Scheduling Algorithms

CPU scheduling deals with the problem of deciding which of the processes in the ready queue is to be allocated the CPU. There are many different CPU scheduling algorithms:

1. First-Come, First-Served (FCFS) Scheduling
2. Shortest-Job-First (SJF) Scheduling
3. Shortest-Remaining-Time-First (SRTF) Scheduling
4. Priority Scheduling
5. Round-Robin (RR) Scheduling

First-Come, First-Served (FCFS) Scheduling

- It is the simplest CPU scheduling algorithm. In this scheme the process that requests the CPU first is allocated the CPU first.
- The implementation of the FCFS policy is easily managed with a FIFO queue.
- The average waiting time in FCFS policy is quite long.
- The FCFS scheduling algorithm is non-preemptive.
- It is not suitable for time sharing systems.

Convoy Effect

- FCFS may suffer from the **convoy effect** if the burst time of the first job is the highest among all.
- If the CPU gets the processes of the higher burst time at the front end of the ready queue then the processes of lower burst time may get blocked which means they may never get the CPU if the job in the execution has a very high burst time. This is called **convoy effect** or **starvation**.

Shortest-Job-First (SJF) Scheduling

- It is also known as “Shortest-Next-CPU-Burst” algorithm.
- In this scheme 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.
- The SJF scheduling algorithm is also non-preemptive.
- This algorithm is optimal; it gives the minimum average waiting time.
- SJF is priority scheduling where priority is the inverse of predicted next CPU burst time

Shortest Remaining Time First (SRTF)

- Shortest Remaining Time First scheduling is also known as preemptive SJF scheduling.

- In this scheme the choice arises when a new process arrives at the ready queue while a previous process is still executing.
- If the next CPU burst of newly arrived process is shorter then what is left of the currently executing process, then SRTF algorithm will preempt the currently executing process.

Priority Scheduling

- A priority is associated with each process, and the CPU is allocated to the process with the highest priority.
- Equal priority processes are scheduled in FCFS order.
- Priorities are generally indicated by some fixed range of numbers, such as 0 to 7 or 0 to 4095. There is no general agreement on whether 0 is highest or lowest priority.
- Priority scheduling can be either preemptive or non-preemptive.
- When a process arrives at the ready queue, its priority is compared with the priority of the currently running process.
- A preemptive priority scheduling algorithm will preempt the CPU if the priority of the newly arrived process is higher than the priority of the currently running process.
- A non-preemptive priority scheduling algorithm will simply put the new process at the head of the ready queue.
- Problem ≡ ***Starvation*** – low priority processes may never execute
- Solution ≡ ***Aging*** – as time progresses increase the priority of the process

Round Robin (RR) Scheduling

- The Round Robin (RR) scheduling algorithm is designed especially for the time sharing systems.
- It is preemptive scheduling algorithm.

Multilevel Queue Scheduling (MQS)

- Another class of scheduling algorithms has been created for situations in which processes are easily classified into different groups. For example, a common division is made between foreground (or interactive) processes and background (or batch) processes. These two types of processes have different response-time requirements, and so might have different scheduling needs.
- A MQS 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, priority or process type.
- Each queue has its own scheduling algorithm. For example, separate queues might be used for foreground and background processes. The foreground queue might be scheduled by a RR algorithm, while the background queue is scheduled by an FCFS algorithm.

- In addition, there must be scheduling among the queues, which is commonly implemented as fixed-priority preemptive scheduling.

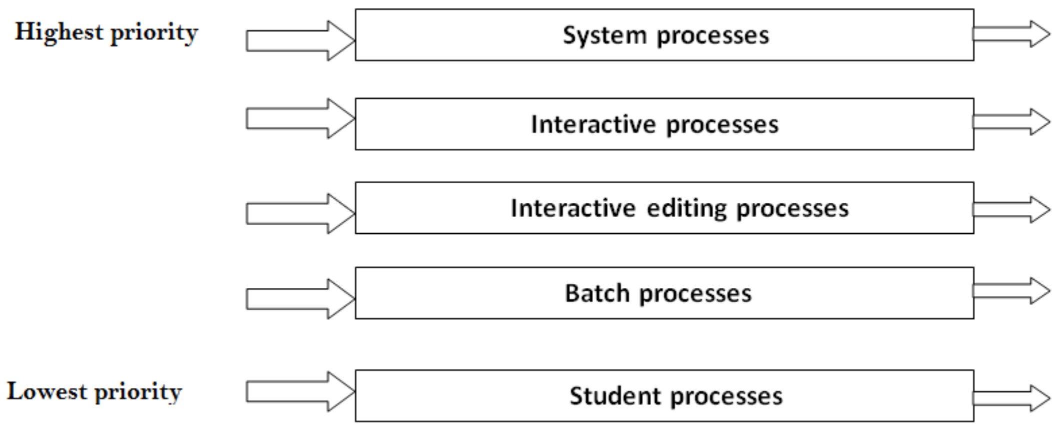

Figure: Multilevel Queue Scheduling

Multilevel Feedback Queue Scheduling

- Normally, in a multilevel queue-scheduling algorithm, processes are permanently assigned to a queue on entry to the system. Processes do not move between queues. If there are separate queues for foreground and background processes, for example, processes do not move from one queue to the other, since processes do not change their foreground or background nature. This setup has 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. This scheme leaves I/O-bound and interactive processes in the higher-priority queues. 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.

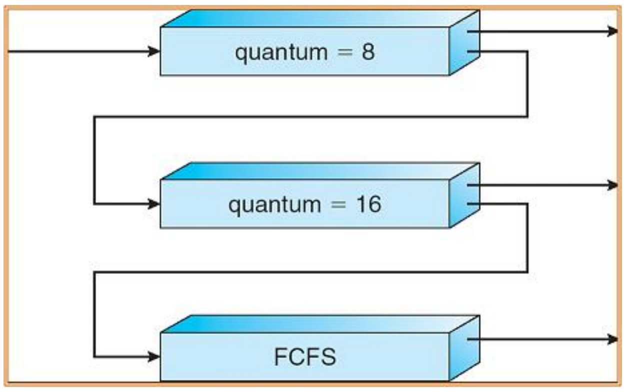

Figure: Multilevel Feedback Queue Scheduling

- For example, consider a multilevel feedback queue scheduler with three queues, numbered from 0 to 2. 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 be executed only if queues 0 and 1 are empty. A process that arrives for queue 1 will preempt a process in queue 2. A process that arrives for queue 0 will, in turn, preempt a process in queue 1. 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, 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.