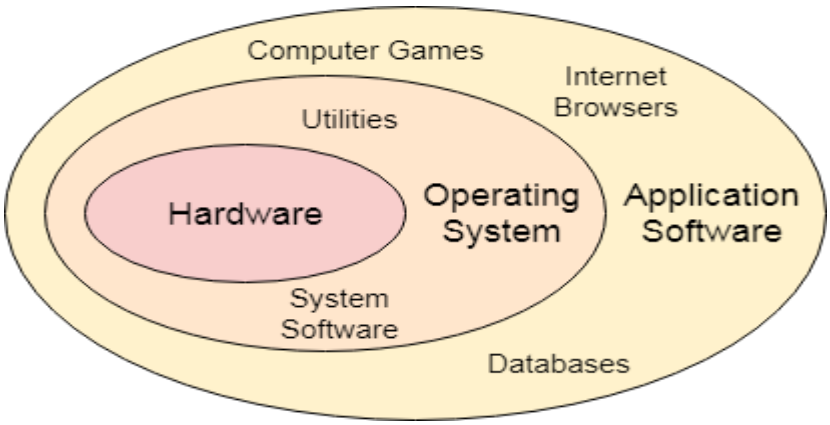
Operating System Definition and Function

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.



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, CPU management, File Management, and many other tasks.

Structure of a Computer System

A Computer System consists of:

- Users (people who are using the computer)
- Application Programs (Compilers, Databases, Games, Video player, Browsers, etc.)
- System Programs (Shells, Editors, Compilers, etc.)
- Operating System (A special program which acts as an interface between user and hardware)
- Hardware (CPU, Disks, Memory, etc)

What does an Operating system do?

1. Process Management
2. Process Synchronization
3. Memory Management
4. CPU Scheduling
5. File Management
6. Security

Types of Operating System

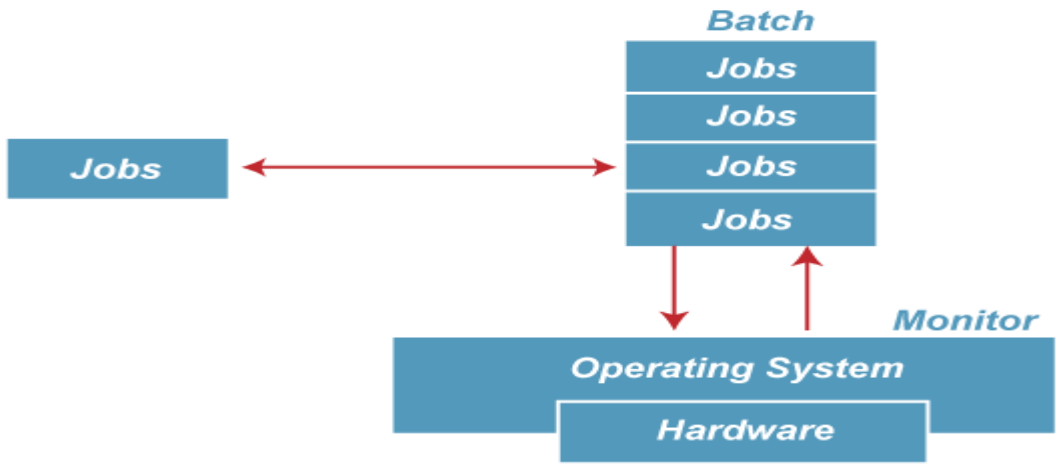
1. Batch Operating System
2. Time-Sharing Operating System
3. Embedded Operating System
4. Multiprogramming Operating System
5. Network Operating System
6. Distributed Operating System
7. Multiprocessing Operating System
8. Real-Time Operating System

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.

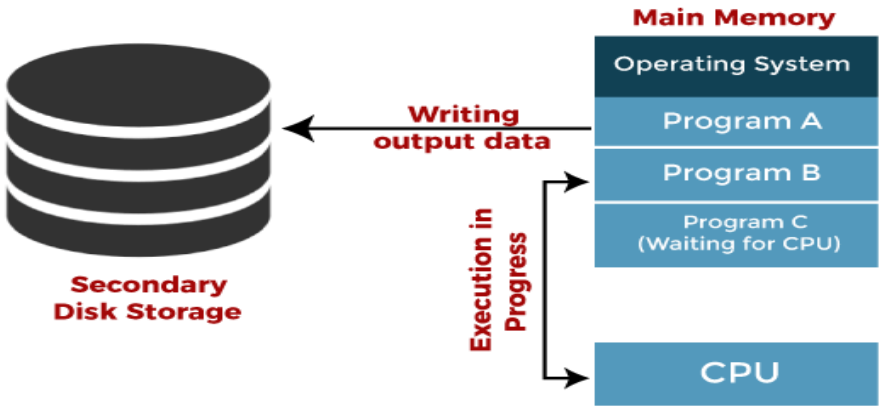


The purpose of this operating system was mainly to transfer control from one job to another as soon as the job was completed. It contained a small set of programs called the resident monitor that always resided in one part of the main memory. The remaining part is used for servicing jobs.

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

Advantages of Multiprogramming OS

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

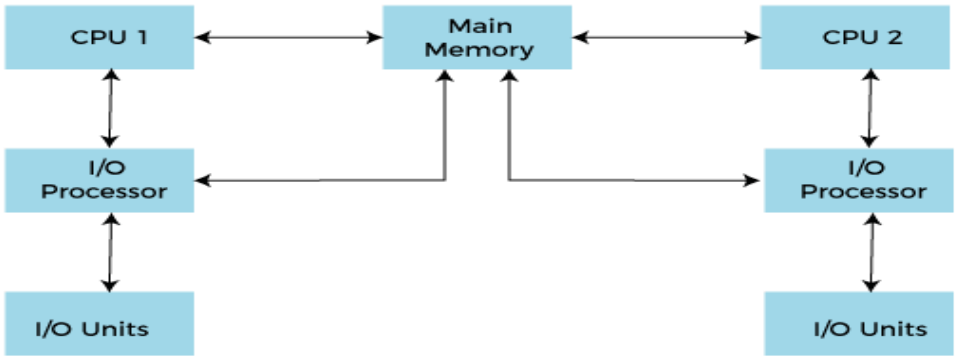
Disadvantages of Multiprogramming OS

- 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.

Multiprocessing Operating System

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.

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.



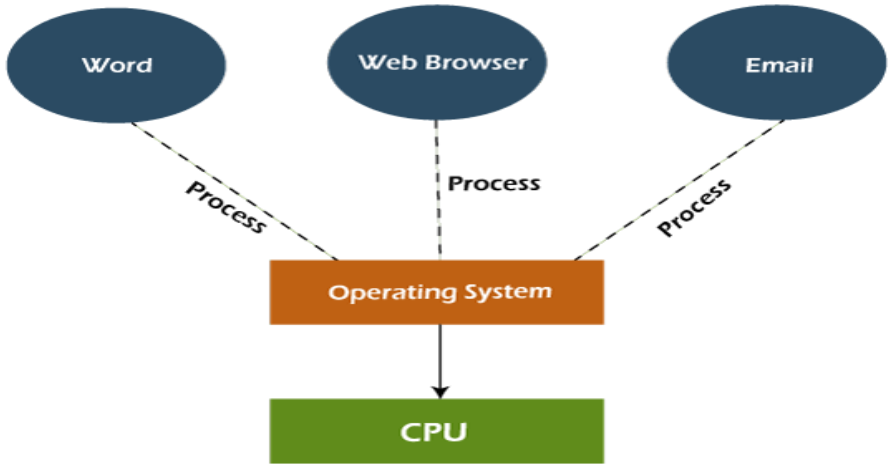
Working of Multiprocessor System

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.

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.

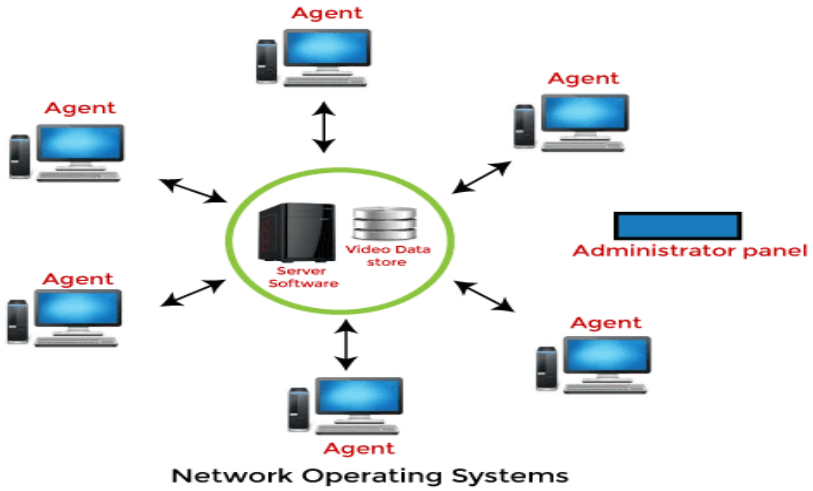
Advantages of Multitasking operating system

- 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



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

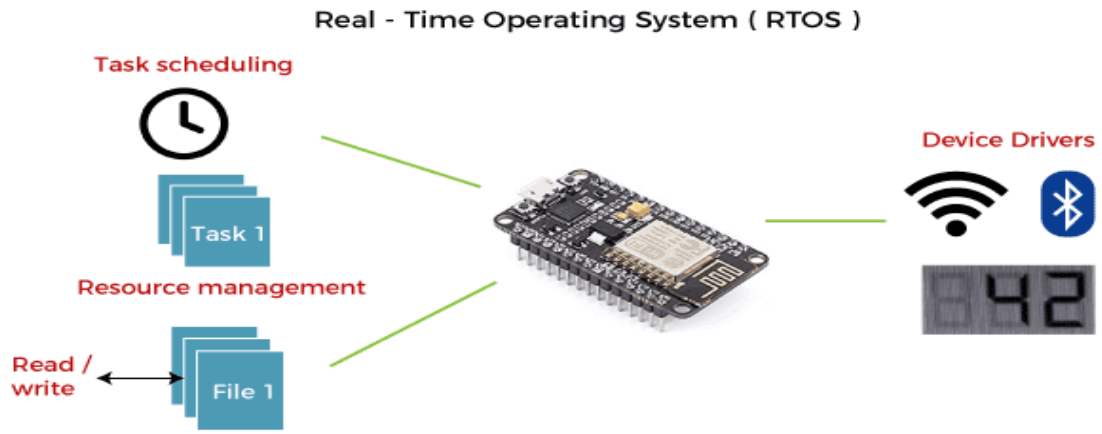
- In this type of operating system, network traffic reduces due to the division between clients and the server.
- This type of system is less expensive to set up and maintain.

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.

Advantages of Real-time operating system:

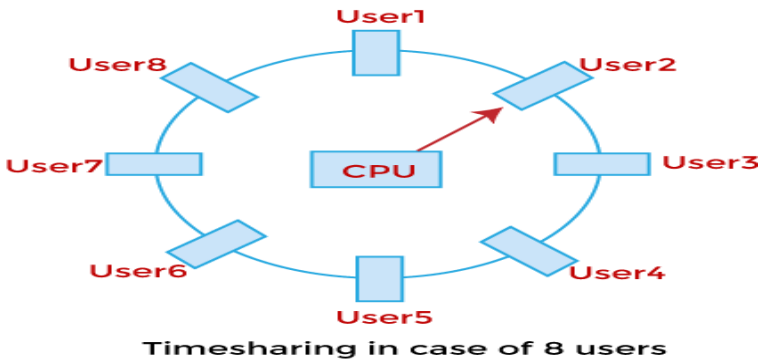
- Easy to layout, develop and execute real-time applications under the real-time operating system.
- In a Real-time operating system, the maximum utilization of devices and systems.

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

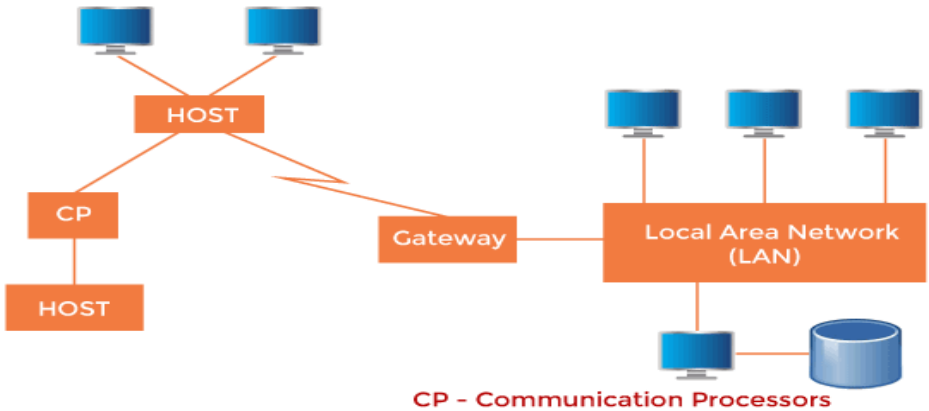
- The time-sharing operating system provides effective utilization and sharing of resources.
- This system reduces CPU idle and response time.

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.

The evolution of operating system, Major achievements

Generations of Operating System

The First Generation (1940 to early 1950s)

When the first electronic computer was developed in 1940, it was created without any operating system. In early times, users have full access to the computer machine and write a program for each task in absolute machine language. The programmer can perform and solve only simple mathematical calculations during the computer generation, and this calculation does not require an operating system.

The Second Generation (1955 - 1965)

The first operating system (OS) was created in the early 1950s and was known as **GMOS. General Motors** has developed OS for the **IBM** computer. The second-generation operating system was based on a single stream batch processing system because it collects all similar jobs in groups or batches and then submits the jobs to the operating system using a punch card to complete all jobs in a machine. At each completion of jobs (either normally or abnormally), control transfer to the operating system that is cleaned after completing one job and then continues to read and initiates the next job in a punch card. After that, new machines were called mainframes, which were very big and used by professional operators.

The Third Generation (1965 - 1980)

During the late 1960s, operating system designers were very capable of developing a new operating system that could simultaneously perform multiple tasks in a single computer program called multiprogramming. The introduction of **multiprogramming** plays a very important role in developing operating systems that allow a CPU to be busy every time by performing different tasks on a computer at the same time. During the third generation, there was a new development of minicomputer's phenomenal growth starting in 1961 with the DEC PDP-1. These PDP's leads to the creation of personal computers in the fourth generation.

The Fourth Generation (1980 - Present Day)

The fourth generation of operating systems is related to the development of the personal computer. However, the personal computer is very similar to the minicomputers that were developed in the third generation. The cost of a personal computer was very high at that time; there were small fractions of minicomputers costs. A major factor related to creating personal computers was the birth of Microsoft and the Windows operating system. Microsoft created the first **window** operating system in 1975. After introducing the Microsoft Windows OS,

Bill Gates and Paul Allen had the vision to take personal computers to the next level. Therefore, they introduced the **MS-DOS** in 1981; however, it was very difficult for the person to understand its cryptic commands. Today, Windows has become the most popular and most commonly used operating system technology. And then, Windows released various operating systems such as Windows 95, Windows 98, Windows XP and the latest operating system, Windows 7. Currently, most Windows users use the Windows 10 operating system. Besides the Windows operating system, Apple is another popular operating system built in the 1980s, and this operating system was developed by Steve Jobs, a co-founder of Apple. They named the operating system Macintosh OS or Mac OS.

Multiprocessor and Multicore System in Operating System

Multicore and multiprocessor systems both serve to accelerate the computing process. A multicore contains multiple cores or processing units in a single CPU. A multiprocessor is made up of several CPUs. A multicore processor does not need complex configurations like a multiprocessor. In contrast, A multiprocessor is much reliable and capable of running many programs. In this article, you will learn about the Multiprocessor and Multicore system in the operating system with their advantages and disadvantages.

What is a Multiprocessor System?

A multiprocessor has multiple CPUs or processors in the system. Multiple instructions are executed simultaneously by these systems. As a result, throughput is increased. If one CPU fails, the other processors will continue to work normally. So, multiprocessors are more reliable.

Shared memory or distributed memory can be used in multiprocessor systems. Each processor in a shared memory multiprocessor shares main memory and peripherals to execute instructions concurrently. In these systems, all CPUs access the main memory over the same bus. Most CPUs will be idle as the bus traffic increases. This type of multiprocessor is also known as the symmetric multiprocessor. It provides a single memory space for all processors.

Each CPU in a distributed memory multiprocessor has its own private memory. Each processor can use local data to accomplish the computational tasks. The processor may use the bus to communicate with other processors or access the main memory if remote data is required.

Advantages and disadvantages of Multiprocessor System

There are various advantages and disadvantages of the multiprocessor system. Some advantages and disadvantages of the multiprocessor system are as follows:

Advantages

There are various advantages of the multiprocessor system. Some advantages of the multiprocessor system are as follows:

1. It is a very reliable system because multiple processors may share their work between the systems, and the work is completed with collaboration.
2. It requires complex configuration.
3. Parallel processing is achieved via multiprocessing.
4. If multiple processors work at the same time, the throughput may increase.
5. Multiple processors execute the multiple processes a few times.

Disadvantages

There are various disadvantages of the multiprocessor system. Some disadvantages of the multiprocessor system are as follows:

1. Multiprocessors work with different systems, so processors require memory space.
2. If one of the processors fails, the work is shared among the remaining processors.
3. These types of systems are very expensive.
4. If any processor is already utilizing an I/O device, additional processors may not utilize the same I/O device that creates deadlock.
5. The operating system implementation is complicated because multiple processors communicate with each other.

What is a Multicore System?

A single computing component with multiple cores (independent processing units) is known as a multicore processor. It denotes the presence of a single CPU with several cores in the system. Individually, these cores may read and run computer instructions. They work in such a way that the computer system appears to have several processors, although they are cores, not processors. These cores may execute normal processors instructions, including add, move data, and branch.

A single processor in a multicore system may run many instructions simultaneously, increasing the overall speed of the system's program execution. It decreases the amount of heat generated by the CPU while enhancing the speed with which instructions are executed.

Multicore processors are used in various applications, including general-purpose, embedded, network, and graphics processing (GPU).

The software techniques used to implement the cores in a multicore system are responsible for the system's performance. The extra focus has been put on developing software that may execute in parallel because you want to achieve parallel execution with the help of many cores'

Advantages and disadvantages of Multicore System

There are various advantages and disadvantages of the multicore system. Some advantages and disadvantages of the multicore system are as follows:

Advantages

There are various advantages of the multicore system. Some advantages of the multicore system are as follows:

1. Multicore processors may execute more data than single-core processors.
2. When you are using multicore processors, the PCB requires less space.
3. It will have less traffic.
4. Multicores are often integrated into a single integrated circuit die or onto numerous dies but packaged as a single chip. As a result, Cache Coherency is increased.
5. These systems are energy efficient because they provide increased performance while using less energy.

Disadvantages

There are various disadvantages of the multicore system. Some disadvantages of the multicore system are as follows:

Advantages and disadvantages of Multicore System

There are various advantages and disadvantages of the multicore system. Some advantages and disadvantages of the multicore system are as follows:

Advantages

There are various advantages of the multicore system. Some advantages of the multicore system are as follows:

1. Multicore processors may execute more data than single-core processors.

2. When you are using multicore processors, the PCB requires less space.
3. It will have less traffic.
4. Multicores are often integrated into a single integrated circuit die or onto numerous dies but packaged as a single chip. As a result, Cache Coherency is increased.
5. These systems are energy efficient because they provide increased performance while using less energy.

Disadvantages

There are various disadvantages of the multicore system. Some disadvantages of the multicore system are as follows:

Features	Multiprocessors	Multicore
Definition	It is a system with multiple CPUs that allows processing programs simultaneously.	A multicore processor is a single processor that contains multiple independent processing units known as cores that may read and execute program instructions.
Execution	Multiprocessors run multiple programs faster than a multicore system.	The multicore executes a single program faster.
Reliability	It is more reliable than the multicore system. If one of any processors fails in the system, the other processors will not be affected.	It is not much reliable than the multiprocessors.
Traffic	It has high traffic than the multicore system.	It has less traffic than the multiprocessors.
Cost	It is more expensive as compared to a multicore system.	These are cheaper than the multiprocessors system.
Configuration	It requires complex configuration.	It doesn't need to be configured.

PROCESS CONCEPT– Processes, PCB

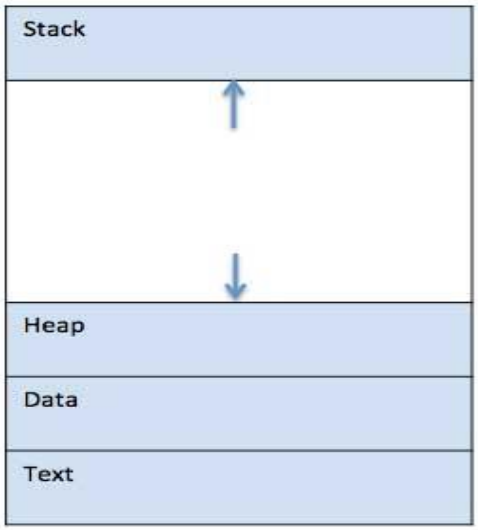
Process

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.

Program

A program is a piece of code which may be a single line or millions of lines. A computer program is usually written by a computer programmer in a programming language. For example, here is a simple program written in C programming language –

```
#include <stdio.h>

int main() {
   printf("Hello, World! \n");
   return 0;
}
```

A computer program is a collection of instructions that performs a specific task when executed by a computer. When we compare a program with a process, we can conclude that a process is a dynamic instance of a computer program.

A part of a computer program that performs a well-defined task is known as an **algorithm**. A collection of computer programs, libraries and related data are referred to as a **software**.

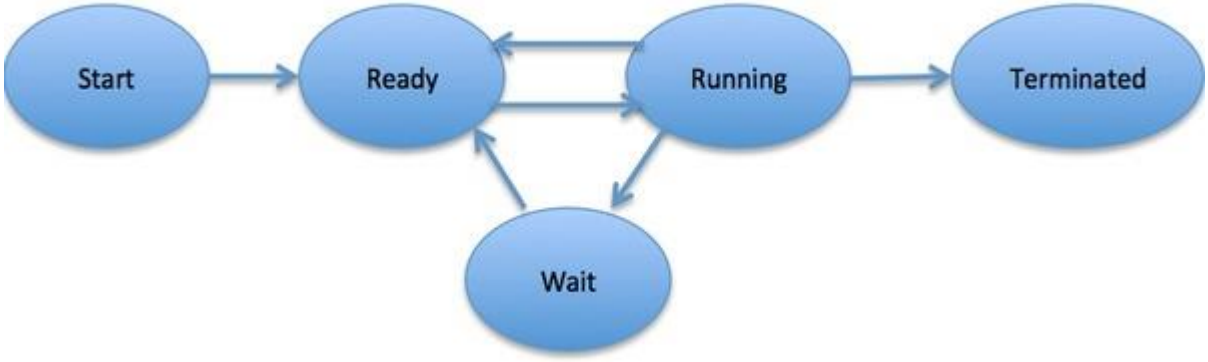
Process Life Cycle

When a process executes, it passes through different states. These stages may differ in different operating systems, and the names of these states are also not standardized.

In general, a process can have one of the following five states at a time.

S.N.	State & Description
1	**Start** This is the initial state when a process is first started/created.
2	**Ready** The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run. Process may come into this state after **Start** state or while running it by but interrupted by the scheduler to assign CPU to some other process.

3	**Running** Once the process has been assigned to a processor by the OS scheduler, the process state is set to running and the processor executes its instructions.
4	**Waiting** Process moves into the waiting state if it needs to wait for a resource, such as waiting for user input, or waiting for a file to become available.
5	**Terminated or Exit** Once the process finishes its execution, or it is terminated by the operating system, it is moved to the terminated state where it waits to be removed from main memory.



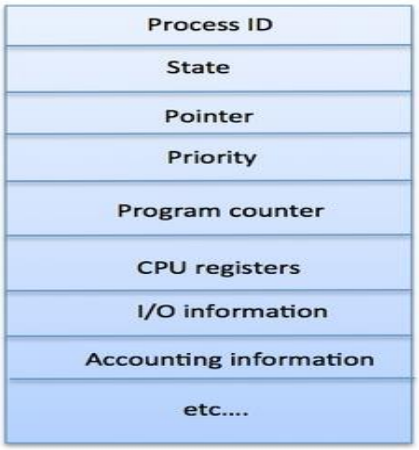
Process Control Block (PCB)

A Process Control Block is a data structure maintained by the Operating System for every process. The PCB is identified by an integer process ID (PID). A PCB keeps all the information needed to keep track of a process as listed below in the table −

S.N.	Information & Description
1	**Process State** The current state of the process i.e., whether it is ready, running, waiting, or whatever.
2	**Process privileges** This is required to allow/disallow access to system resources.
3	**Process ID**

	Unique identification for each of the process in the operating system.
4	**Pointer** A pointer to parent process.
5	**Program Counter** Program Counter is a pointer to the address of the next instruction to be executed for this process.
6	**CPU registers** Various CPU registers where process need to be stored for execution for running state.
7	**CPU Scheduling Information** Process priority and other scheduling information which is required to schedule the process.
8	**Memory management information** This includes the information of page table, memory limits, Segment table depending on memory used by the operating system.
9	**Accounting information** This includes the amount of CPU used for process execution, time limits, execution ID etc.
10	**IO status information** This includes a list of I/O devices allocated to the process.

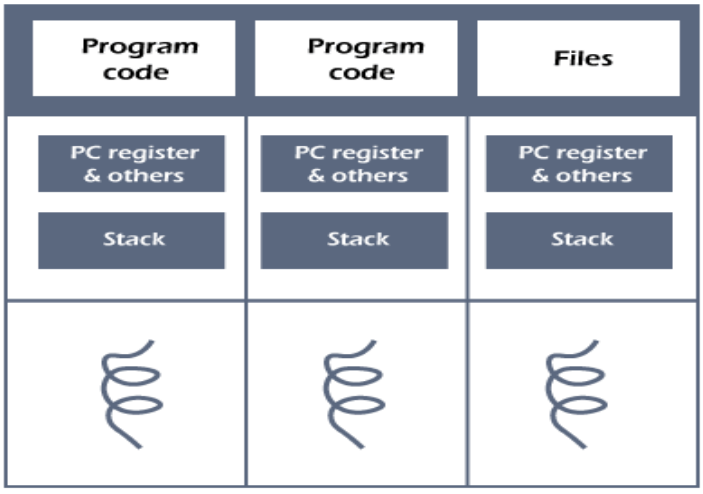
The architecture of a PCB is completely dependent on Operating System and may contain different information in different operating systems. Here is a simplified diagram of a PCB −



The PCB is maintained for a process throughout its lifetime, and is deleted once the process terminates.

Threads in Operating System

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 operating system

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 nothing 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.

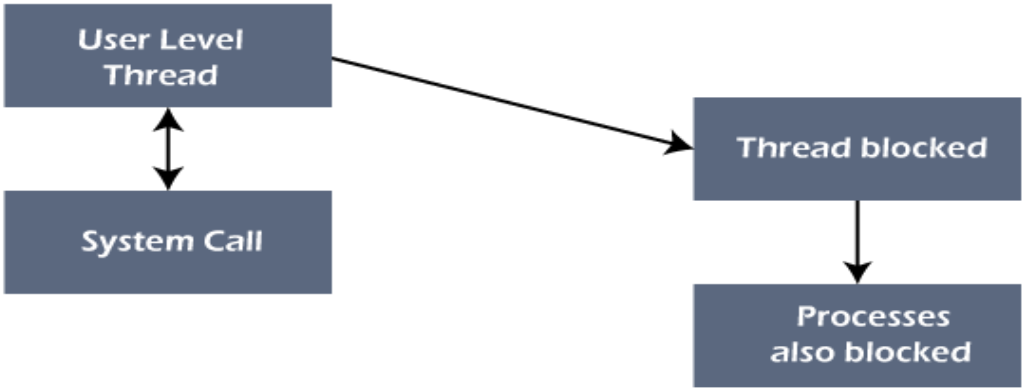
Advantages of User-level threads

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.

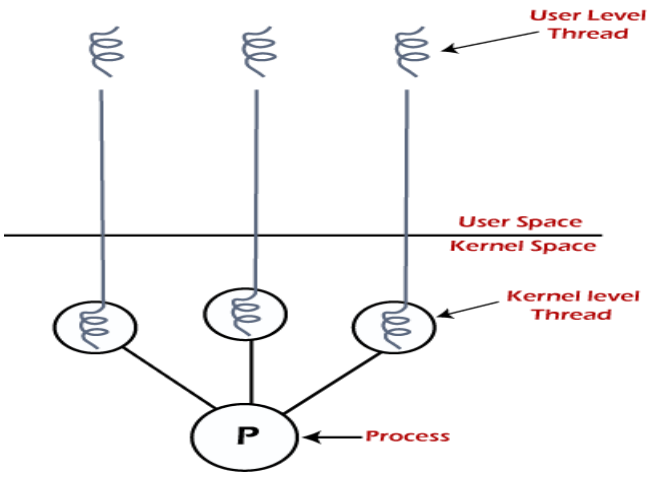
Disadvantages of User-level threads

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.

Components of Threads

Any thread has the following components.

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

Benefits of Threads

- **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.

Process Scheduling in OS: Long, Medium, Short Term Scheduler

What is Process Scheduling?

Process Scheduling is an OS task that schedules processes of different states like ready, waiting, and running.

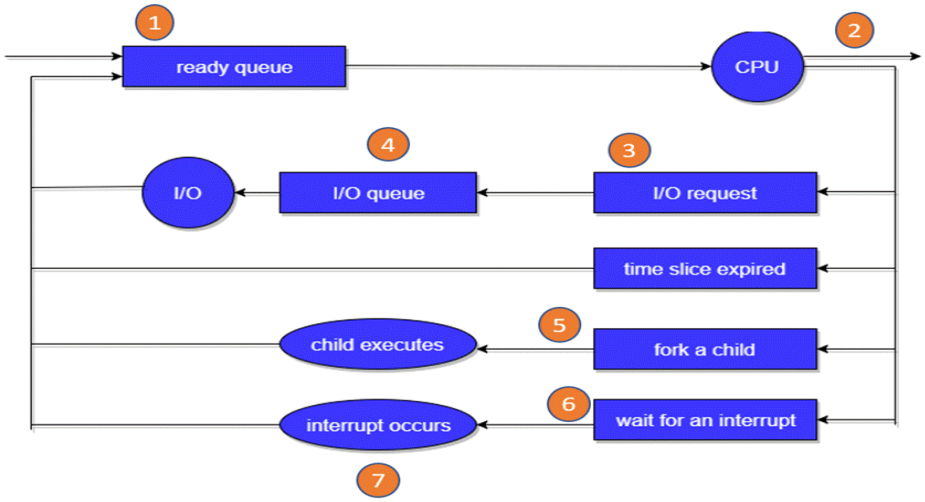
Process scheduling allows OS to allocate a time interval of CPU execution for each process. Another important reason for using a process scheduling system is that it keeps the CPU busy all the time. This allows you to get the minimum response time for programs.

Process Scheduling Queues

Process Scheduling Queues help you to maintain a distinct queue for each and every process states and PCBs. All the process of the same execution state are placed in the same queue. Therefore, whenever the state of a process is modified, its PCB needs to be unlinked from its existing queue, which moves back to the new state queue.

Three types of operating system queues are:

1. **Job queue** – It helps you to store all the processes in the system.
2. **Ready queue** – This type of queue helps you to set every process residing in the main memory, which is ready and waiting to execute.
3. **Device queues** – It is a process that is blocked because of the absence of an I/O device.



Process Scheduling Queues

In the above-given Diagram,

- Rectangle represents a queue.
- Circle denotes the resource
- Arrow indicates the flow of the process.

1. Every new process first put in the Ready queue .It waits in the ready queue until it is finally processed for execution. Here, the new process is put in the ready queue and wait until it is selected for execution or it is dispatched.
2. One of the processes is allocated the CPU and it is executing
3. The process should issue an I/O request
4. Then, it should be placed in the I/O queue.
5. The process should create a new subprocess
6. The process should be waiting for its termination.
7. It should remove forcefully from the CPU, as a result interrupt. Once interrupt is completed, it should be sent back to ready queue.

Two State Process Model

Two-state process models are:

- Running State
- Not Running State

Running

In the Operating system, whenever a new process is built, it is entered into the system, which should be running.

Not Running

The process that are not running are kept in a queue, which is waiting for their turn to execute. Each entry in the queue is a point to a specific process.

Scheduling Objectives

Here, are important objectives of Process scheduling

- Maximize the number of interactive users within acceptable response times.
- Achieve a balance between response and utilization.
- Avoid indefinite postponement and enforce priorities.
- It also should give reference to the processes holding the key resources.

Type of Process Schedulers

A scheduler is a type of system software that allows you to handle process scheduling.

There are mainly three types of Process Schedulers:

1. Long Term Scheduler
2. Short Term Scheduler
3. Medium Term Scheduler

Long Term Scheduler

Long term scheduler is also known as a **job scheduler**. This scheduler regulates the program and select process from the queue and loads them into memory for execution. It also regulates the degree of multi-programing.

However, the main goal of this type of scheduler is to offer a balanced mix of jobs, like Processor, I/O jobs., that allows managing multiprogramming.

Medium Term Scheduler

Medium-term scheduling is an important part of **swapping**. It enables you to handle the swapped out-processes. In this scheduler, a running process can become suspended, which makes an I/O request.

A running process can become suspended if it makes an I/O request. A suspended processes can't make any progress towards completion. In order to remove the process from memory and make space for other processes, the suspended process should be moved to secondary storage.

Short Term Scheduler

Short term scheduling is also known as **CPU scheduler**. The main goal of this scheduler is to boost the system performance according to set criteria. This helps you to select from a group of processes that are ready to execute and allocates CPU to one of them. The dispatcher gives control of the CPU to the process selected by the short term scheduler.

Difference between Schedulers

Long-Term Vs. Short Term Vs. Medium-Term

Long-Term	Short-Term	Medium-Term
Long term is also known as a job scheduler	Short term is also known as CPU scheduler	Medium-term is also called swapping scheduler.
It is either absent or minimal in a time-sharing system.	It is insignificant in the time-sharing order.	This scheduler is an element of Time-sharing systems.
Speed is less compared to the short term scheduler.	Speed is the fastest compared to the short-term and medium-term scheduler.	It offers medium speed.
Allow you to select processes from the loads and pool back into the memory	It only selects processes that is in a ready state of the execution.	It helps you to send process back to memory.
Offers full control	Offers less control	Reduce the level of multiprogramming

Process creation, Process termination operating system notes

Process Creation and Process termination are used to create and terminate processes respectively. Details about these are given as follows −

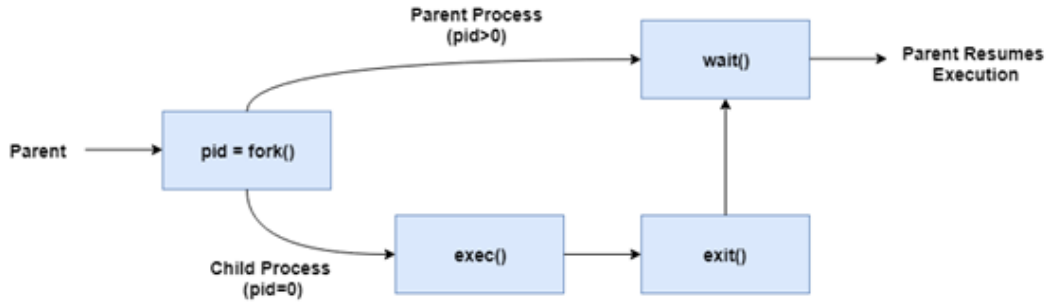
Process Creation

A process may be created in the system for different operations. Some of the events that lead to process creation are as follows −

- User request for process creation
- System Initialization
- Batch job initialization
- Execution of a process creation system call by a running process

A process may be created by another process using fork(). The creating process is called the parent process and the created process is the child process. A child process can have only one parent but a parent process may have many children. Both the parent and child processes have the same memory image, open files and environment strings. However, they have distinct address spaces.

A diagram that demonstrates process creation using fork() is as follows −



Process Creation Using fork()

Process Termination

Process termination occurs when the process is terminated The exit() system call is used by most operating systems for process termination.

Some of the causes of process termination are as follows −

- A process may be terminated after its execution is naturally completed. This process leaves the processor and releases all its resources.
- A child process may be terminated if its parent process requests for its termination.
- A process can be terminated if it tries to use a resource that it is not allowed to. For example - A process can be terminated for trying to write into a read only file.

- If an I/O failure occurs for a process, it can be terminated. For example - If a process requires the printer and it is not working, then the process will be terminated.
- In most cases, if a parent process is terminated then its child processes are also terminated. This is done because the child process cannot exist without the parent process.
- If a process requires more memory than is currently available in the system, then it is terminated because of memory scarcity.

Inter Process Communication (IPC) in OS

What is Inter Process Communication?

Inter process communication (IPC) is used for exchanging data between multiple threads in one or more processes or programs. The Processes may be running on single or multiple computers connected by a network. The full form of IPC is Inter-process communication.

It is a set of programming interface which allow a programmer to coordinate activities among various program processes which can run concurrently in an operating system. This allows a specific program to handle many user requests at the same time.
Since every single user request may result in multiple processes running in the operating system, the process may require to communicate with each other. Each IPC protocol approach has its own advantage and limitation, so it is not unusual for a single program to use all of the IPC methods.

Approaches for Inter-Process Communication

Here, are few important methods for interprocess communication:

Pipes

Pipe is widely used for communication between two related processes. This is a half-duplex method, so the first process communicates with the second process. However, in order to achieve a full-duplex, another pipe is needed.

Message Passing:

It is a mechanism for a process to communicate and synchronize. Using message passing, the process communicates with each other without resorting to shared variables.

IPC mechanism provides two operations:

- Send (message)- message size fixed or variable
- Received (message)

Message Queues:

A message queue is a linked list of messages stored within the kernel. It is identified by a message queue identifier. This method offers communication between single or multiple processes with full-duplex capacity.

Direct Communication:

In this type of inter-process communication process, should name each other explicitly. In this method, a link is established between one pair of communicating processes, and between each pair, only one link exists.

Indirect Communication:

Indirect communication establishes like only when processes share a common mailbox each pair of processes sharing several communication links. A link can communicate with many processes. The link may be bi-directional or unidirectional.

Shared Memory:

Shared memory is a memory shared between two or more processes that are established using shared memory between all the processes. This type of memory requires to protected from each other by synchronizing access across all the processes.

FIFO:

Communication between two unrelated processes. It is a full-duplex method, which means that the first process can communicate with the second process, and the opposite can also happen

PROCESS SYNCHRONIZATION: Background, Critical section Problem

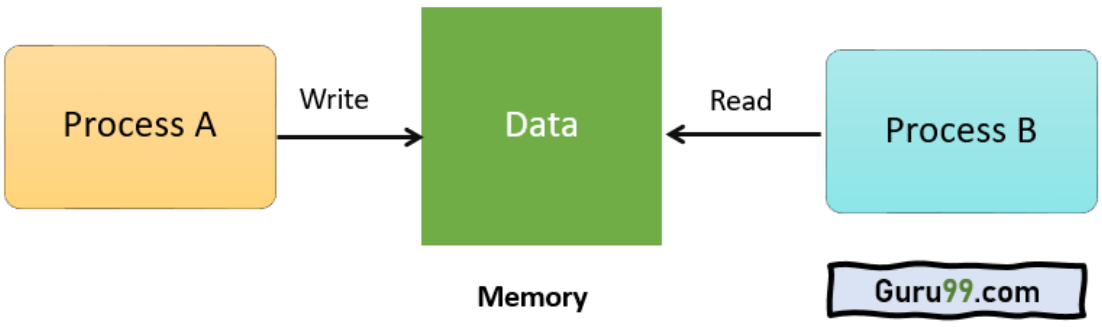
What is Process Synchronization?

Process Synchronization is the task of coordinating the execution of processes in a way that no two processes can have access to the same shared data and resources.

It is specially needed in a multi-process system when multiple processes are running together, and more than one processes try to gain access to the same shared resource or data at the same time.
This can lead to the inconsistency of shared data. So the change made by one process not necessarily reflected when other processes accessed the same shared data. To avoid this type of inconsistency of data, the processes need to be synchronized with each other.

How Process Synchronization Works?

For Example, process A changing the data in a memory location while another process B is trying to read the data from the **same** memory location. There is a high probability that data read by the second process will be erroneous.



- **Entry Section:** It is part of the process which decides the entry of a particular process.
- **Critical Section:** This part allows one process to enter and modify the shared variable.

- **Exit Section:** Exit section allows the other process that are waiting in the Entry Section, to enter into the Critical Sections. It also checks that a process that finished its execution should be removed through this Section.
- **Remainder Section:** All other parts of the Code, which is not in Critical, Entry, and Exit Section, are known as the Remainder Section.

What is Critical Section Problem?

A critical section is a segment of code which can be accessed by a signal process at a specific point of time. The section consists of shared data resources that required to be accessed by other processes.

- The entry to the critical section is handled by the wait() function, and it is represented as P().
- The exit from a critical section is controlled by the signal() function, represented as V().

In the critical section, only a single process can be executed. Other processes, waiting to execute their critical section, need to wait until the current process completes its execution.

Rules for Critical Section

The critical section need to must enforce all three rules:

- **Mutual Exclusion:** Mutual Exclusion is a special type of binary semaphore which is used for controlling access to the shared resource. It includes a priority inheritance mechanism to avoid extended priority inversion problems. Not more than one process can execute in its critical section at one time.
- **Progress:** This solution is used when no one is in the critical section, and someone wants in. Then those processes not in their reminder section should decide who should go in, in a finite time.
- **Bound Waiting:** When a process makes a request for getting into critical section, there is a specific limit about number of processes can get into their critical section. So, when the limit is reached, the system must allow request to the process to get into its critical section.