Priority Scheduling

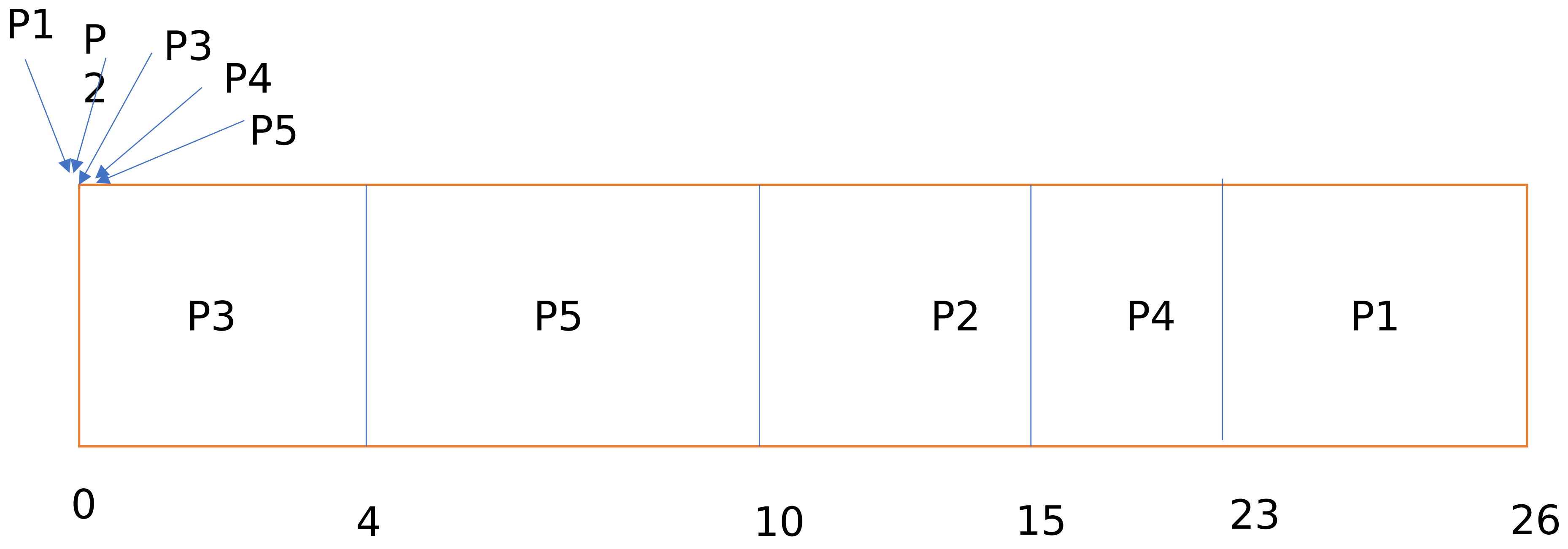
- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority).
    - Preemptive
    - nonpreemptive
- SJF is a priority scheduling where priority is the predicted next CPU burst time.
- Problem

  Starvation – low priority processes may never execute.
- Solution

  Aging – as time progresses increase (or redefine) the priority of process.

Q1. Using non preemptive priority scheduling technique compute the starting time of process P3 and P4; finishing time of P2 and P5; waiting time of process P3 and turn around time of process P5 for following batch of jobs

Process	Burst Time (ns)	Priority
P1	3	9
P2	5	3
P3	4	0
P4	8	8
P5	6	2



Starting Time of P1 = 23
P2= 10
P3 = 0
P4 = 15
P5 = 4

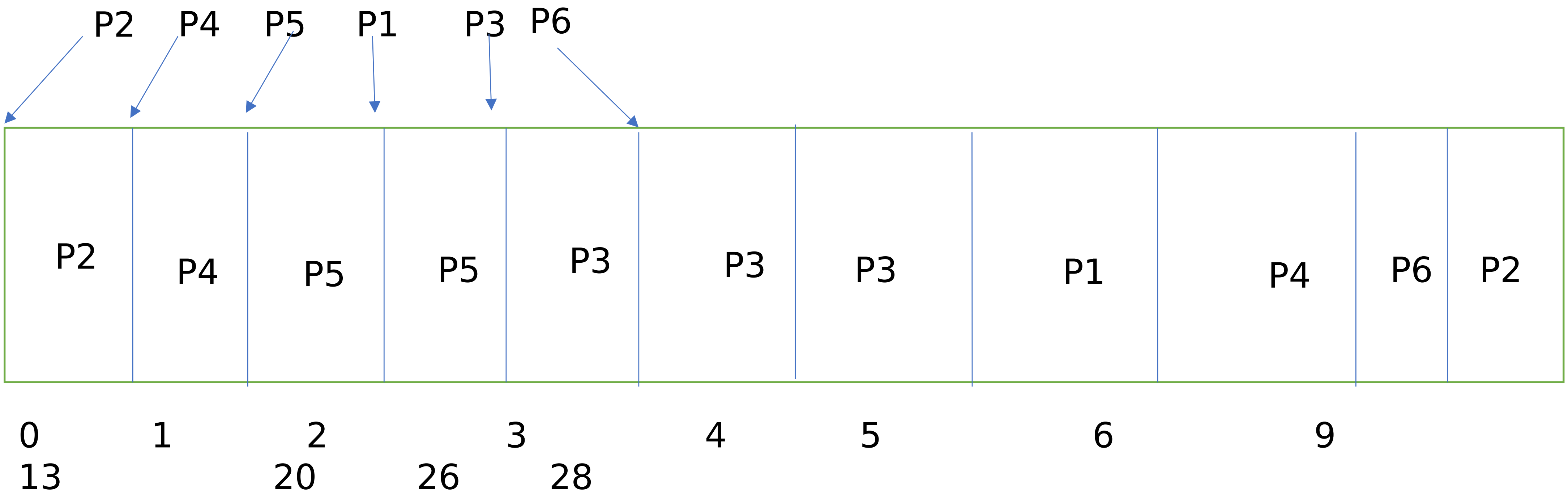
Finishing Time of P1 = 26
P2= 15
P3 = 4
P4 = 23
P5 = 10

Waiting Time of P1 = 23
P2= 10
P3 = 0
P4 = 15
P5 = 4

Turn Around Time of P1 = 26
P2=
15
P3 =
4
P4 = 23
P5 = 10

Q2. Using Preemptive priority scheduling technique compute the AWT and ATT for following batch of jobs

Process	Burst Time (ns)	Priority	Arrival Time
P1	4	3	3
P2	3	8	0
P3	5	0	4
P4	8	4	1
P5	2	2	2
P6	6	5	5



P2 P4 P5 P1 P3 P6

P2	P4	P5	P5	P3	P3	P3	P1	P4	P6	P2

0 1 2 3 4 5 6 9

13 20 26 28

Waiting Time of P1 = 6 (Finishing time-(arrival time + burst time))

P2 = 25

P3 = 0

P4 = 11

P5 = 0

P6 = 15

AWT = 57/6 = 9.5 ns

Turn Around Time of P1 = 10 (waiting time + bur time) or

(Finishing- Arriv

P2 = 28

P3 = 5

P4 = 19

P5 = 2

P6 = 21

AWT = 85/6 = 14.16 ns

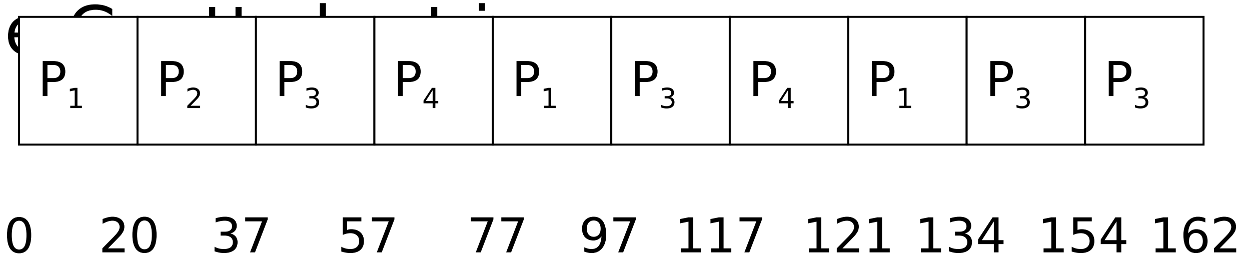
Round Robin (RR)

- 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 no process waits more than (*n*-1)*q* time units.
- Performance
  - *q* large ⇒ FIFO
  - *q* small ⇒ *q* must be large with respect to context switch, otherwise overhead is too high.

Example: RR with Time Quantum = 20

Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

- The Gantt chart is

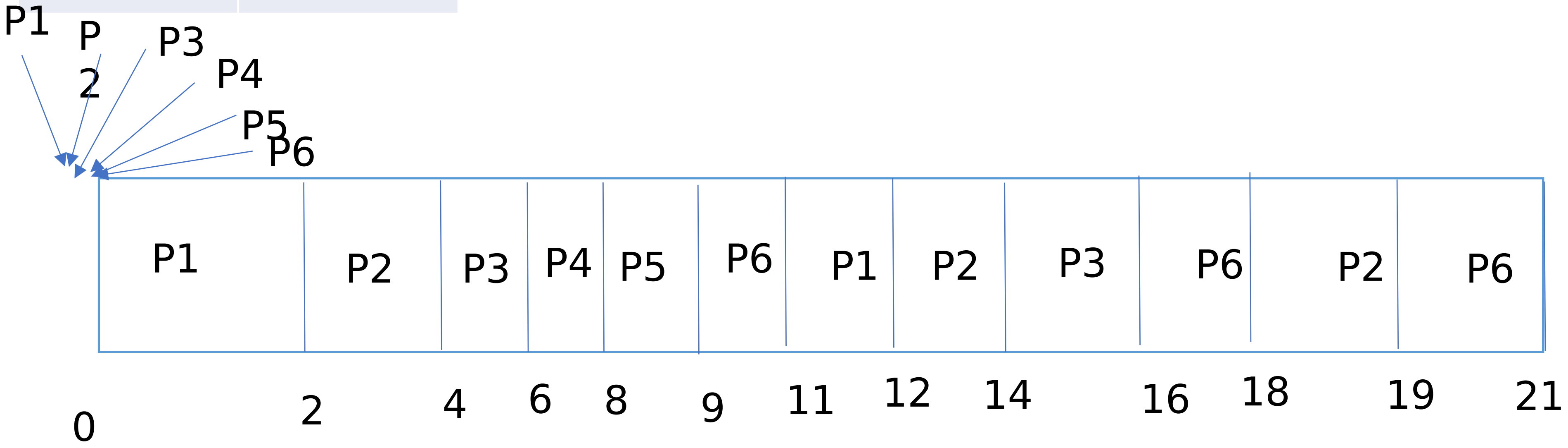


Process	Burst Time (ms)
P1	~~3~~ ~~1~~
P2	~~5~~ ~~3~~ ~~1~~
P3	~~4~~ ~~2~~
P4	~~2~~ 0
P5	~~1~~ 0
P6	~~6~~ ~~4~~ ~~2~~

Calculate following:

1. Waiting time of P3 and P5
2. Turn Around Time of P2 and P4
3. No. of context switching excluding first and last one
4. AWT and ATT

Time quantum is 2 ms.



P1	P2	P3	P4	P5	P6	P1	P2	P3	P6	P2	P6

0 2 4 6 8 9 11 12 14 16 18 19 21

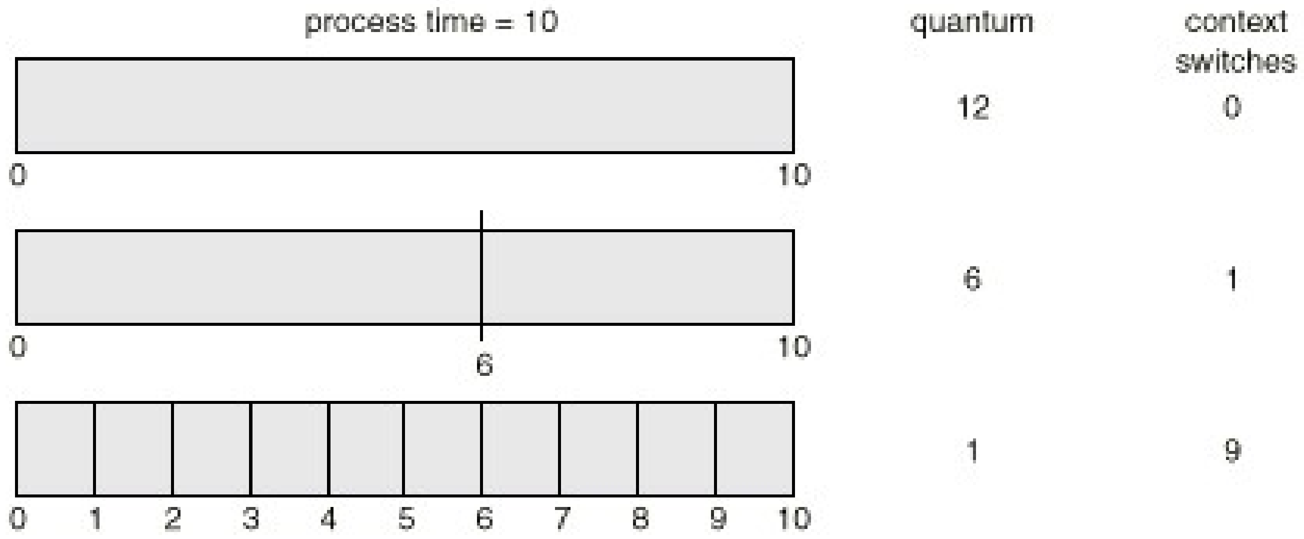
Waiting Time of P1 = 9
P2 = 14
P3 = 12
P4 = 6
P5 = 8
P6 = 15
AWT = 64/6 =10.66 ms

Turn Around Time of P1 = 12
P2 = 19
P3 = 16
P4 = 8
P5 = 9
P6 = 21
ATT = 85/6 =14.16 ms

Total number of context switching = 13
Excluding first and last = 11

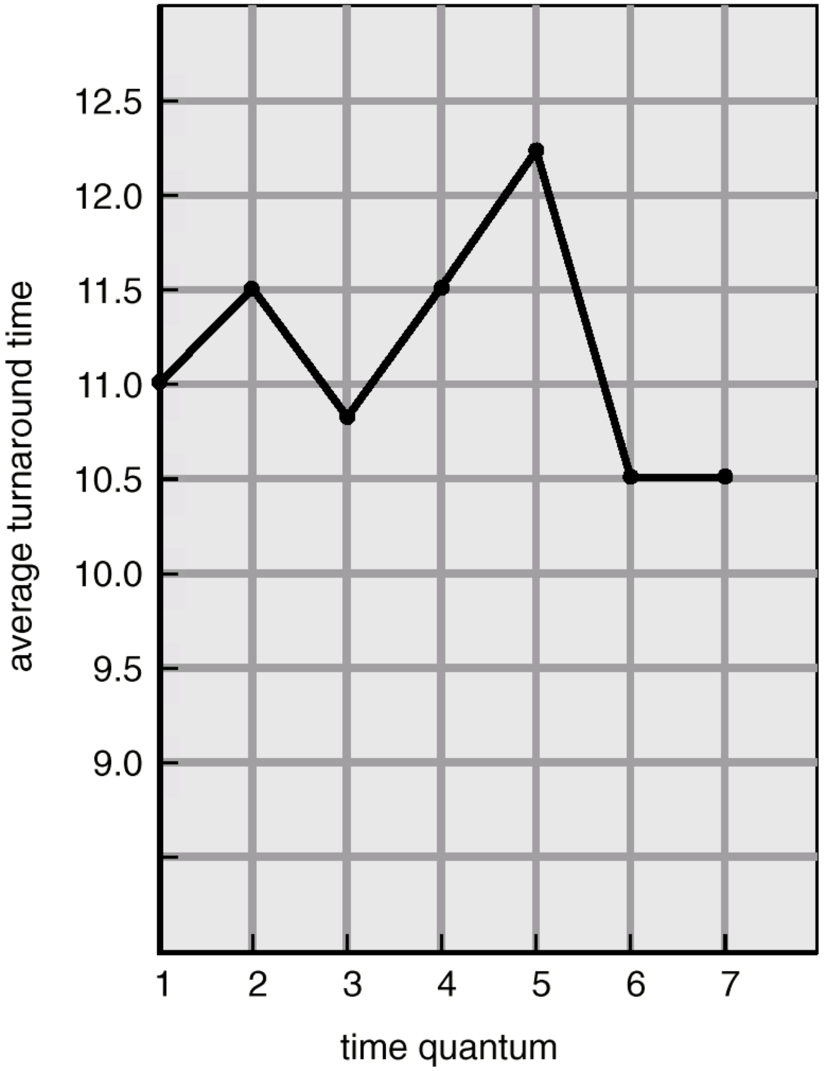
Typically, higher average turnaround than SJF, but better *response*.

How a Smaller Time Quantum Increases Context Switches



quantum	context switches
12	0
6	1
1	9

Turnaround Time Varies With The Time Quantum



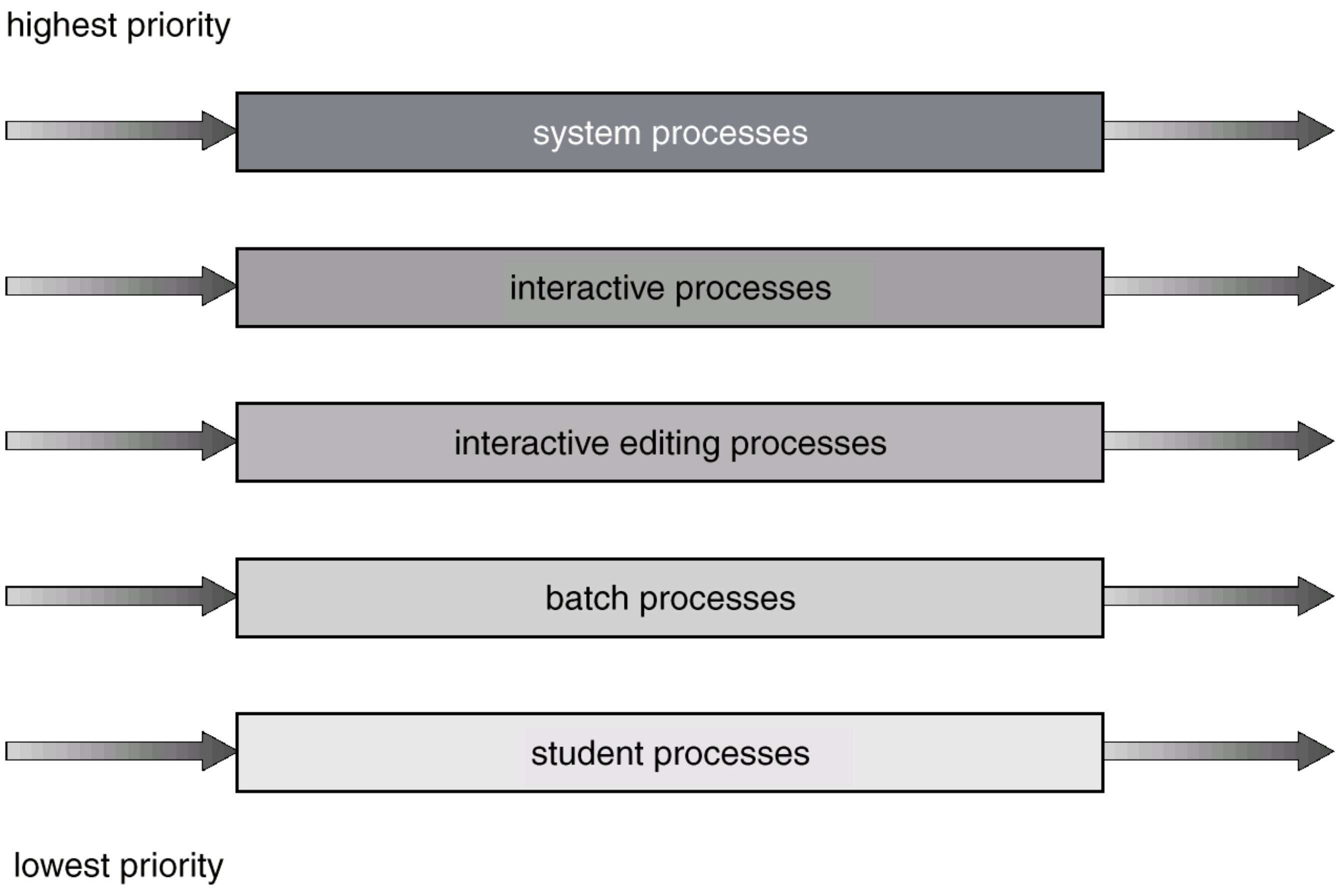
process	time
P_1	6
P_2	3
P_3	1
P_4	7

Topics for Self-Study

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.
  - Fixed priority scheduling; i.e., serve all from foreground then from background. Possibility of starvation.
  - 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
  - 20% to background in FCFS

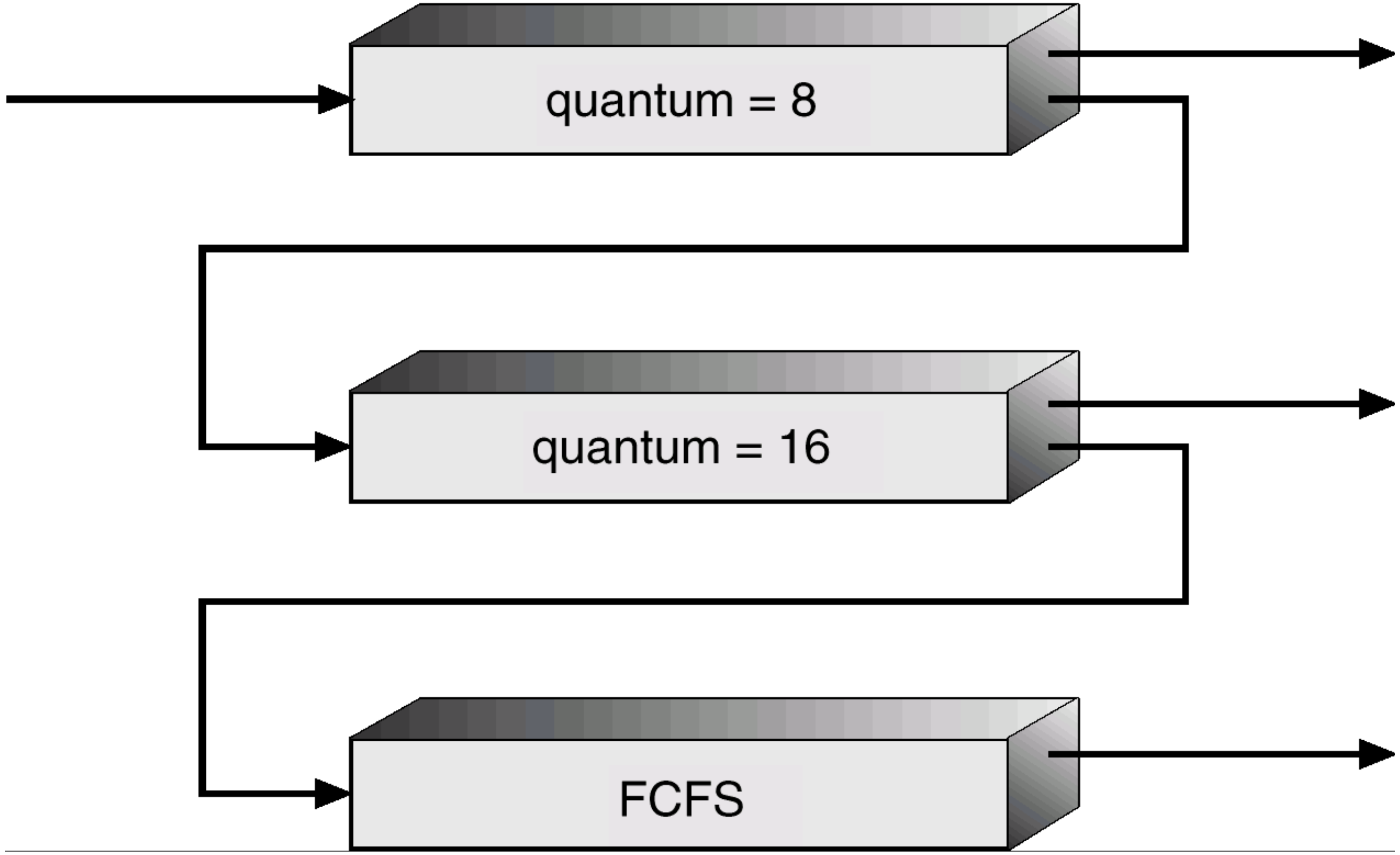
Multilevel Queue Scheduling



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:
  - number of queues
  - scheduling algorithms for each queue
  - method used to determine when to upgrade a process
  - method used to determine when to demote a process
  - method used to determine which queue a process will enter when that process needs service

Multilevel Feedback Queues



Example of Multilevel Feedback Queue

- Three queues:
  - Q_0 – time quantum 8 milliseconds
  - Q_1 – time quantum 16 milliseconds
  - Q_2 – FCFS
- Scheduling
  - A new job enters queue Q_0 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 Q_1.
  - At Q_1 job is again served FCFS and receives 16 additional milliseconds. If it still does not complete, it is preempted and moved to queue Q_2.