**Unit-1**

**Introduction:**

**Definition:-**

An operating system acts as an intermediary between the user of a computer and computer hardware. The purpose of an operating system is to provide an environment in which a user can execute programs conveniently and efficiently.

An operating system is software that manages computer hardware. The hardware must provide appropriate mechanisms to ensure the correct operation of the computer system and to prevent user programs from interfering with the proper operation of the system.

In other words, an operating system is a type of software without which you cannot operate or run a computer. It acts as an intermediary or translation system between computer hardware and application programs installed on the computer.

**Functions of Operating system**:-

* Convenience: An OS makes a computer more convenient to use.
* Efficiency: An OS allows the computer system resources to be used efficiently.
* Ability to Evolve: An OS should be constructed in such a way as to permit the effective development, testing, and introduction of new system functions at the same time without interfering with service.
* Throughput: An OS should be constructed so that It can give maximum throughput(Number of tasks per unit time).

Major Functionalities of Operating System:-

* **Memory management**: It manages both the primary and secondary memory such as RAM, ROM, hard disk, pen drive, etc. It checks and decides the allocations and deallocation of memory space to different processes. When a user interacts with a system, the CPU is supposed to read or write operations, in this case, OS decides the amount of memory to be allocated for loading the program instructions and data into RAM. After this program is terminated, the memory area is again free and is ready to be allocated to other programs by the OS.
* **Processor Management**: It facilitates processor management, where it decides the order for the processes to access the processor as well as decides the processing time to be allocated for each process. Besides this, it monitors the status of processes, frees the processor when a process is executed then allocates it to a new process.
* **Device/ hardware management**: The operating system also contains drivers to manage devices. A driver is a type of translation software that allows the operating system to communicate with devices, and there are different drivers for different devices as each device speaks a different language.
* **Run software applications**: It offers the environment to run or use software applications developed to perform specific tasks, for example, Ms Word, Ms Excel, Photoshop, etc.
* **Data management**:- It helps in data management by offering and displaying directories for data management. You can view and manipulate files, folders, e.g., you can move, copy, name, or rename, delete a file or a folder.
* **Evaluates the system's health**: It gives us an idea about the performance of the hardware of the system. For example, you can see how busy the CPU is, how fast the data is retrieved from the hard disk, etc.
* **Provides user interface**: It acts as an interface between the user and the hardware. It can be a GUI where you can see and click elements on the screen to perform various tasks. It enables you to communicate with the computer even without knowing the computer's language.
* **I/O management**: It manages the input output devices and makes the I/O process smooth and effective. For example, it receives the input provided by the user through an input device and stores it in the main memory. Then it directs the CPU to process this input and accordingly provides the output through an output device such as a monitor.
* **Security**: It has a security module to protect the data or information stored in the memories of the computer against malware and unauthorized access. Thus, it not only manages your data but also helps to protect it.
* **Time Management**: It helps CPU in time management. The Kernel OS keeps checking the frequency of processes that requests CPU time. When two or more processes that are equally important compete for the CPU time, then the CPU time is sliced into segments and allocated to these processes in a round-robin fashion to prevent a single process from monopolizing the CPU.
* **Deadlock Prevention**: Sometimes a resource that is supposed to be shared by two or more processes is held by one process due to which the resource cannot continue. This situation is known as deadlock. The OS does not let this situation arise by carefully distributing the resources among the different processes.
* **Interrupt Handling**: OS also responds to interrupts, which are signals generated by a program or a device to seek the attention of the CPU. The OS checks the priority of the interrupt, and if it is more important than the currently running process, it stops the execution of the current process and preserves this state of CPU then executes the requested process. Thereafter the CPU returns to the same state where it was stopped.

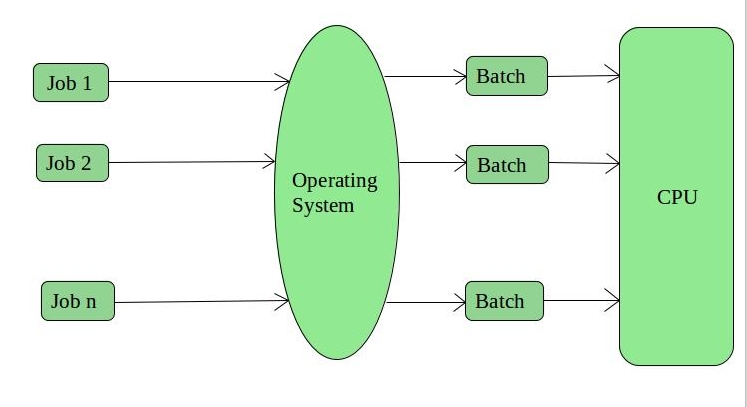
**The process operating system as User Interface:**

* User
* System and application programs
* Operating system
* Hardware

**Types of Operating System:-**

* **Batch OS**
* **Time-Sharing OS**
* **Distributed OS**
* **Real-time OS**
* **Network OS**
* **Multitasking OS**
* **Multiprocessing**
* **Batch Operating System –**

This type of operating system does not interact with the computer directly. There is an operator which takes similar jobs having the same requirement and groups them into batches. It is the responsibility of the operator to sort jobs with similar needs.



**Advantages of Batch Operating System:**

* It is very difficult to guess or know the time required for any job to complete. Processors of the batch systems know how long the job would be when it is in queue.
* Multiple users can share the batch systems
* The idle time for the batch system is very less
* It is easy to manage large work repeatedly in batch systems

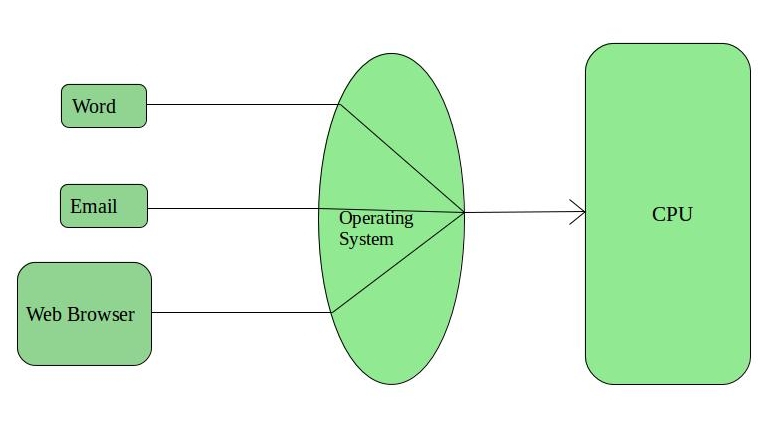
**Disadvantages of Batch Operating System:**

* The computer operators should be well known with batch systems
* Batch systems are hard to debug
* It is sometimes costly
* The other jobs will have to wait for an unknown time if any job fails

**Examples of Batch based Operating System: Payroll System, Bank Statements, etc.**

* **Time-Sharing Operating Systems –**

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.



**Advantages of Time-Sharing OS:**

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

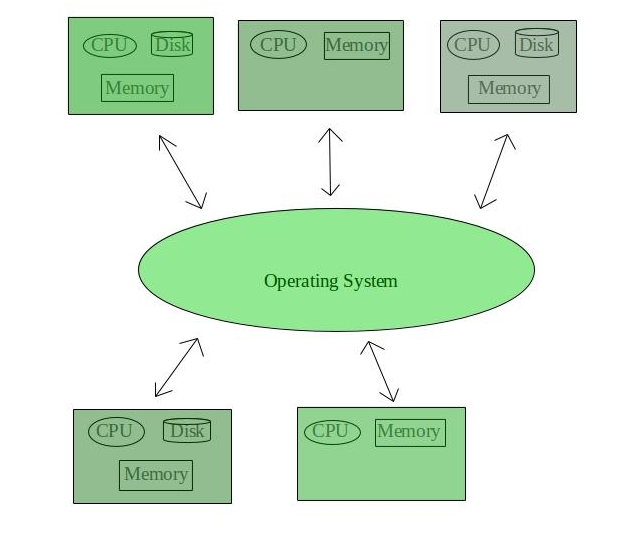
**Disadvantages of Time-Sharing OS:**

* 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.

**Examples of Time**-Sharing OS are Multics, Unix, etc.

* **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.



### **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.

**Examples of Distributed Operating systems are**- LOCUS, etc.

* **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 time interval required to process and respond to inputs is very small. This time interval is called **response time**.

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.

**Two types of Real-Time Operating systems are as follows:**

* Hard Real-Time Systems:

These OSs are meant for applications where time constraints are very strict and even the shortest possible delay is not acceptable. These systems are built for saving life like automatic parachutes or airbags which are required to be readily available in case of an accident. Virtual memory is rarely found in these systems.

* Soft Real-Time Systems:

These OSs are for applications where time-constraint is less strict.

**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.
* **Network Operating System:-**

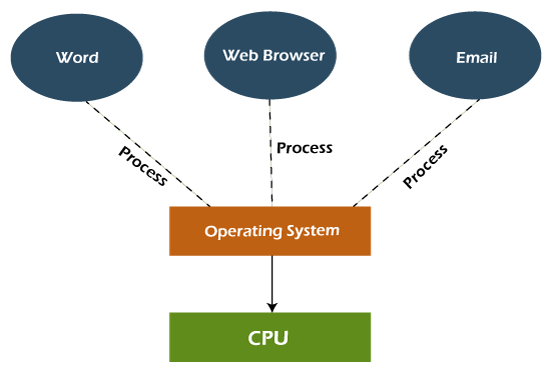
An Operating system, which includes software and associated protocols to communicate with other computers via a network conveniently and cost-effectively, is called Network Operating System.

**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.

**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.

**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.

**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.

**System call:-**

In computing, a system call (commonly abbreviated to ‘syscall’) is the programmatic way in which a computer program requests a service from the kernel of the operating system on which it is executed. This may include hardware-related services(for example, accessing a hard disk drive or accessing the device’s camera), creation and execution of new processes, and communication with integral kernel services such as ‘process scheduling’. System calls provide an essential interface between a process and the operating system. In most systems, system calls can only be made from ‘userspace’ processes, while in some systems,’ OS/360 and successors’ for example, privileged system code also issues 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:- it is the system call that is used to direct the processes. Some process control examples including creating,load,abort,end,execute,process,terminate the process etc.

Device management:- it 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:- it is a system call that is used to maintain information. There are some examples of information maintenance, including getting system data, setting time or date, getting time or date, setting system data, etc.

Communication:- communication is a system call that is used for communication. There are some examples of communication, including creating, deleting communication connections, sending, receiving messages, etc.

**System programs:-** It can be defined as the act of building systems software using system programming languages. According to the computer hierarchy, one which comes, at last, is hardware. Then it is the operating system, system programs, and finally application programs. Program development and execution can be done conveniently in system programs. Some of the system programs are simply user interfaces, others are complex. It traditionally lies between the user interface and system calls.

**System programs can be divided into these categories:-**

**1. File management:** A file is a collection of specific information stored in the memory of a computer system. File management is defined as the process of manipulating files in the computer system, its management includes the process of creating, modifying, and deleting files.

**2. Status information:** information like date, time amount of available memory, or disk space is asked by some users. Others provide detailed performance, logging, and debugging information which is more complex.

**3. File modification:** for modifying the contents of files we use this. For files stored on disks or other storage devices, we used different types of editors.

**4. Programming-language support:** for common programming languages we use compilers, assemblers, debuggers, and interpreters which are already provided to the users. It provides all support to users.

**5. Programing loading and execution:** when the program is ready after the assembling and compilation, it muswt be loaded into memory for execution. A loader is a part of an operating system that is responsible for loading programs and libraries**.**

**6. Communications:** Virtual connections among processes, users, and computer systems are provided by programs. Users can send messages to another user on their screen, user can send e-mail, browsing on web pages, remote login, the transformation of files from one user to another.

**Some examples of system programs in os are:**

**· Windows 10**

**· Mac OS X**

**· Ubuntu**

**· Linux**

**· Android**

**Process management :-**

**Process Concept-**

The java.lang.Process is a subclass of Object class and it can describe the processes that are started by the exec() method of Runtime class. A Process object controls the process and gets information about it. The Process class is an abstract class, therefore, it cannot be instantiated. The important method s of the Process class are destroy(), exitValue(), getErrorStream(), waitFor(), getInputStream() and getOutputStream()**.**

## **Syntax:-**

**public abstract class Process extends Object**

Example-

import java.util.concurrent.\*;

public class ProcessTest {

public static void main(String[] args) throws Exception {

Runtime runtime = Runtime.getRuntime();

System.out.println("Launching of Notepad Application");

**Process process = runtime.exec("Notepad.exe"); // Launch a Notepad application**

**System.out.println("Wait for 5 seconds");**

**p.waitFor(5, TimeUnit.SECONDS);**

**System.out.println("Exit of Notepad Application");**

**process.destroy(); // destroy the application**

}

}

n the above program, we are implementing a Process class. Whenever we can call the **exec("Notepad.exe")** method of **Runtime** class, it launches the **notepad application** and destroys the application after 5 seconds.

## **Output**

Launching of Notepad Application

Wait for 5 seconds

Exit of Notepad Application

**Process Scheduling:**

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 a Multiprogramming operating system. 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.

## **Process Scheduling Queues**

**T**he OS maintains all 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 the 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 the unavailability of an I/O device constitute this queue

****

**COOPERATING PROCESS IN OS**

**COOPERATING PROCESS:**

The processes executing in a system may be either independent or co-operating processes.

**Independent process:**

· A process is independent if it cannot affect other processes or be affected by them.

· Any process that does not share data with others is independent

**Cooperating process:**

· If it can affect Or affected by the other processes executing in the system

· Shares data with other processes

· Cooperation is done to provide information sharing, computational speedups, modularity, and convenience.

· To allow cooperation there should be some mechanism for communication called IPC: Inter-Process Comm.) and to synchronize their actions.

**Advantages of the cooperating process:**

· Information sharing-Several users may which to share the same information e.g. a shared file. The O/S needs to provide a way of allowing concurrent access.

· Computational speedup-Some problems can be solved quicker by subdividing them into smaller tasks that can be executed in parallel on several processors.

· Modularity-The solution of a problem is structured into parts with well-defined interfaces, and where the parts run in parallel.

· Convenience-A user may be running multiple processes to achieve a single goal, or where a utility may invoke multiple components, which interconnect via a pipe structure that attaches the stdout of one stage to stdin of the next etc.

**Example**:

Below is a very simple example of two cooperating processes. The problem is called the Producer Consumer problem and it uses two processes, the producer and the consumer.

**Producer Process**: It produces information that will be consumed by the consumer.

**Consumer Process**: It consumes information produced by the producer.

* Both processes run concurrently. If the consumer has nothing to consume, it waits.
* There are two versions of the producer. In version one, the producer can produce an infinite amount of items. This is called the Unbounded Buffer Producer-Consumer Problem.
* In the other version, there is a fixed limit to the buffer size.
* When the buffer is full, the producer must wait until there is some space in the buffer before it can produce a new item.
* To allow producer and consumer processes to run concurrently, we must have available a buffer of items that can be filled by the producer and emptied by the consumer.
* A producer can produce one item while the consumer is consuming another item.
* The producer and consumer must be synchronized so that the consumer does not try to consume an item that has not yet been produced.
* In this situation, the consumer must wait until an item is produced.
* The unbounded-buffer producer i.e. the consumer problem places no practical limit on the size of the buffer.
* The consumer may have to wait for new items, but the producer can always produce new items.
* The bounded-buffer producer-consumer problem assumes that there is fixed buffer size.
* In this case, the consumer must wait if the buffer is empty and the producer must wait if the buffer is full.
* The buffer may be either provided by the operating system through the use of IPC (Inter-Process Communication) or explicitly coded by the application programmer with the use of shared memory

**Example**:

A print program produces characters that are consumed by the printer driver. A compiler may produce

assembly code, which is consumed by an assembler. The assembler, in turn, may produce

object modules, which are consumed by the loader.

**Reasons for needing cooperating processes**

There may be many reasons for the requirement of cooperating processes. Some of these are given as follows −

**Modularity**

Modularity involves dividing complicated tasks into smaller subtasks. These subtasks can completed by different cooperating processes. This leads to faster and more efficient completion of the required tasks.

**Information Sharing**

Sharing of information between multiple processes can be accomplished using cooperating processes. This may include access to the same files. A mechanism is required so that the processes can access the files in parallel to each other.

**Convenience**

There are many tasks that a user needs to do such as compiling, printing, editing, etc. It is convenient if these tasks can be managed by cooperating processes.

**Computation Speedup**

Subtasks of a single task can be performed parallelly using cooperating processes. This increases the computation speedup as the task can be executed faster. However, this is only possible if the system has multiple processing elements.

**Methods of Cooperation**

Cooperating processes can coordinate with each other using shared data or messages. Details about these are given as follows −

**Cooperation by Sharing**

The cooperating processes can cooperate with each other using shared data such as memory, variables, files, databases, etc. The critical section is used to provide data integrity and writing is mutually exclusive to prevent inconsistent data.

**Cooperation by Communication**

The cooperating processes can cooperate with each other using messages. This may lead to deadlock if each process is waiting for a message from the other to perform an operation. Starvation is also possible if a process never receives a message.

**A cooperating process is one that can affect or be affected by other process executing in the system cooperating process an:**

**Directly share a logical address data space (i.e. code & data) -** This may result in data inconsistency. It is implemented on threads.

**Share data only through files/ messages** - So we will deal with various to order....orderly execution of cooperating process so that data consistency is maintained.

## **THREADS**

## **Java Threads**

Threads allow a program to operate more efficiently by doing multiple things at the same time.

Threads can be used to perform complicated tasks in the background without interrupting the main program.

## Creating a Thread

There are two ways to create a thread.

It can be created by extending the Thread class and overriding its run() method:

### Extend Syntax

public class Main extends Thread {

public void run() {

System.out.println("This code is running in a thread");

}

}

Another way to create a thread is to implement the Runnable interface:

### Implement Syntax

public class Main implements Runnable {

public void run() {

System.out.println("This code is running in a thread");

}

}

## Running Threads

If the class extends the Thread class, the thread can be run by creating an instance of the class and call its start() method:

### Extend Example

public class Main extends Thread {

public static void main(String[] args) {

Main thread = new Main();

thread.start();

System.out.println("This code is outside of the thread");

}

public void run() {

System.out.println("This code is running in a thread");

}}

# 

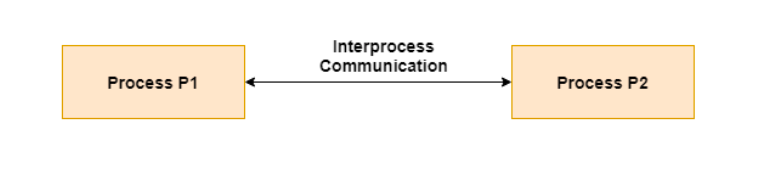
# 

# 

# **Interprocess Communication**

Interprocess 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 interprocess communication is as follows −



## **Synchronization in Interprocess Communication**

Synchronization is a necessary part of interprocess communication. It is either provided by the interprocess 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.

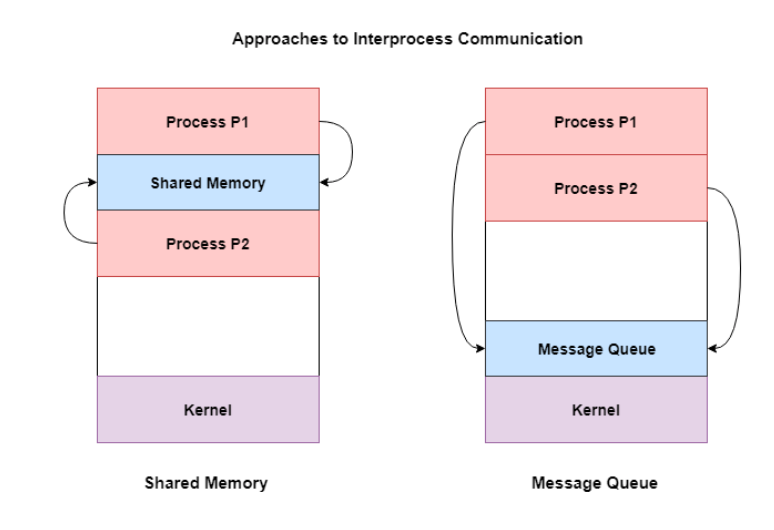
## 

## **Approaches to Interprocess Communication**

The different approaches to implement interprocess 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 interprocess 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 interprocess 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 interprocess communication and are used by most operating systems.

A diagram that demonstrates message queue and shared memory methods of interprocess communication is as follows −



## **CPU scheduling**

**CPU Scheduling** is a process of determining which process will own CPU for execution while another process is on hold. The main task of CPU scheduling is to make sure that whenever the CPU remains idle, the OS at least select one of the processes available in the ready queue for execution. The selection process will be carried out by the CPU scheduler. It selects one of the processes in memory that are ready for execution.

**Types of CPU Scheduling**

Here are two kinds of Scheduling methods:

### **Preemptive Scheduling**

In Preemptive Scheduling, the tasks are mostly assigned with their priorities. Sometimes it is important to run a task with a higher priority before another lower priority task, even if the lower priority task is still running. The lower priority task holds for some time and resumes when the higher priority task finishes its execution.

### **Non-Preemptive Scheduling**

In this type of scheduling method, the CPU has been allocated to a specific process. The process that keeps the CPU busy will release the CPU either by switching context or terminating. It is the only method that can be used for various hardware platforms. That’s because it doesn’t need special hardware (for example, a timer) like preemptive scheduling.

## **Important CPU scheduling Terminologies**

* **Burst Time/Execution Time:** It is a time required by the process to complete execution. It is also called running time.
* **Arrival Time:** when a process enters in a ready state
* **Finish Time:** when process complete and exit from a system
* **Multiprogramming:** A number of programs which can be present in memory at the same time.
* **Jobs:** It is a type of program without any kind of user interaction.
* **User:** It is a kind of program having user interaction.
* **Process:** It is the reference that is used for both job and user.
* **CPU/IO burst cycle:** Characterizes process execution, which alternates between CPU and I/O activity. CPU times are usually shorter than the time of I/O.

There are mainly six types of process scheduling algorithms

1. First Come First Serve (FCFS)

2. Shortest-Job-First (SJF) Scheduling

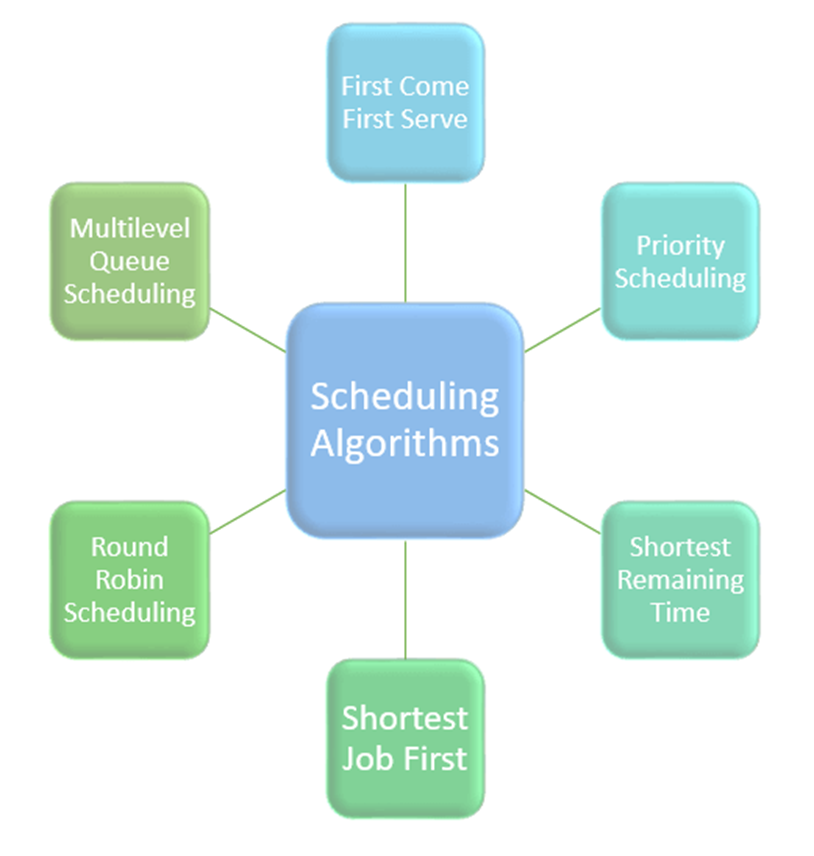
**Types of CPU scheduling Algorithm**

4. Shortest Remaining Time

5. Priority Scheduling

6. Round Robin Scheduling

7. Multilevel Queue Scheduling

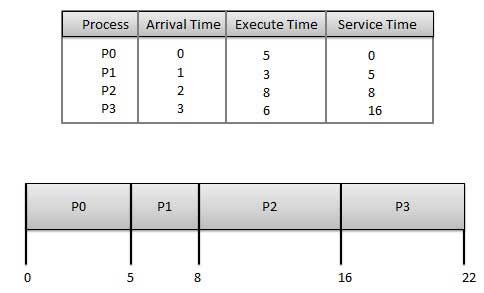


## **Scheduling algorithms**

## **First Come First Serve**

First Come First Serve is the full form of FCFS. It is the easiest and most simple CPU scheduling algorithm. In this type of algorithm, the process which requests the CPU gets the CPU allocation first. This scheduling method can be managed with a FIFO queue.

As the process enters the ready queue, its PCB (Process Control Block) is linked with the tail of the queue. So, when the CPU becomes free, it should be assigned to the process at the beginning of the queue.



**The wait time** of each process is as follows −

| **Process** | **Wait Time : Service Time - Arrival Time** |
| --- | --- |
| P0 | 0 - 0 = 0 |
| P1 | 5 - 1 = 4 |
| P2 | 8 - 2 = 6 |
| P3 | 16 - 3 = 13 |

Average Wait Time: (0+4+6+13) / 4 = 5.75

### **Characteristics of FCFS method:**

* It offers a non-preemptive and pre-emptive scheduling algorithm.
* Jobs are always executed on a first-come, first-serve basis
* It is easy to implement and use.
* However, this method is poor in performance, and the general wait time is quite high.

## **Shortest Remaining Time**

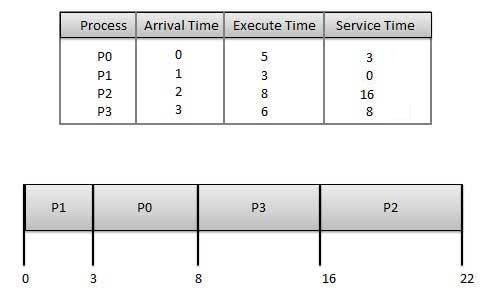
The full form of SRT is Shortest remaining time. It is also known as SJF preemptive scheduling. In this method, the process will be allocated to the task, which is closest to its completion. This method prevents a newer ready state process from holding the completion of an older process.

### **SRT scheduling method:**

* This is also known as a shortest job first, or SJF
* This is a non-preemptive, pre-emptive scheduling algorithm.
* The best approach to minimize waiting time.
* Easy to implement in Batch systems where required CPU time is known in advance.
* Impossible to implement in interactive systems where the required CPU time is not known.
* The processer should know in advance how much time the process will take.

Given: Table of processes, and their Arrival time, Execution time

| **Process** | **Arrival Time** | **Execution Time** | **Service Time** |
| --- | --- | --- | --- |
| P0 | 0 | 5 | 0 |
| P1 | 1 | 3 | 5 |
| P2 | 2 | 8 | 14 |
| P3 | 3 | 6 | 8 |



**Waiting tim**e of each process is as follows −

| **Process** | **Waiting Time** |
| --- | --- |
| P0 | 0 - 0 = 0 |
| P1 | 5 - 1 = 4 |
| P2 | 14 - 2 = 12 |
| P3 | 8 - 3 = 5 |

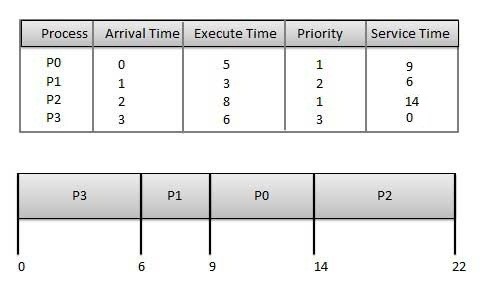
Average Wait Time: (0 + 4 + 12 + 5)/4 = 21 / 4 = 5.25

## **Priority Based Scheduling**

* Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
* Each process is assigned a priority. The process with the highest priority is to be executed first and so on.
* Processes with the same priority are executed on a first come first-served basis.
* Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Given: Table of processes, and their Arrival time, Execution time, and priority. Here we are considering 1 as the lowest priority.

| **Process** | **Arrival Time** | **Execution Time** | **Priority** | **Service Time** |
| --- | --- | --- | --- | --- |
| P0 | 0 | 5 | 1 | 0 |
| P1 | 1 | 3 | 2 | 11 |
| P2 | 2 | 8 | 1 | 14 |
| P3 | 3 | 6 | 3 | 5 |



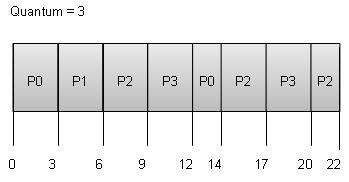
Waiting time of each process is as follows −

| **Process** | **Waiting Time** |
| --- | --- |
| P0 | 0 - 0 = 0 |
| P1 | 11 - 1 = 10 |
| P2 | 14 - 2 = 12 |
| P3 | 5 - 3 = 2 |

Average Wait Time: (0 + 10 + 12 + 2)/4 = 24 / 4 = 6

## **Round-Robin Scheduling**

* Round Robin is the preemptive process scheduling algorithm.
* Each process is provided a fixed time to execute, it is called a quantum.
* Once a process is executed for a given time period, it is preempted and another process executes for a given time period.
* Context switching is used to save states of preempted processes.



Wait time of each process is as follows −

| **Process** | **Wait Time : Service Time - Arrival Time** |
| --- | --- |
| P0 | (0 - 0) + (12 - 3) = 9 |
| P1 | (3 - 1) = 2 |
| P2 | (6 - 2) + (14 - 9) + (20 - 17) = 12 |
| P3 | (9 - 3) + (17 - 12) = 11 |

Average Wait Time: (9+2+12+11) / 4 = 8.5.

## **Multiple-Level Queues Scheduling**

This algorithm separates the ready queue into various separate queues. In this method, processes are assigned to a queue based on a specific property of the process, like the process priority, size of the memory, etc.

However, this is not an independent scheduling OS algorithm as it needs to use other types of algorithms in order to schedule the jobs.

### **Characteristic of Multiple-Level Queues Scheduling:**

* Multiple queues should be maintained for processes with some characteristics.
* Every queue may have its separate scheduling algorithms.
* Priorities are given for each queue.

## 

## **The Purpose of a Scheduling algorithm**

Here are the reasons for using a scheduling algorithm:

* The CPU uses scheduling to improve its efficiency.
* It helps you to allocate resources among competing processes.
* The maximum utilization of the CPU can be obtained with multi-programming.
* The processes which are to be executed are in the ready queue.

**Real-Time Scheduling and Algorithm evaluation.**

Real-time [systems](https://www.geeksforgeeks.org/real-time-systems/) are systems that carry real-time tasks. These tasks need to be performed immediately with a certain degree of urgency. In particular, these tasks are related to control of certain events (or) reacting to them. Real-time tasks can be classified as hard real-time tasks and soft real-time tasks.

If a preemptive scheduler is used, the real-time task needs to wait until its corresponding tasks time slice completes. In the case of a non-preemptive scheduler, even if the highest priority is allocated to the task, it needs to wait until the completion of the current task. This task can be slow (or) of the lower priority and can lead to a longer wait.

The scheduling algorithm is classified as follows:

1. **Static table-driven approaches:**   
   These algorithms usually perform a static analysis associated with scheduling and capture the schedules that are advantageous. This helps in providing a schedule that can point out a task with which the execution must be started at run time.
2. **Static priority-driven preemptive approaches:**   
   Similar to the first approach, these type of algorithms also uses static analysis of scheduling. The difference is that instead of selecting a particular schedule, it provides a useful way of assigning priorities among various tasks in preemptive scheduling.
3. **Dynamic planning-based approaches:**   
   Here, the feasible schedules are identified dynamically (at run time). It carries a certain fixed time interval and a process is executed if and only if satisfies the time constraint.
4. **Dynamic best effort approaches:**   
   These types of approaches consider deadlines instead of feasible schedules. Therefore the task is aborted if its deadline is reached. This approach is used widely in most real-time systems.