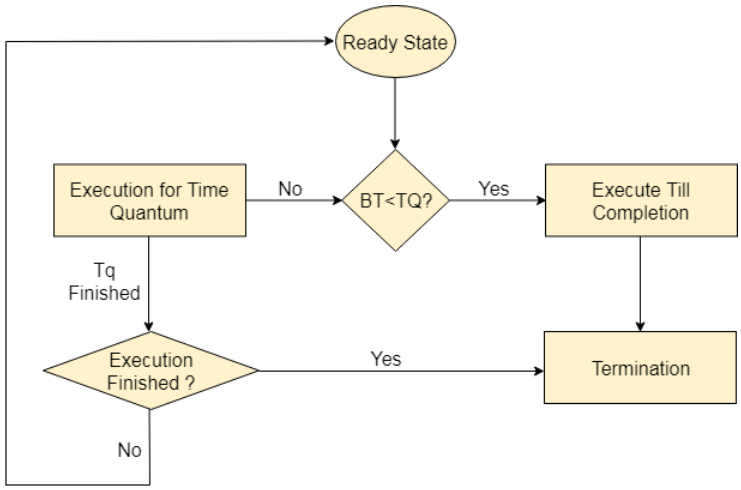
Operating System

Week 6 - Long Descriptive Questions

1.Explain the role of time quantum with respect to Round robin scheduling.

Ans. **Round Robin** scheduling algorithm is one of the most popular scheduling algorithm which can actually be implemented in most of the operating systems. This is the preemptive version of first come first serve scheduling. The Algorithm focuses on Time Sharing. In this algorithm, every process gets executed in a cyclic way. A certain time slice is defined in the system which is called **time quantum**. Each process present in the ready queue is assigned the CPU for that time quantum, if the execution of the process is completed during that time then the process will terminate else the process will go back to the ready queue and waits for the next turn to complete the execution.



Advantages

1. It can be actually implementable in the system because it is not depending on the burst time.
2. It doesn't suffer from the problem of starvation or convoy effect.
3. All the jobs get a fare allocation of CPU.

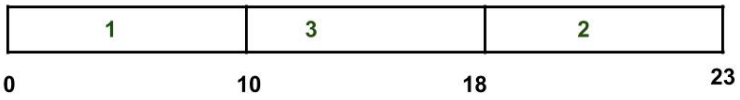
Disadvantages

1. The higher the time quantum, the higher the response time in the system.
2. The lower the time quantum, the higher the context switching overhead in the system.
3. Deciding a perfect time quantum is really a very difficult task in the system.

2.What is the problem encountered in priority scheduling? Explain the solution it ?

Ans. A major problem related to priority scheduling and its solution.
Starvation : Starvation or indefinite blocking is a phenomenon associated with the Priority scheduling algorithms, in which a process ready for the CPU (resources) can wait to run indefinitely because of low priority. In a heavily loaded computer system, a steady stream of higher-priority processes can prevent a low-priority process from ever getting the CPU. There have been rumours that in 1967 Priority Scheduling was used in IBM 7094 at MIT, and they found a low-priority process that had not been submitted till 1973.

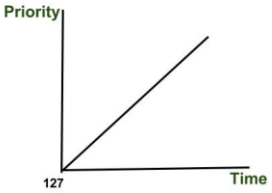
Process	Burst time	Priority
1	10	2
2	5	0
3	8	1



As we see in the above example process having higher priority than other processes getting CPU earlier. We can think of a scenario in which only one process is having very low-priority (for example 127) and we are giving other process with high-priority, this can lead indefinitely waiting for the process for CPU which is having low-priority, this leads to Starvation. Further we have also discuss about the solution of starvation.
Differences between Deadlock and Starvation in OS are as follows:

1. Deadlock occurs when none of the processes in the set is able to move ahead due to occupancy of the required resources by some other process as shown in the figure below, on the other hand, Starvation occurs when a process waits for an indefinite period of time to get the resource it requires.
2. Another name for deadlock is Circular Waiting. Another name for starvation is Lived lock.
3. When deadlock occurs no process can make progress, while in starvation apart from the victim process other processes can progress or proceed.

Solution to Starvation: Aging
Aging is a technique of gradually increasing the priority of processes that wait in the system for a long time. For example, if priority range from 127(low) to 0(high), we could increase the priority of a waiting process by 1 Every 15 minutes. Eventually, even a process with an initial priority of 127 would take no more than 32 hours for the priority 127 process to age to a priority-0 process.



3.Differentiate between Preemptive and Non-Preemptive.

Ans.
Let us see the difference between Preemptive Scheduling and Non-Preemptive Scheduling:

Preemptive Scheduling	**Non-Preemptive Scheduling**
The CPU is allocated to the processes for a certain amount of time.	The CPU is allocated to the process till it ends it's execution or switches to waiting state.
The executing process here is interrupted in the middle of execution.	The executing process here is not interrupted in the middle of execution.
It usually switches the process from ready state to running state, vice-versa, and maintains the ready queue.	It does not switch the process from running state to ready state.
Here, if a process with high priority frequently arrives in the ready queue then the process with low priority has to wait for long, and it may have to starve.	Here, if CPU is allocated to the process with larger burst time then the processes with small burst time may have to starve.
It is quite flexible because the critical processes are allowed to access CPU as they arrive into the ready queue, no matter what process is executing currently.	It is rigid as even if a critical process enters the ready queue the process running CPU is not disturbed.
This is cost associative as it has to maintain the integrity of shared data.	This is not cost associative.
This scheduling leads to more context switches.	This scheduling leads to less context switches compared to preemptive scheduling.

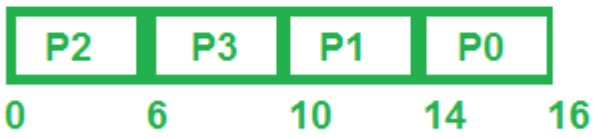
Preemptive scheduling is better than non-preemptive scheduling or vice-versa can't be said definitely. It depends on how scheduling minimizes the average waiting time of the processes and maximizes CPU utilization.

Example :

Process	Arrival Time	CPU Burst Time (in millisec.)
P0	3	2
P1	2	4
P2	0	6
P3	1	4

Preemptive Scheduling

Process	Arrival Time	CPU Burst Time (in millisec.)
P0	3	2
P1	2	4
P2	0	6
P3	1	4



Non-Preemtive Scheduling

4.List out the CPU scheduling criteria and explain it.

Ans.

Different CPU scheduling algorithms have different properties and the choice of a particular algorithm depends on the various factors. Many criteria have been suggested for comparing CPU scheduling algorithms.

The criteria include the following:

1. CPU utilisation: The main objective of any CPU scheduling algorithm is to keep the CPU as busy as possible. Theoretically, CPU utilisation can range from 0 to 100 but in a real-time system, it varies from 40 to 90 percent depending on the load upon the system.

2. Throughput:A measure of the work done by the CPU is the number of processes being executed and completed per unit of time. This is called throughput. The throughput may vary depending upon the length or duration of the processes.
3. Turnaround time: For a particular process, an important criterion is how long it takes to execute that process. The time elapsed from the time of submission of a process to the time of completion is known as the turnaround time. Turn-around time is the sum of times spent waiting to get into memory, waiting in the ready queue, executing in CPU, and waiting for I/O. The formula to calculate Turn Around Time = Compilation Time – Arrival Time.
4. Waiting time: A scheduling algorithm does not affect the time required to complete the process once it starts execution. It only affects the waiting time of a process i.e. time spent by a process waiting in the ready queue. The formula for calculating Waiting Time = Turnaround Time – Burst Time.
5. Response time: In an interactive system, turn-around time is not the best criteria. A process may produce some output fairly early and continue computing new results while previous results are being output to the user. Thus another criteria is the time taken from submission of the process of request until the first response is produced. This measure is called response time. The formula to calculate Response Time = CPU Allocation Time(when the CPU was allocated for the first) – Arrival Time
6. Completion time: This is the time when the process completes its execution.

5.Write short notes on Multilevel feedback queue scheduling.

Ans.

Multilevel Feedback Queue Scheduling (MLFQ) CPU Scheduling is like Multilevel Queue(MLQ) Scheduling but in this processes can move between the queues. And thus, much more efficient than multilevel queue scheduling.

Characteristics of Multilevel Feedback Queue Scheduling:

- In a multilevel queue-scheduling algorithm, processes are permanently assigned to a queue on entry to the system and processes are not allowed to move between queues.
- As the processes are permanently assigned to the queue, this setup has the advantage of low scheduling overhead,
- But on the other hand disadvantage of being inflexible.

Advantages of Multilevel Feedback Queue Scheduling:

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

Disadvantages of Multilevel Feedback Queue Scheduling:

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