UNIT 1 : Introduction to OS

What do you mean by Operating System?

An Operating System is a program that acts as an intermediary between a user of a computer and the computer hardware.

The purpose of an Operating System is to provide an environment in which a user can execute program.

An Operating System is an important part of almost every Computer System.

It is basically a control program that controls the execution of user programs to prevent errors and improper use of the computer.

Computer System Architecture:

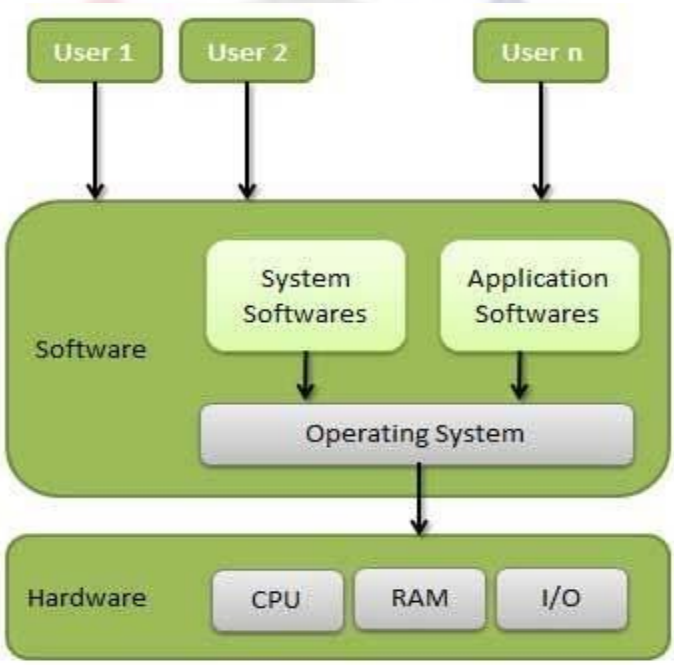
Computer system can be divided into four components

Hardware – provides basic computing resources CPU, memory, I/O devices

Operating system Controls and coordinates use of hardware among various applications and users

Application programs – define the ways in which the system resources are used to solve the computing problems of the users

Word processors, compilers, web browsers, database systems, video games Users People, machines, other computers



Most systems use a single general-purpose processor (PDAs through mainframes)

Most systems have special-purpose processors as well Multiprocessors systems growing in use and importance

Also known as parallel systems, tightly-coupled systems

Services Provided by OS :

An operating system provides an environment for the execution of the program. It provides some services to the programs.

The various services provided by an operating system are as follows:

• Program Execution: The system must be able to load a program into memory and to run that program. The program must be able to terminate this execution either normally or abnormally.

• I/O Operation: A running program may require I/O. This I/O may involve a file or a I/O device for specific device. Some special function can be desired. Therefore the operating system must provide a means to do I/O.

• File System Manipulation: The programs need to create and delete files by name and read and write files. Therefore the operating system must maintain each and every files correctly.

• Communication: The communication is implemented via shared memory or by the technique of message passing in which packets of information are moved between the processes by the operating system.

• Error detection: The operating system should take the appropriate actions for the occurrences of any type like arithmetic overflow, access to the illegal memory location and too large user CPU time. • Research Allocation: When multiple users are logged on to the system the resources must be allocated to each of them. For current distribution of the resource among the various processes the operating system uses the CPU scheduling run times which determine which process will be allocated with the resource. • Accounting: The operating system keep track of which users use how many and which kind of computer resources. • Protection: The operating system is responsible for both hardware as well as software protection. The operating system protects the information stored in a multiuser computer system.

Types of OS :

Operating systems are there from the very first computer generation and they keep evolving with time. In this chapter, we will discuss some of the important types of operating systems which are most commonly used.

Batch operating system

The users of a batch operating system do not interact with the computer directly. Each user prepares his job on an off-line device like punch cards and submits it to the computer operator. To speed up processing, jobs with similar needs are batched together and run as a group. The programmers leave their programs with the operator and the operator then sorts the programs with similar requirements into batches.

The problems with Batch Systems are as follows −

- Lack of interaction between the user and the job.
- CPU is often idle, because the speed of the mechanical I/O devices is slower than the CPU.
- Difficult to provide the desired priority.

Time-sharing operating systems

Time-sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing.

The main difference between Multiprogrammed Batch Systems and Time-Sharing Systems is that in case of Multiprogrammed batch systems, the objective is to maximize processor use, whereas in Time-Sharing Systems, the objective is to minimize response time.

Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response. For example, in a transaction processing, the processor executes each user program in a short burst or quantum of computation. That is, if **n** users are present, then each user can get a time quantum. When the user submits the command, the response time is in few seconds at most.

The operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time. Computer systems that were designed primarily as batch systems have been modified to time-sharing systems.

Advantages of Timesharing operating systems are as follows −

- Provides the advantage of quick response.

- Avoids duplication of software.
- Reduces CPU idle time.

Disadvantages of Time-sharing operating systems are as follows −

- Problem of reliability.
- Question of security and integrity of user programs and data.
- Problem of data communication.

Distributed operating System

Distributed systems use multiple central processors to serve multiple real-time applications and multiple users. Data processing jobs are distributed among the processors accordingly.

The processors communicate with one another through various communication lines (such as high-speed buses or telephone lines). These are referred as **loosely coupled systems** or distributed systems. Processors in a distributed system may vary in size and function. These processors are referred as sites, nodes, computers, and so on.

The advantages of distributed systems are as follows −

- With resource sharing facility, a user at one site may be able to use the resources available at another.
- Speedup the exchange of data with one another via electronic mail.
- If one site fails in a distributed system, the remaining sites can potentially continue operating.
- Better service to the customers.
- Reduction of the load on the host computer.
- Reduction of delays in data processing.

Network operating System

A Network Operating System runs on a server and provides the server the capability to manage data, users, groups, security, applications, and other networking functions. The primary purpose of the network operating

system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or to other networks.

Examples of network operating systems include Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

The advantages of network operating systems are as follows −

- Centralized servers are highly stable.
- Security is server managed.
- Upgrades to new technologies and hardware can be easily integrated into the system.
- Remote access to servers is possible from different locations and types of systems.

The disadvantages of network operating systems are as follows −

- High cost of buying and running a server.
- Dependency on a central location for most operations.
- Regular maintenance and updates are required.

Real Time operating System

A real-time system is defined as a data processing system in which the time interval required to process and respond to inputs is so small that it controls the environment. The time taken by the system to respond to an input and display of required updated information is termed as the **response time**. So in this method, the response time is very less as compared to online processing.

Real-time systems are used when there are rigid time requirements on the operation of a processor or the flow of data and real-time systems can be used as a control device in a dedicated application. A real-time operating system must have well-defined, fixed time constraints, otherwise the system will fail. For example, Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.

There are two types of real-time operating systems.

Hard real-time systems

Hard real-time systems guarantee that critical tasks complete on time. In hard real-time systems, secondary storage is limited or missing and the data is stored in ROM. In these systems, virtual memory is almost never found.

Soft real-time systems

Soft real-time systems are less restrictive. A critical real-time task gets priority over other tasks and retains the priority until it completes. Soft real-time systems have limited utility than hard real-time systems. For example, multimedia, virtual reality, Advanced Scientific Projects like undersea exploration and planetary rovers, etc.

OS Structure :

An operating system is a construct that allows the user application programs to interact with the system hardware. Since the operating system is such a complex structure, it should be created with utmost care so it can be used and modified easily. An easy way to do this is to create the operating system in parts. Each of these parts should be well defined with clear inputs, outputs and functions.

Simple Structure

There are many operating systems that have a rather simple structure. These started as small systems and rapidly expanded much further than their scope. A common example of this is MS-DOS. It was designed simply for a niche amount for people. There was no indication that it would become so popular.

It is better that operating systems have a modular structure, unlike MS-DOS. That would lead to greater control over the computer system and its various applications. The modular structure would also allow the programmers to hide information as required and implement internal routines as they see fit without changing the outer specifications.

Layered Structure

One way to achieve modularity in the operating system is the layered approach. In this, the bottom layer is the hardware and the topmost layer is the user interface.

As seen from the image, each upper layer is built on the bottom layer. All the layers hide some structures, operations etc from their upper layers.

One problem with the layered structure is that each layer needs to be carefully defined. This is necessary because the upper layers can only use the functionalities of the layers below them.

1. **Microkernels**

This structures the operating system by removing all nonessential portions of the kernel and implementing them as system and user level programs.

- Generally they provide minimal process and memory management, and a communications facility.
- Communication between components of the OS is provided by message passing.

The *benefits* of the microkernel are as follows:

- Extending the operating system becomes much easier.
- Any changes to the kernel tend to be fewer, since the kernel is smaller.
- The microkernel also provides more security and reliability.

Main *disadvantage* is poor performance due to increased system overhead from message passing.

2. **Modular structure or approach:**

   It is considered as the best approach for an OS. It involves designing of a modular kernel. The kernel has only set of core components and other services are added as dynamically loadable modules to the kernel either during run time or boot time. It resembles layered structure due to the fact that each kernel has defined and protected interfaces but it is more flexible than the layered structure as a module can call any other module.

UNIT 2 : System Structure

OPERATING SYSTEM AND USER INTERFACE

As already mentioned, in addition to the hardware, a computer also needs a set of programs—an operating system—to control the devices. This page will discuss the following:

There are different kinds of operating systems: such as Windows, Linux and Mac OS

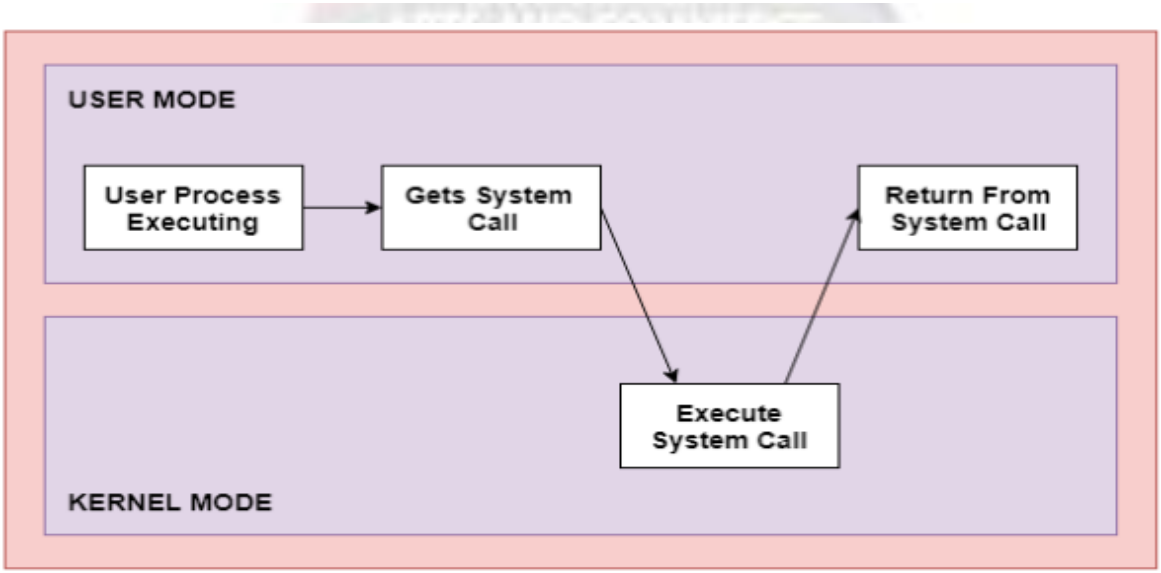
- **There are also different versions of these operating systems,** e.g. Windows 7, 8 and 10
- **Operating systems can be used with different user interfaces (UI)**: text user interfaces (TUI) and graphical user interfaces (GUI) as examples
- **Graphical user interfaces have many similarities in different operating systems**: such as the start menu, desktop etc.

When you can recognize the typical parts of each operating system's user interface, you will mostly be able to use both Windows and Linux as well as e.g. Mac OS.

What is System Call in Operating System?

A **system call** is a mechanism that provides the interface between a process and the operating system. It is a programmatic method in which a computer program requests a service from the kernel of the OS.

System call offers the services of the operating system to the user programs via API (Application Programming Interface). System calls are the only entry points for the kernel system.



Types of System Calls

There are mainly five types of system calls. These are explained in detail as follows −

Process Control

These system calls deal with processes such as process creation, process termination etc.

File Management

These system calls are responsible for file manipulation such as creating a file, reading a file, writing into a file etc.

Device Management

These system calls are responsible for device manipulation such as reading from device buffers, writing into device buffers etc.

Information Maintenance

These system calls handle information and its transfer between the operating system and the user program.

Communication

These system calls are useful for interprocess communication. They also deal with creating and deleting a communication connection.

UNIT 3 : Process Management

Write the Definition of Process.

Ans.: Process Definition : In general, a process is a program in execution. The execution of a process progresses in a sequential fashion. Thus a process is an active entity with a program counter, specifying the next instruction to be executed and a set of associated resources.

Explain the different States of Process with example. OR Describe about the Life Cycle of a Process.

Ans.: Process States : The various stages through which a process passes is called its Life Cycle and this Life Cycle of a process can be divided in to several stages called as "Process States". When a process starts execution, it goes through one state to another state.

Each process may be in one of the following states : (i) New (ii) Ready (iii) Running (iv) Waiting (v) Terminated

New: The process is being created.

• Ready: The process is waiting to be assigned to a processor.

• Running: Instructions are being executed.

• Waiting: The process is waiting for some event to occur.

• Terminated: The process has finished execution.

Explain the Process Control Block (PCB) with diagram.

Ans.: Process Control Block (PCB) : To control the various processes the Operating System maintains a table called the Process Table. The Process Table contains one entry per process. These entries are referred to as "Process Control Block". It contains many pieces of information related with a specific process including the Process Number, Process State, Program Counter, CPU Registers, CPU Scheduling Information, Memory Management Information, Accounting Informa tion, and I/O Status Information.

What is the requirement of CPU Scheduling? Explain.

Ans.: CPU Scheduling : CPU Scheduling is the basis of multi Programmed Operating System. By switching the CPU among the processes, the operating system can make the computer more productive. The main objective of Scheduling is to increase CPU utilization and higher throughput.

What are the different types of Scheduling Queues?

Ans.: The Scheduling Queues in the Systems are : (1) Job Queue : As processes enters in the System, they are put into a job queue. (2) Ready Queue : The processes that are residing in the main memory and are ready and waiting to execute are kept in the ready queue. (3) Device Queue : This queue contains a list of processes waiting for I/O devices.

Write about the different types of Schedulers.

Ans.: Types of schedulers : There are basically three types of schedulers :

(1) Long Term Scheduler : This Scheduler determines which job will be submitted for immediate processing. It selects from the job pool and loads them into memory.

(2) Short Term Scheduler : It is also called a CPU Scheduler. It allocates processes belonging to ready queue to CPU for immediate processing.

(3) Medium Term Scheduler : It is also called as Memory Scheduler. During the execution processes are swapped-out and swapped-in by the Medium Term Scheduler

Context Switch

Context Switch When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process via a context switch Context of a process represented in the PCB Context-switch time is overhead; the system does no useful work while switching Time dependent on hardware support

Process Creation

Parent process create children processes, which, in turn create other processes, forming a tree of processes Generally, process identified and managed via a process identifier (pid) Resource sharing Parent and children share all resources Children share subset of parent's resources Parent and child share no resources Execution Parent and children execute concurrently Parent waits until children terminate Address space Child duplicate of parent Child has a program loaded into it UNIX examples fork system call creates new process exec system call used after a fork to replace the process' memory space with a new program

Process Termination

Process executes last statement and asks the operating system to delete it (exit) Output data from child to parent (via wait) Process' resources are deallocated by operating system Parent may terminate execution of children processes (abort) Child has exceeded allocated resources Task assigned to child is no longer required If parent is exiting Some operating system do not allow child to continue if its parent terminates All children terminated - cascading termination

Interprocess Communication

Processes within a system may be independent or cooperating Cooperating process can affect or be affected by other processes, including sharing data Reasons for cooperating processes: Information sharing Computation speedup Modularity Convenience Cooperating processes need interprocess communication (IPC) Two models of IPC Shared memory Message passing

Interprocess Communication –

Message Passing Mechanism for processes to communicate and to synchronize their actions Message system – processes communicate with each other without resorting to shared variables IPC facility provides two operations: send(message) – message size fixed or variable receive(message) If P and Q wish to communicate, they need to: establish a communication link between them exchange messages via send/receive Implementation of communication link physical (e.g., shared memory, hardware bus) logical (e.g., logical properties)

UNIT 4 : CPU Scheduling

What is 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.

CPU scheduling Concept / 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.

CPU Scheduling Criteria

A CPU scheduling algorithm tries to maximize and minimize the following:

Maximize:

CPU utilization: CPU utilization is the main task in which the operating system needs to make sure that CPU remains as busy as possible. It can range from 0 to 100 percent. However, for the RTOS, it can be range from 40 percent for low-level and 90 percent for the high-level system.

Throughput: The number of processes that finish their execution per unit time is known Throughput. So, when the CPU is busy executing the process, at that time, work is being done, and the work completed per unit time is called Throughput.

Minimize:

Waiting time: Waiting time is an amount that specific process needs to wait in the ready queue.

Response time: It is an amount to time in which the request was submitted until the first response is produced.

Turnaround Time: Turnaround time is an amount of time to execute a specific process. It is the calculation of the total time spent waiting to get into the memory, waiting in the queue and, executing on the CPU. The period between the time of process submission to the completion time is the turnaround time.

Interval Timer

Timer interruption is a method that is closely related to preemption. When a certain process gets the CPU allocation, a timer may be set to a specified interval. Both timer interruption and preemption force a process to return the CPU before its CPU burst is complete.

Most of the multi-programmed operating system uses some form of a timer to prevent a process from tying up the system forever.

What is Dispatcher?

It is a module that provides control of the CPU to the process. The Dispatcher should be fast so that it can run on every context switch. Dispatch latency is the amount of time needed by the CPU scheduler to stop one process and start another.

Functions performed by Dispatcher:

- Context Switching
- Switching to user mode
- Moving to the correct location in the newly loaded program.

Scheduling Algorithms

To decide which process to execute first and which process to execute last to achieve maximum CPU utilization, computer scientists have defined some algorithms, they are:

1. First Come First Serve(FCFS) Scheduling
2. Shortest-Job-First(SJF) Scheduling
3. Priority Scheduling
4. Round Robin(RR) Scheduling
5. Multilevel Queue Scheduling
6. Multilevel Feedback Queue Scheduling

1. First Come, First Served Scheduling (FCFS) Algorithm:

In the "First come first serve" scheduling algorithm, as the name suggests, the process which arrives first, gets executed first, or we can say that the process which requests the CPU first, gets the CPU allocated first.

- First Come First Serve, is just like **FIFO**(First in First out) Queue data structure, where the data element which is added to the queue first, is the one who leaves the queue first.
- This is used in Batch Systems.
- It's **easy to understand and implement** programmatically, using a Queue data structure, where a new process enters through the **tail** of the queue, and the scheduler selects process from the **head** of the queue.
- A perfect real life example of FCFS scheduling is **buying tickets at ticket counter**.

Calculating Average Waiting Time

For every scheduling algorithm, **Average waiting time** is a crucial parameter to judge it's performance.

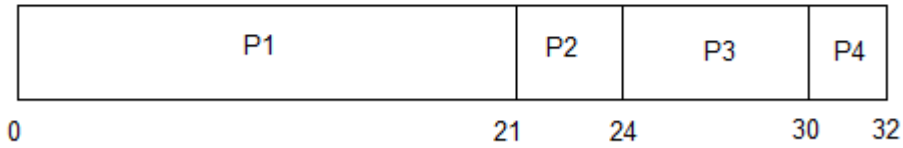
AWT or Average waiting time is the average of the waiting times of the processes in the queue, waiting for the scheduler to pick them for execution.

Lower the Average Waiting Time, better the scheduling algorithm.

Consider the processes P1, P2, P3, P4 given in the below table, arrives for execution in the same order, with **Arrival Time** 0, and given **Burst Time**, let's find the average waiting time using the FCFS scheduling algorithm.

PROCESS	BURST TIME
P1	21
P2	3
P3	6
P4	2

The average waiting time will be = (0 + 21 + 24 + 30)/4 = 18.75 ms



This is the GANTT chart for the above processes

The average waiting time will be 18.75 ms

For the above given proccesses, first **P1** will be provided with the CPU resources,

- Hence, waiting time for **P1** will be 0
- **P1** requires 21 ms for completion, hence waiting time for **P2** will be 21 ms
- Similarly, waiting time for process **P3** will be execution time of **P1** + execution time for **P2**, which will be (21 + 3) ms = 24 ms.
- For process **P4** it will be the sum of execution times of **P1**, **P2** and **P3**.

The **GANTT chart** above perfectly represents the waiting time for each process.

Problems with FCFS Scheduling

Below we have a few shortcomings or problems with the FCFS scheduling algorithm:

1. It is **Non Pre-emptive** algorithm, which means the **process priority** doesn't matter.

   If a process with very least priority is being executed, more like **daily routine backup** process, which takes more time, and all of a sudden some other high priority process arrives, like **interrupt to avoid system crash**, the high priority process will have to wait, and hence in this case, the system will crash, just because of improper process scheduling.

2. Not optimal Average Waiting Time.

3. Resources utilization in parallel is not possible, which leads to **Convoy Effect**, and hence poor resource(CPU, I/O etc) utilization.

2. Shortest Job First(SJF) Scheduling

Shortest Job First scheduling works on the process with the shortest **burst time** or **duration** first.

- This is the best approach to minimize waiting time.
- This is used in Batch Systems.
- It is of two types:
  1. Non Pre-emptive
  2. Pre-emptive
- To successfully implement it, the burst time/duration time of the processes should be known to the processor in advance, which is practically not feasible all the time.
- This scheduling algorithm is optimal if all the jobs/processes are available at the same time. (either Arrival time is 0 for all, or Arrival time is same for all)

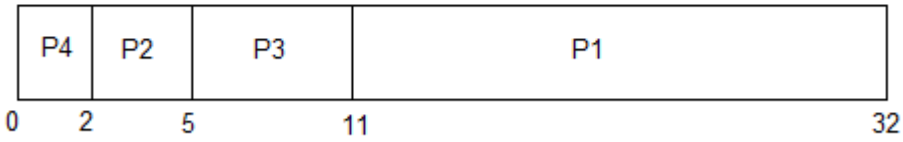
Non Pre-emptive Shortest Job First

Consider the below processes available in the ready queue for execution, with **arrival time** as 0 for all and given **burst times**.

PROCESS	BURST TIME
P1	21
P2	3
P3	6
P4	2

In Shortest Job First Scheduling, the shortest Process is executed first.

Hence the GANTT chart will be following :



Now, the average waiting time will be = (0 + 2 + 5 + 11)/4 = 4.5 ms

As you can see in the **GANTT chart** above, the process **P4** will be picked up first as it has the shortest burst time, then **P2**, followed by **P3** and at last **P1**.

We scheduled the same set of processes using the First come first serve algorithm in the previous tutorial, and got average waiting time to be 18.75 ms, whereas with SJF, the average waiting time comes out 4.5 ms.

Problem with Non Pre-emptive SJF

If the **arrival time** for processes are different, which means all the processes are not available in the ready queue at time 0, and some jobs arrive after some time, in such situation, sometimes process with short burst time have to wait for the current process's execution to finish, because in Non Pre-emptive SJF, on arrival of a process with short duration, the existing job/process's execution is not halted/stopped to execute the short job first.

This leads to the problem of **Starvation**, where a shorter process has to wait for a long time until the current longer process gets executed. This happens if shorter jobs keep coming, but this can be solved using the concept of **aging**.

Pre-emptive Shortest Job First

In Preemptive Shortest Job First Scheduling, jobs are put into ready queue as they arrive, but as a process with **short burst time** arrives, the existing process is preempted or removed from execution, and the shorter job is executed first.

The average waiting time will be,((5-3)+(6-2)+(12-1))/4=8.75

The average waiting time for preemptive shortest job first scheduling is less than both,non preemptive SJF scheduling and FCFS scheduling

As you can see in the **GANTT chart** above, as **P1** arrives first, hence it's execution starts immediately, but just after 1 ms, process **P2** arrives with a **burst time** of 3 ms which is less than the burst time of **P1**, hence the process **P1**(1 ms done, 20 ms left) is preemptied and process **P2** is executed.

As **P2** is getting executed, after 1 ms, **P3** arrives, but it has a burst time greater than that of **P2**, hence execution of **P2** continues. But after another millisecond, **P4** arrives with a burst time of 2 ms, as a result **P2**(2 ms done, 1 ms left) is preemptied and **P4** is executed.

After the completion of **P4**, process **P2** is picked up and finishes, then **P2** will get executed and at last **P1**.

The Pre-emptive SJF is also known as **Shortest Remaining Time First**, because at any given point of time, the job with the shortest remaining time is executed first.

3. Priority CPU Scheduling

In this tutorial we will understand the priority scheduling algorithm, how it works and its advantages and disadvantages.

In the Shortest Job First scheduling algorithm, the priority of a process is generally the inverse of the CPU burst time, i.e. the larger the burst time the lower is the priority of that process.

In case of priority scheduling the priority is not always set as the inverse of the CPU burst time, rather it can be internally or externally set, but yes the scheduling is done on the basis of priority of the process where the process which is most urgent is processed first, followed by the ones with lesser priority in order.

Processes with same priority are executed in FCFS manner.

The priority of process, when internally defined, can be decided based on **memory requirements**, **time limits** ,**number of open files**, **ratio of I/O burst to CPU burst** etc.

Whereas, external priorities are set based on criteria outside the operating system, like the importance of the process, funds paid for the computer resource use, makrte factor etc.

Types of Priority Scheduling Algorithm

Priority scheduling can be of two types:

1. **Preemptive Priority Scheduling**: If the new process arrived at the ready queue has a higher priority than the currently running process, the CPU is preempted, which means the processing of the current process is stoped and the incoming new process with higher priority gets the CPU for its execution.

2. **Non-Preemptive Priority Scheduling**: In case of non-preemptive priority scheduling algorithm if a new process arrives with a higher priority than the current running process, the incoming process is put at the head of the ready queue, which means after the execution of the current process it will be processed.

Example of Priority Scheduling Algorithm

Consider the below table fo processes with their respective CPU burst times and the priorities.

PROCESS	BURST TIME	PRIORITY
P1	21	2
P2	3	1
P3	6	4
P4	2	3

The GANTT chart for following processes based on Priority scheduling will be,

The average waiting time will be, (0 + 3 + 24 + 26)/4 = 13.25 ms

As you can see in the GANTT chart that the processes are given CPU time just on the basis of the priorities.

Problem with Priority Scheduling Algorithm

In priority scheduling algorithm, the chances of **indefinite blocking** or **starvation**.

A process is considered blocked when it is ready to run but has to wait for the CPU as some other process is running currently.

But in case of priority scheduling if new higher priority processes keeps coming in the ready queue then the processes waiting in the ready queue with lower priority may have to wait for long durations before getting the CPU for execution.

In 1973, when the IBM 7904 machine was shut down at MIT, a low-priority process was found which was submitted in 1967 and had not yet been run.

Using Aging Technique with Priority Scheduling

To prevent starvation of any process, we can use the concept of **aging** where we keep on increasing the priority of low-priority process based on the its waiting time.

For example, if we decide the aging factor to be **0.5** for each day of waiting, then if a process with priority **20**(which is comparitively low priority) comes in the ready queue. After one day of waiting, its priority is increased to **19.5** and so on.

Doing so, we can ensure that no process will have to wait for indefinite time for getting CPU time for processing.

4. Round Robin Scheduling

Round Robin(RR) scheduling algorithm is mainly designed for time-sharing systems. This algorithm is similar to FCFS scheduling, but in Round Robin(RR) scheduling, preemption is added which enables the system to switch between processes.

- A fixed time is allotted to each process, called a **quantum**, for execution.
- Once a process is executed for the given time period that process is preempted and another process executes for the given time period.
- Context switching is used to save states of preempted processes.
- This algorithm is simple and easy to implement and the most important is thing is this algorithm is starvation-free as all processes get a fair share of CPU.
- It is important to note here that the length of time quantum is generally from 10 to 100 milliseconds in length.

Some important characteristics of the Round Robin(RR) Algorithm are as follows:

1. Round Robin Scheduling algorithm resides under the category of Preemptive Algorithms.
2. This algorithm is one of the oldest, easiest, and fairest algorithm.
3. This Algorithm is a real-time algorithm because it responds to the event within a specific time limit.
4. In this algorithm, the time slice should be the minimum that is assigned to a specific task that needs to be processed. Though it may vary for different operating systems.
5. This is a hybrid model and is clock-driven in nature.
6. This is a widely used scheduling method in the traditional operating system.

Important terms

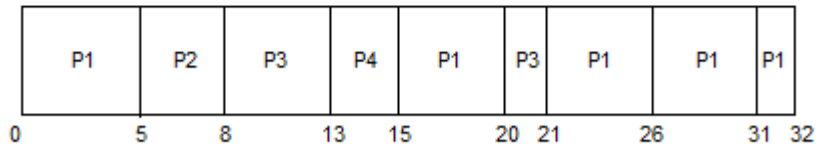
1. **Completion Time** It is the time at which any process completes its execution.
2. **Turn Around Time** This mainly indicates the time Difference between completion time and arrival time. The Formula to calculate the same is: **Turn Around Time = Completion Time – Arrival Time**

3. **Waiting Time(W.T):** It Indicates the time Difference between turn around time and burst time. And is calculated as **Waiting Time = Turn Around Time – Burst Time**

Let us now cover an example for the same:

PROCESS	BURST TIME
P1	21
P2	3
P3	6
P4	2

The GANTT chart for round robin scheduling will be,



The average waiting time will be, 11 ms.

In the above diagram, arrival time is not mentioned so it is taken as 0 for all processes.

Note: If arrival time is not given for any problem statement then it is taken as 0 for all processes; if it is given then the problem can be solved accordingly.

Explanation

The value of time quantum in the above example is 5.Let us now calculate the Turn around time and waiting time for the above example :

Processes	Burst Time	Turn Around Time Turn Around Time = Completion Time – Arrival Time	Waiting Time Waiting Time = Turn Around Time – Burst Time
P1	21	32-0=32	32-21=11
P2	3	8-0=8	8-3=5
P3	6	21-0=21	21-6=15
P4	2	15-0=15	15-2=13

Average waiting time is calculated by adding the waiting time of all processes and then dividing them by no.of processes.

average waiting time = waiting time of all processes/ no.of processes

average waiting time=11+5+15+13/4 = 44/4= **11ms**

Multilevel Queue Scheduling Algorithm

Another class of scheduling algorithms has been created for situations in which processes are easily classified into different groups.

For example, A common division is made between foreground(or interactive) processes and background (or batch) processes. These two types of processes have different response-time requirements, and so might have different scheduling needs. In addition, foreground processes may have priority over background processes.

A multi-level queue scheduling algorithm partitions the ready queue into several separate queues. The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority, or process type. Each queue has its own scheduling algorithm.

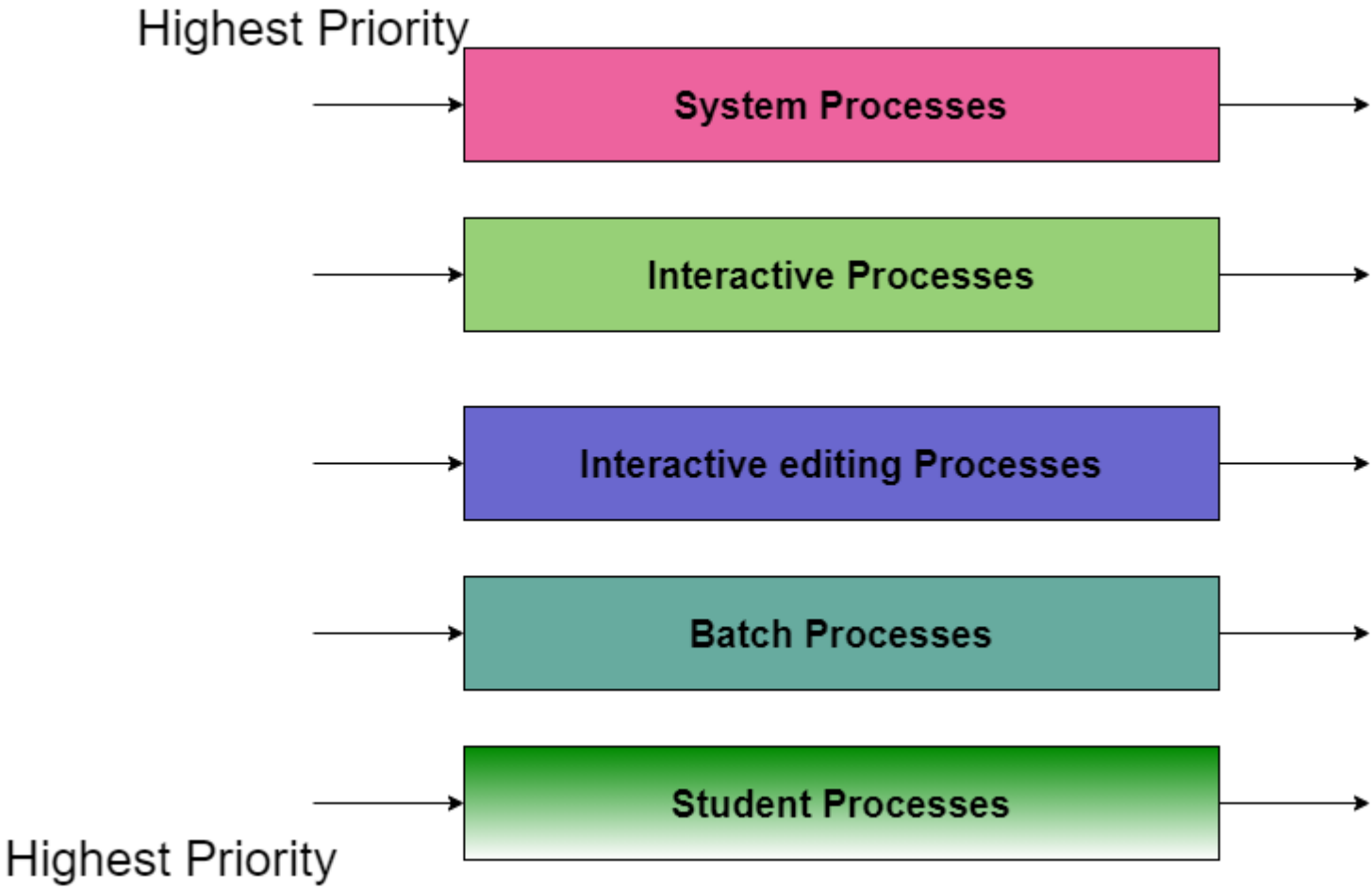
For example, separate queues might be used for foreground and background processes. The foreground queue might be scheduled by the Round Robin algorithm, while the background queue is scheduled by an FCFS algorithm.

In addition, there must be scheduling among the queues, which is commonly implemented as fixed-priority preemptive scheduling. **For example,** The foreground queue may have absolute priority over the background queue.

Let us consider an example of a multilevel queue-scheduling algorithm with five queues:

1. System Processes
2. Interactive Processes
3. Interactive Editing Processes
4. Batch Processes
5. Student Processes

Each queue has absolute priority over lower-priority queues. No process in the batch queue, for example, could run unless the queues for system processes, interactive processes, and interactive editing processes were all empty. If an interactive editing process entered the ready queue while a batch process was running, the batch process will be preempted.



In this case, if there are no processes on the higher priority queue only then the processes on the low priority queues will run. For **Example:** Once processes on the system queue, the Interactive queue, and Interactive editing queue become empty, only then the processes on the batch queue will run.

The Description of the processes in the above diagram is as follows:

- **System Process** The Operating system itself has its own process to run and is termed as System Process.
- **Interactive Process** The Interactive Process is a process in which there should be the same kind of interaction (basically an online game).
- **Batch Processes** Batch processing is basically a technique in the Operating system that collects the programs and data together in the form of the **batch** before the **processing** starts.
- **Student Process** The system process always gets the highest priority while the student processes always get the lowest priority.

In an operating system, there are many processes, in order to obtain the result we cannot put all processes in a queue; thus this process is solved by Multilevel queue scheduling.

Multilevel Feedback Queue Scheduling

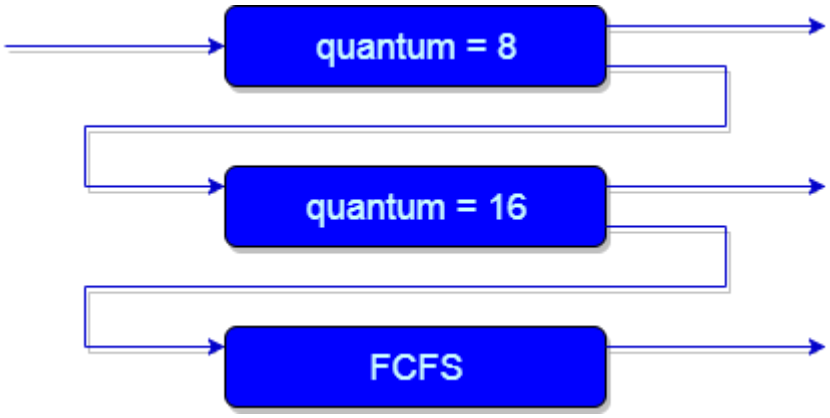
In a multilevel queue-scheduling algorithm, processes are permanently assigned to a queue on entry to the system. Processes do not move between queues. This setup has the advantage of low scheduling overhead, but the disadvantage of being inflexible.

Multilevel feedback queue scheduling, however, allows a process to move between queues. The idea is to separate processes with different CPU-burst characteristics. If a process uses too much CPU time, it will be moved to a lower-priority queue. Similarly, a process that waits too long in a lower-priority queue may be moved to a higher-priority queue. This form of aging prevents starvation.

In general, a multilevel feedback queue scheduler is defined by the following parameters:

- The number of queues.
- The scheduling algorithm for each queue.
- The method used to determine when to upgrade a process to a higher-priority queue.
- The method used to determine when to demote a process to a lower-priority queue.
- The method used to determine which queue a process will enter when that process needs service.

The definition of a multilevel feedback queue scheduler makes it the most general CPU-scheduling algorithm. It can be configured to match a specific system under design. Unfortunately, it also requires some means of selecting values for all the parameters to define the best scheduler. Although a multilevel feedback queue is the **most general scheme**, it is also the **most complex**.



An example of a multilevel feedback queue can be seen in the above figure.

Explanation:

First of all, Suppose that queues 1 and 2 follow round robin with time quantum 8 and 16 respectively and queue 3 follows FCFS. One of the implementations of Multilevel Feedback Queue Scheduling is as follows:

1. If any process starts executing then firstly it enters queue 1.

2. In queue 1, the process executes for 8 unit and if it completes in these 8 units or it gives CPU for I/O operation in these 8 units unit than the priority of this process does not change, and if for some reasons it again comes in the ready queue than it again starts its execution in the Queue 1.
3. If a process that is in queue 1 does not complete in 8 units then its priority gets reduced and it gets shifted to queue 2.
4. Above points 2 and 3 are also true for processes in queue 2 but the time quantum is 16 units. Generally, if any process does not complete in a given time quantum then it gets shifted to the lower priority queue.
5. After that in the last queue, all processes are scheduled in an FCFS manner.
6. It is important to note that a process that is in a lower priority queue can only execute only when the higher priority queues are empty.
7. Any running process in the lower priority queue can be interrupted by a process arriving in the higher priority queue.

Also, the above implementation may differ for the example in which the last queue will follow **Round-robin Scheduling.**

In the above Implementation, there is a problem and that is; Any process that is in the lower priority queue has to suffer starvation due to some short processes that are taking all the CPU time.

And the solution to this problem is : There is a solution that is to boost the priority of all the process after regular intervals then place all the processes in the highest priority queue.

The need for Multilevel Feedback Queue Scheduling(MFQS)

Following are some points to understand the need for such complex scheduling:

- This scheduling is more flexible than Multilevel queue scheduling.
- This algorithm helps in reducing the response time.
- In order to optimize the turnaround time, the SJF algorithm is needed which basically requires the running time of processes in order to schedule them. As we know that the running time of processes is not known in advance. Also, this scheduling mainly runs a process for a time quantum and after that, it can change the priority of the process if the process is long. Thus this scheduling algorithm mainly learns from the past behavior of the processes and then it can predict the future behavior of the processes. In this way, MFQS tries to run a shorter process first which in return leads to optimize the turnaround time.

Advantages of MFQS

- This is a flexible Scheduling Algorithm
- This scheduling algorithm allows different processes to move between different queues.
- In this algorithm, A process that waits too long in a lower priority queue may be moved to a higher priority queue which helps in preventing starvation.

Disadvantages of MFQS

- This algorithm is too complex.
- As processes are moving around different queues which leads to the production of more CPU overheads.
- In order to select the best scheduler this algorithm requires some other means to select the values