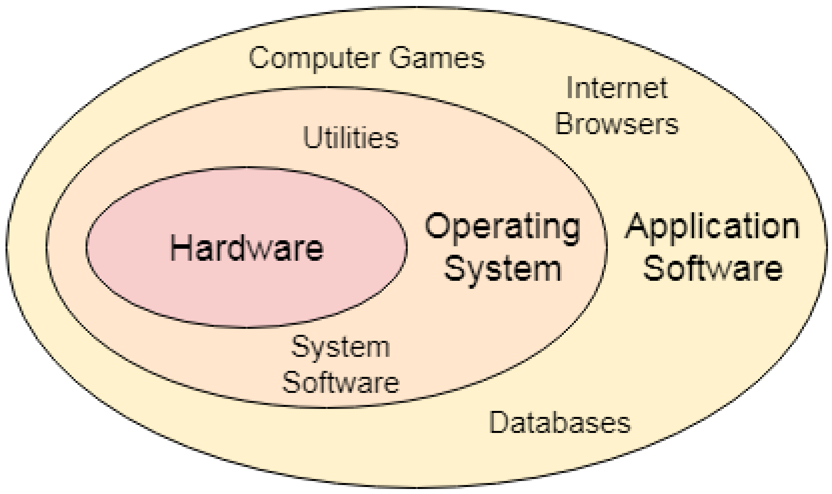
Unit 1

Introduction to Operating System

- Operating System Tutorial provides the basic and advanced concepts of operating system.
- Our Operating system tutorial is designed for beginners, professionals and GATE aspirants.
- Operating System can be defined as an interface between user and the hardware.
- It provides an environment to the user so that, the user can perform its task in convenient and efficient way.
- The Operating System Tutorial is divided into various parts based on its functions such as Process Management, Process Synchronization, Deadlocks and File Management

Operating System Definition and Function

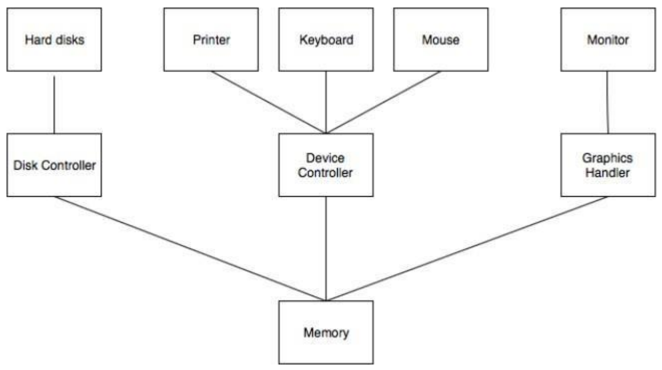
- In the Computer System (comprises of Hardware and software), Hardware can only understand machine code (in the form of 0 and 1) which doesn't make any sense to a naive user.
- We need a system which can act as an intermediary and manage all the processes and resources present in the system.



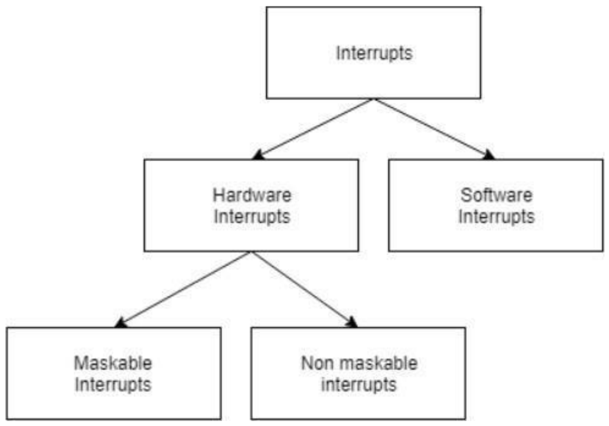
- An **Operating System** can be defined as an **interface between user and hardware**.
- It is responsible for the execution of all the processes, Resource Allocation management, File Management and many other tasks.
- The purpose of an operating system is to provide an environment in which a user can execute programs in convenient and efficient manner.

Computer System Organization

- The computer system is a combination of many parts such as peripheral devices, secondary memory, CPU etc. This can be explained more clearly using a diagram.



- The salient points about the above figure displaying Computer System Organization is
  - The I/O devices and the CPU both execute concurrently. Some of the processes are scheduled for the CPU and at the same time, some are undergoing input/output operations.
  - There are multiple device controllers, each in charge of a particular device such as keyboard, mouse, printer etc.
  - There is buffer available for each of the devices. The input and output data can be stored in these buffers.
  - The data is moved from memory to the respective device buffers by the CPU for I/O operations and then this data is moved back from the buffers to memory.
  - The device controllers use an interrupt to inform the CPU that I/O operation is completed.
- Interrupt Handling
  - An interrupt is a necessary part of Computer System Organization as it is triggered by hardware and software parts when they need immediate attention.
  - An interrupt can be generated by a device or a program to inform the operating system to halt its current activities and focus on something else.
  - The types of interrupts are better explained using the following diagram –

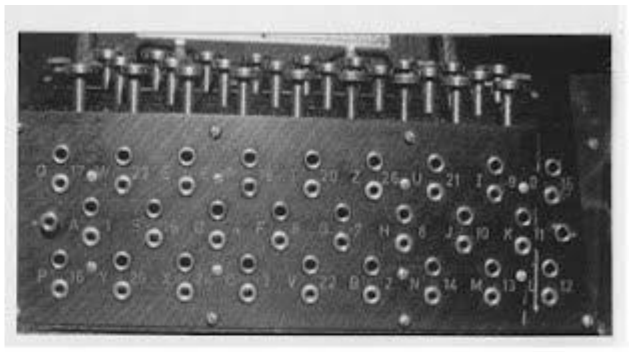


- Hardware and software interrupts are two types of interrupts. Hardware interrupts are triggered by hardware peripherals while software interrupts are triggered by software function calls.
- Hardware interrupts are of further two types. Maskable interrupts can be ignored or disabled by the CPU while this is not possible for non maskable interrupts.

<u>History of the Operating System</u>

The First Generation (1940's to early 1950's)

- When electronic computers were first introduced in the 1940's they were created without any operating systems.
- All programming was done in absolute machine language, often during this generation computers were generally used to solve simple math calculations; operating systems were not necessarily needed.



The Second Generation (1955-1965)

- The first operating system was introduced in the early 1950's, it was called GMOS and was created by General Motors for IBM's machine the 701.
- Operating systems in the 1950's were called single-stream batch processing systems because the data was submitted in groups.
- These new machines were called mainframes, and they were used by professional operators in large computer rooms.
- Since there was such as high price tag on these machines, only government agencies or large corporations were able to afford them.

The Third Generation (1965-1980)

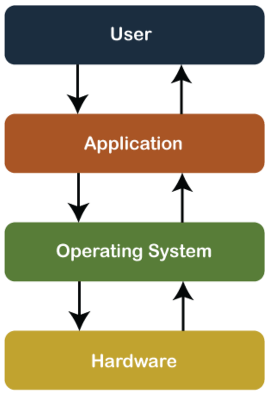
- By the late 1960's operating systems designers were able to develop the system of multiprogramming in which a computer program will be able to perform multiple jobs at the same time.
- The introduction of multiprogramming was a major part in the development of operating systems because it allowed a CPU to be busy nearly 100 percent of the time that it was in operation.
- These microcomputers help create a whole new industry and the development of more PDP's.
- These PDP's helped lead to the creation of personal computers which are created in the fourth generation.

The Fourth Generation (1980-Present Day)

- The fourth generation of operating systems saw the creation of personal computing.
- Although these computers were very similar to the minicomputers developed in the third generation, personal computers cost a very small fraction of what minicomputers cost.
- A personal computer was so affordable that it made it possible for a single individual could be able to own one for personal use while minicomputers where still at such a high price that only corporations could afford to have them.
- One of the major factors in the creation of personal computing was the birth of Microsoft and the Windows operating system.
- The windows Operating System was created in 1975 when Paul Allen and Bill Gates had a vision to take personal computing to the next level.
- They introduced the MS-DOS in 1981 although it was effective it created much difficulty for people who tried to understand its cryptic commands.
- Windows went on to become the largest operating system used in technology today with releases of Windows 95, Windows 98, Windows XP (Which is currently the most used operating system to this day), and their newest operating system Windows 7.
- Along with Microsoft, Apple is the other major operating system created in the 1980's. Steve Jobs, co founder of Apple, created the Apple Macintosh which was a huge success due to the fact that it was so user friendly.
- Windows developments throughout the later years were influenced by the Macintosh and it created a strong competition between the two companies.
- Today all of our electronic devices run off of operating systems, from our computers and smart phones, to ATM machines and motor vehicles. And as technology advances, so do operating systems.

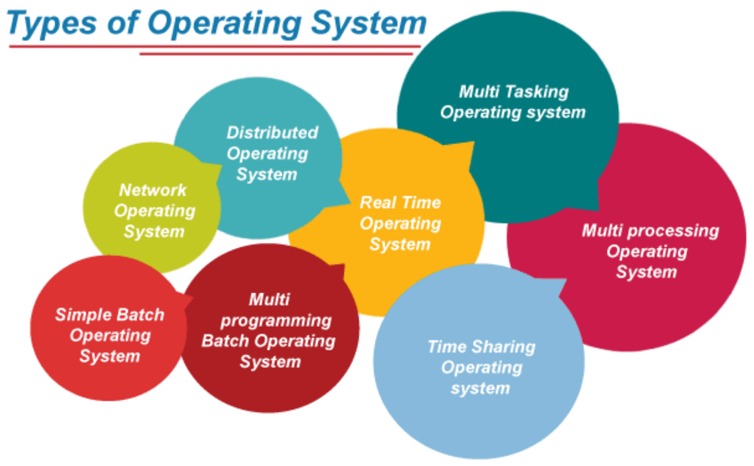
Operating System

- The operating system is a system program that serves as an interface between the computing system and the end-user.
- Operating systems create an environment where the user can run any programs or communicate with software or applications in a comfortable and well-organized way.
- Furthermore, an operating is a software program that manages and controls the execution of application programs, software resources and computer hardware.
- It also helps manage the software/hardware resource, such as file management, memory management, input/ output and many peripheral devices like a disk drive, printers, etc. These are the popular operating system: Mac OS, VMS, OS/400 etc.



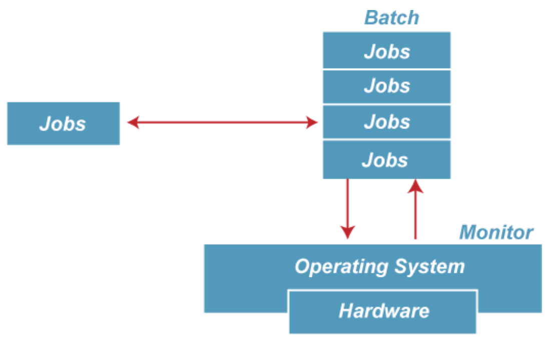
Types of Operating Systems (OS)

- An operating system is a well-organized collection of programs that manages the computer hardware.
- It is a type of system software that is responsible for the smooth functioning of the computer system.
- It is a type of system software that is responsible for the smooth functioning of the computer system.


Types of Operating System
Multi Tasking Operating system
Distributed Operating System
Network Operating System
Real Time Operating System
Multi processing Operating System
Simple Batch Operating System
Multi programming Batch Operating System
Time Sharing Operating system

Batch Operating System

- In the 1970s, Batch processing was very popular.
- In this technique, similar types of jobs were batched together and executed in time. People were used to having a single computer which was called a mainframe.
- In Batch operating system, access is given to more than one person; they submit their respective jobs to the system for the execution.
- The system put all of the jobs in a queue on the basis of first come first serve and then executes the jobs one by one. The users collect their respective output when all the jobs get executed.



- The purpose of this operating system was mainly to transfer control from one job to another as soon as the job was completed. It contained a small set of programs called the resident monitor that always resided in one part of the main memory.
- The remaining part is used for servicing jobs.

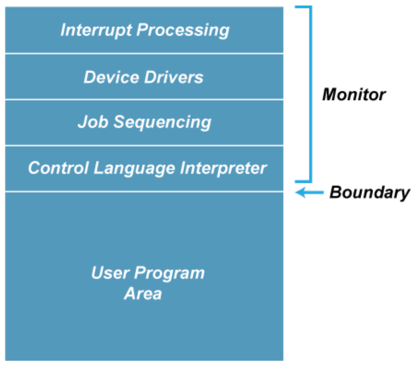
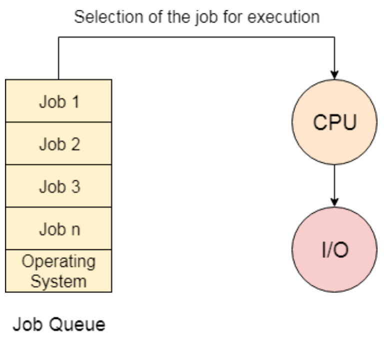


Figure: Memory Layout of the resident monitor

- Advantages of Batch OS
    - The use of a resident monitor improves computer efficiency as it eliminates CPU time between two jobs.
- Disadvantages of Batch OS
- **Starvation**
    - Batch processing suffers from starvation.
    - **For Example:**



There are five jobs J1, J2, J3, J4, and J5, present in the batch. If the execution time of J1 is very high, then the other four jobs will never be executed, or they will have to wait for a very long time. Hence the other processes get starved.

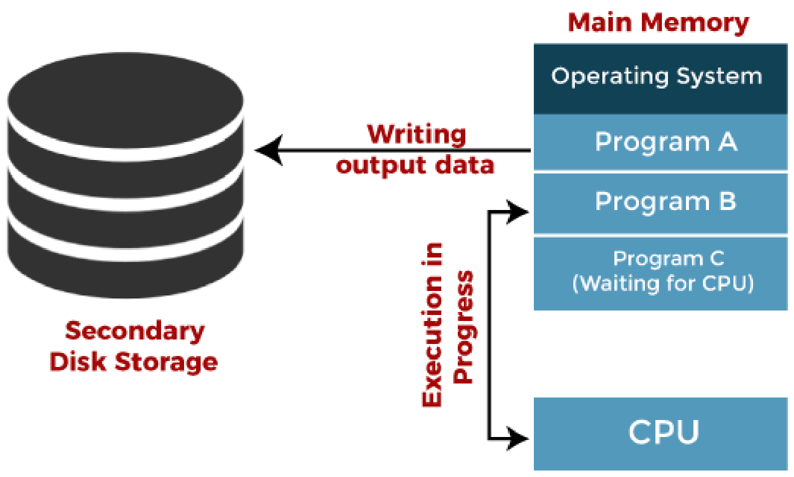
Not Interactive

- Batch Processing is not suitable for jobs that are dependent on the user's input.
- If a job requires the input of two numbers from the console, then it will never get it in the batch processing scenario since the user is not present at the time of execution.

Multiprogramming Operating System

- Multiprogramming is an extension to batch processing where the CPU is always kept busy.

- Each process needs two types of system time: CPU time and IO time.
- In a multiprogramming environment, when a process does its I/O, The CPU can start the execution of other processes. Therefore, multiprogramming improves the efficiency of the system.



Jobs in multiprogramming system

<u>Advantages of Multiprogramming OS</u>

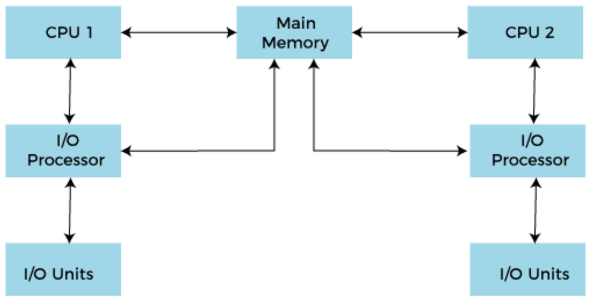
- Throughout the system, it increased as the CPU always had one program to execute.
- Response time can also be reduced.

<u>Disadvantages of Multiprogramming OS</u>

- Multiprogramming systems provide an environment in which various systems resources are used efficiently, but they do not provide any user interaction with the computer system.

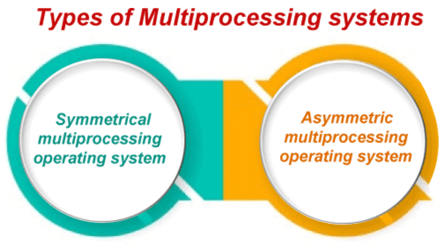
<u>Multiprocessing Operating System</u>

- In Multiprocessing, Parallel computing is achieved.
- There are more than one processors present in the system which can execute more than one process at the same time.
- This will increase the throughput of the system.



Working of Multiprocessor System

- In Multiprocessing, Parallel computing is achieved.
- More than one processor present in the system can execute more than one process simultaneously, which will increase the throughput of the system.


Types of Multiprocessing systems
Symmetrical multiprocessing operating system
Asymmetric multiprocessing operating system

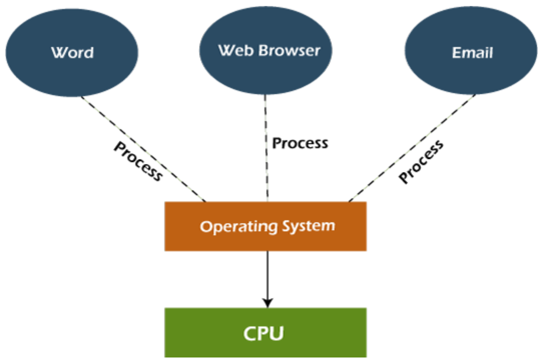
Advantages of Multiprocessing operating system:

- **Increased reliability:**
  - Due to the multiprocessing system, processing tasks can be distributed among several processors.
  - This increases reliability as if one processor fails, the task can be given to another processor for completion.
- **Increased throughout:**
  - As several processors increase, more work can be done in less.

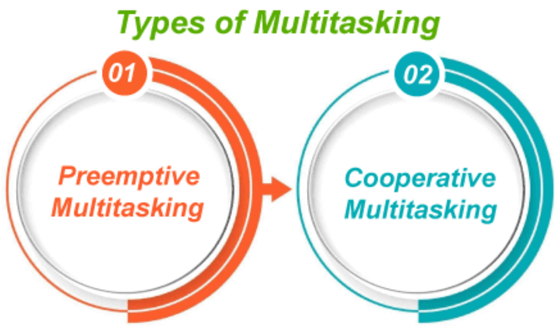
Disadvantages of Multiprocessing operating System

- Multiprocessing operating system is more complex and sophisticated as it takes care of multiple CPUs simultaneously.

Multitasking Operating System



- The multitasking operating system is a logical extension of a multiprogramming system that enables **multiple** programs simultaneously.
- It allows a user to perform more than one computer task at the same time.



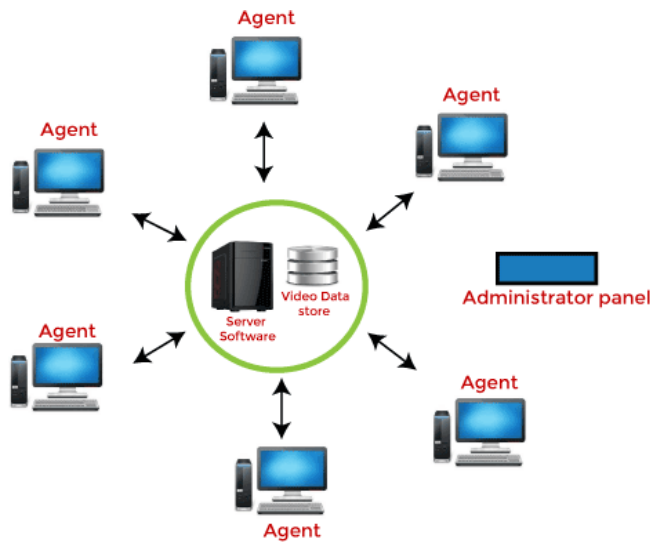
Advantages of Multitasking operating system

- This operating system is more suited to supporting multiple users simultaneously.
- The multitasking operating systems have well-defined memory management.

Disadvantages of Multitasking operating system

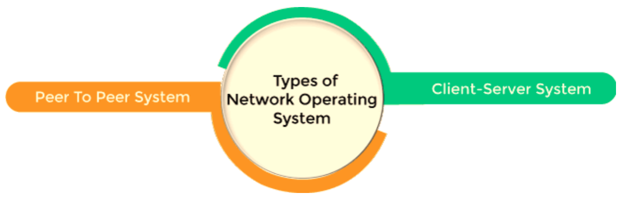
- The multiple processors are busier at the same time to complete any task in a multitasking environment, so the CPU generates more heat.

Network Operating System



Network Operating Systems

- An Operating system, which includes software and associated protocols to communicate with other computers via a network conveniently and cost-effectively, is called Network Operating System.



Advantages of Network Operating System

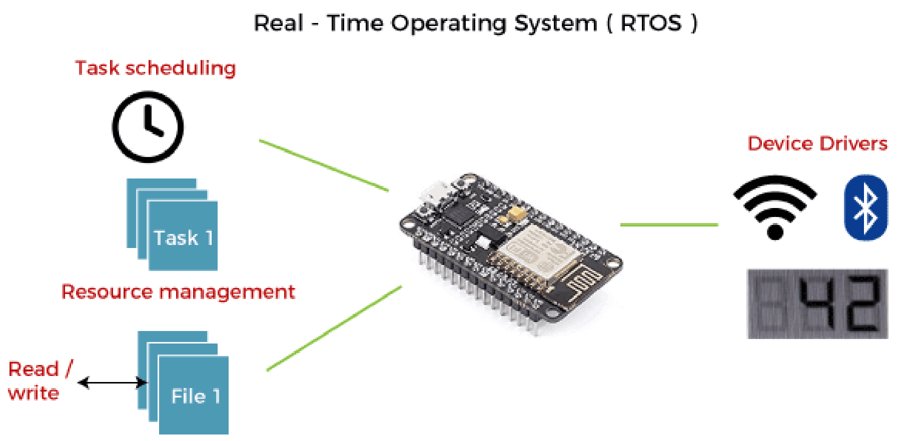
- In this type of operating system, network traffic reduces due to the division between clients and the server.
- This type of system is less expensive to set up and maintain.

Disadvantages of Network Operating System

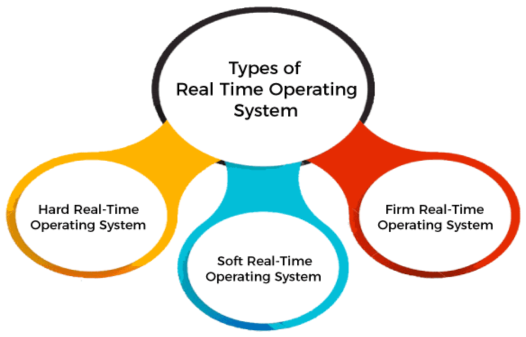
- In this type of operating system, the failure of any node in a system affects the whole system.
- Security and performance are important issues. So trained network administrators are required for network administration.

Real Time Operating System

- In Real-Time Systems, each job carries a certain deadline within which the job is supposed to be completed, otherwise, the huge loss will be there, or even if the result is produced, it will be completely useless.



- The Application of a Real-Time system exists in the case of military applications, if you want to drop a missile, then the missile is supposed to be dropped with a certain precision.


Types of Real Time Operating System
Hard Real-Time Operating System
Soft Real-Time Operating System
Firm Real-Time Operating System

Advantages of Real-time operating system:

- Easy to layout, develop and execute real-time applications under thereal-time operating system.
- In a Real-time operating system, the maximum utilization of devices and systems.

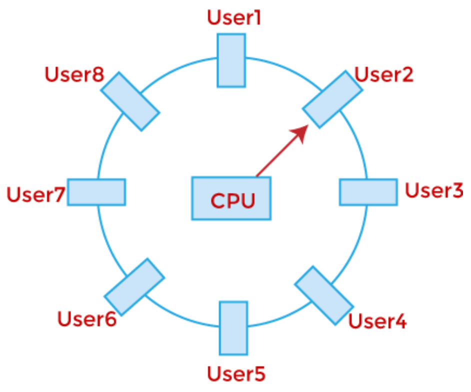
Disadvantages of Real-time operating system:

- Real-time operating systems are very costly to develop.

- Real-time operating systems are very complex and can consume critical CPU cycles.

Time-Sharing Operating System

- In the Time Sharing operating system, computer resources are allocated in a time-dependent fashion to several programs simultaneously.
- Thus it helps to provide a large number of user's direct access to the main computer.
- It is a logical extension of multiprogramming. In time-sharing, the CPU is switched among multiple programs given by different users on a scheduled basis.



Timesharing in case of 8 users

- A time-sharing operating system allows many users to be served simultaneously, so sophisticated CPU scheduling schemes and Input/output management are required.
- Time-sharing operating systems are very difficult and expensive to build.

Advantages of Time Sharing Operating System

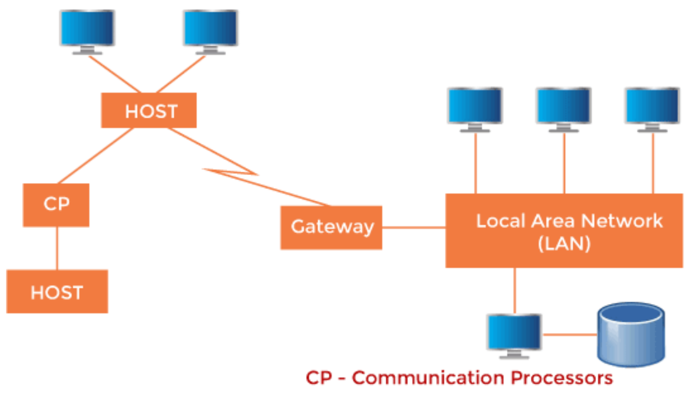
- The time-sharing operating system provides effective utilization and sharing of resources.
- This system reduces CPU idle and response time.

Disadvantages of Time Sharing Operating System

- Data transmission rates are very high in comparison to other methods.
- Security and integrity of user programs loaded in memory and data need to be maintained as many users access the system at the same time.

Distributed Operating System

- The Distributed Operating system is not installed on a single machine, it is divided into parts, and these parts are loaded on different machines.
- A part of the distributed Operating system is installed on each machine to make their communication possible.
- Distributed Operating systems are much more complex, large, and sophisticated than Network operating systems because they also have to take care of varying networking protocols.



A Typical View of a Distributed System

Advantages of Distributed Operating System

- The distributed operating system provides sharing of resources.
- This type of system is fault-tolerant.

Disadvantages of Distributed Operating System

- Protocol overhead can dominate computation cost.

Functions of Operating System

- Memory Management
- Processor Management
- Device Management
- File Management
- Network Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

Memory Management

- Memory management refers to management of Primary Memory or Main Memory.
- Main memory is a large array of words or bytes where each word or byte has its own address.
- Main memory provides a fast storage that can be accessed directly by the CPU.
- For a program to be executed, it must in the main memory. An Operating System does the following activities for memory management –
    - Keeps tracks of primary memory, i.e., what part of it are in use by whom, what part are not in use.
    - In multiprogramming, the OS decides which process will get memory when and how much.
    - Allocates the memory when a process requests it to do so.
    - De-allocates the memory when a process no longer needs it or has been terminated.

Processor Management

- In multiprogramming environment, the OS decides which process gets the processor when and for how much time.
- This function is called **process scheduling**.
- An Operating System does the following activities for processor management –
    - Keeps tracks of processor and status of process. The program responsible for this task is known as **traffic controller**.
    - Allocates the processor (CPU) to a process.
    - De-allocates processor when a process is no longer required.

Device Management

- An Operating System manages device communication via their respective drivers.
- It does the following activities for device management –
    - Keeps tracks of all devices. Program responsible for this task is known as the **I/O controller**.
    - Decides which process gets the device when and for how much time.
    - Allocates the device in the efficient way.
    - De-allocates devices.

File Management

- A file system is normally organized into directories for easy navigation and usage.
- These directories may contain files and other directions.
- An Operating System does the following activities for file management –
    - Keeps track of information, location, uses, status etc. The collective facilities are often known as **file system**.
    - Decides who gets the resources.
    - Allocates the resources.
    - De-allocates the resources.

Other Important Activities

Following are some of the important activities that an Operating System performs –

- **Security –**
    - By means of password and similar other techniques, it prevents unauthorized access to programs and data.
- **Control over system performance –**
    - Recording delays between request for a service and response from the system.
- **Job accounting –**
    - Keeping track of time and resources used by various jobs and users.
- **Error detecting aids –**
    - Production of dumps, traces, error messages, and other debugging and error detecting aids.
- **Coordination between other software's and users –**
    - Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

System Calls in Operating System (OS)

- A system call is a way for a user program to interface with the operating system.
- The program requests several services, and the OS responds by invoking a series of system calls to satisfy the request.
- A system call can be written in assembly language or a high-level language like **C** or **Pascal**.
- System calls are predefined functions that the operating system may directly invoke if a high-level language is used.

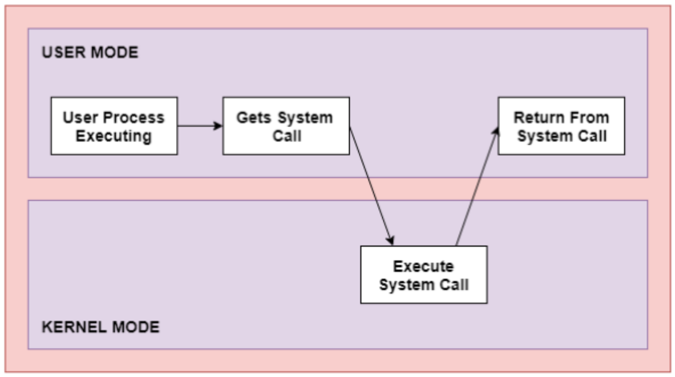
Definition System Call

- A system call is a method for a computer program to request a service from the kernel of the operating system on which it is running.
- A system call is a method of interacting with the operating system via programs.
- A system call is a request from computer software to an operating system's kernel.

The **Application Program Interface (API)** connects the operating system's functions to user programs.

It acts as a link between the operating system and a process, allowing user-level programs to request operating system services.

The kernel system can only be accessed using system calls. System calls are required for any programs that use resources.



How is system calls made

- When computer software needs to access the operating system's kernel, it makes a system call.
- The system call uses an API to expose the operating system's services to user programs.
- It is the only method to access the kernel system.
- All programs or processes that require resources for execution must use system calls, as they serve as an interface between the operating system and user programs.

Below are some examples of how a system call varies from a user function.

1. A system call function may create and use kernel processes to execute the asynchronous processing.
2. A system call has greater authority than a standard subroutine. A system call with kernel-mode privilege executes in the kernel protection domain.
3. System calls are not permitted to use shared libraries or any symbols that are not present in the kernel protection domain.
4. The code and data for system calls are stored in global kernel memory.

Need of system call in OS:

There are various situations where you must require system calls in the operating system. Following of the situations are as follows:

1. It is must require when a file system wants to create or delete a file.
2. Network connections require the system calls to sending and receiving data packets.
3. If you want to read or write a file, you need to system calls.
4. If you want to access hardware devices, including a printer, scanner, you need a system call.
5. System calls are used to create and manage new processes.

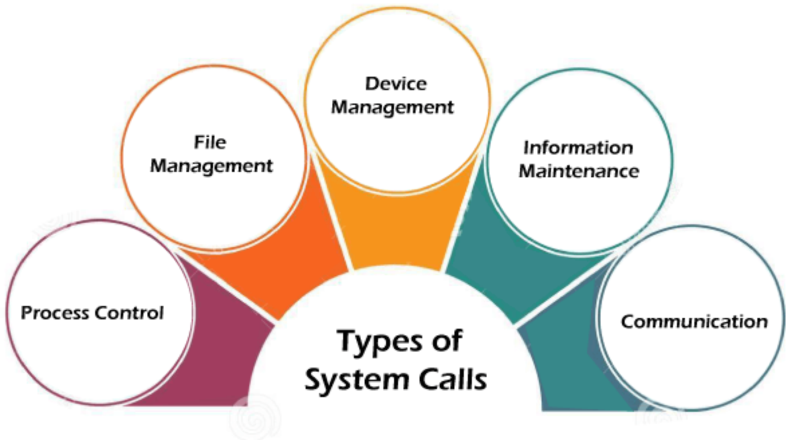
System Call Working:

- The Applications run in an area of memory known as user space.
- A system call connects to the operating system's kernel, which executes in kernel space.
- When an application creates a system call, it must first obtain permission from the kernel.
- It achieves this using an interrupt request, which pauses the current process and transfers control to the kernel.
- If the request is permitted, the kernel performs the requested action, like creating or deleting a file.
- As input, the application receives the kernel's output.
- The application resumes the procedure after the input is received.
- When the operation is finished, the kernel returns the results to the application and then moves data from kernel space to user space in memory.

- A simple system call may take few nanoseconds to provide the result, like retrieving the system date and time.
- A more complicated system call, such as connecting to a network device, may take a few seconds.
- Most operating systems launch a distinct kernel thread for each system call to avoid bottlenecks.
- Modern operating systems are multi-threaded, which means they can handle various system calls at the same time.

Types of System Calls

There are commonly five types of system calls. These are as follows:



1. **Process Control**
2. **File Management**
3. **Device Management**
4. **Information Maintenance**
5. **Communication**

Process Control

- Process control is the system call that is used to direct the processes. Some process control examples include creating, load, abort, end, execute, process, terminate the process, etc.

File Management

- File management is a system call that is used to handle the files. Some file management examples include creating files, delete files, open, close, read, write, etc.

Device Management

- Device management is a system call that is used to deal with devices. Some examples of device management include read, device, write, get device attributes, release device, etc.

Information Maintenance

- Information maintenance is a system call that is used to maintain information.
- There are some examples of information maintenance, including getting system data, set time or date, get time or date, set system data, etc.

Communication

- Communication is a system call that is used for communication.
- There are some examples of communication, including creates, delete communication connections, send, receive messages, etc.

Operating System Structure

- An operating system is a design that enables user application programs to communicate with the hardware of the machine.
- The operating system should be built with the utmost care because it is such a complicated structure and should be simple to use and modify.
- Partially developing the operating system is a simple approach to accomplish this.
- Each of these components needs to have distinct inputs, outputs, and functionalities.

  - Simple Structure
  - Monolithic Structure
  - Layered Approach Structure
  - Micro-Kernel Structure
  - Exo-Kernel Structure
  - Virtual Machines

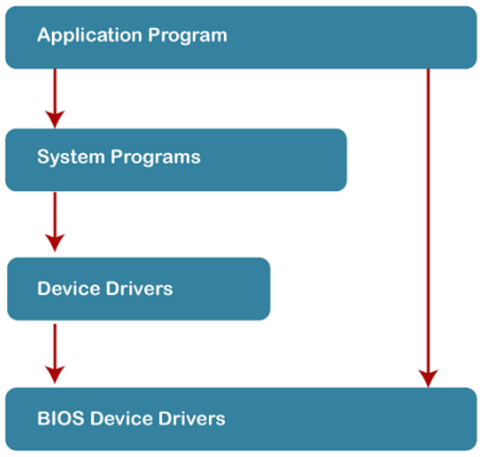
SIMPLE STRUCTURE

- It is the most straightforward operating system structure, but it lacks definition and is only appropriate for usage with tiny and restricted systems.
- Since the interfaces and degrees of functionality in this structure are clearly defined, programs are able to access I/O routines, which may result in unauthorized access to I/O procedures.

This organizational structure is used by the MS-DOS operating system:

- There are four layers that make up the MS-DOS operating system, and each has its own set of features.
- These layers include ROM BIOS device drivers, MS-DOS device drivers, application programs, and system programs.
- The MS-DOS operating system benefits from layering because each level can be defined independently and, when necessary, can interact with one another.
- If the system is built in layers, it will be simpler to design, manage, and update. Because of this, simple structures can be used to build constrained systems that are less complex.
- When a user program fails, the operating system as whole crashes.
- Because MS-DOS systems have a low level of abstraction, programs and I/O procedures are visible to end users, giving them the potential for unwanted access.

The following figure illustrates layering in simple structure



Advantages of Simple Structure:

- Because there are only a few interfaces and levels, it is simple to develop.
- Because there are fewer layers between the hardware and the applications, it offers superior performance.

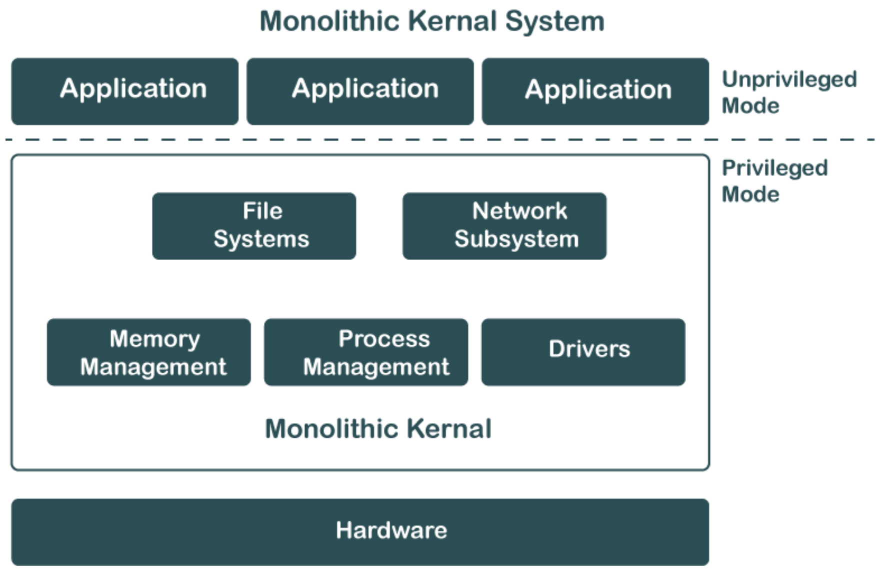
Disadvantages of Simple Structure:

- The entire operating system breaks if just one user program malfunctions.
- Since the layers are interconnected, and in communication with one another, there is no abstraction or data hiding.
- The operating system's operations are accessible to layers, which can result in data tampering and system failure.

MONOLITHIC STRUCTURE

- The monolithic operating system controls all aspects of the operating system's operation, including file management, memory management, device management, and operational operations.
- The core of an operating system for computers is called the kernel (OS).
- All other System components are provided with fundamental services by the kernel.
- The operating system and the hardware use it as their main interface.
- When an operating system is built into a single piece of hardware, such as a keyboard or mouse, the kernel can directly access all of its resources.

- The monolithic operating system is often referred to as the monolithic kernel.
- Multiple programming techniques such as batch processing and time-sharing increase a processor's usability.
- Working on top of the operating system and under complete command of all hardware, the monolithic kernel performs the role of a virtual computer.
- This is an old operating system that was used in banks to carry out simple tasks like batch processing and time-sharing, which allows numerous users at different terminals to access the Operating System.
- The following diagram represents the monolithic structure:



Advantages of Monolithic Structure:

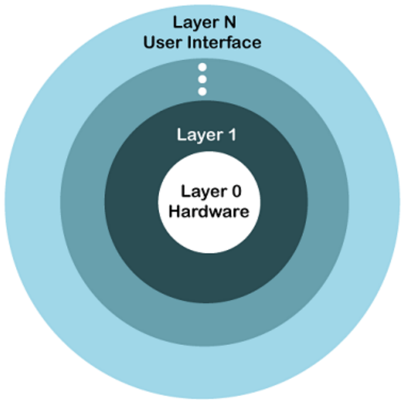
- Because layering is unnecessary and the kernel alone is responsible for managing all operations, it is easy to design and execute.
- Due to the fact that functions like memory management, file management, process scheduling, etc., are implemented in the same address area, the monolithic kernel runs rather quickly when compared to other systems. Utilizing the same address speeds up and reduces the time required for address allocation for new processes.

Disadvantages of Monolithic Structure:

- The monolithic kernel's services are interconnected in address space and have an impact on one another, so if any of them a malfunction, the entire system does as well.
- It is not adaptable. Therefore, launching a new service is difficult.

LAYERED STRUCTURE

- The OS is separated into layers or levels in this kind of arrangement. Layer 0 (the lowest layer) contains the hardware, and layer 1 (the highest layer) contains the user interface (layer N).
- These layers are organized hierarchically, with the top-level layers making use of the capabilities of the lower-level ones.
- The functionalities of each layer are separated in this method, and abstraction is also an option.
- Because layered structures are hierarchical, debugging is simpler, therefore all lower-level layers are debugged before the upper layer is examined.
- As a result, the present layer alone has to be reviewed since all the lower layers have already been examined.
- The image below shows how OS is organized into layers:



Advantages of Layered Structure:

- Work duties are separated since each layer has its own functionality, and there is some amount of abstraction.

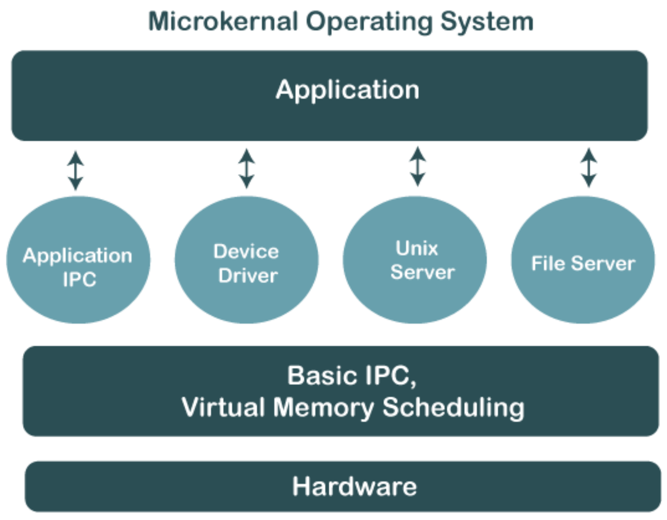
- Debugging is simpler because the lower layers are examined first, followed by the top layers.

Disadvantages of Layered Structure:

- Performance is compromised in layered structures due to layering.
- Construction of the layers requires careful design because upper layers only make use of lower layers' capabilities.

MICRO-KERNEL STRUCTURE

- The operating system is created using a micro-kernel framework that strips the kernel of any unnecessary parts.
- Systems and user applications are used to implement these optional kernel components.
- So, Micro-Kernels is the name given to these systems that have been developed.
- Each Micro-Kernel is created separately and is kept apart from the others.
- As a result, the system is now more trustworthy and secure.
- If one Micro-Kernel malfunctions, the remaining operating system is unaffected and continues to function normally.
- The image below shows Micro-Kernel Operating System Structure:



Advantages of Micro-Kernel Structure:

- It enables portability of the operating system across platforms.
- Due to the isolation of each Micro-Kernel, it is reliable and secure.

- The reduced size of Micro-Kernels allows for successful testing.
- The remaining operating system remains unaffected and keeps running properly even if a component or Micro-Kernel fails.

Disadvantages of Micro-Kernel Structure:

- The performance of the system is decreased by increased inter-module communication.
- The construction of a system is complicated.

EXOKERNEL

- An operating system called Exokernel was created at MIT with the goal of offering application-level management of hardware resources.
- The exokernel architecture's goal is to enable application-specific customization by separating resource management from protection. Exokernel size tends to be minimal due to its limited operability.
- Because the OS sits between the programs and the actual hardware, it will always have an effect on the functionality, performance, and breadth of the apps that are developed on it.
- By rejecting the idea that an operating system must offer abstractions upon which to base applications, the exokernel operating system makes an effort to solve this issue.
- The goal is to give developers as few restriction on the use of abstractions as possible while yet allowing them the freedom to do so when necessary.
- Because of the way the exokernel architecture is designed, a single tiny kernel is responsible for moving all hardware abstractions into unreliable libraries known as library operating systems.
- Exokernels differ from micro- and monolithic kernels in that their primary objective is to prevent forced abstraction.

Exokernel operating systems have a number of features, including:

- Enhanced application control support.
- Splits management and security apart.
- A secure transfer of abstractions is made to an unreliable library operating system.
- Brings up a low-level interface.
- Operating systems for libraries provide compatibility and portability.

Advantages of Exokernel Structure:

- Application performance is enhanced by it.
- Accurate resource allocation and revocation enable more effective utilization of hardware resources.
- New operating systems can be tested and developed more easily.
- Every user-space program is permitted to 28ustomi its own 28ustomized memory management.

Disadvantages of Exokernel Structure:

- A decline in consistency
- Exokernel interfaces have a complex architecture.

VIRTUAL MACHINES (VMs)

- The hardware of our personal computer, including the CPU, disc drives, RAM, and NIC (Network Interface Card), is abstracted by a virtual machine into a variety of various execution contexts based on our needs, giving us the impression that each execution environment is a separate computer.
- A virtual box is an example of it.
- Using CPU scheduling and virtual memory techniques, an operating system allows us to execute multiple processes simultaneously while giving the impression that each one is using a separate processor and virtual memory.
- System calls and a file system are examples of extra functionalities that a process can have that the hardware is unable to give.
- Instead of offering these extra features, the virtual machine method just offers an interface that is similar to that of the most fundamental hardware.
- A virtual duplicate of the computer system underneath is made available to each process.
- We can develop a virtual machine for a variety of reasons, all of which are fundamentally connected to the capacity to share the same underlying hardware while concurrently supporting various execution environments, i.e., various operating systems.
- Disk systems are the fundamental problem with the virtual machine technique. If the actual machine only has three-disc drives but needs to host seven virtual machines, let's imagine that.
- It is obvious that it is impossible to assign a disc drive to every virtual machine because the program that creates virtual machines would

require a sizable amount of disc space in order to offer virtual memory and spooling.

- The provision of virtual discs is the solution.
- The result is that users get their own virtual machines.
- They can then use any of the operating systems or software programs that are installed on the machine below.
- Virtual machine software is concerned with programming numerous virtual machines simultaneously into a physical machine; it is not required to take into account any user-support software.
- With this configuration, it may be possible to break the challenge of building an interactive system for several users into two manageable chunks.

Advantages of Virtual Machines:

- Due to total isolation between each virtual machine and every other virtual machine, there are no issues with security.
- A virtual machine may offer an architecture for the instruction set that is different from that of actual computers.
- Simple availability, accessibility, and recovery convenience.

Disadvantages of Virtual Machines:

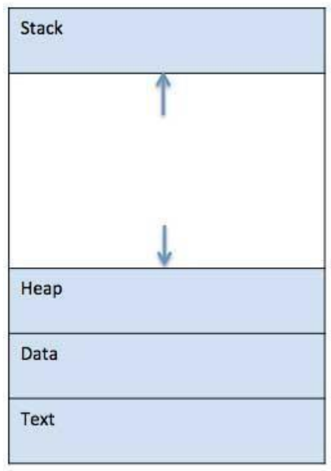
- Depending on the workload, operating numerous virtual machines simultaneously on a host computer may have an adverse effect on one of them.
- When it comes to hardware access, virtual computers are less effective than physical ones.

Process Management

Process:

- A process is basically a program in execution. The execution of a process must progress in a sequential fashion.
- A process is defined as an entity which represents the basic unit of work to be implemented in the system.

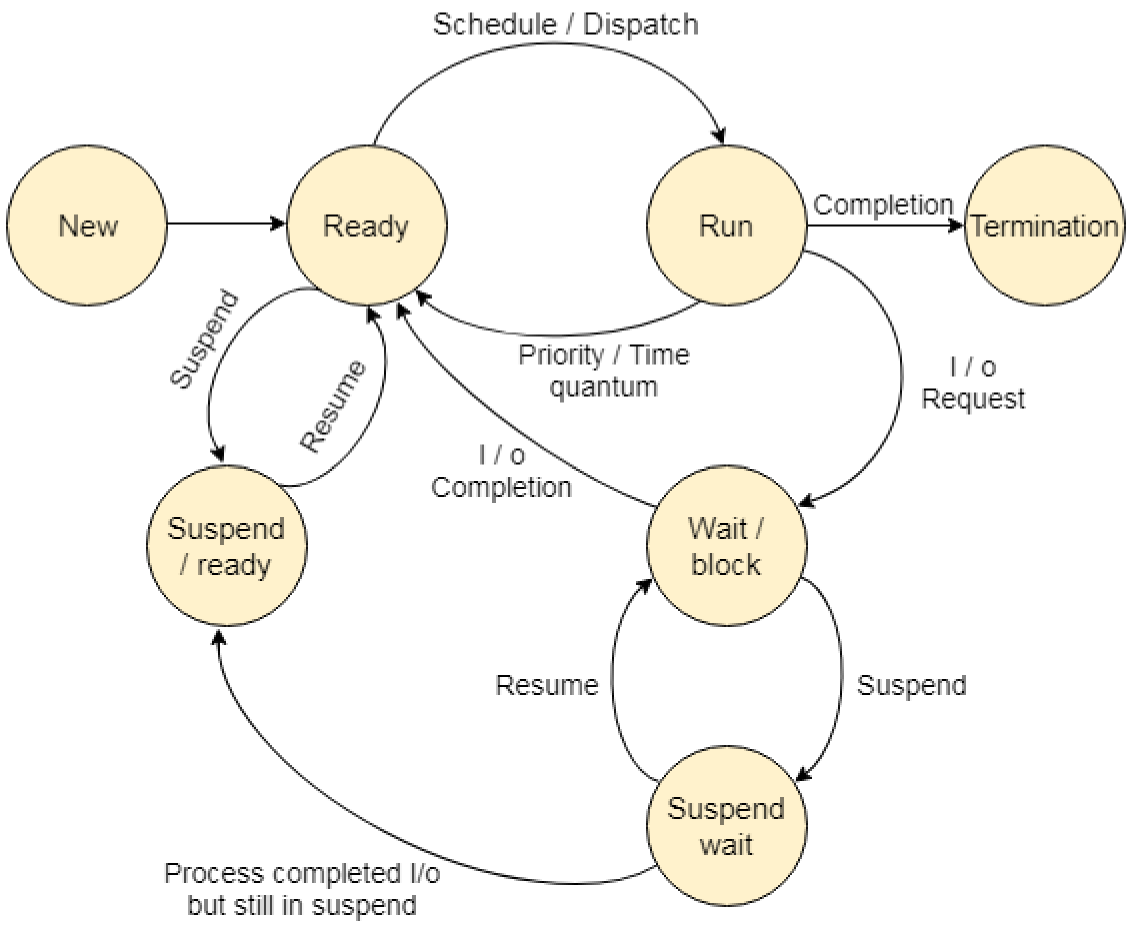
- To put it in simple terms, we write our computer programs in a text file and when we execute this program, it becomes a process which performs all the tasks mentioned in the program.
- When a program is loaded into the memory and it becomes a process, it can be divided into four sections ─ stack, heap, text and data. The following image shows a simplified layout of a process inside main memory –



S.N.	Component & Description
1	**Stack** The process Stack contains the temporary data such as method/function parameters, return address and local variables.
2	**Heap** This is dynamically allocated memory to a process during its run time.
3	**Text** This includes the current activity represented by the value of Program Counter and the contents of the processor's registers.
4	**Data** This section contains the global and static variables.

Process States

State Diagram



- The process, from its creation to completion, passes through various states. The minimum number of states is five.
- The names of the states are not standardized although the process may be in one of the following states during execution.

1. New

- A program which is going to be picked up by the OS into the main memory is called a new process.

2. Ready

- Whenever a process is created, it directly enters in the ready state, in which, it waits for the CPU to be assigned.

- The OS picks the new processes from the secondary memory and put all of them in the main memory.
- The processes which are ready for the execution and reside in the main memory are called ready state processes.
- There can be many processes present in the ready state.

3. Running

- One of the processes from the ready state will be chosen by the OS depending upon the scheduling algorithm.
- Hence, if we have only one CPU in our system, the number of running processes for a particular time will always be one.
- If we have n processors in the system then we can have n processes running simultaneously.

4. Block or wait

- From the Running state, a process can make the transition to the block or wait state depending upon the scheduling algorithm or the intrinsic behavior of the process.
- When a process waits for a certain resource to be assigned or for the input from the user then the OS move this process to the block or wait state and assigns the CPU to the other processes.

5. Completion or termination

- When a process finishes its execution, it comes in the termination state. All the context of the process (Process Control Block) will also be deleted the process will be terminated by the Operating system.

6. Suspend ready

- A process in the ready state, which is moved to secondary memory from the main memory due to lack of the resources (mainly primary memory) is called in the suspend ready state.
- If the main memory is full and a higher priority process comes for the execution then the OS have to make the room for the process in the main memory by throwing the lower priority process out into the secondary memory.
- The suspend ready processes remain in the secondary memory until the main memory gets available

7. Suspend wait

- Instead of removing the process from the ready queue, it's better to remove the blocked process which is waiting for some resources in the main memory.
- Since it is already waiting for some resource to get available hence it is better if it waits in the secondary memory and makes room for the higher priority process.
- These processes complete their execution once the main memory gets available and their wait is finished.

Operations on the Process

1. Creation

- Once the process is created, it will be ready and come into the ready queue (main memory) and will be ready for the execution.

2. Scheduling

- Out of the many processes present in the ready queue, the Operating system chooses one process and start executing it. Selecting the process which is to be executed next, is known as scheduling.

3. Execution

- Once the process is scheduled for the execution, the processor starts executing it.
- Process may come to the blocked or wait state during the execution then in that case the processor starts executing the other processes.

4. Deletion/killing

- Once the purpose of the process gets over then the OS will kill the process.
- The Context of the process (PCB) will be deleted and the process gets terminated by the Operating system.

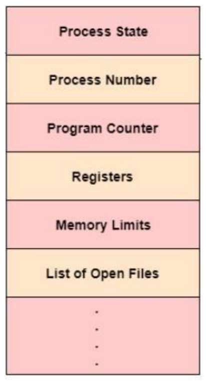
Process Control Block (PCB)

- Process Control Block is a data structure that contains information of the process related to it.
- The process control block is also known as a task control block, entry of the process table, etc.

- It is very important for process management as the data structuring for processes are done in terms of the PCB. It also defines the current state of the operating system.

Structure of the Process Control Block

- The process control stores many data items that are needed for efficient process management.
- Some of these data items are explained with the help of the given diagram



Process Control Block (PCB)

The following are the data items –

Process State

- This specifies the process state i.e. new, ready, running, waiting or terminated.

Process Number

- This shows the number of the particular process.

Program Counter

- This contains the address of the next instruction that needs to be executed in the process.

Registers

- This specifies the registers that are used by the process. They may include accumulators, index registers, stack pointers, general purpose registers etc.

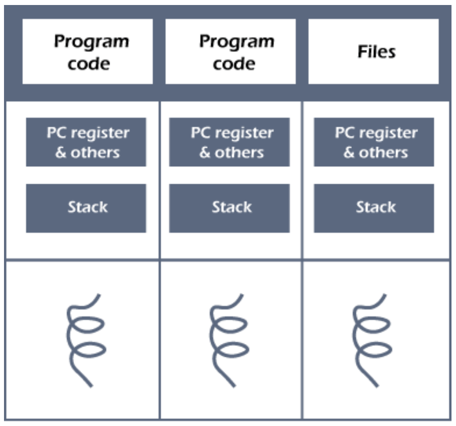
List of Open Files

- These are the different files that are associated with the process

- **CPU Scheduling Information**
    - The process priority, pointers to scheduling queues etc. is the CPU scheduling information that is contained in the PCB.
    - This may also include any other scheduling parameters.
- **Memory Management Information**
    - The memory management information includes the page tables or the segment tables depending on the memory system used.
    - It also contains the value of the base registers, limit registers etc.
- **I/O Status Information**
    - This information includes the list of I/O devices used by the process, the list of files etc.
- **Accounting information**
    - The time limits, account numbers, amount of CPU used, process numbers etc. are all a part of the PCB accounting information.
- **Location of the Process Control Block**
    - The process control block is kept in a memory area that is protected from the normal user access.
    - This is done because it contains important process information. Some of the operating systems place the PCB at the beginning of the kernel stack for the process as it is a safe location.

Threads in Operating System (OS)

- A thread is a single sequential flow of execution of tasks of a process so it is also known as thread of execution or thread of control.
- There is a way of thread execution inside the process of any operating system.
- Apart from this, there can be more than one thread inside a process.
- Each thread of the same process makes use of a separate program counter and a stack of activation records and control blocks.
- Thread is often referred to as a lightweight process.



Three threads of same process

- The process can be split down into so many threads. **For example**, in a browser, many tabs can be viewed as threads.
- MS Word uses many threads - formatting text from one thread, processing input from another thread, etc.

Need of Thread:

- It takes far less time to create a new thread in an existing process than to create a new process.
- Threads can share the common data; they do not need to use Inter-Process communication.
- Context switching is faster when working with threads.
- It takes less time to terminate a thread than a process.

Types of Threads

In the operating system, there are two types of threads.

1. Kernel level thread.
2. User-level thread.

User-level thread

- The OS does not recognize the user-level thread.
- User threads can be easily implemented and it is implemented by the user.
- If a user performs a user-level thread blocking operation, the whole process is blocked.

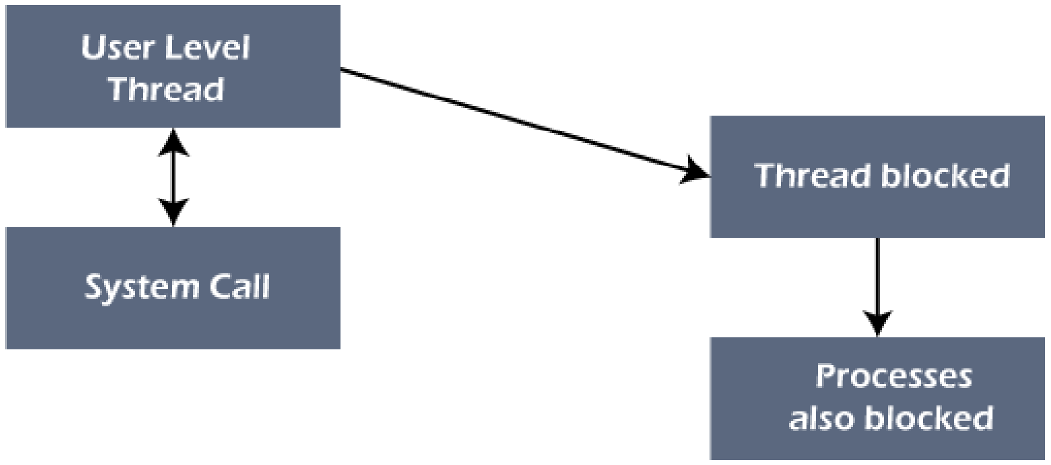
- The kernel level thread does not know anything about the user level thread.
- The kernel-level thread manages user-level threads as if they are single-threaded processes? Examples: Java thread, POSIX threads, etc.

Advantages of User-level threads

1. The user threads can be easily implemented than the kernel thread.
2. User-level threads can be applied to such types of operating systems that do not support threads at the kernel-level.
3. It is faster and efficient.
4. Context switch time is shorter than the kernel-level threads.
5. It does not require modifications of the operating system.
6. User-level threads representation is very simple. The register, PC, stack, and mini thread control blocks are stored in the address space of the user-level process.
7. It is simple to create, switch, and synchronize threads without the intervention of the process.

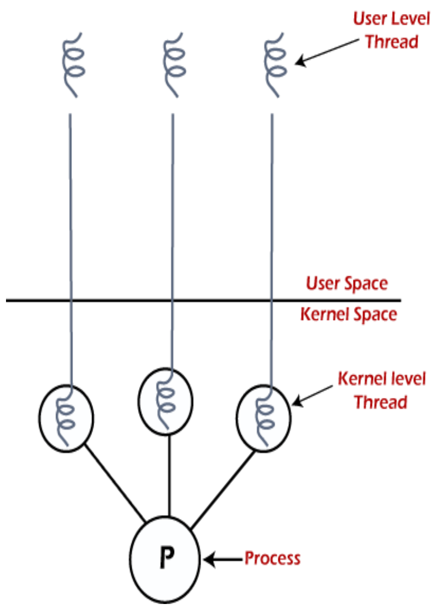
Disadvantages of User-level threads

1. User-level threads lack coordination between the thread and the kernel.
2. If a thread causes a page fault, the entire process is blocked.



Kernel level thread

- The kernel thread recognizes the operating system.
- There is a thread control block and process control block in the system for each thread and process in the kernel-level thread.
- The kernel-level thread is implemented by the operating system.
- The kernel knows about all the threads and manages them.
- The kernel-level thread offers a system call to create and manage the threads from user-space.
- The implementation of kernel threads is more difficult than the user thread.
- Context switch time is longer in the kernel thread. If a kernel thread performs a blocking operation, the Banky thread execution can continue. Example: Window Solaris.



Advantages of Kernel-level threads

1. The kernel-level thread is fully aware of all threads.
2. The scheduler may decide to spend more CPU time in the process of threads being large numerical.

3. The kernel-level thread is good for those applications that block the frequency.

Disadvantages of Kernel-level threads

1. The kernel thread manages and schedules all threads.
2. The implementation of kernel threads is difficult than the user thread.
3. The kernel-level thread is slower than user-level threads.

<u>Components of Threads</u>

Any thread has the following components.

1. Program counter
2. Register set
3. Stack space

<u>Benefits of Threads</u>

- **Enhanced throughput of the system:**
    - When the process is split into many threads and each thread is treated as a job, the number of jobs done in the unit time increases.
    - That is why the throughput of the system also increases.
- **Effective Utilization of Multiprocessor system:**
    - When you have more than one thread in one process, you can schedule more than one thread in more than one processor.
- **Faster context switch:**
    - The context switching period between threads is less than the process context switching.
    - The process context switch means more overhead for the CPU.
- **Responsiveness:**

  - When the process is split into several threads, and when a thread completes its execution, that process can be responded to as soon as possible.
- **Communication:**
  - Multiple-thread communication is simple because the threads share the same address space, while in process; we adopt just a few exclusive communication strategies for communication between two processes.
- **Resource sharing:**
  - Resources can be shared between all threads within a process, such as code, data, and files.
  - Note: The stack and register cannot be shared between threads. There is a stack and register for each thread.

<u>Process Scheduling</u>

<u>Definition</u>

- The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.
- Process scheduling is an essential part of Multiprogramming operating systems.
- Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

<u>Categories of Scheduling</u>

<u>There are two categories of scheduling:</u>

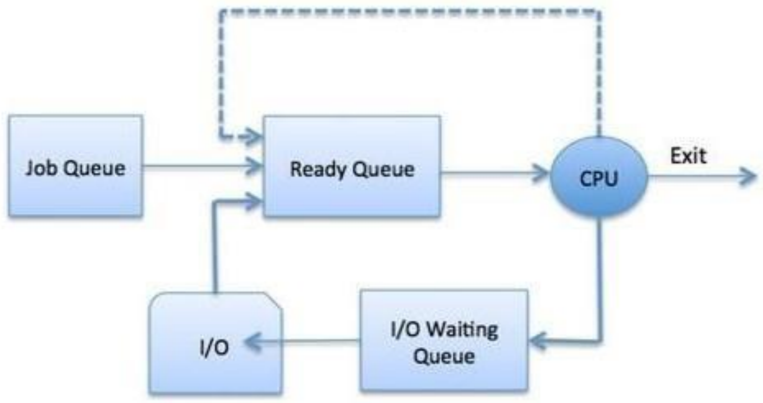
1. **Non-preemptive:**
   a. Here the resource can't be taken from a process until the process completes execution.
   b. The switching of resources occurs when the running process terminates and moves to a waiting state.
2. **Preemptive:**
   a. Here the OS allocates the resources to a process for a fixed amount of time.

b. During resource allocation, the process switches from running state to ready state or from waiting state to ready state.
c. This switching occurs as the CPU may give priority to other processes and replace the process with higher priority with the running process.

Process Scheduling Queues

- The OS maintains all Process Control Blocks (PCBs) in Process Scheduling Queues.
- The OS maintains a separate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue.
- When the state of a process is changed, its PCB is unlinked from its current queue and moved to its new state queue.
- The Operating System maintains the following important process scheduling queues –

- **Job queue** – this queue keeps all the processes in the system.
- **Ready queue** – this queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.
- **Device queues** – the processes which are blocked due to unavailability of an I/O device constitute this queue.



- The OS can use different policies to manage each queue (FIFO, Round Robin, Priority, etc.).
- The OS scheduler determines how to move processes between the ready and run queues which can only have one entry per processor core on the system; in the above diagram, it has been merged with the CPU.

Two-State Process Model

Two-state process model refers to running and non-running states which are described below –

S.N.	State & Description
1	**Running** - When a new process is created, it enters into the system as in the running state.
2	**Not Running** - Processes that are not running are kept in queue, waiting for their turn to execute. - Each entry in the queue is a pointer to a particular process. Queue is implemented by using linked list. - Use of dispatcher is as follows. When a process is interrupted, that process is transferred in the waiting queue. - If the process has completed or aborted, the process is discarded. - In either case, the dispatcher then selects a process from the queue to execute.

Schedulers

- Schedulers are special system software which handles process scheduling in various ways.
- Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types –
  - Long-Term Scheduler
  - Short-Term Scheduler
  - Medium-Term Scheduler

Long Term Scheduler

- It is also called a **job scheduler**.
- A long-term scheduler determines which programs are admitted to the system for processing.
- It selects processes from the queue and loads them into memory for execution.
- Process loads into the memory for CPU scheduling.

- The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound.
- It also controls the degree of multiprogramming.
- If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.
- On some systems, the long-term scheduler may not be available or minimal.
- Time-sharing operating systems have no long term scheduler.
- When a process changes the state from new to ready, then there is use of long-term scheduler.

Short Term Scheduler

- It is also called as **CPU scheduler**.
- Its main objective is to increase system performance in accordance with the chosen set of criteria.
- It is the change of ready state to running state of the process.
- CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.
- Short-term schedulers, also known as dispatchers, make the decision of which process to execute next.
- Short-term schedulers are faster than long-term schedulers.

Medium Term Scheduler

- Medium-term scheduling is a part of **swapping**.
- It removes the processes from the memory.
- It reduces the degree of multiprogramming.
- The medium-term scheduler is in-charge of handling the swapped out-processes.
- A running process may become suspended if it makes an I/O request.
- Suspended processes cannot make any progress towards completion.
- In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage.
- This process is called **swapping**, and the process is said to be swapped out or rolled out.
- Swapping may be necessary to improve the process mix.

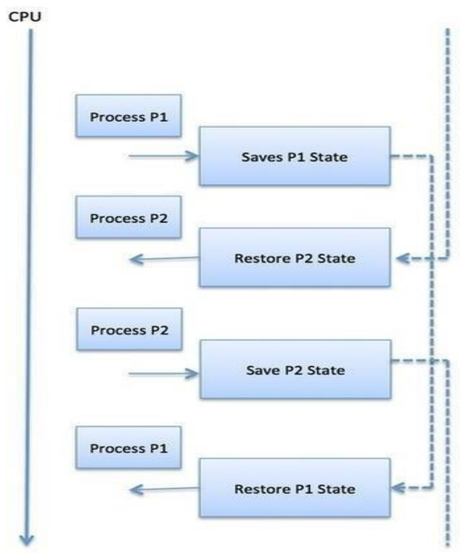
Comparison among Scheduler

S.N.	Long-Term Scheduler	Short-Term Scheduler	Medium-Term Scheduler

1	It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler.
2	Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short and long term scheduler.
3	It controls the degree of multiprogramming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming.
4	It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time sharing systems.
5	It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

Context Switching

- A context switching is the mechanism to store and restore the state or context of a CPU in Process Control block so that a process execution can be resumed from the same point at a later time.
- Using this technique, a context switcher enables multiple processes to share a single CPU.
- Context switching is an essential part of a multitasking operating system features.
- When the scheduler switches the CPU from executing one process to execute another, the state from the current running process is stored into the process control block.
- After this, the state for the process to run next is loaded from its own PCB and used to set the PC, registers, etc.
- At that point, the second process can start executing.



- Context switches are computationally intensive since register and memory state must be saved and restored.
- To avoid the amount of context switching time, some hardware systems employ two or more sets of processor registers.
- When the process is switched, the following information is stored for later use.

  - Program Counter
  - Scheduling information
  - Base and limit register value
  - Currently used register
  - Changed State
  - I/O State information
  - Accounting information

<u>Context Switching in OS (Operating System)</u>

- The Context switching is a technique or method used by the operating system to switch a process from one state to another to execute its function using CPUs in the system.
- When switching performs in the system, it stores the old running process's status in the form of registers and assigns the CPU to a new process to execute its tasks.
- While a new process is running in the system, the previous process must wait in a ready queue.

- The execution of the old process starts at that point where another process stopped it.
- It defines the characteristics of a multitasking operating system in which multiple processes shared the same CPU to perform multiple tasks without the need for additional processors in the system.

The need for Context switching

- A context switching helps to share a single CPU across all processes to complete its execution and store the system's tasks status.
- When the process reloads in the system, the execution of the process starts at the same point where there is conflicting.

Following are the reasons that describe the need for context switching in the Operating system.

1. The switching of one process to another process is not directly in the system. A context switching helps the operating system that switches between the multiple processes to use the CPU's resource to accomplish its tasks and store its context. We can resume the service of the process at the same point later. If we do not store the currently running process's data or context, the stored data may be lost while switching between processes.
2. If a high priority process falls into the ready queue, the currently running process will be shut down or stopped by a high priority process to complete its tasks in the system.
3. If any running process requires I/O resources in the system, the current process will be switched by another process to use the CPUs. And when the I/O requirement is met, the old process goes into a ready state to wait for its execution in the CPU. Context switching stores the state of the process to resume its tasks in an operating system. Otherwise, the process needs to restart its execution from the initials level.
4. If any interrupts occur while running a process in the operating system, the process status is saved as registers using context switching. After

resolving the interrupts, the process switches from a wait state to a ready state to resume its execution at the same point later, where the operating system interrupted occurs.

5. A context switching allows a single CPU to handle multiple process requests simultaneously without the need for any additional processors.

Example of Context Switching

- Suppose that multiple processes are stored in a Process Control Block (PCB).
- One process is running state to execute its task with the use of CPUs. As the process is running, another process arrives in the ready queue, which has a high priority of completing its task using CPU.
- Here we used context switching that switches the current process with the new process requiring the CPU to finish its tasks.
- While switching the process, a context switch saves the status of the old process in registers.
- When the process reloads into the CPU, it starts the execution of the process when the new process stops the old process.
- If we do not save the state of the process, we have to start its execution at the initial level.
- In this way, context switching helps the operating system to switch between the processes, store or reload the process when it requires executing its tasks.

Context switching triggers

Following are the three types of context switching triggers as follows.

1. Interrupts
2. Multitasking
3. Kernel/User switch

Interrupts:

- A CPU requests for the data to read from a disk, and if there are any interrupts, the context switching automatic switches a part of the hardware that requires less time to handle the interrupts.

Multitasking:

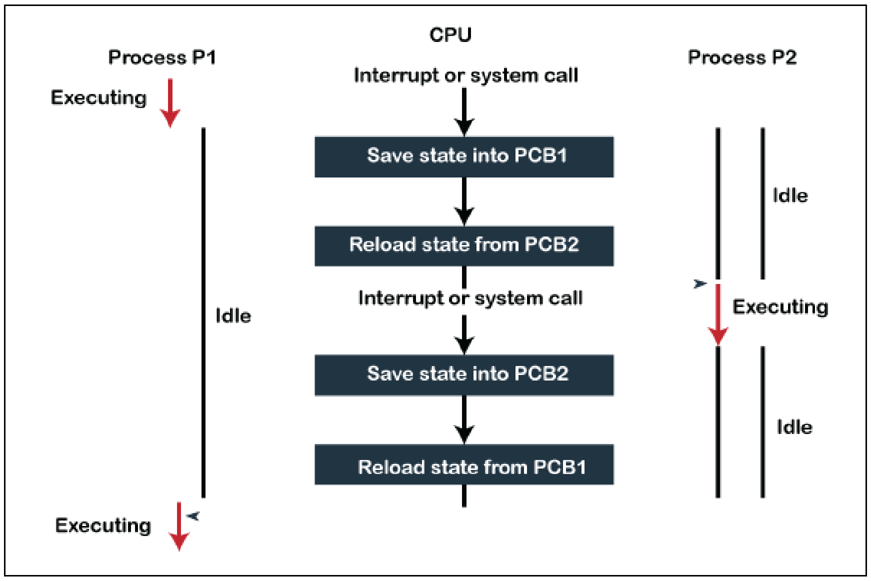
- A context switching is the characteristic of multitasking that allows the process to be switched from the CPU so that another process can be run.
- When switching the process, the old state is saved to resume the process's execution at the same point in the system.

Kernel/User Switch:

- It is used in the operating systems when switching between the user mode and the kernel/user mode is performed.

Steps for Context Switching

- There are several steps involves in context switching of the processes.
- The following diagram represents the context switching of two processes, P1 to P2, when an interrupt, I/O needs, or priority-based process occurs in the ready queue of PCB.



- As we can see in the diagram, initially, the P1 process is running on the CPU to execute its task, and at the same time, another process, P2, is in the ready state.
- If an error or interruption has occurred or the process requires input/output, the P1 process switches its state from running to the waiting state.
- Before changing the state of the process P1, context switching saves the context of the process P1 in the form of registers and the program counter to the **PCB1**.
- After that, it loads the state of the P2 process from the ready state of the **PCB2** to the running state.

The following steps are taken when switching Process P1 to Process 2:

1. First, thes context switching needs to save the state of process P1 in the form of the program counter and the registers to the PCB (Program Counter Block), which is in the running state.
2. Now update PCB1 to process P1 and moves the process to the appropriate queue, such as the ready queue, I/O queue and waiting queue.
3. After that, another process gets into the running state, or we can select a new process from the ready state, which is to be executed, or the process has a high priority to execute its task.
4. Now, we have to update the PCB (Process Control Block) for the selected process P2. It includes switching the process state from ready to running state or from another state like blocked, exit, or suspend.
5. If the CPU already executes process P2, we need to get the status of process P2 to resume its execution at the same time point where the system interrupt occurs.

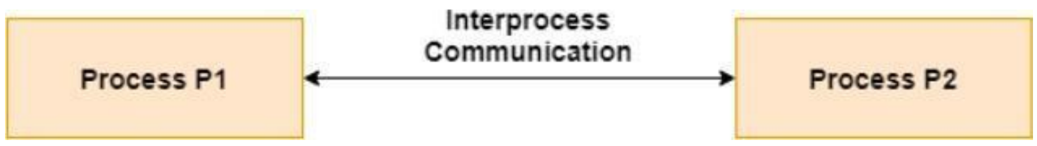
- Similarly, process P2 is switched off from the CPU so that the process P1 can resume execution.
- P1 process is reloaded from PCB1 to the running state to resume its task at the same point.

- Otherwise, the information is lost, and when the process is executed again, it starts execution at the initial level.

Inter Process Communication:

- Inter process communication is the mechanism provided by the operating system that allows processes to communicate with each other.
- This communication could involve a process letting another process know that some event has occurred or the transferring of data from one process to another.

A diagram that illustrates inter process communication is as follows –



Synchronization in Inter Process Communication

- Synchronization is a necessary part of inter process communication. It is either provided by the inter process control mechanism or handled by the communicating processes.
- Some of the methods to provide synchronization are as follows –

- **Semaphore**

- A semaphore is a variable that controls the access to a common resource by multiple processes.
- The two types of semaphores are binary semaphores and counting semaphores.

- **Mutual Exclusion**

- Mutual exclusion requires that only one process thread can enter the critical section at a time.
- This is useful for synchronization and also prevents race conditions.

- **Barrier**

- A barrier does not allow individual processes to proceed until all the processes reach it.
- Many parallel languages and collective routines impose barriers.

- **Spinlock**

- This is a type of lock. The processes trying to acquire this lock wait in a loop while checking if the lock is available or not.
- This is known as busy waiting because the process is not doing any useful operation even though it is active.

<u>Approaches to Inter process Communication</u>

The different approaches to implement inter process communication are given as follows –

- **Pipe**

- A pipe is a data channel that is unidirectional. Two pipes can be used to create a two-way data channel between two processes.
- This uses standard input and output methods.
- Pipes are used in all POSIX systems as well as Windows operating systems.

- **Socket**

- The socket is the endpoint for sending or receiving data in a network.
- This is true for data sent between processes on the same computer or data sent between different computers on the same network.
- Most of the operating systems use sockets for inter process communication.

- **File**

- A file is a data record that may be stored on a disk or acquired on demand by a file server.
- Multiple processes can access a file as required.
- All operating systems use files for data storage.

- **Signal**

- Signals are useful in inter process communication in a limited way.
- They are system messages that are sent from one process to another.

- Normally, signals are not used to transfer data but are used for remote commands between processes.

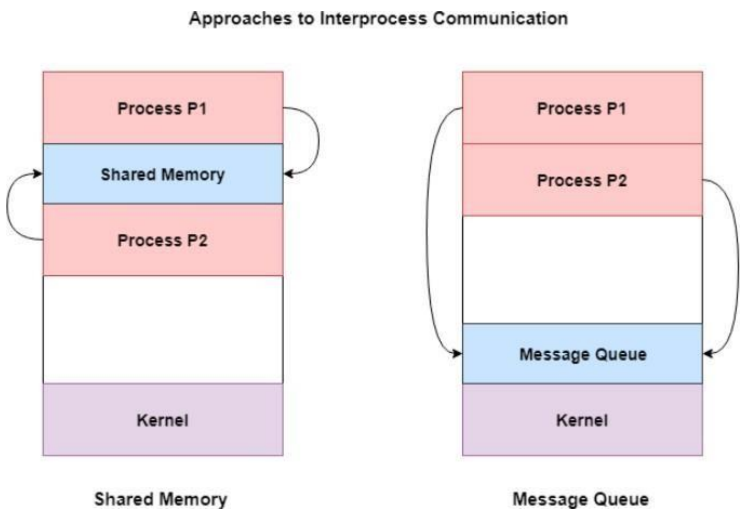
- **Shared Memory**

- Shared memory is the memory that can be simultaneously accessed by multiple processes.
- This is done so that the processes can communicate with each other.
- All POSIX systems, as well as Windows operating systems use shared memory.

- **Message Queue**

- Multiple processes can read and write data to the message queue without being connected to each other.
- Messages are stored in the queue until their recipient retrieves them.
- Message queues are quite useful for inter process communication and are used by most operating systems.

A diagram that demonstrates message queue and shared memory methods of inter process communication is as follows –



Unit 1

2 Marks:

1. What is system call? Mention different types of system call.

2. Define thread. State the major advantage of thread.

3. Define context switch.

4. What is the need for an operating system? Give example of a operating systems

5. Define term CPU scheduling and throughput.

6. What is multiprogramming?

7. Mention the types of Inter process communication.

8. Define operating system list with its goals.

9. Mention any two applications real time systems.

10. What do you mean by process?

11. Define throughput and response time.

12. Define schedulers. List its types.

4 Marks:

1. What are the components of process control block? Explain

2. List out advantages and disadvantages of real time systems.

3. What are process states? Explain the state transition diagram

4. Explain the activities of an operating system in connection with process management and file management.

5. What is a real time system? Explain.

6. Explain multiprogramming system with its features.

7. Explain batch operating system with advantages and disadvantages.

8. Explain process control block with neat diagram.

9. Explain monolithic and Microkernel operating system.

10. Explain system call.