Introduction to Operating System

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 in a convenient and efficient manner. .

The operating system as User Interface –

1. User
2. System and application programs
3. Operating system
4. Hardware

The Operating system must support the following tasks. The task are:

1. Provides the facilities to create, modification of programs and data files using an editor.
2. Access to the compiler for translating the user program from high-level language to machine language.
3. Provide a loader program to move the compiled program code to the computer's memory for execution.
4. Provide routines that handle the details of I/O programming.
5. The operating system coordinates the use of the hardware among the various system programs and application programs for various users

I/O System Management –

The module that keeps track of the status of devices is called the I/O traffic controller.

The I/O subsystem consists of

- A memory Management component that includes buffering, caching and spooling.
- A general device driver interface.
- Drivers for specific hardware devices.

Assembler:-An assembler is a program that converts assembly language into machine code. It takes the basic commands and operations from assembly code and converts them into binary code that can be recognized by a specific type of processor.

Compiler:- Compiler transforms code written in a high-level programming language into the machine code, at once, before program runs, whereas an Interpreter coverts each high-level program statement, one by one, into the machine code, during program run.

Loader:- The loader is a program that places programs into memory and prepares them for execution i.e. the assembler outputs the machine language translation of a program on a secondary device and a loader places it in the core.

Important functions of an operating System

Security – It uses password protection to protect user data and similar other techniques and prevents unauthorized access to programs and user data.

Error detecting aids –
Operating system constantly monitors the system to detect errors and avoid the malfunctioning of computer system.

Coordination between other software and users –
It coordinate and assign interpreters, compilers, assemblers and other software to the various users of the computer systems.

Management
OS plays a vital role in management of memory, processors, devices, files etc via deciding the order in which allocation of memory, processors should be done and deallocation when a process is either terminated or no more required. It keeps track of where information is stored, user access settings and status of every file also known as file system.

Types of operating system

S.no.	Type of O.S.	Definition	Advantages	Disadvantages
1	Batch O.S.	There is an operator which takes similar jobs having the same requirement and group them into batches without interacting the computer directly	**1.** Multiple users can share the batch systems **2.**Idle time for the batch system is very less **3.**Easy to manage large work repeatedly in batch systems	**1.**Hard to debug **2.**Sometimes costly **3.**Other jobs will have to wait for an unknown time if any job fails
2	Time sharing O.S.	Each task is given some time known as quantum(After this time interval is over OS switches over to the next task.) to execute so that all the tasks work smoothly.These systems are also known as Multitasking Systems.	**1.**Each task gets an equal opportunity **2.**Fewer chances of duplication of software **3.**CPU idle time can be reduced	**1.**Reliability problem **2.**One must have to take care of the security and integrity of user programs and data **3.**Data communication problem

3	Distributed O.S.	Various autonomous interconnected computers communicate with each other using a shared communication network. Independent systems possess their own memory unit and CPU. These are referred to as “loosely coupled systems”	**1.**Failure of one will not affect the other network communication **2.**Highly fast and durable **3.**Load on host computer reduces	**1.**Failure of the main network will stop the entire communication **2.**Language used to establish this system is not well defined yet **3.**Expensive , highly complex and not understood well yet
4	Network O.S.	These systems run on a server and allow shared access of files, printers, security, applications, and other networking functions over a small private network.These are referred to as tightly coupled systems	**1.**Highly stable centralized servers **2.**Security concerns are handled through servers **3.**New technologies and hardware up-gradation are easily integrated into the system	**1.**Servers are costly **2.**User has to depend on a central location for most operations **3.**Maintenance and updates are required regularly

5	Real time O.S.	These systems fulfil the need of strict time requirements like missile systems, air traffic control systems, robots, etc. Two types of Real-Time Operating System which are as follows: **1.**Hard Real-Time Systems **2.**Soft Real-Time Systems:	**1.**Maximum consumption **2.**Task shifting **3.**Error free	**1.**Limited task **2.**Complex Algorithms **3.**Use heavy system resources

Real Time Systems

Hard real-time system:-This type of system can never miss its deadline as it may have disastrous consequences. Example: Flight controller system.

Soft real-time system:-This type of system can miss its deadline occasionally with some acceptably low probability as it has no disastrous consequences. Example: Telephone switches.

Terms related to real time system:

Job – A job is a small piece of work that can be assigned to a processor and may or may not require resources.

Task – A set of related jobs that jointly provide some system functionality.

Release time of a job – It is the time at which job becomes ready for execution.

Execution time of a job – It is the time taken by job to finish its execution.

Deadline of a job – It is the time by which a job should finish its execution. Deadline is of two types: absolute deadline and relative deadline.

Response time of a job – It is the length of time from release time of a job to the instant when it finishes.

Relative Deadline - Maximum allowable response time of a job.

Absolute deadline - Relative deadline + release time.

Resources which are essential for execution of a job are called active resources such as processors while which may or may not be required for execution are called passive resources such as mutex.
Two resources are identical if they can be used interchangeably else they are heterogeneous.

There are two types of tasks in real-time systems:

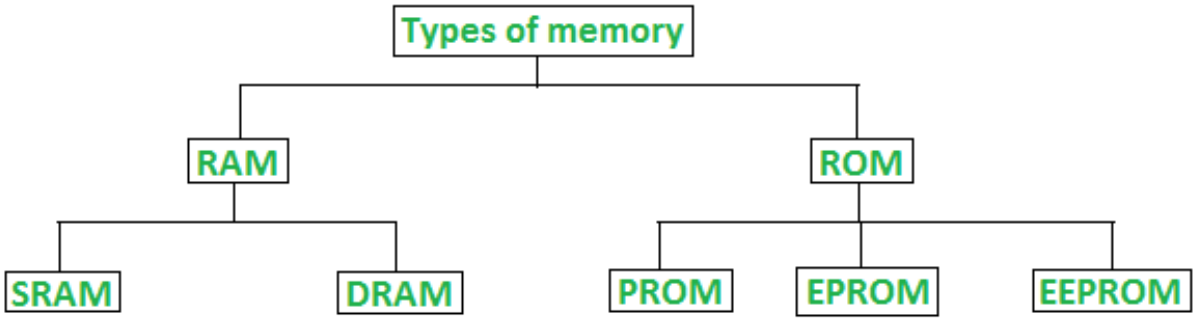
1.Periodic tasks:-jobs are released at regular intervals.

2.Dynamic tasks:-sequential program that is invoked by the occurrence of an event.

Difference between Multiprogramming, multitasking, multithreading and multiprocessing

No.	Characteristic	Multi programming	Multi processing	Multithreading	Multi tasking
1	Definition	The concurrent residency of more than one program in the main memory is called as multi programming.	The availability of more than one processor per system, which can execute several set of instructions in parallel is called as multi processing.	A process is divided into several different sub-processes called threads, which has its own path of execution. This concept is called as multi threading.	The execution of more than one task simultaneously is called as multi tasking.
2	Number of CPU:	One	More than one	Can be one or more than one	One
3	Job processing time:	More time.	Less time.	Moderate amount of time.	Moderate amount of time.
4	No.of process being executed:	One process is executed at a time.	More than one process can be executed at a time	Various components of the same process are being executed at a time.	One by one job is being executed at a time.
5	Throughput:	Less.	Maximum.	Moderate.	Moderate
6	Efficiency:	Less	Maximum	Moderate	Moderate

Memory



Classification of computer memory

RAM :-

It is also called as *read write memory* or the *main memory* or the *primary memory*.It is a volatile memory as the data loses when the power is turned off.

Read Only Memory (ROM) –

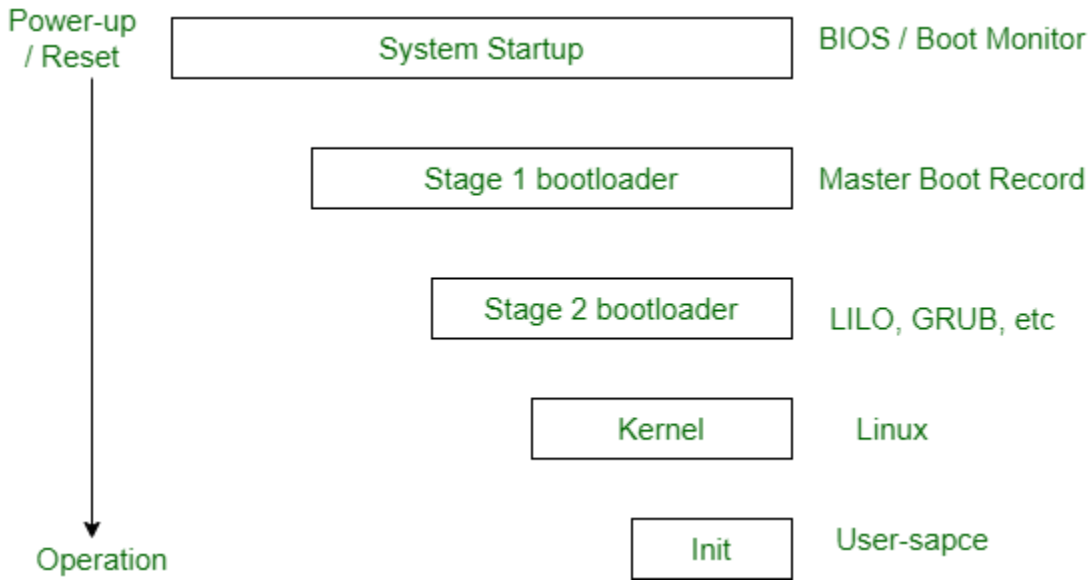
Stores crucial information essential to operate the system, like the program essential to boot the computer.It is not volatile**.**

A major difference between 32-bit processors and 64-bit processors is the number of calculations per second they can perform, which affects the speed at which they can complete tasks. 64-bit processors can come in dual-core, quad-core, six-core, and eight-core versions for home computing. Multiple cores allow for an increased number of calculations per second that can be performed, which can increase the processing power and help make a computer run faster.

What happens when we turn on computer?

The first thing a computer has to do when it is turned on is to start up a special program called an operating system.

An overview of the boot process



The BIOS (Basic Input Output System) chip tells it to look in a fixed place. The **POST** (Power On Self Test) first checks the bios and then tests the CMOS RAM. If there is no problem, then continues to check the CPU, hardware devices. If some errors found then an error message is displayed on the screen or a number of beeps are heard known as POST beep codes.

Boot Block in Operating System

On most of the computer systems, a small piece of code known as bootstrap program locates the kernel, loads it into main memory and starts its execution and this program is stored in ROM because it doesn't require initialization and as the location is fixed so the processor can start executing when powered up or reset. Also, ROM is basically read-only memory and hence it cannot be affected by the computer virus.
But the problem is that changing the bootstrap code basically requires changes in the ROM hardware chips and hence nowadays there is a tiny bootstrap loader program in the boot whose only job is to bring the full bootstrap program from the disk through

which we can easily change the full bootstrap program and the new version can be easily written onto the disk.

The full bootstrap program is stored in the **boot blocks** at a fixed location on the disk which has a boot partition called boot disk. The code in the boot ROM instructs the read controller to read the boot blocks into the memory and then starts the execution of code. The full bootstrap program is basically able to load the complete OS from a non-fixed location on disk to start the os running and even though it is very small.

UEFI(Unified Extensible Firmware Interface)

UEFI like BIOS is a firmware that runs when the computer is booted. It initializes the hardware and loads the os into the memory. However, being the more modern solution and overcoming various limitations of BIOS, UEFI is all set to replace the former.

Limitations of BIOS

- Can boot from drives of less than 2 TB. 3+ TB drives are now standard, and a system with a BIOS can’t boot from them.
- Runs in 16-bit processor mode, and has only 1 MB space to execute.
- Can’t initialize multiple hardware devices at once, thus leading to slow booting process.

Advantages of UEFI over BIOS

- Breaking Out Of Size Limitations
- Speed and performance : UEFI can run in 32-bit or 64-bit mode and has more addressable address space than BIOS.
- More User-Friendly Interface
- Security: It allows only authentic drivers and services to load at boot time, to make sure that no malware can be loaded at computer startup.

System Structure

Kernel:-

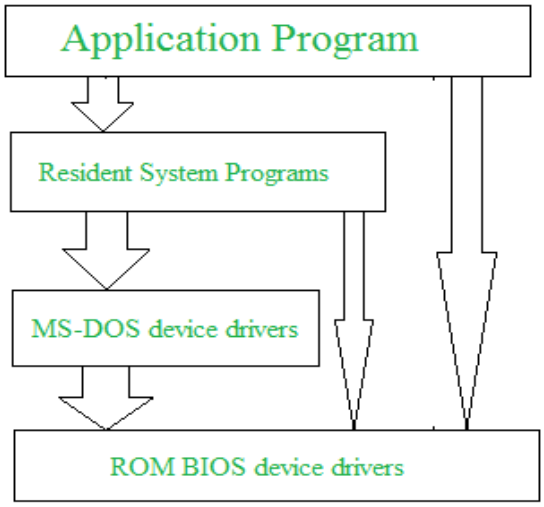
A Kernel is a computer program that is the heart and core of an Operating System. Whenever a system starts, the Kernel is the first program that is loaded after the bootloader because the Kernel has to handle the rest of the system for the Operating System. The Kernel remains in the memory until the Operating System is shut-down.

The Kernel is responsible for low-level tasks such as disk management, memory management, task management, etc.

Types of Operating System:-

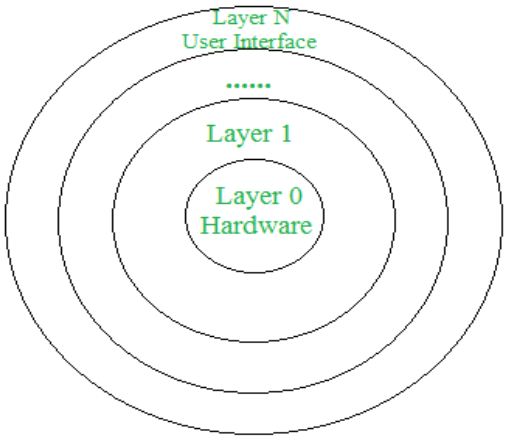
Simple structure:

Such operating systems do not have well defined structure and are small, simple and limited systems. The interfaces and levels of functionality are not well separated. MS-DOS is an example of such an operating system. MS-DOS application programs are able to access the basic I/O routines. These types of operating systems cause the entire system to crash if one of the user programs fails.



Layered structure:

An OS can be broken into pieces and retain much more control on the system. In this structure the OS is broken into a no.of layers (levels). The bottom layer (layer 0) is the hardware and the topmost layer (layer N) is the user interface.These layers are so designed that each layer uses the functions of the lower level layers only.



Advantage:

This simplifies the debugging process as if lower level layers are debugged and an error occurs during debugging then the error must be on that layer only as the lower level layers have already been debugged.

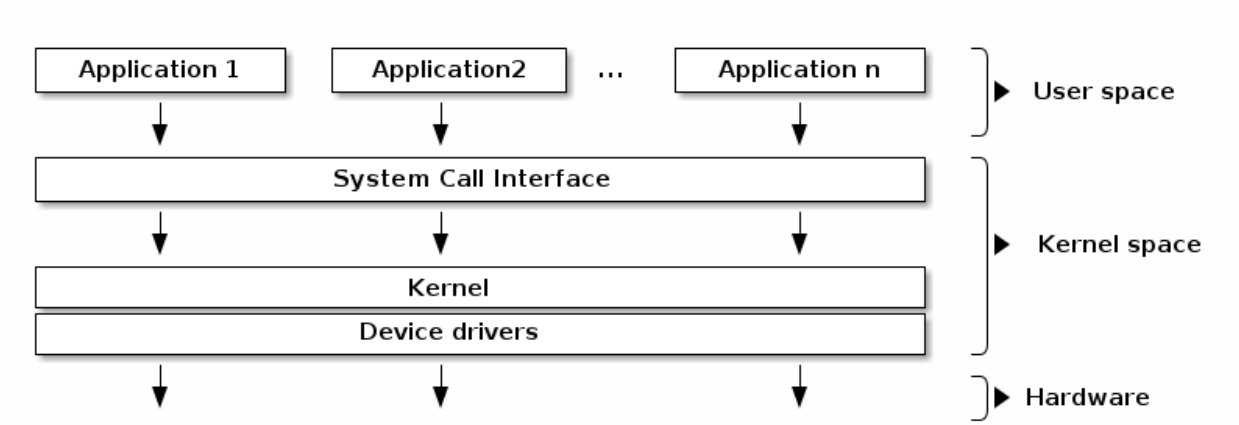
Disadvantage:

The main disadvantage of this structure is that at each layer, the data needs to be modified and passed on which adds overhead to the system. Moreover careful planning of the layers is necessary as a layer can use only lower level layers.

ex.UNIX.

Monolithic Kernel:

This kernel provides CPU scheduling, memory management, file management and other operating system functions through system calls. As both services are implemented under the same address space, this makes operating system execution faster.



Advantage:

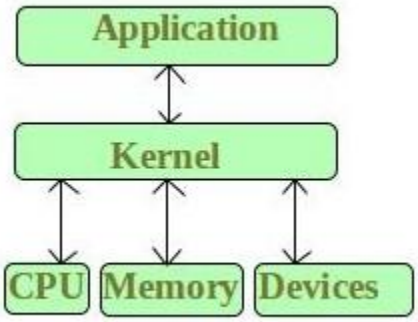
1. It provides CPU scheduling, memory management, file management and other operating system functions through system calls.

Disadvantages:

1. If anyone service fails it leads to entire system failure**.**

Micro-kernel:

This structure designs the operating system by removing all non-essential components from the kernel and implementing them as system and user programs. This results in a smaller kernel called the micro-kernel.



Advantages:
1. The architecture of this kernel is small and isolated hence it can function better.
2. Expansion of the system is easier, it is simply added in the system application without disturbing the kernel.

Disadvantages:
1.Inter process-Communication makes the process completion slow.

Kernel I/O Subsystem in Operating System

Kernel provides many services related to I/O which are as follows :-

1. **I/O Scheduling –** OS developers implement schedules by maintaining a wait queue of the request for each device. When an application issue a blocking I/O system call, the request is placed in the queue after which the scheduler rearranges the order to improve the efficiency of system.

2. **Buffering –** A *buffer* is a memory area that stores data being transferred between two devices or between a device and an application. Buffering

helps in coping with a speed mismatch, adaptation for data having different data-transfer sizes and supporting copy semantics for I/O application.

3. **Caching –** A *cache* is a region of fast memory that holds a copy of data. Buffers may hold only the existing copy of a data item while a cache holds a copy on faster storage of an item that resides elsewhere.

4. **Spooling and Device Reservation –** A *spool* is a buffer that holds the output of a device, such as a printer. The output of all applications wishing to print their output concurrently is spooled in a separate disk file. When an application finishes printing then the spooling system queues the corresponding spool file for output to the printer.
5. **Error Handling –** An OS that uses protected memory can guard against many kinds of hardware and application errors so that a complete system failure is not the usual result of each minor mechanical glitch.

6. **I/O Protection** – To prevent illegal I/O access, we define all I/O instructions to be privileged instructions. The user cannot issue I/O instruction directly.

System call is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. A system call is a way for programs to interact with the operating system.
System call provides the services of the operating system to the user programs via Application Program Interface(API).

Services Provided by System Calls :

1. Process creation and management
2. Main memory management
3. File Access, Directory and File system management
4. Device handling(I/O)
5. Protection
6. Networking, etc.

Process Management

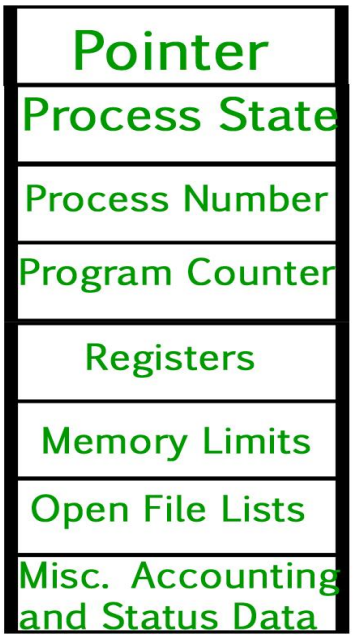
Program vs Process
A process is a program in execution. For example, when we write a program in C or C++ and compile it, the compiler creates binary code. The original code and binary code are both programs. When we actually run the binary code, it becomes a process.

Text Section*:* A Process, sometimes known as the Text Section, also includes the current activity represented by the value of the *Program Counter*.

Stack*:* The stack contains the temporary data, such as function parameters, returns addresses, and local variables.

Data Section*:* Contains the global variable.

Heap Section*:* Dynamically allocated memory to process during its run time.



Process Control Block

Attribute of a process:

1. **Process id:** unique number which will be given to process in a computer.
2. **Program counter :** It will contain the next instruction to be executed.
3. **Process state :** this stores the current states of a process ex. New,ready,waiting etc.
4. **Priority :** It shows the importance of the process. Processes with higher priority will be executed first compared to the processes with lower priority.
5. **General purpose registers :** They should be stored (stages of general purpose registers) so that when the process resumes the previous state (the last state where it is paused) can be regained/restored.
6. **List of the open files**
7. **List of the open devices**

Process control Block :- In this block the entire information about the attributes of a process is stored. Every process gets one PCB.

States of a process:
States are in b/w the creation and death of the process.

1. New: Newly Created Process (or) being-created process.

2. Ready: After creation process moves to Ready state, i.e. the the process is ready for execution.

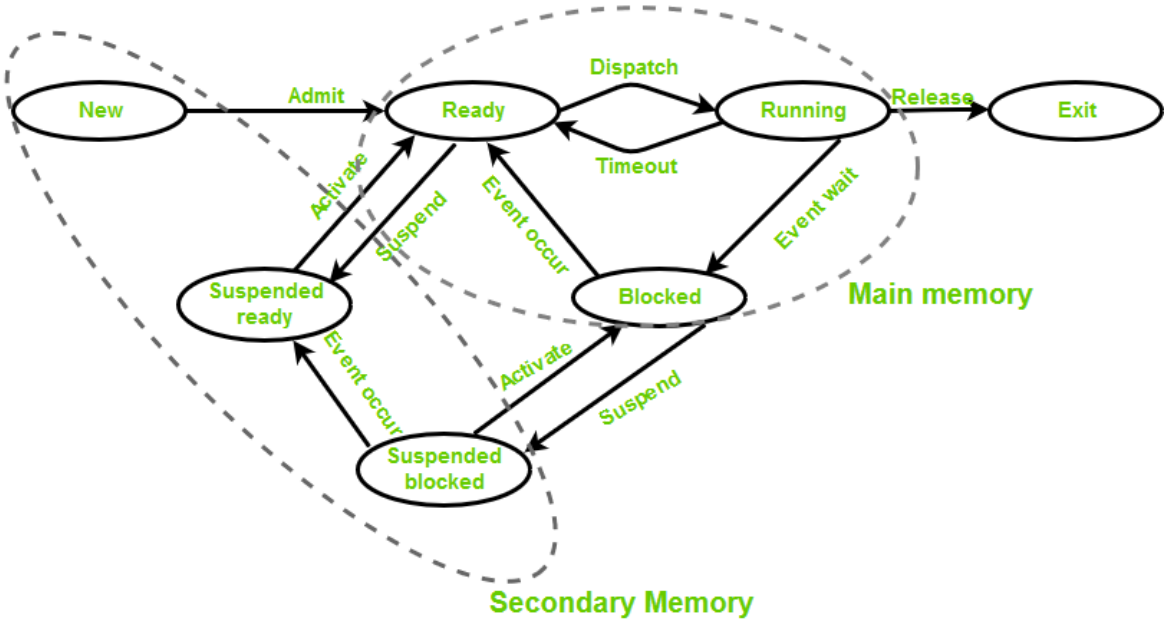
3. Run: Currently running process in CPU (only one process at a time can be under execution in a single processor).

4. Wait (or Block): When a process requests I/O access.

5. Complete (or Terminated): The process completed its execution.

6. Suspended Ready: When the ready queue becomes full, some processes are moved to suspended ready state

7. Suspended Block: When the waiting queue becomes full.



Context Switching

The process of saving the context of one process and loading the context of another process is known as Context Switching. In simple terms, it is like loading and unloading the process from the running state to the ready state.

CPU-Bound vs I/O-Bound Processes:
A CPU-bound process requires more CPU time or spends more time in the running state.
An I/O-bound process requires more I/O time and less CPU time. An I/O-bound process spends more time in the waiting state.

Types of schedulers:

1. **Long term – performance –** Makes a decision about how many processes should stay in the ready state, this decides the degree of multiprogramming. Once a decision is taken it lasts for a long time hence called a long term scheduler.
2. **Short term – Context switching time –** Decides which process to be executed next then calling dispatcher. A dispatcher is a software that moves processes from ready to run and vice versa. In other words, it is context switching.
3. **Medium term – Swapping time –** Suspension decision is taken by medium term scheduler. Medium term scheduler is used for swapping that is moving the process from main memory to secondary and vice versa.

Multiprogramming – We have many processes ready to run. There are two types of multiprogramming:

1. **Pre-emption –** Process is forcefully removed from the CPU. Pre-emption is also called time sharing or multitasking.
2. **Non pre-emption –** Processes are not removed until they complete the execution.

Degree of multiprogramming –

The number of processes that can reside in the ready state at maximum decides the degree of multiprogramming, e.g., if the degree of programming = 100, this means 100 processes can reside in the ready state at maximum.

Important parameters of processes :-

1.Arrival time :- Time at which the process comes to the ready queue.

2.Burst time :- The amount of time required by the process to complete(CPU time)

3.Completion time :- The time at which process finishes.

4.Turn around time :- The difference b/w the completion time and arrival time.

5.Waiting time :- The duration of the time the process waited in the ready queue.

6.Response time :- The first time the process starts its execution minus arrival time.

CPU Scheduling Algorithms

FCFS :-

Criteria :- Arrival Time
Mode :- Non-preemptive

First come, First served (FCFS), is the simplest scheduling algorithm. FIFO simply queues processes in the order that they arrive in the ready queue.
In this, the process that comes first will be executed first and next process starts only after the previous gets fully executed.

Implementation:

1- Input the processes along with their burst time(bt)
and arrival time(at)

2- Find waiting time for all other processes i.e. for
a given process i:
wt[i] = (bt[0] + bt[1] +..... bt[i-1]) - at[i]

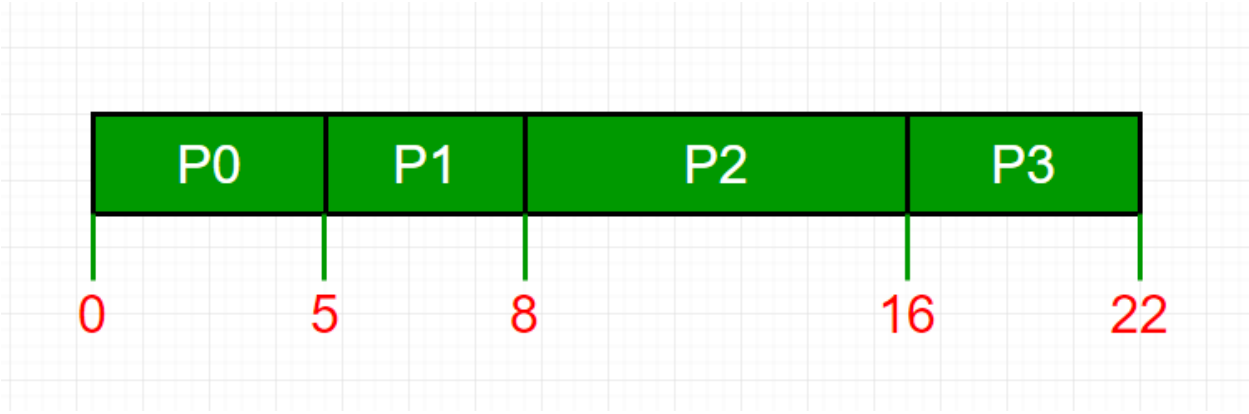
3- Now find **turnaround time**
= waiting_time + burst_time for all processes

4- **Average waiting time** =
total_waiting_time / no_of_processes

5- **Average turnaround time** =
total_turn_around_time / no_of_processes

Ex:-

Processes	Burst time	Arrival Time	Service Time
P0	5	0	0
P1	3	1	5
P2	8	2	8
P3	6	3	16



Disadvantage : Cannot utilize resources in parallel : Results in Convoy effect

Convoy effect :-

Suppose there is one CPU intensive (large burst time) process in the ready queue, and several other processes with relatively less burst times but are Input/Output (I/O) bound (Need I/O operations frequently).

Steps are as following below:

- The I/O bound processes are first allocated CPU time. As they are less CPU intensive, they quickly get executed and goto I/O queues.
- Now, the CPU intensive process is allocated CPU time. As its burst time is high, it takes time to complete.
- While the CPU intensive process is being executed, the I/O bound processes complete their I/O operations and are moved back to ready queue.
- However, the I/O bound processes are made to wait as the CPU intensive process still hasn't finished. **This leads to I/O devices being idle.**
- When the CPU intensive process gets over, it is sent to the I/O queue so that it can access an I/O device.
- Meanwhile, the I/O bound processes get their required CPU time and move back to I/O queue.
- However, they are made to wait because the CPU intensive process is still accessing an I/O device. As a result, **the CPU is sitting idle now**.

Hence in Convoy Effect, one slow process slows down the performance of the entire set of processes, and leads to wastage of CPU time and other devices.

Starvation or indefinite blocking is phenomenon, in which a process ready to run for CPU can wait indefinitely because of low priority. In heavily loaded computer system, a steady stream of higher-priority processes can prevent a low-priority process from ever getting the CPU.

Shortest Job First (or SJF):-

Criteria :- Burst time
Mode :- Non-preemptive

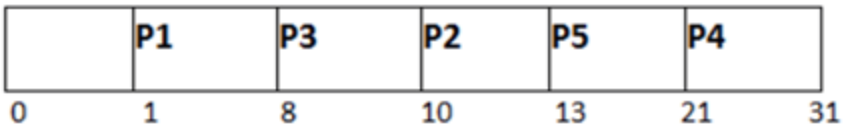
Shortest job first (SJF) or shortest job next, is a scheduling policy that selects the waiting process with the smallest execution time to execute next.

Algorithm:

1. Sort all the process according to the arrival time.
2. Then select that process which has minimum arrival time and minimum Burst time.
3. After completion of the process make a pool of processes which have a arrival time less than or equal to the completion of the previous process and select that process among the pool which is having minimum Burst time.

*Convoy effect can also be observed in the SJF if the process with higher Burst time will arrive a bit earlier in the ready queue and it is the only option at that time.
Ex:-

PID	Arrival Time	Burst Time
1	1	7
2	3	3
3	6	2
4	7	10
5	9	8



Avg Waiting Time = 27/5

PID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	1	7	8	7	0
2	3	3	13	10	7
3	6	2	10	4	2
4	7	10	31	24	14
5	9	8	21	12	4

Advantages:

1. Maximum throughput
2. Minimum Avg. waiting time and Turn around time.

Disadvantages:

1. Starvation of longer jobs.
2. It is not implementable because Burst time of process cannot be known ahead.

Throughput :- Number of processes completed per unit time.

Shortest Remaining Time First (SRTF)

Criteria :- Shortest time
Mode :- Preemptive

This is the process with the execution of the smallest amount of time remaining until completion is selected.

Implementation:

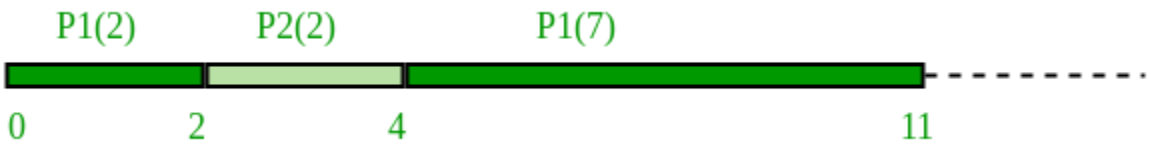
1- Traverse until all process gets completely

executed.
a) Find process with minimum remaining time at every single time lap.
b) Reduce its time by 1.
c) Check if its remaining time becomes 0
d) Increment the counter of process completion.
e) Completion time of current process = current_time +1;
e) Calculate waiting time for each completed process.
wt[i]= Completion time - arrival_time-burst_time
f)Increment time lap by one.

2- Find turnaround time (waiting_time+burst_time).

Ex:-

Process	Duration	Order	Arrival Time
P1	9	1	0
P2	2	2	2



Advantage:

1- Short processes are handled very quickly.

2- The system also requires very little overhead since it only makes a decision when a process completes or a new process is added.

3- When a new process is added the algorithm only needs to compare the currently executing process with the new process, ignoring all other processes currently waiting to execute.

Disadvantage:

1- Like shortest job first, it has the potential for process starvation.

2- Long processes may be held off indefinitely if short processes are continually added.

Shortest Job First CPU Scheduling with predicted burst time

We may not know the length of the next CPU burst, but we may be able to predict its value. We expect the next CPU burst will be similar in length to the previous ones.

There are two methods by which we can predict the burst time of the process :

1. Static method – We can predict the Burst-Time by two factors :

- **Process size –**
  Let say we have Process P_{old} having size 200 KB which is already executed and its Burst-time is 20 Units of time, now let's say we have a New Process P_{new} having size 201 KB which is yet to be executed. Then Burst time $_{\text{new process}}$=Burst time $_{\text{old process}}$
- **Process type –**
  We can predict Burst-Time depending on the Type of Process. Operating System process(like scheduler, dispatcher, segmentation, fragmentation) are faster than User process(Gaming, application softwares). Burst-Time for any New O.S process can be predicted from any old O.S process of similar type and same for User process.

2. Dynamic method – Let t_i be the actual Burst-Time of i_{th} process and T_{n+1} be the predicted Burst-time for $n+1_{th}$ process.

- **Simple average –** Given n processes (P_1, P_2… P_n)

  $T_{n+1} = 1/n(\Sigma_{i=1 \text{ to } n}\ t_i)$

- **Exponential average (Aging) –**

  $T_{n+1} = \alpha t_n + (1 - \alpha)T_n$

  where α = is smoothing factor and 0 <= α <= 1 ,

  t_n = actual burst time of n_{th} process,

  T_n = predicted burst time of n_{th} process.

Smoothening factor (α) – It controls the relative weight of recent and past history in our prediction.

Longest Remaining Time First (LRTF) :-

Criteria :- longest time
Mode :- Preemptive

In this scheduling algorithm, we find the process with the maximum remaining time and then process it.

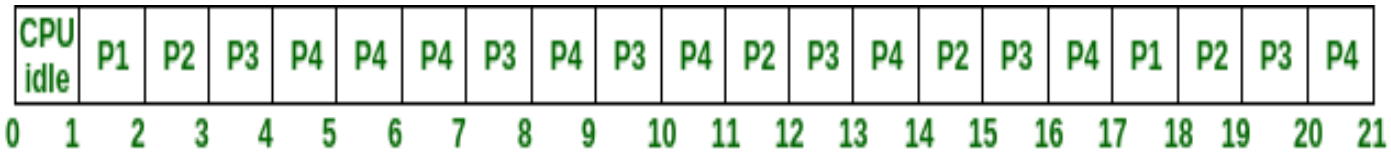
Procedure:

- **Step-1:** First, sort the processes in increasing order of their Arrival Time.
- **Step-2:** Choose the process having least arrival time but with most Burst Time. Then process it for 1 unit. Check if any other process arrives upto that time of execution or not.
- **Step-3:** Repeat the above both steps until you execute all the processes.

Ex:-

Process	Arrival time	Burst Time
P1	1 ms	2 ms
P2	2 ms	4 ms
P3	3 ms	6 ms
P4	4 ms	8 ms

1. At t = 1, Available Process : P1. So, select P1 and execute 1 ms.
2. At t = 2, Available Process : P1, P2. So, select P2 and execute 1 ms (since BT(P1)=1 which is less than BT(P2) = 4)
3. At t = 3, Available Process : P1, P2, P3. So, select P3 and execute 1 ms (since, BT(P1) = 1 , BT(P2) = 3 , BT(P3) = 6).
4. Repeat the above steps until the execution of all processes.



Advantage :

All the processes get completed by the time the longest job reaches its completion.

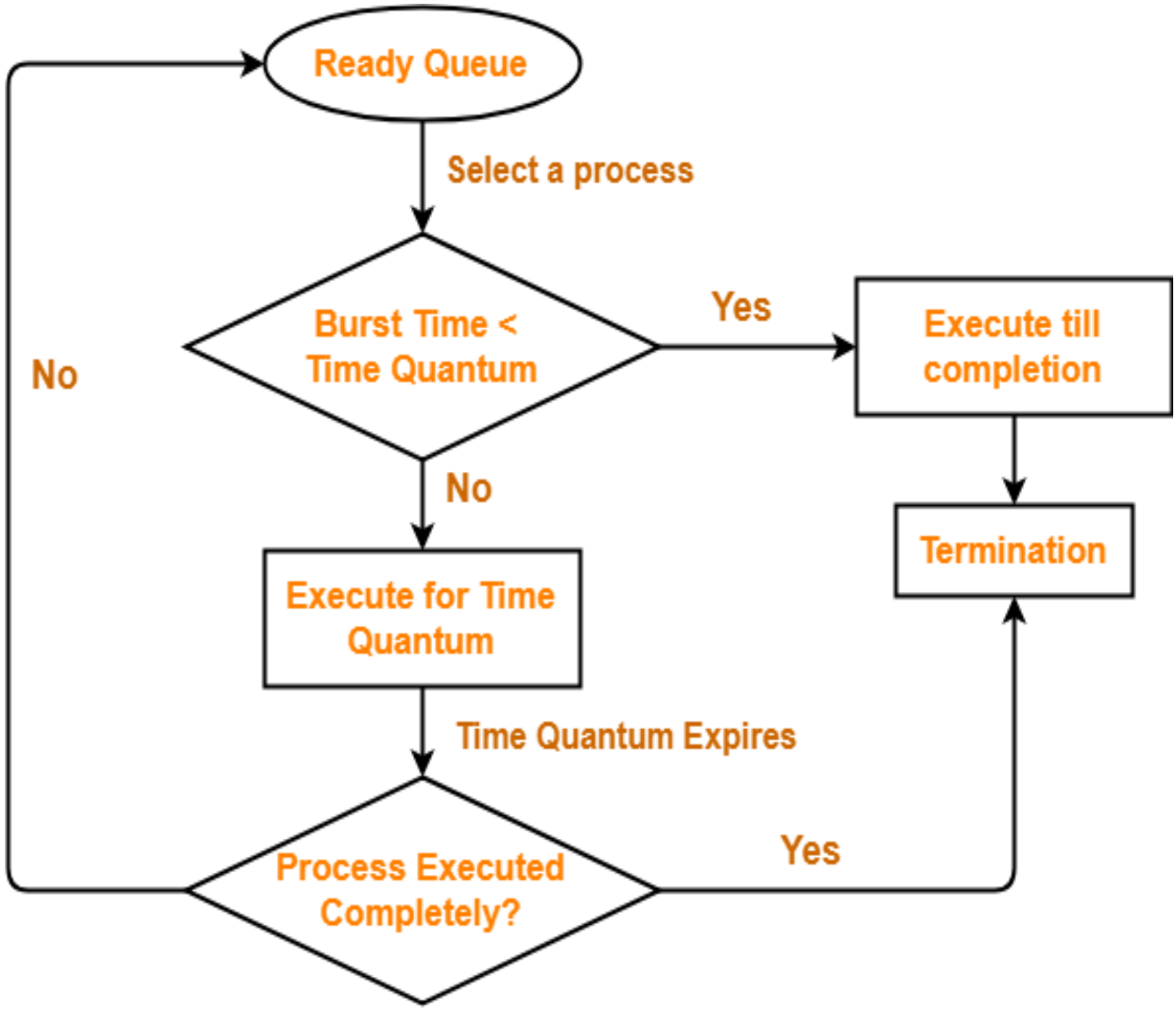
Disadvantage :

The average waiting **time** and turnaround **time** are too high, even if burst **time** is less for each process.

Round Robin scheduling :-

Criteria :- Time quantum + Arrival time
Mode :- Preemptive

Round Robin is a CPU scheduling algorithm where each process is assigned a fixed time slot in a cyclic way.It is simple, easy to implement, and starvation-free as all processes get fair share of CPU.It is the most commonly used technique in CPU scheduling as a core.



Ex:-

Process	Arrival Time	Burst Time
P1	0	5
P2	1	4
P3	2	2
P4	3	1

Process	Arrival Time	Burst Time	Completion time	Turn Around Time	Waiting time
P1	0	5	12	12	7
P2	1	4	11	10	6
P3	2	2	6	4	2
P4	3	1	9	6	5

Advantages:

- Each process is served by CPU for a fixed time, so priority is the same for each one
- Starvation does not occur because of its cyclic nature.

Disadvantages:

- Throughput depends on quantum time.
- If we want to give some process priority, we cannot.

Priority Scheduling :-

Criteria :- Priority
Mode :- Non-preemptive

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.

Implementation –

1. First input the processes with their arrival time, burst time and priority.

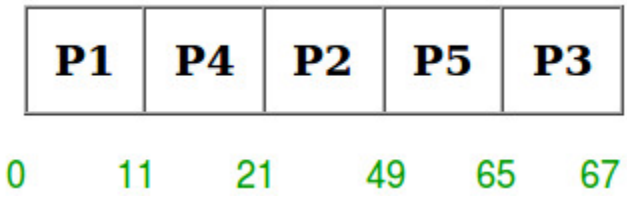
2. Sort the processes, according to arrival time if two process arrival time is same then sort according process priority if two process priority are same then sort according to process number.
3. Now simply apply the FCFS algorithm.

A problem with priority scheduling is indefinite blocking or starvation. A solution to the problem of indefinite blockage of the low-priority process is aging.

Aging is a technique of gradually increasing the priority of processes that wait in the system for a long period of time.

Ex:-

Process	Arrival Time	Burst Time	Priority
P1	0	11	2
P2	5	28	0
P3	12	2	3
P4	2	10	1
P5	9	16	4



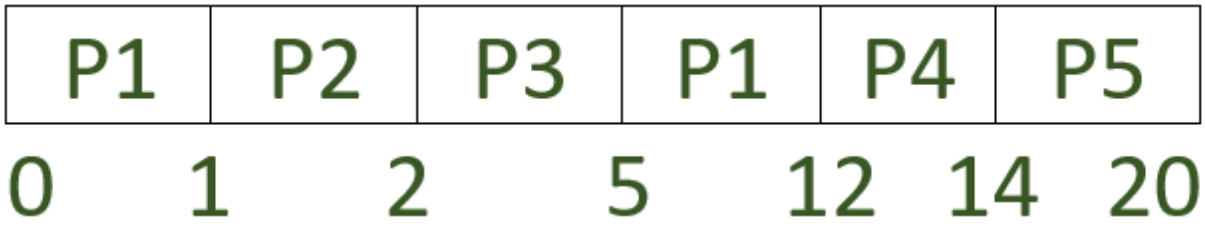
Preemptive Priority Scheduling :-

Criteria :- Priority
Mode :- preemptive

In Preemptive Priority Scheduling, at the time of arrival of a process in the ready queue, its Priority is compared with the priority of the other processes present in the ready queue as well as with the one which is being executed by the CPU at that point of time. The One with the highest priority among all the available processes will be given the CPU next.

Ex:-

Process	Arrival Time	Burst Time	Priority
P1	0	8	3
P2	1	1	1
P3	2	3	2
P4	3	2	3
P5	4	6	4



Highest Response Ratio Next (HRRN) :-

Criteria :- Response ratio
Mode :- Non-preemptive

In this scheduling, we find the response ratio of all available processes and select the one with the highest Response Ratio.

Response Ratio = (W + S)/S

W=waiting time for a process so far.

S= service time/burst time

Implementation of HRRN Scheduling –

1. Input the number of processes, their arrival times and burst times.
2. Sort them according to their arrival times.
3. At any given time calculate the response ratios and select the appropriate process to be scheduled.
4. Calculate the turnaround time as completion time – arrival time.
5. Calculate the waiting time as turnaround time – burst time.
6. Sum up the waiting and turn around times of all processes and divide by the number of processes to get the average waiting and turn around time.

HRRN not only favours shorter jobs but also limits the waiting time of longer jobs

Advantages :

1. Aging without service increases ratio, longer jobs can get past shorter jobs.

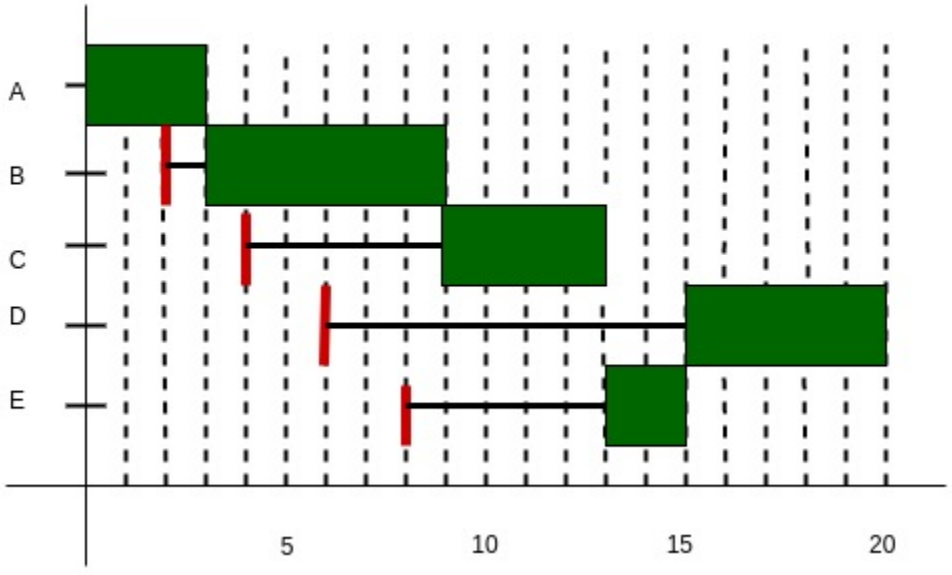
Disadvantage :

1. Like SJF, HRRN is also not implementable as we cannot know the burst time of the process ahead.

Scheduling Example

Thread	Arrival TIme	CPU Burst Length
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

Gantt Chart -

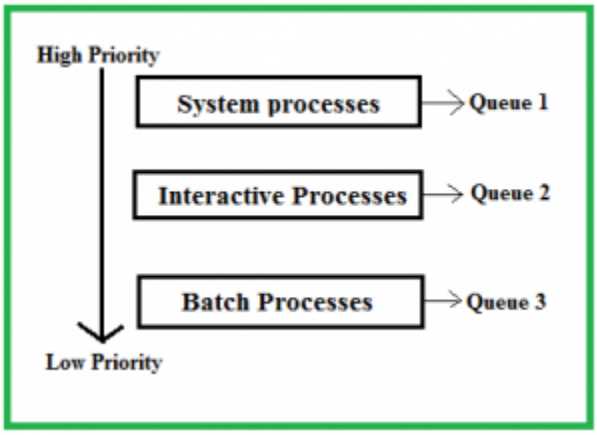


Explanation –

- At t = 0 we have only one process available, so A gets scheduled.
- Similarly at t = 3 we have only one process available, so B gets scheduled.
- Now at t = 9 we have 3 processes available, C, D and E. Since, C, D and E were available after 4, 6 and 8 units respectively. Therefore, waiting time for C, D and E are (9 – 4 =)5, (9 – 6 =)3, and (9 – 8 =)1 unit respectively.
- Using the formula given above we calculate the Response Ratios of C, D and E respectively as 2.25, 1.6 and 1.5.
- Clearly C has the highest Response Ratio and so it gets scheduled
- Next at t = 13 we have 2 jobs available D and E.
- Response Ratios of D and E are 2.4 and 3.5 respectively.
- So process E is selected next and process D is selected last.

Multilevel Queue (MLQ) Scheduling :-

Ready Queue is divided into separate queues for each class of processes.



All three different types of processes have their own queue. Each queue has its own Scheduling algorithm. For example, queue 1 and queue 2 use Round Robin while queue 3 can use FCFS to schedule their processes.

Scheduling among the queues :

1. **Fixed priority preemptive scheduling method** – Each queue has absolute priority over lower priority queue.
2. **Time slicing** – In this method each queue gets a certain portion of CPU time and can use it to schedule its own processes.

Ex:-

Process	Arrival Time	CPU Burst Time	Queue Number
P1	0	4	1
P2	0	3	1
P3	0	8	2
P4	10	5	1

Advantages:

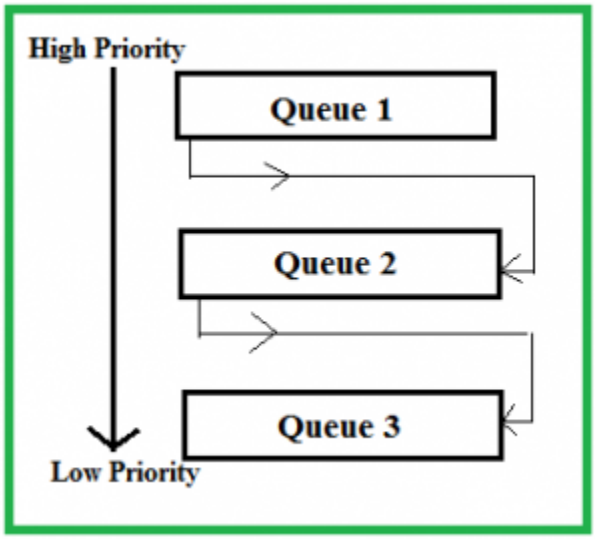
- The processes are permanently assigned to the queue, so it has the advantage of low scheduling overhead.

Disadvantages:

- Some processes may starve for CPU if some higher priority queues are never becoming empty.
- It is inflexible in nature.

Multilevel Feedback Queue Scheduling (MLFQ) Scheduling :-

This Scheduling is like Multilevel Queue(MLQ) Scheduling but in this process can move between the queues.It keeps analyzing the behavior (time of execution) of processes and according to which changes the priority.



Ex :-

Now let us suppose that queue 1 and 2 follow round robin with time quantum 4 and 8 respectively and queue 3 follow FCFS.One implementation of MFQS is given below –

1. When a process starts executing then it first enters queue 1.
2. In queue 1 process executes for 4 unit and if it completes in this 4 unit or it gives CPU for I/O operation in this 4 unit than the priority of this process does not change and if it again comes in the ready queue than it again starts its execution in Queue 1.
3. If a process in queue 1 does not complete in 4 unit then its priority gets reduced and it shifted to queue 2.

4. Above points 2 and 3 are also true for queue 2 processes but the time quantum is 8 unit.In a general case if a process does not complete in a time quantum than it is shifted to the lower priority queue.
5. In the last queue, processes are scheduled in FCFS manner.
6. A process in lower priority queue can only execute only when higher priority queues are empty.
7. A process running in the lower priority queue is interrupted by a process arriving in the higher priority queue.

Advantages:

1. It is more flexible.
2. It allows different processes to move between different queues.
3. It prevents starvation by moving a process that waits too long for a lower priority queue to the higher priority queue.

Disadvantages:

1. For the selection of the best scheduler, it requires some other means to select the values.
2. It produces more CPU overheads.
3. It is the most complex algorithm.