M.Sc. IV Sem. (Mathematics)

Paper 2nd - Fundamentals of Computer Science - II

Unit V

Reference Book : **C. Ritchie, *Operating Systems incorporating UNIX and Windows,* BPB Publications, New Delhi.**

Topic : Scheduling Algorithms

A process scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms.

These algorithms are either non-preemptive or preemptive. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

Scheduling algorithms are of two types :

1) **Preemptive Scheduling.**
2) **Non-Preemptive Scheduling.**

Preemptive Scheduling :

Preemptive scheduling is used when a process switches from running state to ready state or from waiting state to ready state. The resources (mainly CPU cycles) are allocated to the process for the limited amount of time and then is taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining. That process stays in ready queue till it gets next chance to execute.

Algorithms based on preemptive scheduling are: Round Robin (RR), Shortest Remaining Time First (SRTF) etc.

Non-Preemptive Scheduling :

Non-preemptive Scheduling is used when a process terminates, or a process switches from running to waiting state. In this scheduling, once the resources (CPU cycles) is allocated to a process, the process holds the CPU till it gets terminated or it reaches a waiting state. In case of non-preemptive, scheduling does not interrupt a process running CPU in middle of the execution. Instead, it waits till the process complete its CPU burst time and then it can allocate the CPU to another process.

Algorithms based on non-preemptive scheduling are: Shortest Job First, First Come First Served, Highest Response Ratio Next etc.

Low Level Scheduling Policies (Algorithms) :

The following are the low level scheduling policies :

1. **First Come First Served (FCFS)**
2. **Shortest Job First (SJF)**
3. **Round Robin (RR)**
4. **Shortest Remaining Time (SRT)**
5. **Highest Response Ratio Next (HRRN)**
6. **Multi-level Feedback Queues (MFQ)**

1. First Come First Served (FCFS) :

Also known as First-in-First-out (FIFO). With this policy the process that requests the C.P.U. first, is allocated the C.P.U. first. The FCFS policy simply assigns the processor to the process which first in the READY queue (i.e. it has been waiting for the longest time). This is a non-preemptive scheme, since it is actioned only when the current process relinquishes control. FCFS policy can be illustrated by the following example :

	(A)	(B)	(C)
Job	Estimated Run Time	Waiting Time	Ratio B/A
1	2	0	0
2	60	2	0.03
3	1	62	62
4	3	63	21
5	50	66	1.32

Table : Illustration of FCFS Policy

This policy is unfair to short processes, as we can see from column (C) that this ratio for small jobs 3 and 4 is very large, while being reasonable for long jobs.

2. Shortest Job First (SJF) :

Also known as Shortest Job Next. This non-preemptive policy reverses the bias against short jobs found in FCFS scheme by selecting the process from the READY queue which has the shortest estimated run time. It is essentially priority scheme where the priority is the inverse of the estimated run time. The queue jumping effect in SJF is called starvation. SJF policy can be illustrated by the following example :

	(A)	(B)	(C)
Job	Estimated Run Time	Waiting Time	Ratio B/A
3	1	0	0
1	2	1	0.5
4	3	3	1.0
5	50	6	0.1
2	60	56	0.9

Table : Illustration of SJF Policy

SJF is more applicable to batch working.

3. Round Robin (RR) :

In the Round Robin scheme, a process is selected for running from the ready queue in First-in-First-out scheme. However, if the process runs beyond a certain fixed length of time, called the time quantum, it is interrupted and returned to the end of the ready queque. A time quantum is generally from 10 to 100 milli-seconds. In other words, each active process is given a 'time slice' in rotation. The RR scheme is illustrated in the following figure :

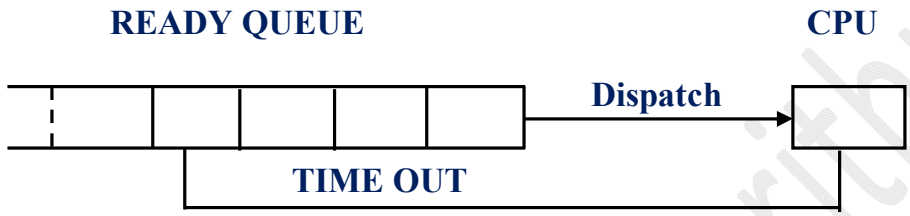


Fig. : Round Robin Scheduling

The RR scheme is preemptive but preemption occurs only by expiry of the time quantum. If the quantum size is increased sufficiently, the scheduling approaches FCFS, while if the quantum is too small, scheduling overhead in the form of context switch time.

4. Shortest Remaining Time (SRT) :

SRT is a preemptive version of SJF. At the time of dispatching the shortest queued process, say Job A will be started; however, if during the running of this process another job arrives whose run time is shorter than Job A's remaining run time then Job A will be preempted to allow the new job to start. SRT favours short jobs even more than SJF, since a currently running long job could be preempted by a new shorter one. It is also affected by the danger of starvation.

5. Highest Response Ratio Next (HRRN) :

This scheme is derived from the SJF method modified to reduce the SJF's bias against long jobs and to avoid the danger of starvation. In effect HRN derives a dynamic priority value based on the estimated run time and the incurred waiting time. The priority for each process is calculated from the formula :

$$\text{Priority (P)} = \frac{\text{Time waiting} + \text{Run time}}{\text{Run time}}.$$

The process with the highest priority value will be selected for running. When processes first appear in the ready queue, the time waiting will be zero and hence P will be equal to 1 for all processes. After a short period of waiting, the shorter jobs will be favoured. This technique guarantees that a job cannot be starved.

6. Multi-level Feedback Queues (MFQ) :

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.

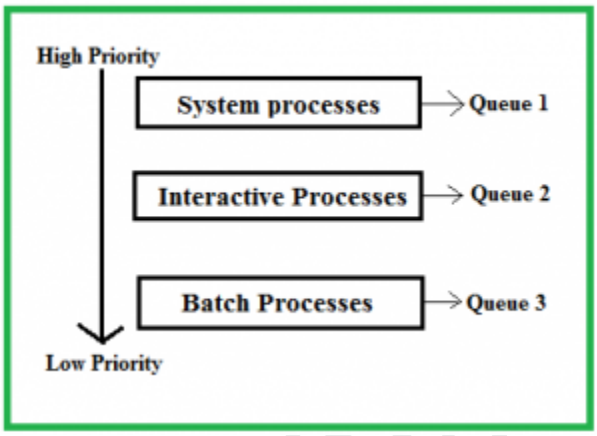
Multilevel feedback queue scheduling, however, allows a process to move between queues. The idea is to separate processes with different CPU-burst characteristics. If a process uses too much CPU time, it will be moved to a lower-priority queue. Similarly, a process that waits too long in a lower-priority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

For example, let us take three different types of processes : System processes, Interactive processes and Batch Processes. All three processes have their own queue. Now, look at the below figure.



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