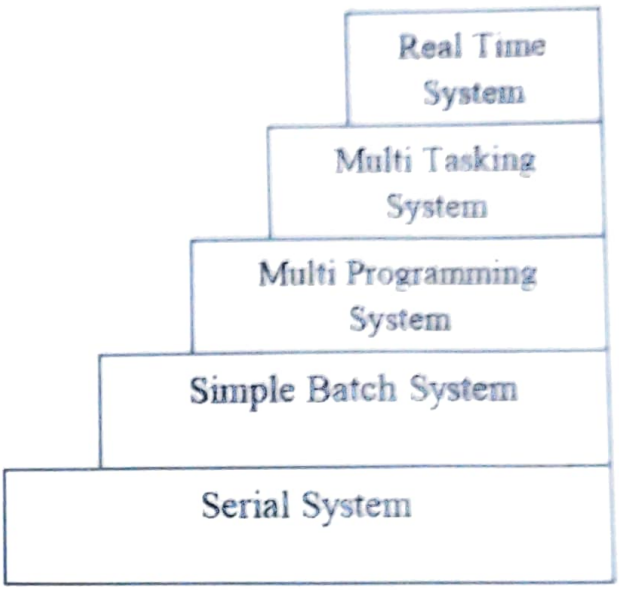

Abstract view of the components of a computer system.

**JOB:** When we combined program, input data and control instructions together, it is called Job.



# Classification of Operating System

- Decreasing Memory Size
- Increasing waiting time And Response time
- Decreasing speed of CPU
- Decreasing CPU utilization
- Increasing CPU idle time

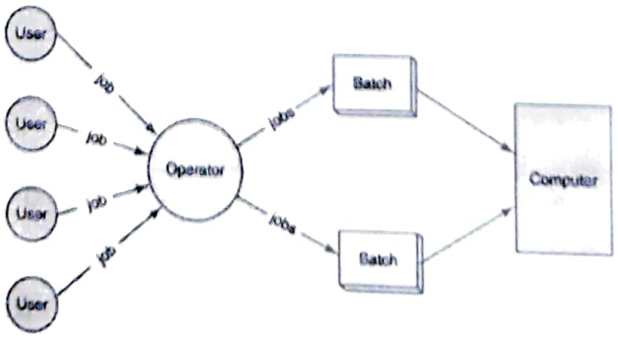



## Serial System

In this system, Operating System is not needed. Programs interact directly with computer hardware.

## Batch System

- Batch means set of jobs with similar need.
- In this system, Operator makes Batch of jobs and loads it into punch card.
- Similar types of jobs batch together and execute at a time.




**Advantages**

- ✓ Save time from activities like loading compiler.
- ✓ No manual intervention by user is needed.

**Disadvantages**

- ✓ Limited Memory

- ✓ Interaction of I/O directly with CPU.
- ✓ CPU is often idle.

## Multiprogramming Operating System

- ❖ Main objective of multiprogramming is to maximize CPU utilization.
- ❖ More than one process can reside in main memory which are ready to execute i.e. more than one process is in ready state.
- ❖ In this system if any running process performs I/O or other event which does not require CPU, then instead of sitting idle, CPU performs context switching and picks another process for execution.

**Advantages**

- ✓ High CPU utilization.
- ✓ Less waiting time, response time etc.
- ✓ Useful in current scenario when load is high.

**Disadvantages**

- ✓ Process scheduling is difficult.
- ✓ Main memory management is required.
- ✓ Problems like memory fragmentation may occur.

## Time Sharing or Multi-Tasking System

- ❖ It is also called fair share system or multi programming with round robin System.
- ❖ It is extension of multi Programming System.
- ❖ In this system CPU switches between processes so quickly that it gives an illusion that all executing at same time.
- ❖ This system can be used to handle multiple interactive tasks.

**Advantages**

- ✓ High CPU utilization.
- ✓ Less waiting time, response time etc.
- ✓ Useful in current scenario when load is high.

**Disadvantages**

- ✓ Process scheduling is difficult.
- ✓ Main memory management is required.
- ✓ Problems like memory fragmentation may occur.

# Multiprocessing System

- ❖ A system is called multiprocessing system if two or more CPU within a single computer communicate with each other and share system bus memory and I/O devices.
- ❖ It provides true parallel execution of processes.

## Type of Multiprocessing system

1. **Asymmetric Multiprocessing System:** In this system, each processor is assigned a specific task. A master processor controls the system. The other processors called slave processors either look to master for instruction or have predefined tasks i.e. master-slave relationship hold in this type of system.
2. **Symmetric Multiprocessing System:** In this system each processor performs all tasks within the operating system i.e. all processor are peers, no master-slave relationship exists between processors.

**Advantages:**

- ✓ **Increased Throughput:** By increasing number of processors, we expect to get more work done in less time.
- ✓ **Economy of Scale:** multiprocessor system can cost less than equivalent multiple single processor systems.
- ✓ **Increased Reliability:** Since work is distributed among several processors; failure of one processor will not halt the system only slow it down.

**Disadvantages**

- ✓ It is complex system.
- ✓ Process scheduling is difficult in this system.
- ✓ It required large size of Main memory.
- ✓ Overhead reduces throughput.

# Real Time System

- A real time system has well-defined, fixed time constraints. Processing must be done within the defined constraints, or the system will fail.
- A real time system functions correctly only if it returns the correct result within its time constraints.
- This system is used when we require quick response against input.
- In this system, task is completed within specified time.
- It is classified in following two types-

1. **Hard Real Time System:** In hard Real Time system task will be fail if response time against input is more than specified time.
2. **Soft Real Time System:** In Soft Real Time system inaccurate result will be found if response time against input is more than specified time.

# Interactive System

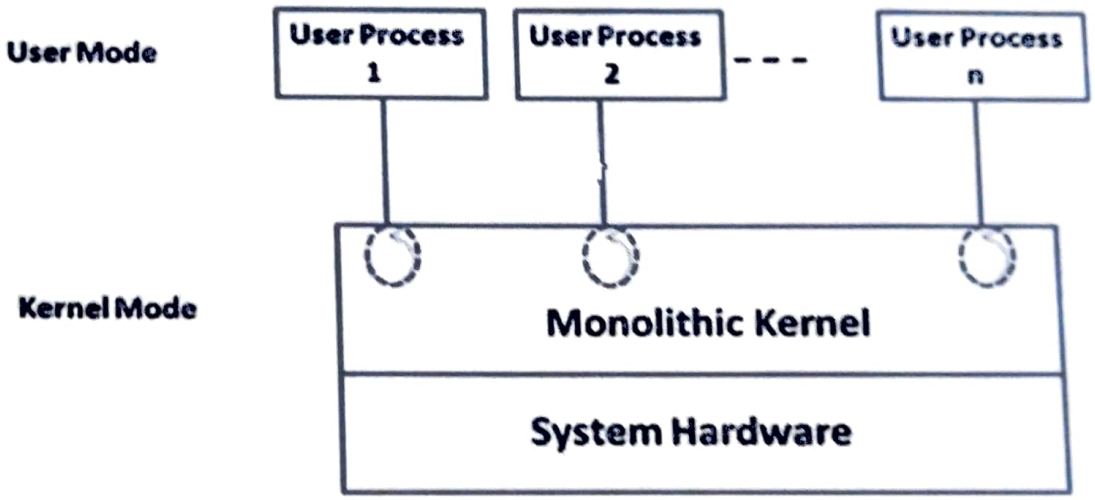
Time sharing requires an interactive computer system, which provides direct communication between the user and the system. The user gives instructions to the Operating System or to a program directly, using an input device such as a keyboard or a mouse, and waits for immediate results on an output device. Response time in interactive system should be short.

# Multi-user operating system

Multi-user operating system is a computer operating system that allows multiple users on different computers or terminals to access a single system with one OS on it. These programs are often quite complicated and must be able to properly manage the necessary tasks required by the different users connected to it. The users will typically be at terminals or computers that give them access to the system through a network, as well as other machines on the system such as printers. **Example** Ubuntu,Unix etc.

# Monolithic Kernel

- Monolithic means all in one piece.
- In monolithic kernel user services and kernel services are implemented under same address space, It increases size of the kernel thus increase size of operating system. Example: Linux




**Advantage**

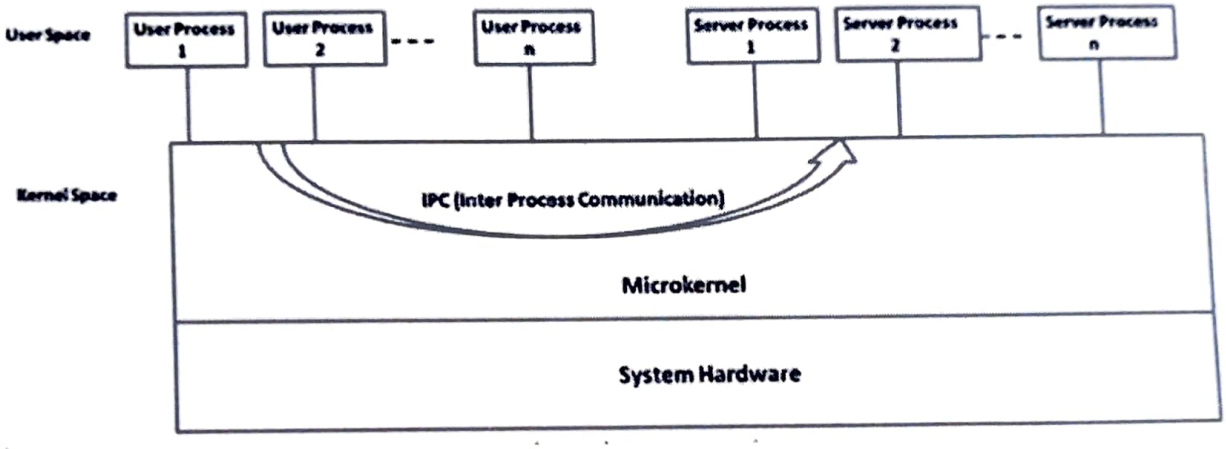
- ✓ Execution is faster.

**Disadvantages**

- ✓ Size is larger.
- ✓ Hard to extend.
- ✓ Hard to port.
- ✓ More prone to errors and bugs.

# Microkernel

- In microkernel, user services and kernel services are implemented in different address space, therefore it also reduces the size of kernel as well as the size of Operating system.
- In this architecture, only the most important services are present inside kernel and rest of the operating system services are present inside system application program.
- Communication between client process and services running in user address space is established through message passing thus reduce the speed of execution.
  Example: Mach OS




## Advantages

- ✓ Size is smaller.
- ✓ Easy to extend.
- ✓ Easy to port.
- ✓ Less prone to errors and bugs.

## Disadvantage

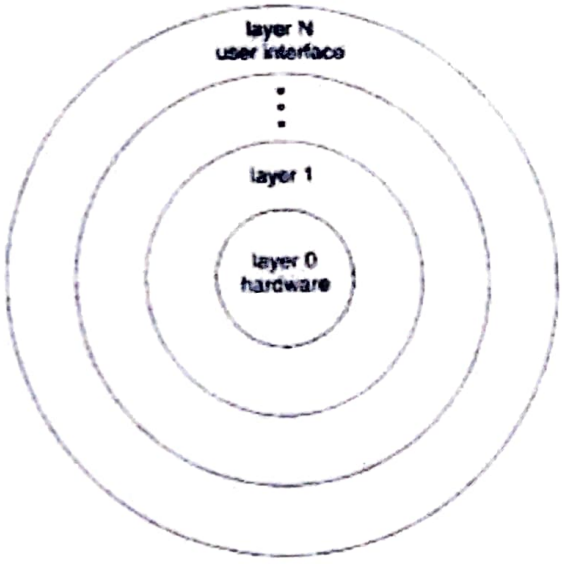
- ✓ Execution is slower.

### Difference between microkernel and monolithic kernel

| Parameter | Microkernel | Monolithic Kernel |
|---|---|---|
| Size | Small | Large |
| Speed of execution | ~~Faster~~ Slower | ~~Slower~~ faster |
| Extendibility | Easy | Hard |
| Portability | Easy | Hard |
| Error prone capability | Less | More |

# Layered Kernel

- In layered approach, the OS consists of several layers where each layer has a well-defined functionality and each layer is designed, coded and tested independently.
- Lowest layer interacts and deal with underlying hardware and top most layer provides an interface to the application programs and to the user processes.
- Each layer relies on the service of the layer below it. Communication takes place between adjacent layers.
- Each layer knows what services are provided by the layer above it but the details about how the services are provided are hidden.
- Different layers can be file management layer, process management layer, memory management layer etc.




A layered operating system.

**Advantages**

- ✓ Easier to add new feature or make change in one layer without affecting the other layer.
- ✓ Easy to add new layer when required.
- ✓ A particular layer can be debugged correctly or redesigned without affecting other layers.

**Disadvantages**

- ✓ Overhead incurred to maintain is more, if no of layers is more.
- ✓ If functionality of layers are not properly divided it may cause low system performance.

# Reentrant Kernel

- A reentrant kernel is one that consists of executable code stored in memory such that several processes can use this code at same time without hindering the working of other processes.
- Thus in reentrant kernel several processes may be executed in kernel mode at same time.
- Since multiple concurrent processes execute in kernel mode therefore they also share data that may affect the functionality of other processes. Hence, to avoid this condition, reentrant function and locking mechanism is used.
- Reentrant functions are one which allow processes to modify only the local variables and do not affect global data variables.
- In locking mechanism several processes can execute reentrant function concurrently but only one process at a time can execute non reentrant function, this will ensure that global data is modified by only one process at a time and thus each process has same copy of global data that is being shared among them.
  Example: UNIX.

# Multithreading

**Thread:** Light weight Process is called Thread.

### Thread Vs Process

| Thread | Process |
|---|---|
| Light weight Process | Heavy Weight Process |
| Thread Switching does not need interaction with OS | Process Switching needs interaction with OS |
| Thread can share Code segment, Data Section, File Descriptor, address space, heap etc | Processes own their separate Code segment, Data Section, File Descriptor, address space, heap etc |

### Threads has its own

| Register |
|---|
| Program Counter |
| Stack |

### Threads of same process share

| Data Section |
|---|
| Code Segment |
| Heap |
| Address Space |
| File Descriptor |
| Message Queue |

**Note:** Local variables of the Process are stored in Stack, Global variables are stored in Data Section and dynamically created variables are stored in Heap.

### User Level Thread vs. Kernel Level Thread

| User Level Thread | Kernel Level Thread |
|---|---|
| Implementation is easy | Implementation is difficult |
| OS don't recognized user level thread | Kernel Level Thread is recognized by OS |
| Context switch is fast because it requires no hardware support and less information to be stored during context switching. | Context switch is slow because it requires hardware support and more information to be stored during context switching. |
| If one user level thread performs blocking operation then entire process will be blocked. | If one kernel thread perform blocking operation then another thread can continue execution |

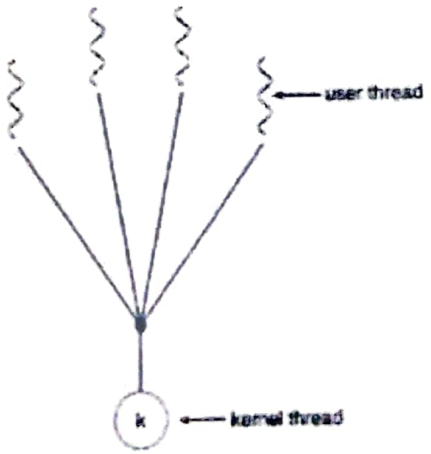
# Benefits of Multithreading Programming

1. **Responsiveness**: Multithreading allow a program to continuerunning even part of it is blocked or is performing a lengthy operation, thereby increasing responsiveness to user.
2. **Resource Sharing:** Since in multithreading multiple threads of a process share code and data that enables an application to have several different threads of activity within the same address space.
3. **Economy:** Allocating memory and resources for process creation is costly because thread share resource of the process to which they belong, it is more economical to create and context switch thread.
4. **Scalability:** Benefit of multithreading can be greatly increased in a multiprocessor architecture, where threads may be running on different processors.

# Multithreading Models

## Many to One Model

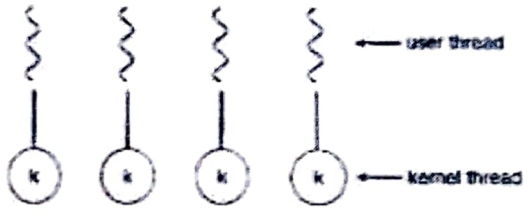
- It maps many user level threads to one kernel level thread.
- In this model Entire Process is blocked if a thread makes blocking system call.
- Since only one thread can access the kernel at a time multiple threads are unable to run in parallel on multiprocessor.




Many-to-one model.

## One to One Model

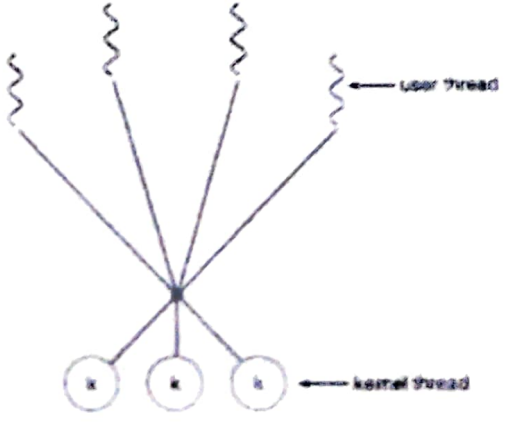
- It maps each user thread to a kernel thread.
- It provides more concurrency because it allows another thread to run in parallel on multiprocessor.
- It allows multiple threads to run in parallel on multiple processors.
- Creating a user thread requires creating the corresponding kernel thread.




One-to-one model.

## Many to Many Model

- It maps many user level threads to a smaller or equal number of kernel level threads.
- Developer can create as many user level threads as required.
- Kernel level threads can run in parallel on a multiprocessor.
- When a thread performs blocking system call the kernel can schedule another thread for execution.




Many-to-many model.