Scheduling Algorithms

First-Come, First-Served (FCFS) Scheduling

Process	Burst Time
P1	24
P2	3
P3	3

Suppose that the processes arrive in the order: *P1* , *P2* , *P3*

The Gantt Chart for the schedule is:

P1	P2	P3
0	24	27 30

Waiting time for *P1* **= 0;** *P2* **= 24;** *P3* **= 27**
Average waiting time: (0 + 24 + 27)/3 = 17
Suppose that the processes arrive in the order
P2 , *P3* , *P1* .
The Gantt chart for the schedule is:

P2	P3	P1
0	3	6 30

Waiting time for *P1* = **6;** *P2* **= 0;** *P3* = **3**
Average waiting time: (6 + 0 + 3)/3 = 3
Much better than previous case.
Convoy effect **short process behind long process**

Shortest-Job-First (SJR) Scheduling

Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time.
Two schemes:
1. non pre- emptive – once CPU given to the process it cannot be preempted until completes its CPU burst.
2. preemptive – if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is know as the Shortest-Remaining-Time-First (SRTF).
SJF is optimal – gives minimum average waiting time for a given set of processes.

Process	Arrival Time	Burst Time
P1	0.0	7
P2	2.0	4
P3	4.0	1
P4	5.0	4

SJF (non-preemptive)

P1	P3	P2	P4
0 7	8	12	16

Average waiting time = [0 +(8-2)+(7-4) +(12-5)] /4 =4

Example of Preemptive SJF

Proces	Arrival Time	Burst Time
P1	0.0	7
P2	2.0	4
P3	4.0	1
P4	5.0	4

SJF (preemptive)

P1	P2	P3	P2	P4	P1

0 2 4 5 7 11 16

Average waiting time = (9 + 1 + 0 +2)/4 =3

Determining Length of Next CPU Burst

Can only estimate the length.
Can be done by using the length of previous CPU bursts, using exponential averaging.

Prediction of the Length of the Next CPU Burst

Pn+1 = a tn +(1-a)Pn
This formula defines an exponential average
Pn stores the past history
tn contents are most recent information
the parameter "a "controls the relative weight of recent and past history of in our prediction
If a =0 then Pn +1 =Pn
That is prediction is constant
If a = 1 then Pn +1 = tn
Prediction is last cpu burst

Priority Scheduling

A priority number (integer) is associated with each process
The CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority).

1. Preemptive
2. nonpreemptive

SJF is a priority scheduling where priority is the predicted next CPU burst time.
Problem ≡ Starvation – low priority processes may never execute.
Solution ≡ Aging – as time progresses increase the priority of the process.

Round Robin (RR)

Each process gets a small unit of CPU time (*time quantum*), usually 10-100 milliseconds.
After this time has elapsed, the process is preempted and added to the end of the ready queue.
If there are *n* processes in the ready queue and the time quantum is *q*, then each process gets 1/*n* of the
CPU time in chunks of at most *q* time units at once. No process waits more than (*n*-1)*q* time units.
Performance

1. *q* large _ FIFO
2. *q* small _ *q* must be large with respect to context switch, otherwise overhead is too high.

Example of RR with Time Quantum = 4

Process	Burst Time
P1	24
P2	*3*
P3	*3*

The Gantt chart is:

P1	P2	P3	P1	P1	P1	P1	P1

0 4 7 10 14 18 22 26 30

Average waiting time = [(30-24)+4+7]/3 = 17/3 =5.66

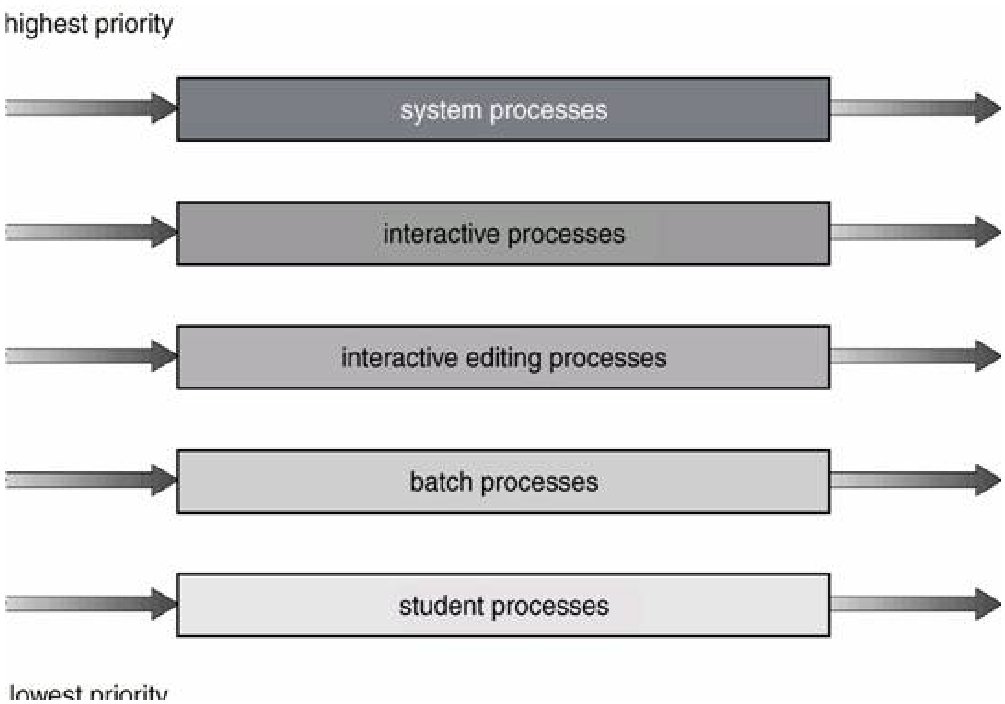
Multilevel Queue

Ready queue is partitioned into separate queues:
foreground (interactive)
background (batch)
Each queue has its own scheduling algorithm,
foreground – RR
background – FCFS
Scheduling must be done between the queues.

1. **Fixed priority scheduling; (i.e., serve all from foreground then from background). Possibility of starvation.**
2. **Time slice – each queue gets a certain amount of CPU time which it can schedule amongst its processes; i.e., 80% to foreground in RR**

1. **20% to background in FCFS**

Multilevel Queue Scheduling

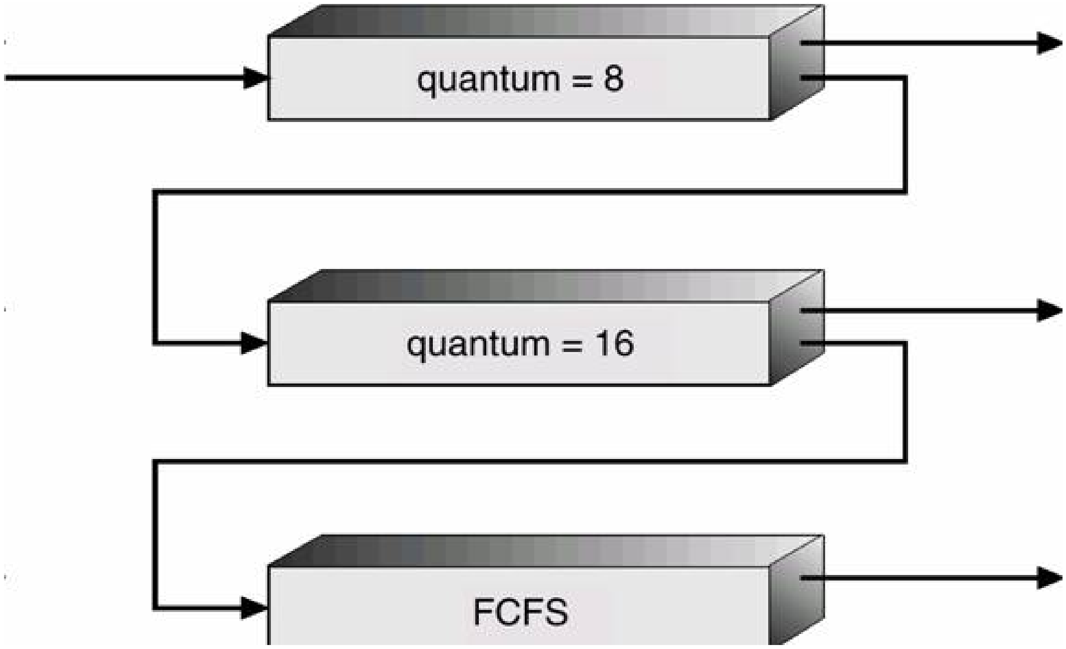


Multilevel Feedback Queue

A process can move between the various queues; aging can be implemented this way.
Multilevel-feedback-queue scheduler defined by the following parameters:

1. **number of queues**
2. **scheduling g algorithms for each queue**
3. **method used to determine when to upgrade a process**
4. **method used to determine when to demote a process**
5. **method used to determine which queue a process will enter when that process needs service**

Example of Multilevel Feedback Queue



Three queues:

1. **$Q0$ – time quantum 8 milliseconds**
2. **$Q1$ – time quantum 16 milliseconds**
3. **$Q2$ – FCFS**

Scheduling

1. **A new job enters queue *Q0* which is served FCFS . When it gains CPU, job receives 8 milliseconds.**
   If it does not finish in 8 milliseconds, job is moved to queue $Q1$.
2. **At $Q1$ job is again served FCFS and receives 16 additional milliseconds. If it still does not complete,**
   it is preempted and moved to queue $Q2$.

[BACK]