CPU SCHEDULING ALGORITHMS

Objectives:

In this lab, student will be able to:

1. Understand the different CPU scheduling algorithms.
2. Compute the turnaround time, response time and waiting time for each process.

Scheduling Criteria & Optimization:

- CPU utilization – keep the CPU as busy as possible
    - Maximize CPU utilization
- Throughput – # of processes that complete their execution per time unit
    - Maximize throughput
- Turnaround time – amount of time to execute a particular process
    - Minimize turnaround time
- Waiting time – amount of time a process has been waiting in the ready queue
    - Minimize waiting time
- Response time – time from the submission of a request until the first response is produced (response time, is the time it takes to start responding, not the time it takes to output the response)
    - Minimize response time

CPU Scheduling algorithms:

(i) First-Come First Served (FCFS) Scheduling:
The process that requests the CPU first is allocated the CPU first.

(ii) Shortest-Job-First (SJF) Scheduling:
This algorithm associates with each process the length of its next CPU burst. When the CPU is available, it is assigned to the process that has the smallest next CPU burst. If the next CPU bursts of two processes are the same, FCFS scheduling is used to break the tie.
Two schemes:

- Non-preemptive – once CPU given to the process it cannot be preempted until it completes its CPU burst.

- Preemptive – if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is known as the Shortest-Remaining-Time-First (SRTF).

(iii) Priority Scheduling:
A priority number (integer) is associated with each process. The CPU is allocated to the process with the highest priority. A smaller value means a higher priority.
Two schemes:

- Preemptive
- Non-preemptive

Note: SJF is a priority scheduling where priority is the predicted next CPU burst time.

(iv) Round-Robin (RR) Scheduling:
The RR scheduling is the Preemptive version of FCFS. In RR scheduling, each process gets a small unit of CPU time (time quantum). Usually 10-100 ms. After quantum expires, the process is preempted and added to the end of the ready queue.

(v) Multilevel Queue (MQ) Scheduling:

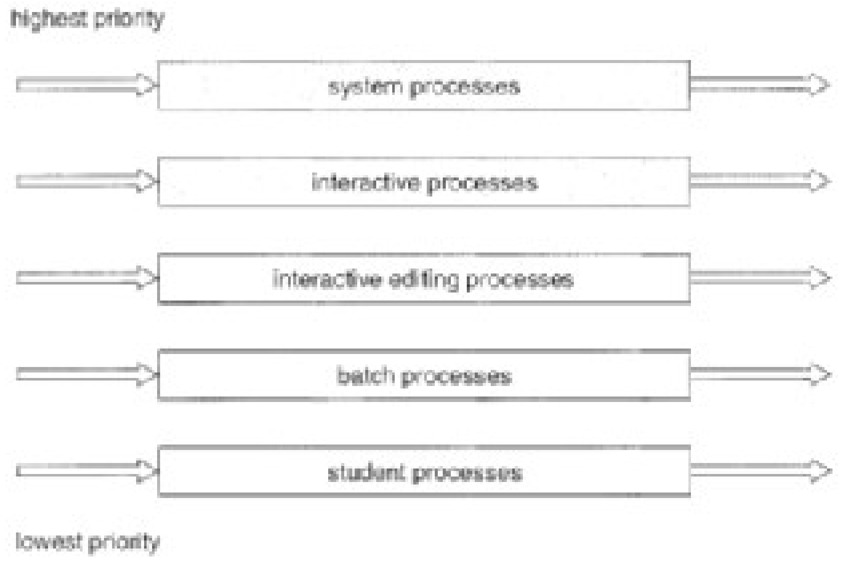

Figure 6.1 Multilevel queue scheduling

MQ scheduling is used when processes can be classified into groups. For example, **foreground** (interactive) processes and **background** (batch) processes. A MQ scheduling algorithm partitions the ready queue into several separate queues:

- foreground (interactive)
- background (batch)

Each process assigned to one queue based on its memory size, process priority, or process type. Each queue has its own scheduling algorithm

- foreground – RR
- background – FCFS

Scheduling must be done between the queues

- Fixed priority scheduling; (i.e., serve all from foreground then from background as shown Fig. 6.1).
- Time slice – each queue gets a certain portion of CPU time which it can schedule amongst its processes; i.e., 80% to foreground in RR and 20% to background in FCFS

(vi) Multilevel Feedback Queue (MFQ) Scheduling:

In MQ scheduling algorithm processes do not move from one queue to the other. In contrast, the MFQ scheduling algorithm, allows a process to move between queues. The idea is to separate processes according to the characteristics of their CPU bursts. If a process uses too much CPU time, it will be moved to a lower-priority queue. This scheme leaves I/O-bound and interactive processes in the higher-priority queues. In addition, a process that waits too long in a lower-priority queue may be moved to a higher-priority queue**.**

Example of Multilevel Feedback Queue:

Consider multilevel feedback queue scheduler with three queues as shown in Fig. 6.2.

- Q_0 – RR with time quantum 8 milliseconds
- Q_1 – RR time quantum 16 milliseconds
- Q_2 – FCFS

MFQ Scheduling

- A process queue in Q_0 *is given a time quantum of* 8 milliseconds. If it does not finish in 8 milliseconds, the job is moved to the tail of queue Q_1.
- When Q_0 is empty, the process at the head of Q_1 is given a quantum of 16 milliseconds. If it does not complete, it is pre-empted and moved to queue Q_2 .

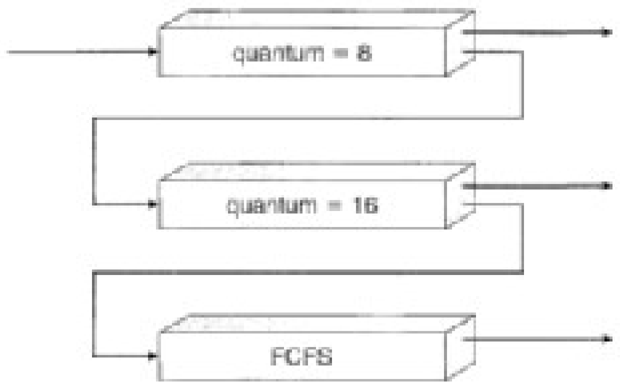

Figure 6.2 Multilevel feedback queues