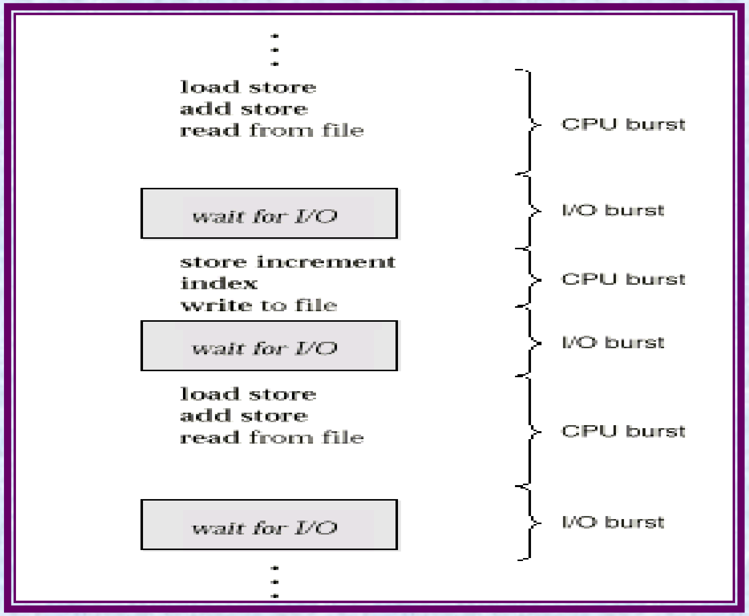
Chapter 6: CPU Scheduling

- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms
- Multiple-Processor Scheduling
- Real-Time Scheduling
- Algorithm Evaluation

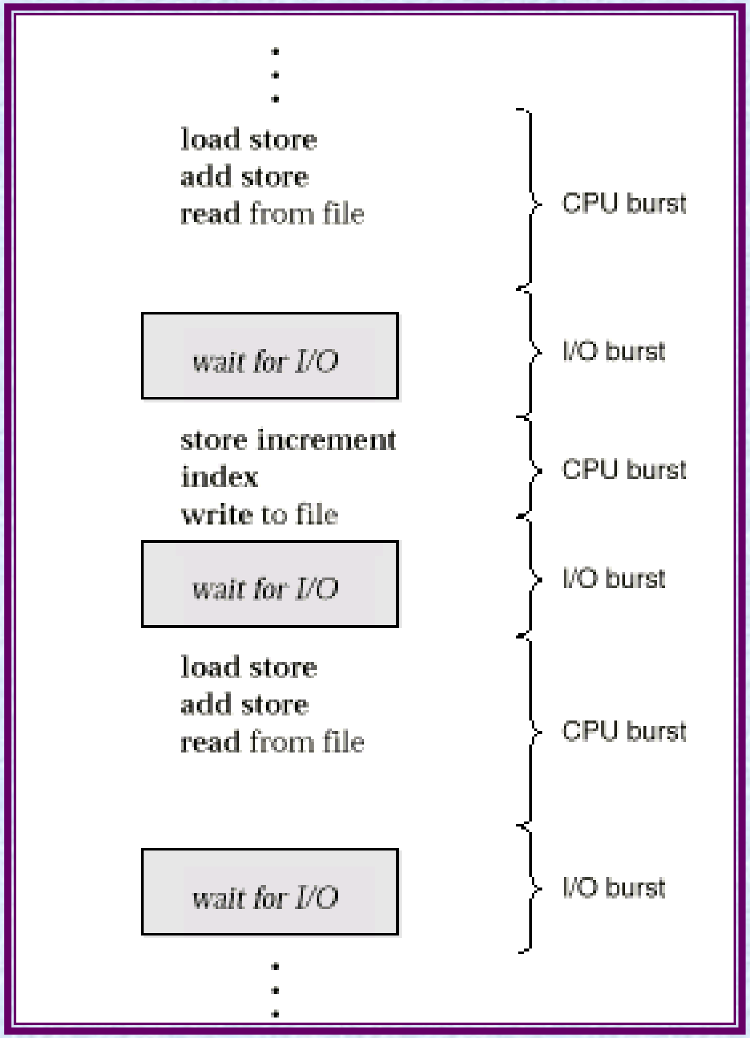
Basic Concepts

- Maximum CPU utilization obtained with multiprogramming
- CPU–I/O Burst Cycle – Process execution consists of a *cycle* of CPU execution and I/O wait.
- CPU burst distribution

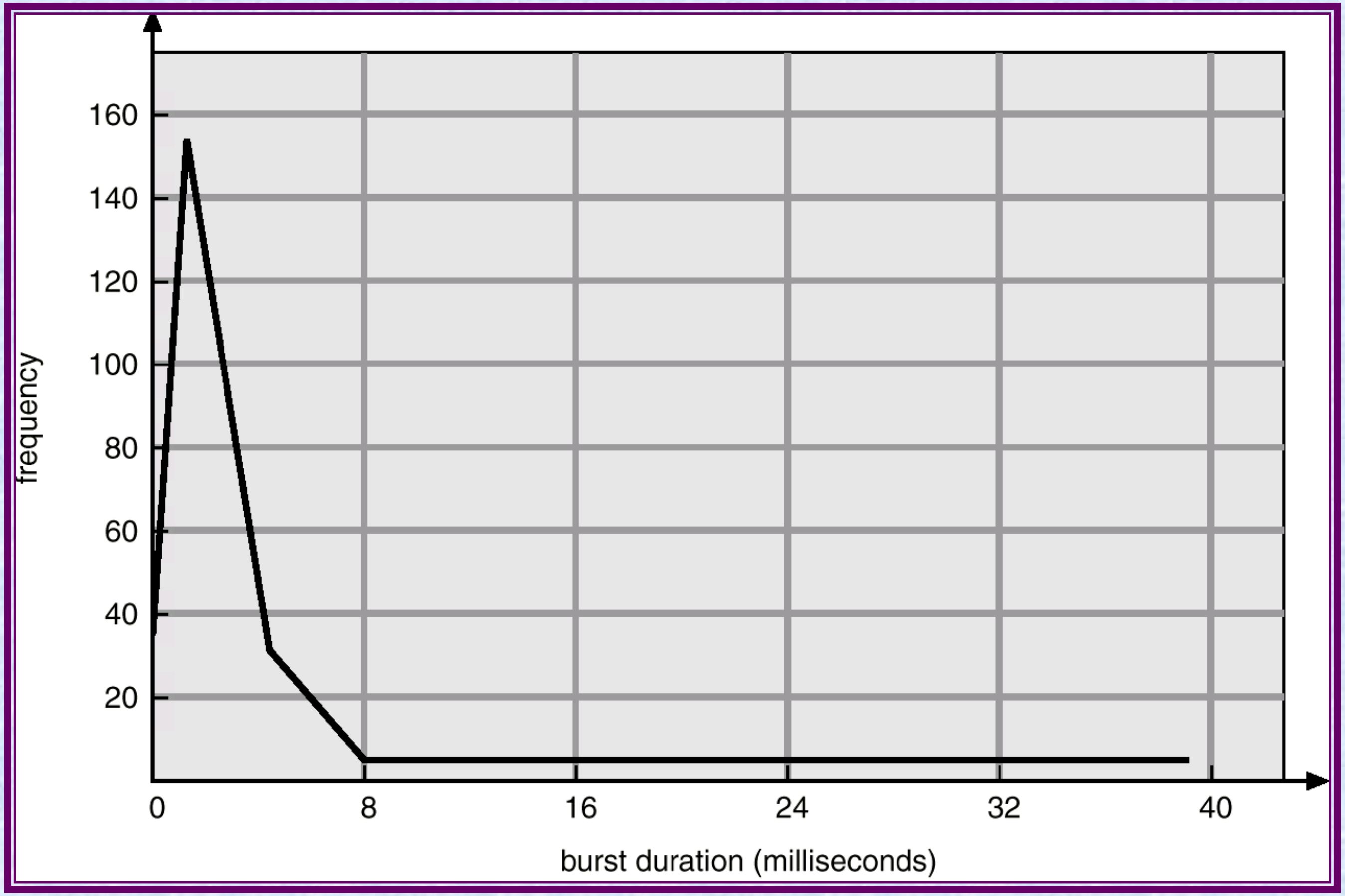
Alternating Sequence of CPU And I/O Bursts



Alternating Sequence of CPU And I/O Bursts



Histogram of CPU-burst Times



CPU Scheduler

- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them.
- CPU scheduling decisions may take place when a process:
  1. Switches from running to waiting state.
  2. Switches from running to ready state.
  3. Switches from waiting to ready.
  4. Terminates.
- Scheduling under 1 and 4 is *nonpreemptive*.
- All other scheduling is *preemptive.*

Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
  - switching context
  - switching to user mode
  - jumping to the proper location in the user program to restart that program
- *Dispatch latency* – time it takes for the dispatcher to stop one process and start another running.

Scheduling Criteria

- CPU utilization – keep the CPU as busy as possible
- Throughput – # of processes that complete their execution per time unit
- Turnaround time – amount of time to execute a particular process
- Waiting time – amount of time a process has been waiting in the ready queue
- Response time – amount of time it takes from when a request was submitted until the first response is produced, **not** output (for time-sharing environment)

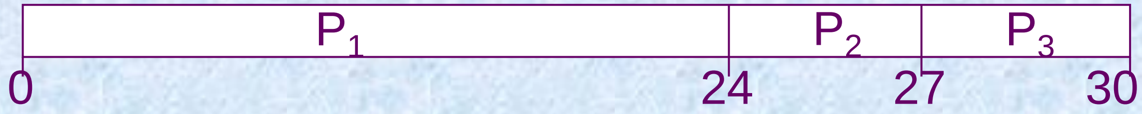
Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

First-Come, First-Served (FCFS) Scheduling

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3
  The Gantt Chart for the schedule is:



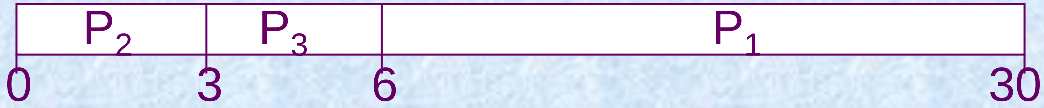
- Waiting time for P_1 = 0; P_2 = 24; P_3 = 27
- Average waiting time: (0 + 24 + 27)/3 = 17

FCFS Scheduling (Cont.)

Suppose that the processes arrive in the order

$$P_2 , P_3 , P_1 .$$

- The Gantt chart for the schedule is:



- Waiting time for $P_1 = 6; P_2 = 0; P_3 = 3$
- Average waiting time: (6 + 0 + 3)/3 = 3
- Much better than previous case.
- *Convoy effect* short process behind long process

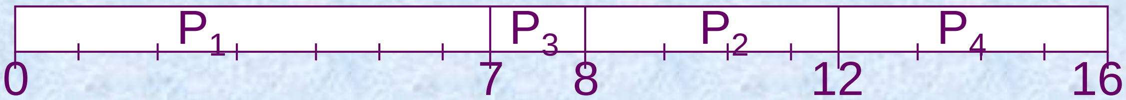
Shortest-Job-First (SJF) Scheduling

- Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time.
- Two schemes:
  - nonpreemptive – once CPU given to the process it cannot be preempted until completes its CPU burst.
  - 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.

Example of Non-Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (non-preemptive)



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

  = 0 + 6 +3 +7 = 16/4 = 4

- Average Turn time = ((0+7) + (6+4) + (3+1) + (7+4))/4
  - » 7 + 10 + 4 +11 = 32 / 4 = 8

Example 1 of Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (preemptive)

P_1	P_2	P_3	P_2	P_4	P_1

0 2 4 5 7 11 16

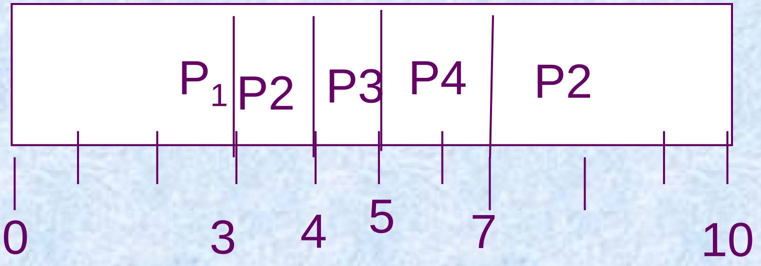
- Average waiting time = (((0-0)+(11-2))+ ((2-2)+(5-4))+(4-4)+(7-5))/4

  = (0 + 9) + (0 + 1) + (0)+ (2)= 12/4 = 3

- Average TAT= ((9+ 7)+ (1+4) + (0+1) + (2+4) = 16+5+1+6 = 28/4 = 7

Example 2 Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	3
P_2	2.0	4
P_3	4.0	1
P_4	5.0	2



- Average waiting time = (0 +(3 – 2)+ (7- 4) + (4- 4)+ (5-5) = 1+3+0+0 = 4 / 4 = 1
- Avg TaT = (0+3) + (4+4) +(0 + 1) +(0+ 2) = 3+8+1+2 =14/4= 3.5

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.

1. $t_n =$ actual lenght of n^{th} CPU burst
2. $\tau_{n+1} =$ predicted value for the next CPU burst
3. α, $0 \leq \alpha \leq 1$
4. Define:

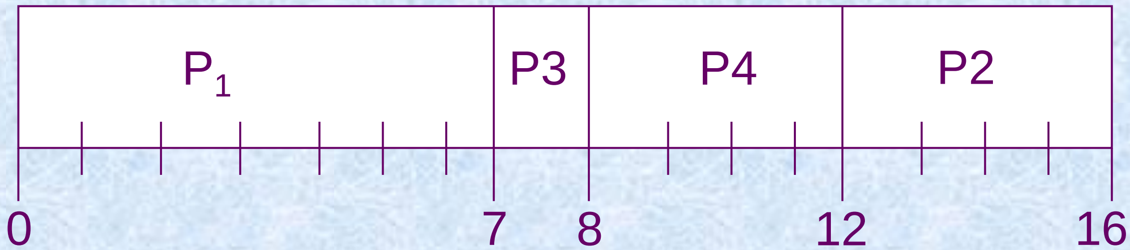
$$\tau_{n=1} = \alpha \, t_n + (1 - \alpha)\tau_n.$$

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 the priority of the process.

Non-Preemptive Priority

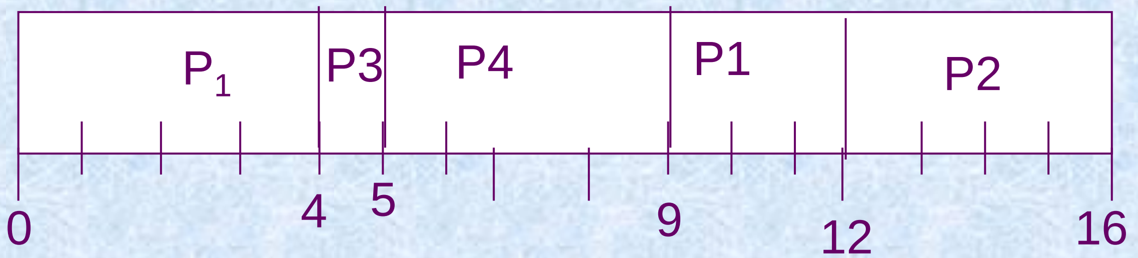
Process	Arrival Time	Burst Time	Priority
P_1	0.0	7	3
P_2	2.0	4	4
P_3	4.0	1	1
P_4	5.0	4	2



- Wt. Time P1 = 0-0 = 0, P2 = 12- 2 =10,
  P3 = 7 – 4 = 3, P4 = 8 – 5 = 3
- Average waiting time = (0 +(12.0 - 2.0)+ (7.0 - 4.0) + (8.0 - 5.0) = 10 + 3 + 3 = 16 / 4 = 4
- Turn Around Time P1= 0 + 7 = 7, P2 = 10 +4 = 14
  P3 = 3 + 1 = 4, P4= 3 +4 = 7
- Avg TaT = (7+14+4+7) /4 = 32/4= 8

Preemptive Priority Example

Process	Arrival Time	Burst Time	Priority
P1	**0.0**	**7**	**3**
P2	**2.0**	**4**	**4**
P3	**4.0**	**1**	**1**
P4	**5.0**	**4**	**2**



- Wt. Time P1 = (0-0) +(9-4) = 5, P2 = 12- 2 =10,

  P3 = 4 – 4 = 0, P4 = 5 – 5 = 0

  Avg. wt. time = 5+10+0+0 = 15/4= 3.75
- TAT Time P1 = 5+7=12, P2 = 10+4=14,

  P3 = 0+1=1, P4 = 0+4=4

  Avg TAT= (12+14+1+4)/4 =31/4= 7.75

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
  - q large $\Rightarrow$ FIFO
  - q small $\Rightarrow$ q must be large with respect to context switch, otherwise overhead is too high.

Example of RR with Time Quantum = 20

Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

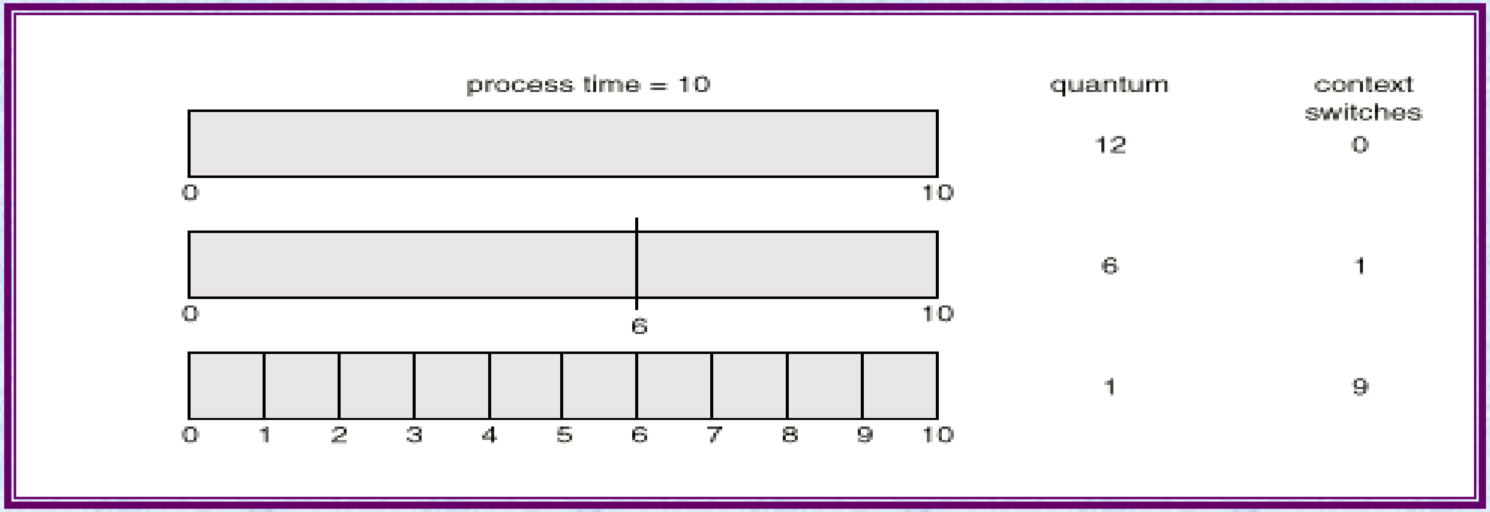
- The Gantt chart is:

P_1	P_2	P_3	P_4	P_1	P_3	P_4	P_1	P_3	P_3

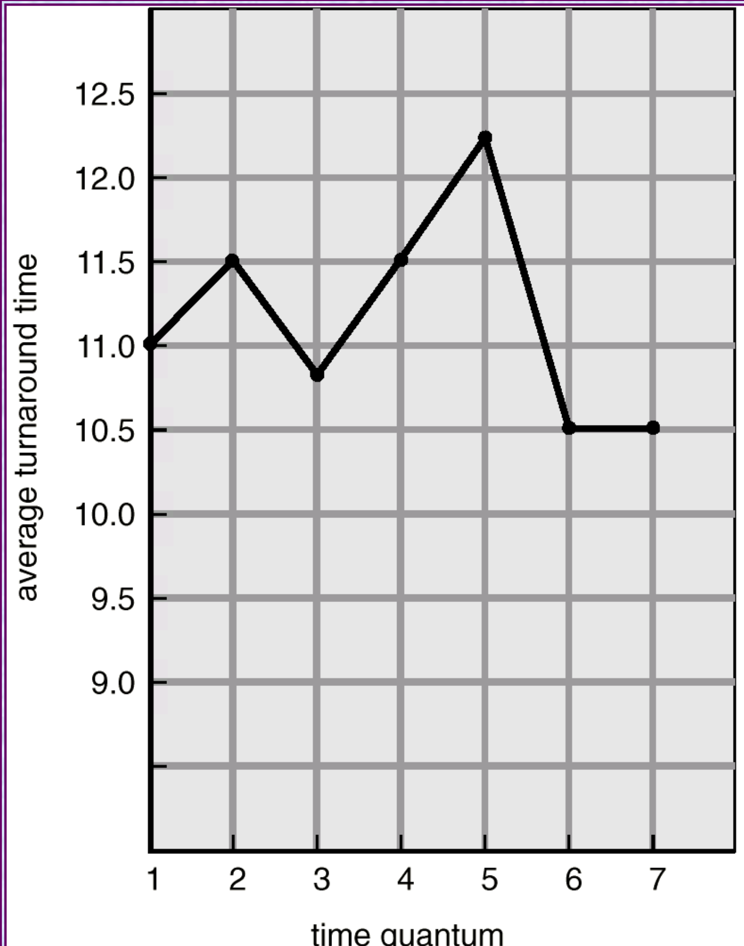
0 20 37 57 77 97 117 121 134 154 162

- Wt. Time P1 = (0-0) +(77-20)+(121-97) = 81, P2 = 20-0 =20, P3 = (37-0) +(97-57)+(134-117) =94, P4 = (57–0)+(117-77) =97
  - Avg. wt. time = 81+20+94+97 = 292/4= 73
- Typically, higher average turnaround than SJF, but better *response*.

Time Quantum and Context Switch Time



Turnaround Time Varies With The Time Quantum

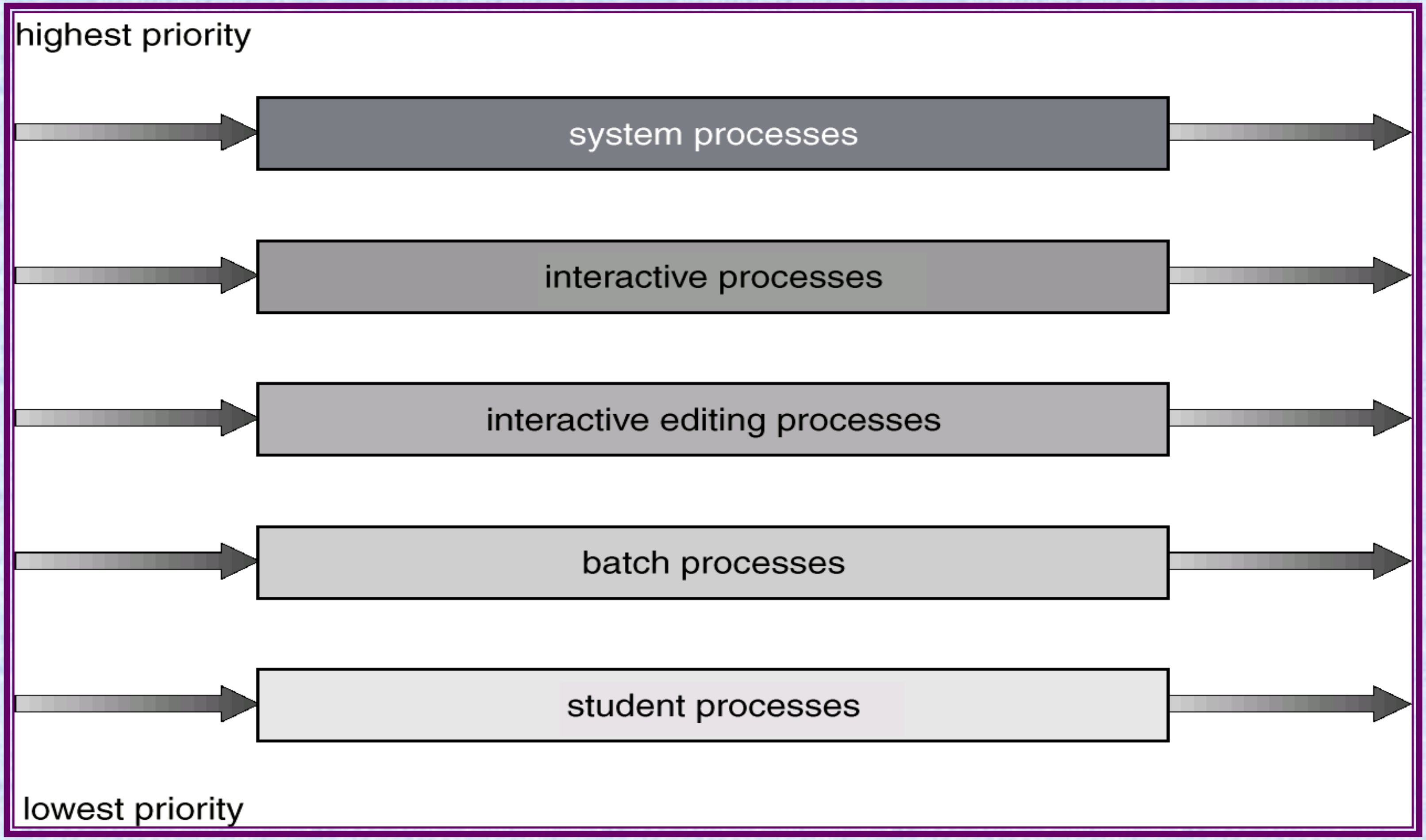


process	time
P_1	6
P_2	3
P_3	1
P_4	7

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



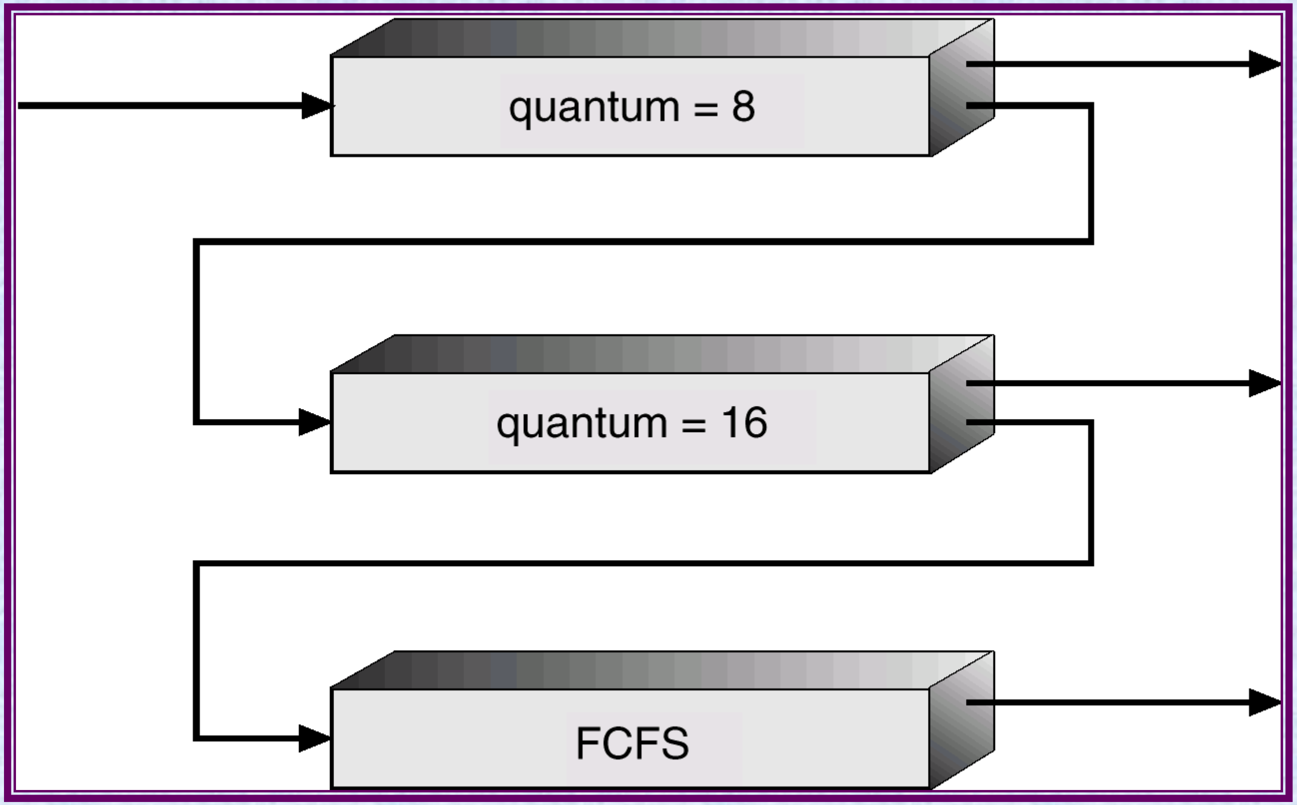
Multilevel Feedback Queue

- 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

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.

Multilevel Feedback Queues



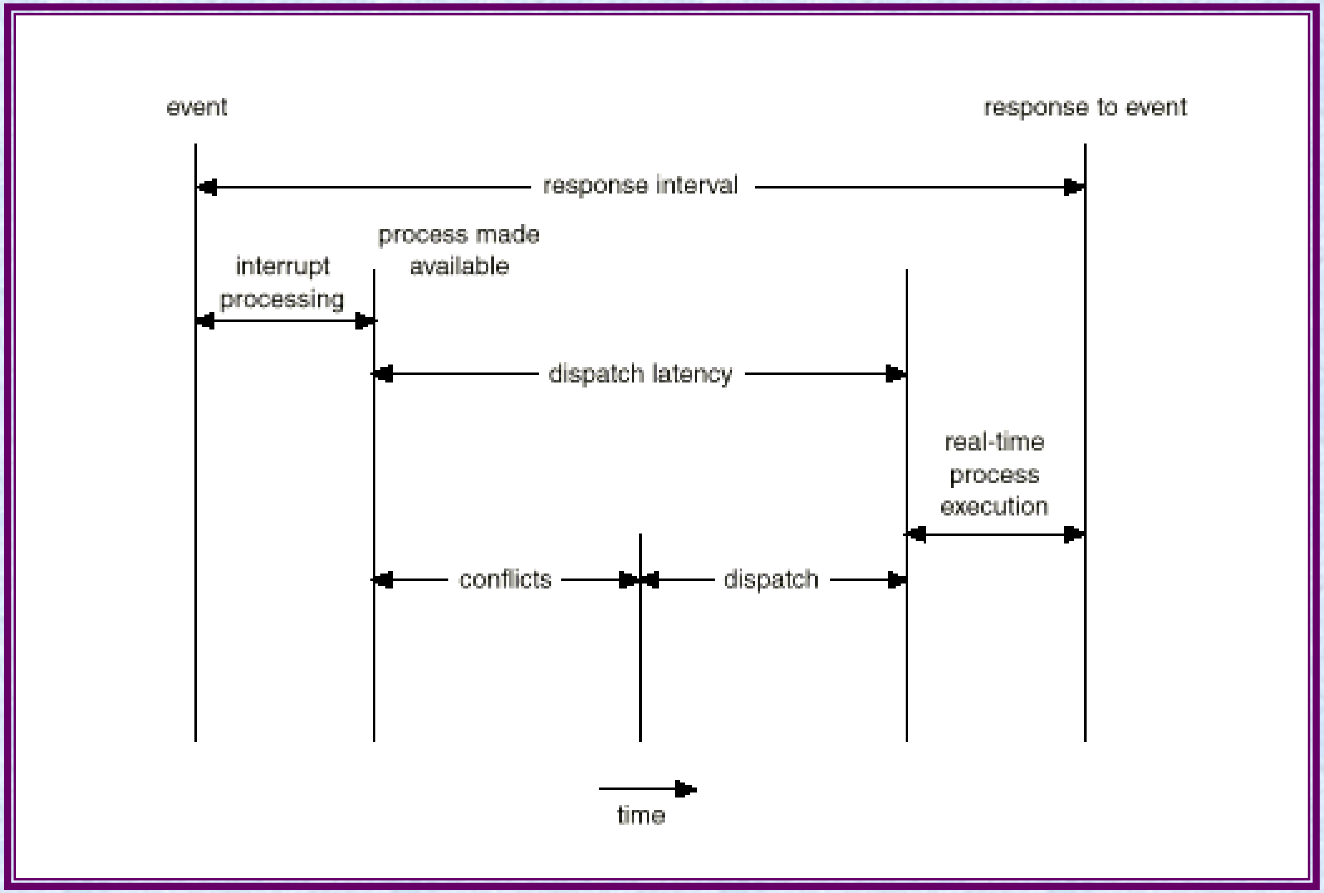
Multiple-Processor Scheduling

- CPU scheduling more complex when multiple CPUs are available.
- *Homogeneous processors* within a multiprocessor.
- *Load sharing*
- *Asymmetric multiprocessing* – only one processor accesses the system data structures, alleviating the need for data sharing.

Real-Time Scheduling

- *Hard real-time* systems – required to complete a critical task within a guaranteed amount of time.
- *Soft real-time* computing – requires that critical processes receive priority over less fortunate ones.

Dispatch Latency



Algorithm Evaluation

- Deterministic modeling – takes a particular predetermined workload and defines the performance of each algorithm for that workload.
- Queueing models
- Simulation
- Implementation

1.Deterministic modeling – takes a particular predetermined workload and defines the performance of each algorithm for that workload

§ .2.Queuing models - no static set of processes; determine the distribution of CPU & I/O bursts, mathematical formula describing the probability of a CPU burst; arrival time distribution.

- Computer system is described as network of servers.
- Each server has a queue of waiting processes.
- The CPU is a server with ready queue; I/O with device queue.
- Knowing the arrival rates and service rates; can compute CPU utilization, average queue length, average wait time, etc called as Queuing – network analysis.

Little’s formula – - $n = \lambda \times w$ where n is average queue length, w is average wt. time in queue and λ is average arrival rate (3/sec).

3. Simulation –

- more accurate;
- Programming a model of the computer system

Software data structure represents the major component.

- 4. Implementation –
  - Most accurate
  - Not always feasible
  - Expensive