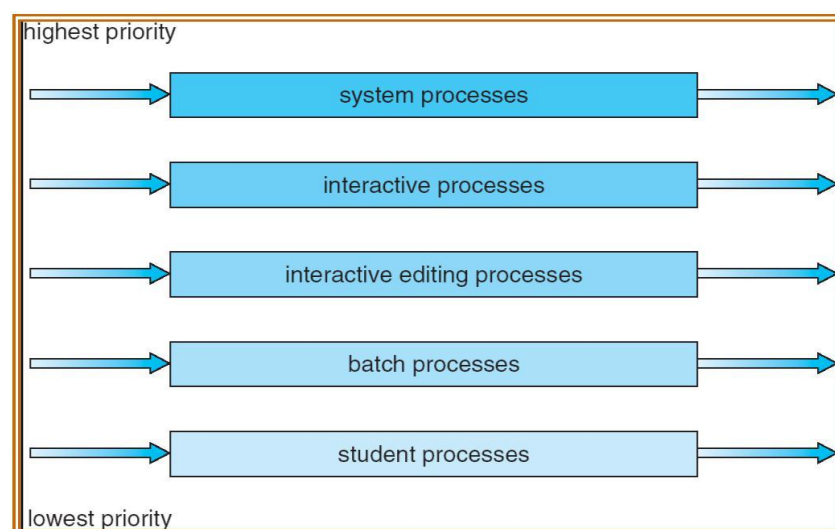


6)

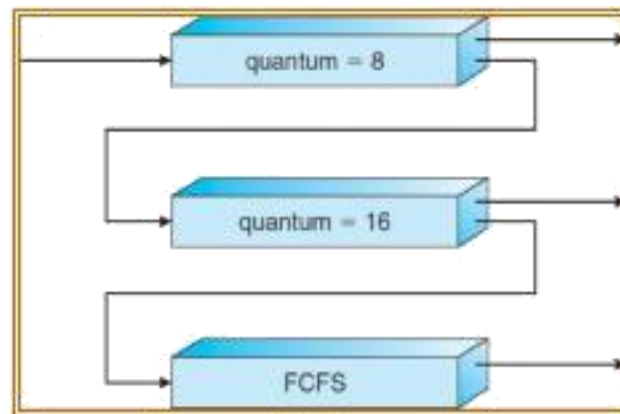
Multilevel Queue Scheduling

- Different types of processes have different response-time requirements and so may have different scheduling needs. In addition, some processes may have priority over other processes.
- A **multilevel queue scheduling algorithm** partitions the ready queue into several separate queues.
- The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority, or process type.
- Each queue has its own scheduling algorithm.
- In addition, there must be scheduling among the queues, which is commonly implemented as fixed-priority preemptive scheduling.
- Let's look at an example of a multilevel queue scheduling algorithm with five queues, listed below in order of priority:
 1. System processes
 2. Interactive processes
 3. Interactive editing processes
 4. Batch processes
 5. Student processes
- Each queue has absolute priority over lower-priority queues. No process in the batch queue, for example, could run unless the queues for system processes, interactive processes, and interactive editing processes were all empty. If an interactive editing process entered the ready queue while a batch process was running, the batch process would be preempted.



Multilevel Feedback-Queue Scheduling

- When the multilevel queue scheduling algorithm is used, processes are permanently assigned to a queue when they enter the system.
- This setup has the advantage of low scheduling overhead, but it is inflexible.
- The **multilevel feedback-queue scheduling algorithm**, in contrast, 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. This form of aging prevents starvation.



- For example, consider a multilevel feedback-queue scheduler with three queues, numbered from 0 to 2 (As shown in above figure).
- The scheduler first executes all processes in queue 0. Only when queue 0 is empty will it execute processes in queue 1.
- Similarly, processes in queue 2 will only be executed if queues 0 and 1 are empty.
- A process that arrives for queue 1 will preempt a process in queue 2. A process in queue 1 will in turn be preempted by a process arriving for queue 0.
- A process entering the ready queue is put in queue 0. A process in queue 0 is given a time quantum of 8 milliseconds. If it does not finish within this time, it is moved to the tail of queue 1.
- If queue 0 is empty, the process at the head of queue 1 is given a quantum of 16 milliseconds. If it does not complete, it is preempted and is put into queue 2.
- Processes in queue 2 are run on an FCFS basis but are run only when queues 0 and 1 are empty.
- This scheduling algorithm gives highest priority to any process with a CPU burst of 8 milliseconds or less. Such a process will quickly get the CPU, finish its CPU burst, and go off to its next I/O burst. Processes that need more than

8 but less than 24 milliseconds are also served quickly, although with lower priority than shorter processes.

- Long processes automatically sink to queue 2 and are served in FCFS order with any CPU cycles left over from queues 0 and 1.
 - 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
-