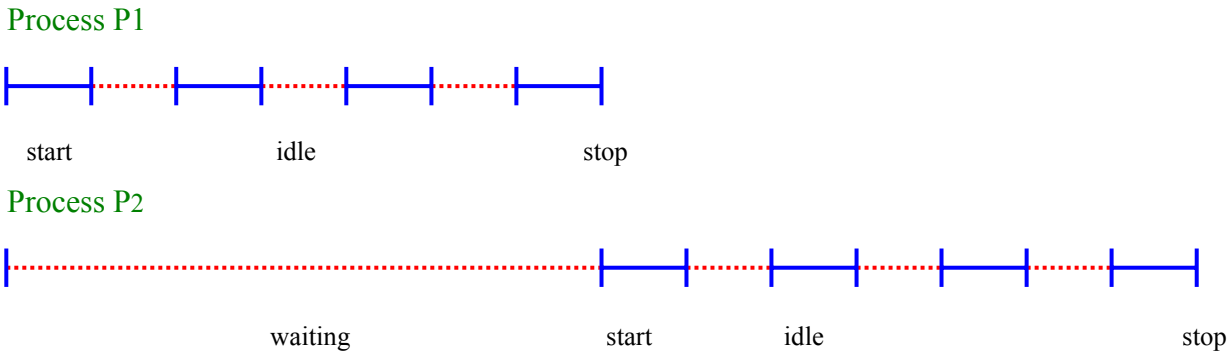
Scheduling Concepts

In Multiprogramming we run many processes at the same time in order to improve CPU utilization. In uni programming CPU sits idles when a running process needs I/O operation, but in multiprogramming when one process needs I/O operation, then operating system gives the CPU to another process.

Benefits of Multiprogramming are:

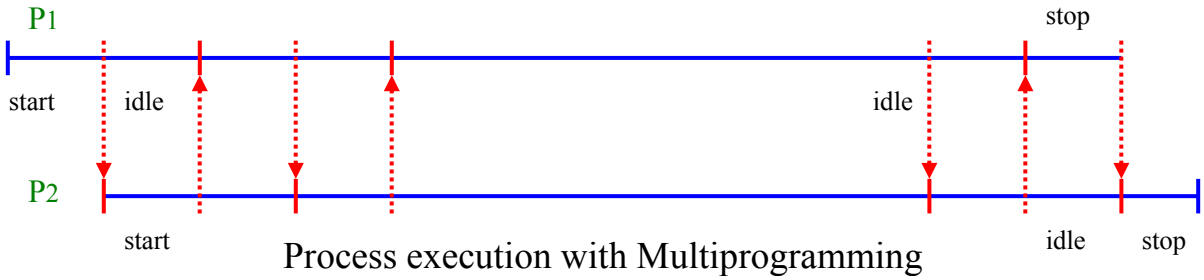
i) **Increased CPU utilization** and
ii) **Higher throughput** (Throughput is the amount of work accomplished in a given time interval).

We have two processes P1 and P2. First we run P1 that takes two minutes to complete, then we run P2 that also takes two minutes to complete. Although the process execution for each process is 1 minute but due to I/O operations both processes took double-time to complete. So our CPU utilization is only 50 percent.



Process execution without Multiprogramming

System performance can be improved greatly if we multi-program process P1 and P2. Process P1 starts execution, as it needs I/O we start executing process P2, as P2 needs I/O till that time process P1 is ready to run again. Now both processes execute like this and the time of completion of both processes is 2 minutes. So, we have improved CPU utilization from 50 percent to 100 percent, and this utilization was achieved due to multiprogramming.



Process execution with Multiprogramming

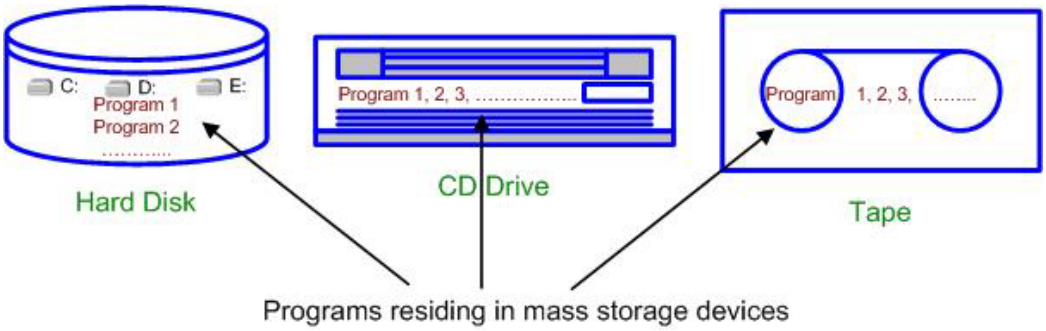
Schedular

When more than one process is run able then Operating System decides which process to run first, so the part of an Operating System concerned with this decision is called the schedular and the algorithm it uses is called the Scheduling Algorithm.

Scheduling Queues

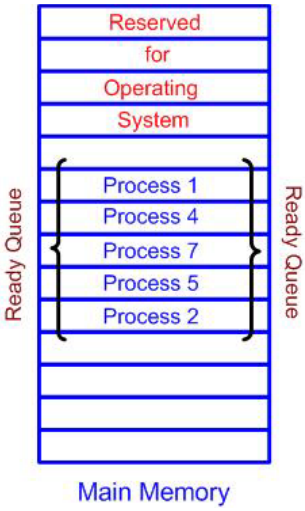
Job Queue

Job Queue consists of all processes residing in mass storage and waiting for the allocation of main memory.



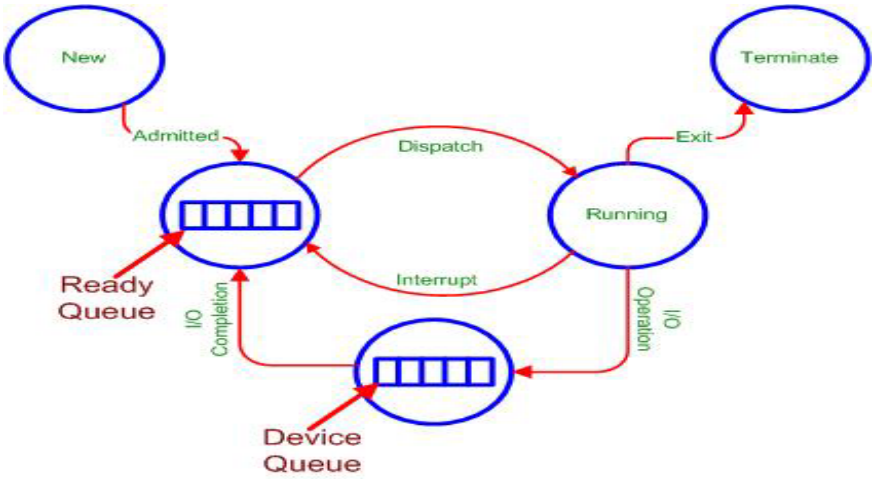
Ready Queue

The processes that are residing in main memory and are ready to execute are kept in a list called the Ready Queue. So, all the processes that are in ready state and waiting for CPU time are considered to be part of ready queue.

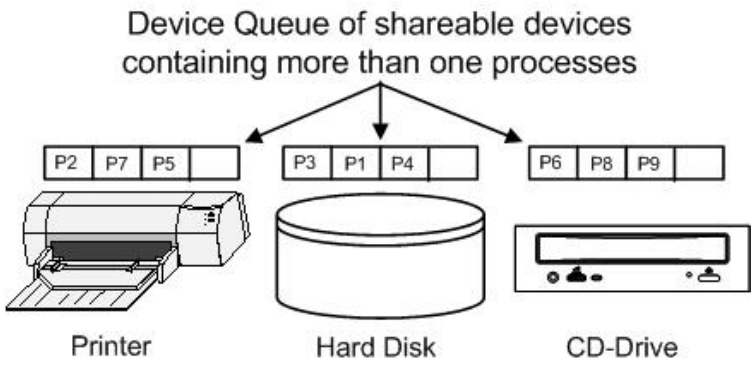


Device Queue

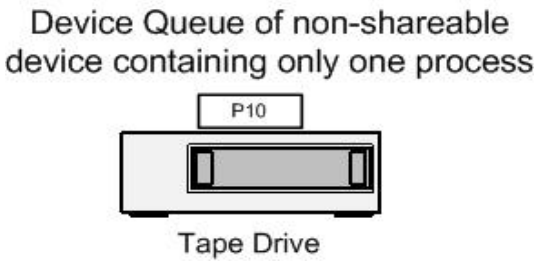
Device Queue consists of processes that are waiting for some I/O device. So, all the processes that are in waiting or blocked state are considered to be part of device queue. Each device has its own device queue.



Shareable Device Queues
Sharable devices such as Printers, Hard Disks, and CDs etc can have several processes in its device queue.



Non-Shareable Device Queues
A non-shareable or dedicated device Like tape drive can never have more than one process in its device queue. So, the device queues of dedicated devices have only one process in their device queues as these devices are of non-shareable nature.



Types of Schedulars

Long Term Schedular or Job Schedular or admission scheduler selects processes present in the Job Queue and loads them into memory for execution.

Short Term Schedular or CPU Schecular selects processes that are ready to execute from the ready queue and allocates the CPU to one of them.

Generally short-term schedular selects new processes for CPU quite often. The short-term schedular must be very fast. If a short term schedular takes 1 msec to decide to execute a process for 10 msec then 1/(10+1) = 9 percent of the CPU is used/wasted just for scheduling the work.

Long term schedulers do not execute frequently. As compared to the short term scheduler, a long term schedular executes less frequently. The long-term schedular controls the degree of Multiprogramming. If the average rate of process creation is equal to the average departure rate of process leaving the system, then it means that the degree of multiprogramming is stable.

It is important that the long-term scheduler select a good process mix of I/O bound and CPU bound processes. Best performance of the system will be achieved if we have a combination of CPU bound and I/O bound processes.

Medium-Term or mid-term Scheduler removes processes from memory in order to reduce the degree of Multiprogramming.

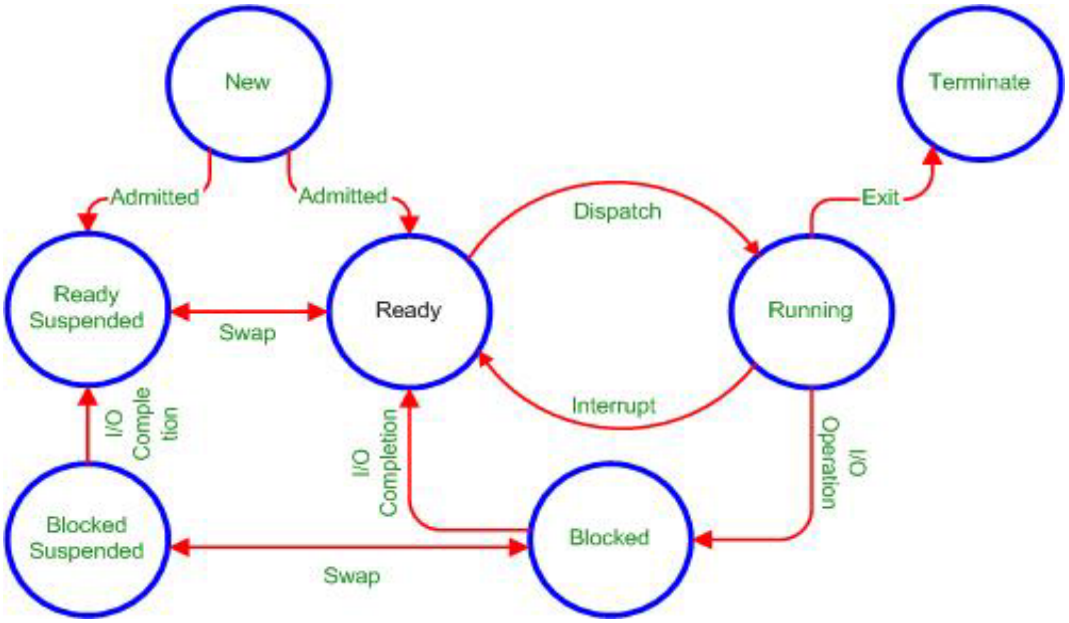
Generally, Medium-Term schedular is used in Time Sharing systems and when a system is heavily loaded with processes then Medium-Term schedular removes some processes from memory and later those processes are brought back into memory so that they continue their execution from where they were left.

The mid-term scheduler, temporarily removes processes from main memory and places them on secondary memory (such as a disk drive) or vice versa. This is commonly referred to as "swapping out" or "swapping in".

The mid-term scheduler may swap out a process when

- It is not active for some time
- Its priority is low
- It has increased page-fault rate
- It is consuming a large amount of memory
- A process has been unblocked and is no longer waiting for a resource.

Process State Diagram with Medium Term Schedular



CPU Bound or CPU Intensive Processes

CPU Bound or CPU Intensive processes spend more of their time doing computations etc and generate I/O requests infrequently.

I/O Bound or I/O Intensive Processes

I/O Bound or I/O Intensive processes spend more of their time doing I/O etc and generate CPU requests infrequently.

CPU Scheduling

When the CPU becomes idle, the Operating System must select one of the processes in the ready queue to be executed, this selection is done through short term or CPU scheduling.

Scheduling decisions take place

1. When a process switches from running to waiting state.
2. When a process switches from running state to the ready state.
3. When a process switches from the waiting state to ready state.
4. When a process terminates.

Non-Preemptive Scheduling Scheme

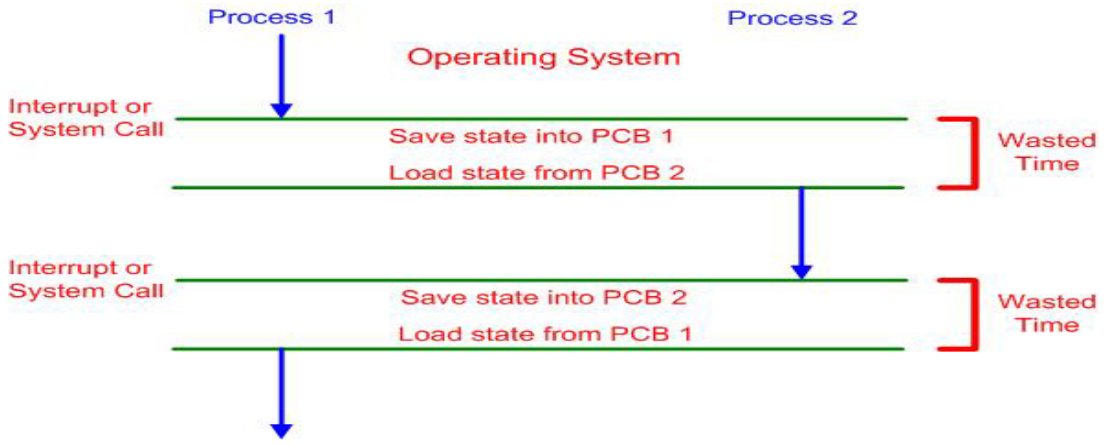
On switching a process from running to waiting state or on process termination, a new process must be selected for execution, and the scheduling scheme used in such a situation is called non-preemptive scheduling scheme.

Preemptive Scheduling Scheme

On switching a process from running to ready state or on switching from waiting to ready state, the scheduling scheme is called preemptive scheduling scheme.

Context Switch

Switching the CPU to another process requires saving the state of the old process and loading the saved state for the new process. This task is known as a context switch. Context Switch time is pure overhead and depends upon the speed of memory and number of registers etc.



Dispatcher

The function that gives control of the CPU to the process selected by the short term or CPU scheduler is called Dispatcher.

Dispatcher function involves

i) Context switching
ii) Switching to user mode
iii) Jumping to the location in the user program to restart that program.

It will be better that a dispatcher should be as fast possible.

Scheduling Algorithms

Criteria for comparing CPU-scheduling Algorithms include the following:

1. ***CPU Utilization***: To keep the CPU as busy as possible we would like the CPU to be utilized as much as we can.

2. ***Throughput***: The number of processes completed per time unit is called throughput.

3. ***Turnaround Time***: It is the sum of period spent waiting to get into memory, waiting in the Ready Queue, executing on the CPU and doing I/O etc. So, turnaround time of a process is the period from the time of submission to the time of completion.

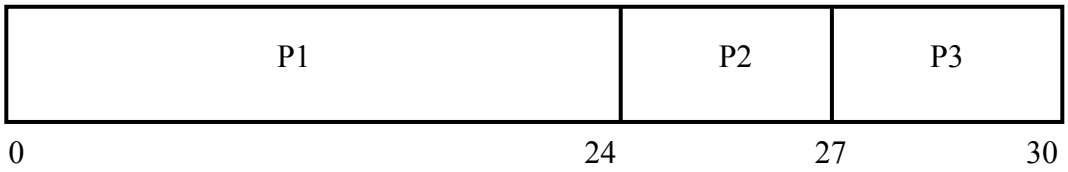
4. ***Waiting Time***: another criterion is the amount of time that a process spends waiting in the ready queue.

5. ***Response Time***: It is an important criterion in interactive systems. Response time is the calculation of time for a process request from its submission to the first response produced.

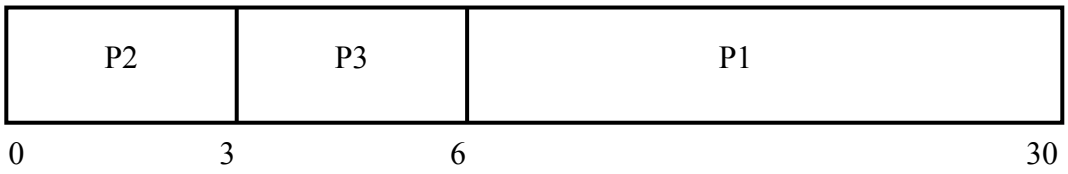
First Come, First Served Scheduling

The simplest algorithm is the First Come, First Served (FCFS) scheduling algorithm. In this scheduling scheme, the process that requests the CPU first is allocated the CPU first. The average waiting time under FCFS algorithm is long.

A set of processes arrives at time 0. The length of CPU Burst for process P1 is 24 ms, 3 ms for P2 and 3 ms for p3.



If these processes are executed using FCFS algorithm then the waiting time for process P1 is 0 ms, 24 ms for P2 and 27 ms for P3. So, the average waiting time is (0+24+27) / 3 = 17 ms. If the process arrive in the order P2, P3 and P1 then the average waiting time is (6+0+3) / 3 = 3 ms.



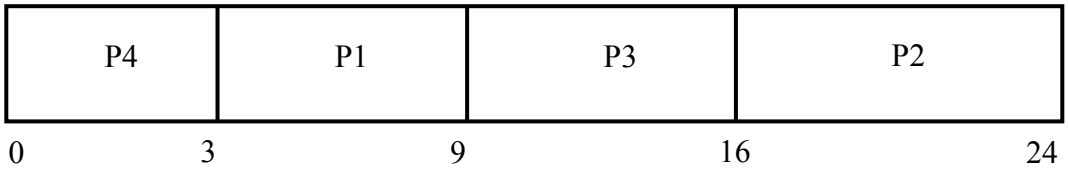
So the average waiting time in FCFS Scheduling Algorithm depends upon the CPU burst times of processes.

Shortest Job First

In the Shortest Job First (SJF) Scheduling the process having the shortest CPU burst time is executed first. Although it is difficult to measure the length of CPU burst among processes but we can predict the values. We expect that the next CPU burst time will be similar in length to the previous ones. So by computing the approximate length of next CPU burst, we pick a process with a shortest predicted CPU burst.

A set of processes have the following CPU burst time in milliseconds

P1 = 6, P2 = 8, P3 = 7 and P4 = 3 ms.



Waiting time for P1 is 3 ms, 16 ms for P2 and 9 ms for P3 and 0 ms for P4. So, the average waiting time is (3+16+9+0) / 4 = 7ms. The average waiting time for this Situation is FCFS scheduling is 10.25 ms.

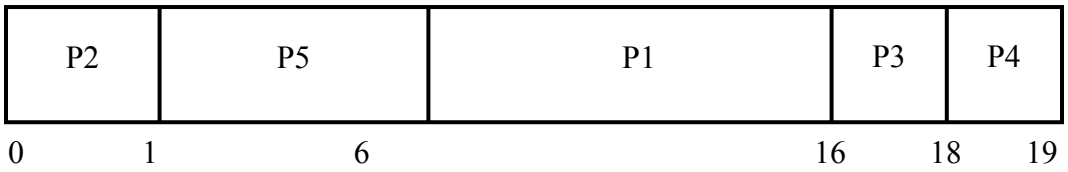
Priority Scheduling

A priority is associated with each process and the CPU is allocated to the process with the highest priority. Processes having equal priority are scheduled in FCFS order. So, the Shortest Job First algorithm is also a priority algorithm because processes having smallest CPU burst are executed first.

A set of processes arrive at time 0 have following CPU burst time and priority.

Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

The processes will be scheduled in the following order using priority scheduling and the average waiting time calculated is 8.2 ms.



Internal or External priorities can be defined in Priority Scheduling. Internally defined priorities use measurable quantities i.e. time limits or memory requirements etc. Externally defined priorities are set by criteria that are external to the Operating System i.e. amount of funds paid, department sponsoring the work etc.

Priority scheduling can be preemptive or non-preemptive. Preemptive priority assigns the CPU to the newly arrived process if its priority is higher than the currently running process. In non-preemptive priority the newly arrived process will be at the head of the ready queue and CPU will be assigned to this process when the currently running process will finish.

Starvation the problem in Priority scheduling is that it will leave low priority processes waiting for CPU for a long time and this problem is called Starvation.

Aging the solution for starvation is aging. In this technique a waiting process priority is increased after a fixed time. So, even a process of low priority will execute, as its priority will be increased within few hours or days.

Round Robin Scheduling

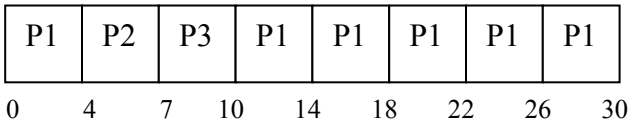
The scheduling algorithm used in time-sharing systems is Round Robin Scheduling. In Round Robin (RR) Algorithm the ready queue is treated as a circular queue. The short-term or CPU schedular allocates CPU to each process for a small unit of time called time quantum or time slice, and one quantum may be from 10 ms to 100 ms.

The RR scheduling algorithm is preemptive and the average waiting time in round robin algorithm is long.

A set of processes arrive at time 0. The length of CPU burst time for process P1 is 24 ms, 3 ms for P2, and 3 ms for P3.

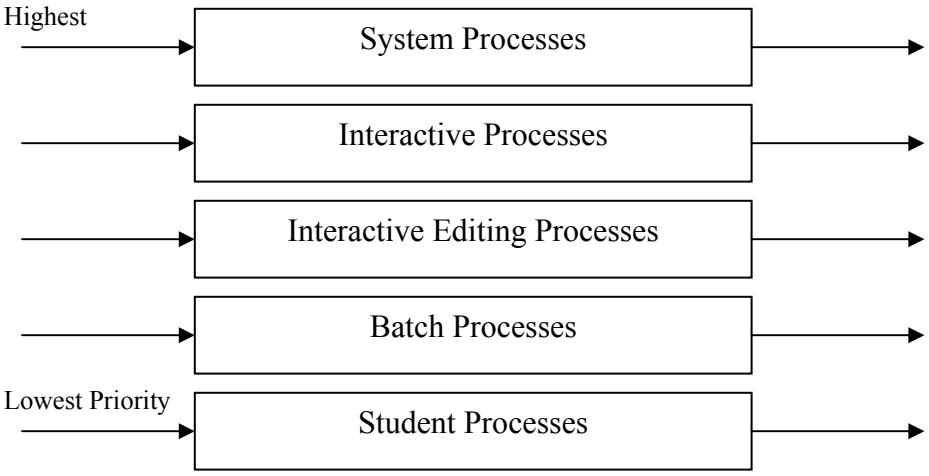
We assume that the time quantum is 4 ms. Process P1 executes for first 4 ms and it requires 20 ms more, but due to 4 ms of time quantum, it will be preempted as CPU will be given the process P2. P2 quits before its time quantum expires and same it happens to P3. Now the CPU is returned to P1 for additional time quantum and the average waiting time like this for RR scheduling in this situation is 17/3 = 5.66 ms.

The performance of RR Scheduling depends on the size of time quantum. Very large time quantum acts like a FCFS policy and very short time quantum e.g. 1 ms is called processor sharing.



Multilevel Queue Scheduling

Multilevel queue scheduling is designed for situations where processes are classified into different groups. Ready queue is partitioned into separate queues and processes stored in any queue depend upon their memory size or process type etc. Each queue then again can have its own scheduling algorithm.



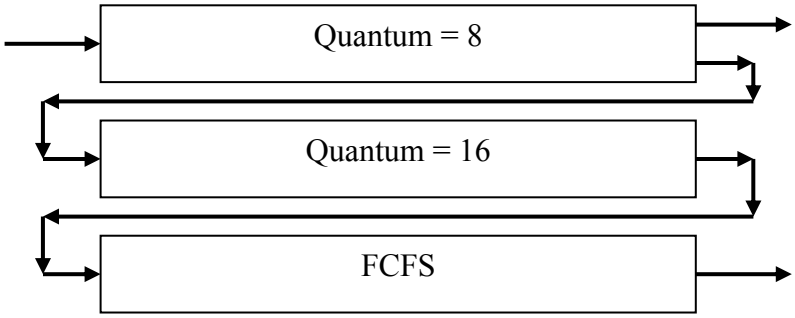
We have a ready queue partitioned into 5 different queues. System processes queue has highest priority whereas student processes queue has lowest priority. A process in a batch queue can not run unless all its upper queues are empty.

Another technique is if we time slice all the queues, i.e. each queue gets a portion of the time which it can then schedule among the various processes in its queue.

Multilevel Feedback Queue Scheduling

In Multilevel Feedback Queue Scheduling, a process can be moved between queues. Generally, processes that are moved between queues depend upon their CPU burst characteristics. If a process uses too much time it is moved to a low priority queue. Similarly, a process waiting for a long time in lower priority queue will be moved to higher priority queue.

A multilevel feedback queue schedular has three queues 0-2. Schedular will first execute processes of queue 0, queue 1 and then queue 2. Processes in queue 0 will be given a time quantum of 8 ms and if a process does not finish in 8 ms, then it will be moved to the tail of queue 1. Processes in queue 1 have time quantum of 16 ms and if a process does not finish within that time, it is moved to queue2 that executes processes on FCFS basis.



Generally a multilevel feedback queue schedular is defined on following considerations.

1. The number of queues
2. Scheduling algorithm for each queue.
3. Decide when to promote a process to a high-priority queue.
4. Decide when to demote a process to a lower-priority queue.
5. Decide which queue, a process should be placed.

Multilevel Feedback queue is a very general scheduling scheme but is also the most complex.

Multiprocessor Scheduling

Multiprocessor Scheduling is used on systems having multiple processors. Scheduling on systems having multiple processors is more complex. Multiprocessor scheduling depends upon the type of processors present in a system.

In **Heterogeneous System** (a system having different processors) each processor has its own queue and its own scheduling algorithm. A program written on one processor will not run on other processor.

In **Homogeneous systems** (a system having identical processors), there is one common ready queue. All processes go into one queue and are scheduled on to any available processor. So this scheme ensures that all processes work almost equally instead of one overloaded with processes and the other sitting idle.

Algorithm Evaluation

In algorithm evaluation, we evaluate different scheduling algorithm and try to find the efficiency of scheduling algorithms depending upon certain conditions. There are many evaluation methods that can be used for this purpose.

Analytic Evaluation

Is an important class of evaluation in which different techniques can be used for finding the efficiency of algorithms.

Deterministic Modeling

In this method we check the performance of different algorithms for a defined workload i.e.

Process	Burst Time
P1	10
P2	29
P3	3
P4	7
P5	12

The average waiting time for this situation in FCFS algorithm is 28 ms, 13 ms in SJF and 23 ms in Round Robin Algorithm.

P1 = 0
P2 = 10+3+7+10+2 = 32
P3 = 20
P4 = 23
P5 = 30+10 = 40

0 + 32 + 20 + 23 + 40 = 115/5 = 23ms

So, Deterministic modeling is a simple way through which the efficiency of different algorithms can be evaluated accurately and quickly.

Queuing Models

Can also be used for comparing scheduling algorithms. The number of processes leaving the queue should be equal so the number of processes entering the queue in the steady System State.

An equation

$$\mathbf{n = \lambda x w}$$

known as Little's formula and can be used for determining the efficiency of scheduling algorithm.

n is the queue length, λ is the average arrival rate for new processes in the queue and **w** is the waiting time.

If 7 processes arrived every second and queue length is 14 then average waiting time per process can be calculated i.e. 2 seconds

$$n = \lambda \times w$$
$$w = n / \lambda$$
$$w = 14 / 7 = 2 \text{ Seconds.}$$

Simulations

Can be used for accurate evaluation of scheduling algorithms. In simulations we monitor the real system and record the sequence of actual events and then we use this sequence in our simulation i.e. a model of the computer system, Programmed.