SCHEDULING ALGORITHMS - II

According to R-R scheduling processes are executed in FCFS order. So, firstly P1(burst time=20ms) is executed but after 4ms it is preempted and new process P2 (Burst time = 3ms) starts its execution whose execution is completed before the time quantum. Then next process P3 (Burst time=4ms) starts its execution and finally remaining part of P1 gets executed with time quantum of 4ms.

Waiting time of Process P1: 0ms + (11 – 4)ms = 7ms
Waiting time of Process P2: 4ms
Waiting time of Process P3: 7ms

Average Waiting time: (7+4+7)/3=6ms

4. 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 the FCFS order.

Assigning priority:
1.To prevent high priority process from running indefinitely the scheduler may decrease the priority of the currently running process at each clock interrupt. If this causes its priority to drop below that of the next highest process, a process switch occurs.
2.Each process may be assigned a maximum time quantum that is allowed to run. When this quantum is used up, the next highest priority process is given a chance to run.

Priorities can be assigned statically or dynamically. For UNIX system there is a command nice for assigning static priority.
It is often convenient to group processes into priority classes and use priority scheduling among the classes but round-robin scheduling within each class.

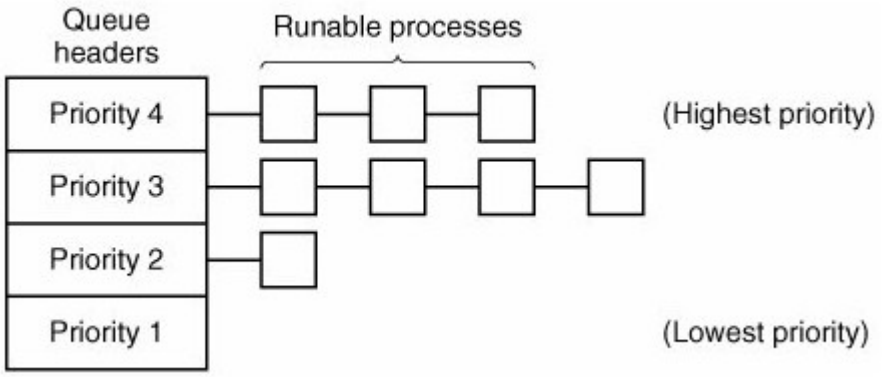
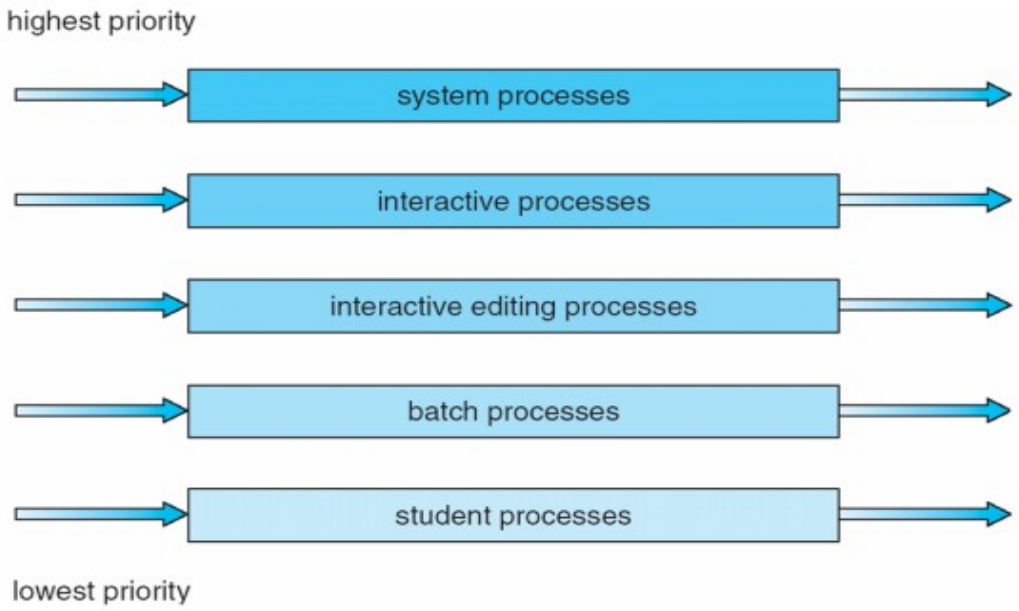


Fig: A scheduling algo. with four Priority classes

Problems in Priority Scheduling:
Starvation:
Low priority process may never execute.
Solution: **Aging**: As time progress increase the priority of Process.

Multilevel Queue Scheduling:

In this scheduling processes are classified into different groups. A common example may be foreground(or Interactive processes) or background (or batch processes).



Ready queue is partitioned into separate queues:
foreground (interactive)
background (batch)

Let us look at an example of a multilevel queue scheduling algorithm with five queues, listed below in the 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 processes in the batch queue, for example could run unless the queue for System processes , interactive processes and interactive editing processes were all empty. If an interactive editing process enters the ready queue while a batch process was running the batch process would be preempted.
Another possibility is to time slice between the queues. For instance foreground queue can be given 80% of the CPU time for RR scheduling among its processes, whereas the background receives 20% of the CPU time.

Guaranteed Scheduling:

•Make real promises to the users about performance.
•If there are n users logged in while you are working, you will receive about 1 /n of the CPU power.
•Similarly, on a single-user system with n processes running, all things being equal, each one should get 1 /n of the CPU cycles.
•To make good on this promise, the system must keep track of how much CPU each process has had since its creation.
•CPU Time entitled= (Time Since Creation)/n
•Then compute the ratio of Actual CPU time consumed to the CPU time entitled.
•A ratio of 0.5 means that a process has only had half of what it should have had, and a ratio of 2.0 means that a process has had twice as much as it was entitled to.
•The algorithm is then to run the process with the lowest ratio until its ratio has moved above its closest competitor.

Lottery Scheduling:

Lottery Scheduling is a probabilistic scheduling algorithm for processes in an operating system.
Processes are each assigned some number of lottery tickets for various system resources such as CPU time.;and the scheduler draws a random ticket to select the next process. The distribution of tickets need not be uniform; granting a process more tickets provides it a relative higher chance of selection. This technique can be used to approximate other scheduling algorithms, such as Shortest job next and Fair-share scheduling.

Lottery scheduling solves the problem of starvation. Giving each process at least one lottery ticket guarantees that it has non-zero probability of being selected at each scheduling operation.

More important process can be given extra tickets to increase their odd of winning. If there are 100 tickets outstanding, & one process holds 20 of them it will have 20% chance of winning each lottery. In the long run it will get 20% of the CPU. A process holding a fraction f of the tickets will get about a fraction of the resource in questions.

Two-Level Scheduling:

Performs process scheduling that involves swapped out processes. Two-level scheduling is needed when memory is too small to hold all the ready processes .
Consider this problem: A system contains 50 running processes all with equal priority. However, the system's memory can only hold 10 processes in memory simultaneously. Therefore, there will always be 40 processes swapped out written on virtual memory on the hard disk
It uses two different schedulers, one lower-level scheduler which can only select among those processes in memory to run. That scheduler could be a Round-robin scheduler. The other scheduler is the higher-level scheduler whose only concern is to swap in and swap out processes from memory. It does its scheduling much less often than the lower-level scheduler since swapping takes so much time. the higher-level scheduler selects among those processes in memory that have run for a long time and swaps them out