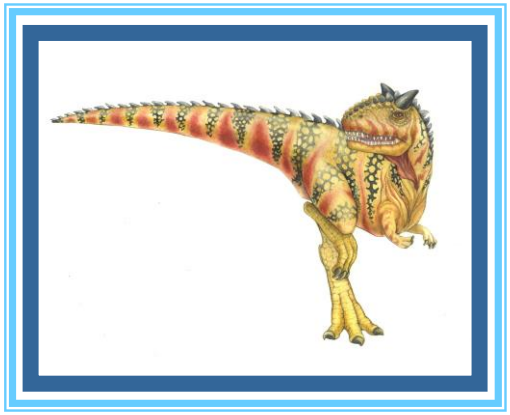
Chapter 6: CPU Scheduling

Conti…

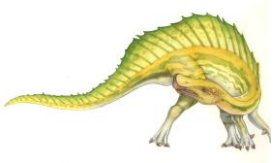


Chapter 6: CPU Scheduling

- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms
- Thread Scheduling
- Multiple-Processor Scheduling
- Real-Time CPU Scheduling
- Operating Systems Examples
- Algorithm Evaluation

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
- SJF is optimal – gives minimum average waiting time for a given set of processes
  - The difficulty is knowing the length of the next CPU request
  - Could ask the user

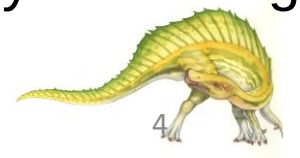


SJF & SRTF Scheduling…

- When the CPU is available it is assigned to the process that has the smallest next CPU burst
- If two processes have the same length next CPU bursts, FCFS scheduling is used to break the tie

Comes in two flavors

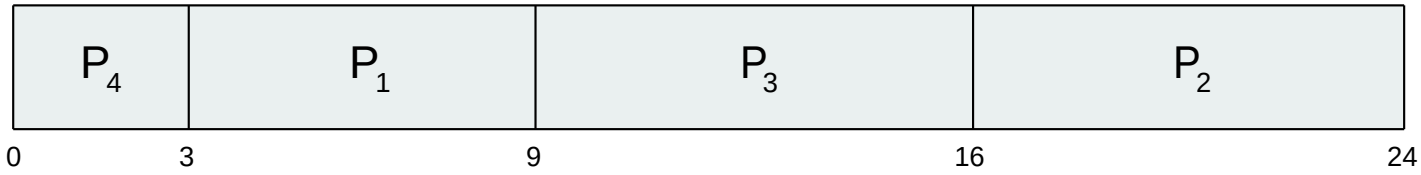
- **Shortest Job First (SJF)**
  - It's a non preemptive algorithm. When a new process arrives having a shorter next CPU burst than what is left of the currently executing process, it ***allows*** the currently running process to finish its CPU burst
- **Shortest Remaining Time First (SRTF)**
  - It's a Preemptive algorithm. When a new process arrives having a shorter next CPU burst than what is left of the currently executing process, it ***preempts*** the currently running process



Example of SJF

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

- SJF scheduling chart

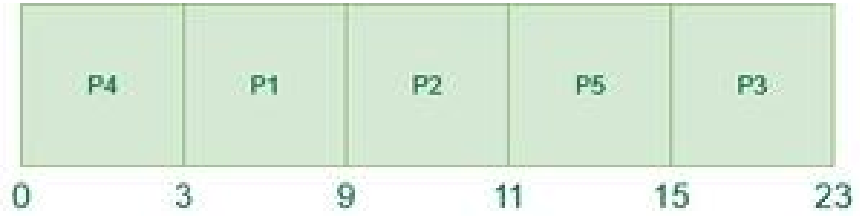


- Average waiting time = (3 + 16 + 9 + 0) / 4 = 7

Example of SJF

Example 1:

Process	Burst Time	Arrival Time
P1	6	2
P2	2	5
P3	8	1
P4	3	0
P5	4	4



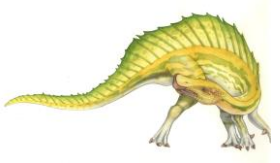
Determining Length of Next CPU Burst

- Can only estimate the length – should be similar to the previous one
  - Then pick process with shortest predicted next CPU burst

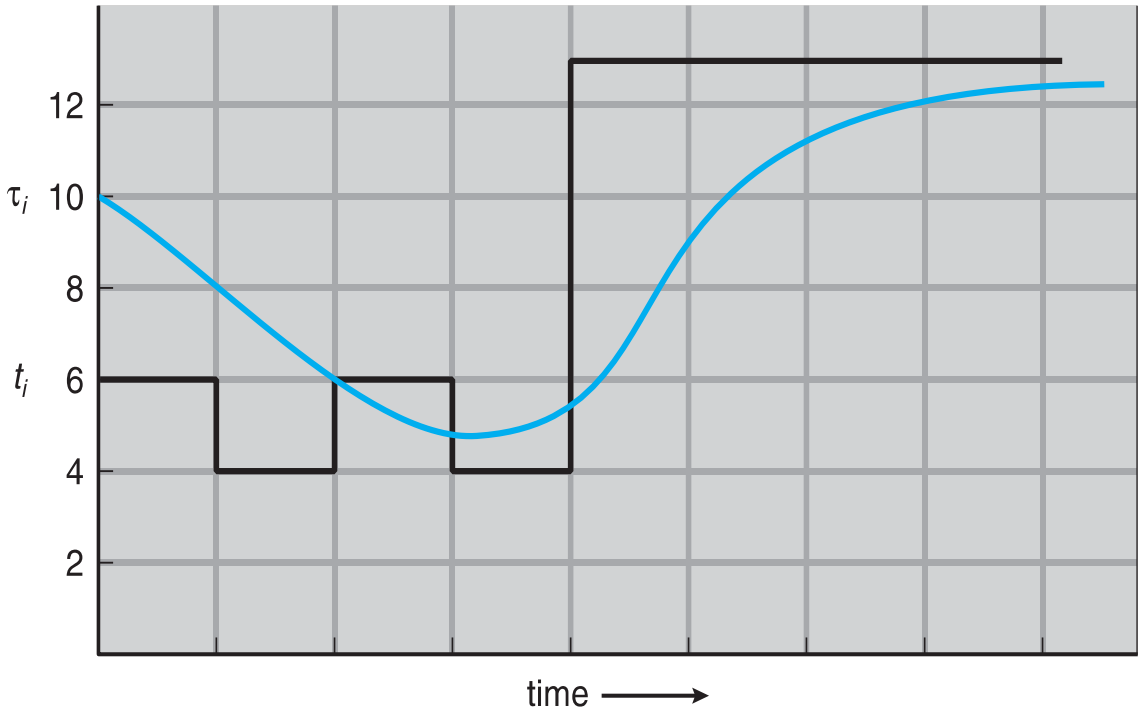
- Can be done by using the length of previous CPU bursts, using exponential averaging

  1. t_n = actual length of n^{th} CPU burst
  2. τ_{n+1} = predicted value for the next CPU burst
  3. $\alpha, 0 \le \alpha \le 1$
  4. Define : $\tau_{n=1} = \alpha\, t_n + (1-\alpha)\tau_n$.

- Commonly, α set to ½
- Preemptive version called **shortest-remaining-time-first**



Prediction of the Length of the Next CPU Burst



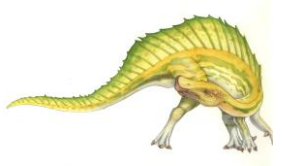
CPU burst (t_i)		6	4	6	4	13	13	13 ...
"guess" (τ_i)	10	8	6	6	5	9	11	12 ...

Examples of Exponential Averaging

- $\alpha = 0$
  - $\tau_{n+1} = \tau_n$
  - Recent history does not count
- $\alpha = 1$
  - $\tau_{n+1} = \alpha\, t_n$
  - Only the actual last CPU burst counts
- If we expand the formula, we get:

$$\tau_{n+1} = \alpha\, t_n + (1 - \alpha)\alpha\, t_{n-1} + \ldots$$
$$+ (1 - \alpha)^j \alpha\, t_{n-j} + \ldots$$
$$+ (1 - \alpha)^{n+1} \tau_0$$

- Since both α and $(1 - \alpha)$ are less than or equal to 1, each successive term has less weight than its predecessor

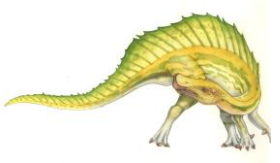


Shortest Remaining Time First (SRTF)

Shortest Remaining Time First (SRTF)

–It's a Preemptive algorithm. When a new process arrives having a shorter next CPU burst than what is left of the currently executing process, it ***preempts*** the currently running process

- For preemptive algo: Waiting time = Finish time – Burst time – Arrival Time

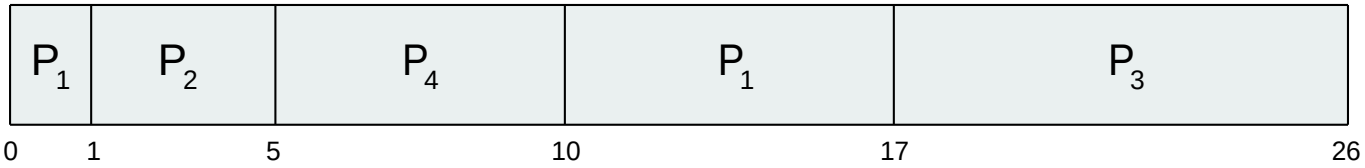


Example of Shortest Remaining Time First (SRTF)

- Now we add the concepts of varying arrival times and preemption to the analysis

Process	*Arrival* Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

- *Preemptive* SJF Gantt Chart



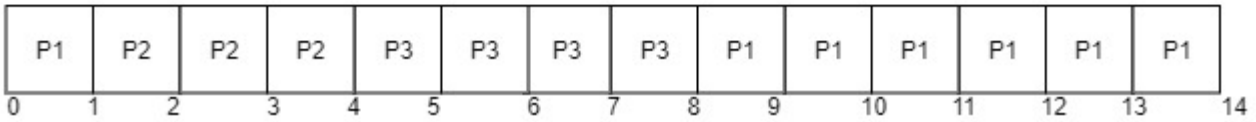
- Average waiting time = [(17-8)+(4-4)+(24-9)+7-5)]/4 = 26/4 = 6.5 msec

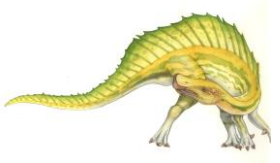
Shortest Remaining Time First

Example-1: Consider the following table of arrival time and burst time for three processes **P1, P2, and P3.**

Process	Burst Time	Arrival Time
P1	7	0
P2	3	1
P3	4	3

The Gantt Chart for SRTF will be:

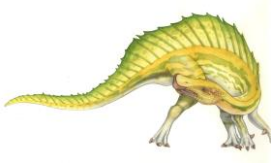




Shortest Remaining Time First

Example-1: Solution

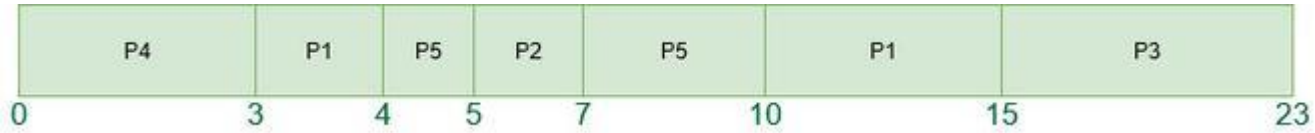
Process	Arrival Time	Burst Time	Completion time	Turn around Time Turn	Waiting Time
P1	0	7	14	14-0=14	14-7=7
P2	1	3	4	4-1=3	3-3=0
P3	3	4	8	8-3=5	5-4=1



Shortest Remaining Time First

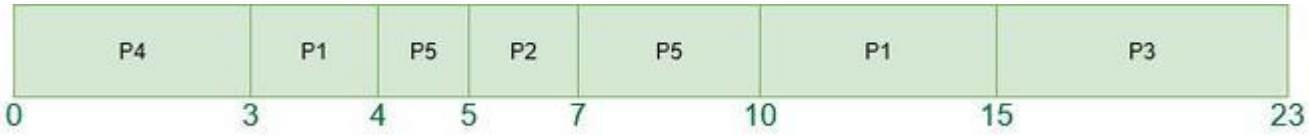
Example-2: Consider the following table of arrival time and burst time for five processes **P1, P2, P3, P4** and **P5**.

Process	Burst Time	Arrival Time
P1	6	2
P2	2	5
P3	8	1
P4	3	0
P5	4	4

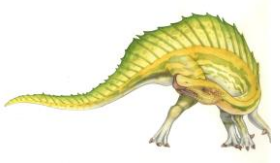


Shortest Remaining Time First

Example-2: Solution



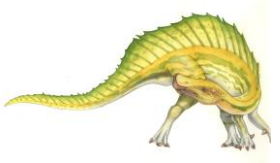
Process	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
P1	2	6	15	15-2 = 13	13-6 = 7
P2	5	2	7	7-5 = 2	2-2 = 0
P3	1	8	23	23-1 = 22	22-8 = 14
P4	0	3	3	3-0 = 3	3-3 = 0
P5	4	4	10	10-4 = 6	6-4 = 2



Shortest Remaining Time First

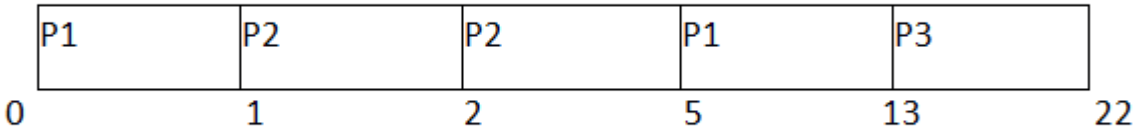
Example-3: Consider the following table of arrival time and burst time for three processes **P1, P2, and P3.**

Process ID	Arrival Time	Burst Time
P1	0	9
P2	1	4
P3	2	9

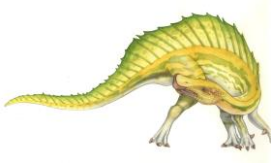


Shortest Remaining Time First

Example-3: Solution

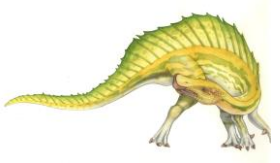


Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
P1	0	9	13	13	4
P2	1	4	5	4	0
P3	2	9	22	20	11



Priority Scheduling

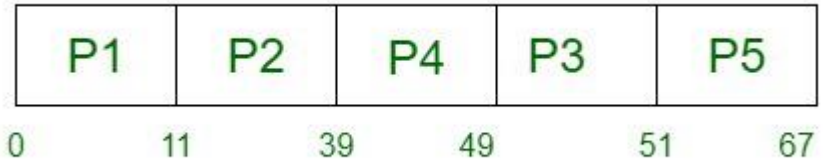
- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority)
  - Preemptive
  - Nonpreemptive
- SJF is priority scheduling where priority is the inverse of predicted next CPU burst time
- Problem ≡ **Starvation** – low priority processes may never execute
- Solution ≡ **Aging** – as time progresses increase the priority of the process



Priority Scheduling

Example-1: Non Pre.emptive

Process	Arrival Time	Burst Time	Priority
P1	0	11	2
P2	5	28	0
P3	12	2	3
P4	2	10	1
P5	9	16	4



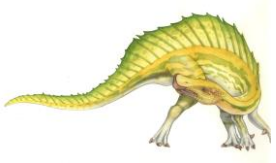
Priority Scheduling

Turn Around Time (T.A.T) = (Completion Time) – (Arrival Time)
Waiting Time (W.T) = (Turn Around Time) – (Burst Time)
Response Time (R.T) = (Process Assign to CPU) – (Arrival Time)

Priority Scheduling

Example-2: Preemptive

Process	Priority	Burst time	Arrival time
P1	1	4	0
P2	2	3	0
P3	1	7	6
P4	3	4	11
P5	2	2	12

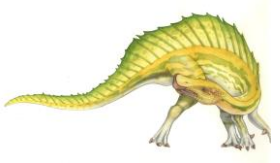


Priority Scheduling

Example-3: Preemptive

Consider the following table of arrival time, Priority, and burst time for five processes **P1, P2, P3, P4,** and **P5**.

Process	Arrival Time	Priority	Burst Time
P1	0	3	3
P2	1	2	4
P3	2	4	6
P4	3	6	4
P5	5	10	2

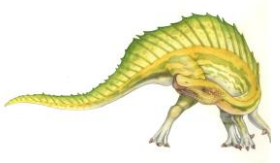


Priority Scheduling

Example-3: Preemptive – Solution

Consider the following table of arrival time, Priority, and burst time for five processes **P1, P2, P3, P4,** and **P5**.

Process	A.T	Priority	B.T	C.T	T.A.T	W.T	R.T
P1	0	3	3	7	7	4	0
P2	1	2(H)	4	5	4	0	0
P3	2	4	6	13	11	5	5
P4	3	6	4	17	14	10	10
P5	5	10(L)	2	19	14	12	12



Priority Scheduling

Example-4: Preemptive

Consider the following table of arrival time, Priority and burst time for seven processes P1, P2, P3, P4, P5, P6 and P7.

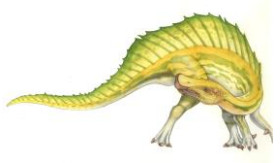
Process	Arrival Time	Priority	Burst Time
P1	0	3	8
P2	1	4	2
P3	3	4	4
P4	4	5	1
P5	5	2	6
P6	6	6	5
P7	10	1	1

Priority Scheduling

Example-4: Preemptive – Solution

Gant Chart

Process	A.T	Priority	B.T	C.T	T.A.T	W.T	R.T
P1	0	3	8	15	15	7	0
P2	1	4	2	17	16	14	14
P3	3	4	4	21	18	14	14
P4	4	5	1	22	18	17	17
P5	5	2	6	12	7	1	0
P6	6	6(L)	5	27	21	16	16
P7	10	1(H)	1	11	1	0	0



Priority Scheduling

Example-5: Preemptive

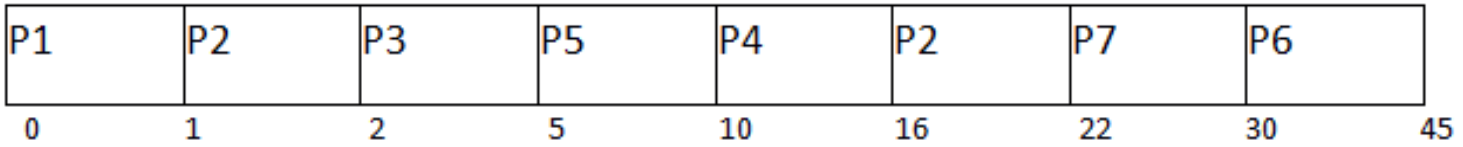
There are 7 processes P1, P2, P3, P4, P5, P6 and P7 given. Their respective priorities, Arrival Times and Burst times are given in the table below…

Process Id	Priority	Arrival Time	Burst Time
1	2	0	1
2	6	1	7
3	3	2	3
4	5	3	6
5	4	4	5
6	10	5	15
7	9	15	8

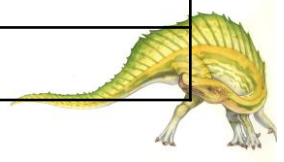
Priority Scheduling

Example-5: Solution

Gant Chart

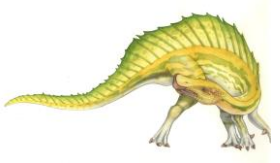


Process Id	Priority	Arrival Time	Burst Time	Completion Time	Turn around Time	Waiting Time
1	2	0	1	1	1	0
2	6	1	7	22	21	14
3	3	2	3	5	3	0
4	5	3	6	16	13	7
5	4	4	5	10	6	1
6	10	5	15	45	40	25
7	9	6	8	30	24	16



Round Robin (RR)

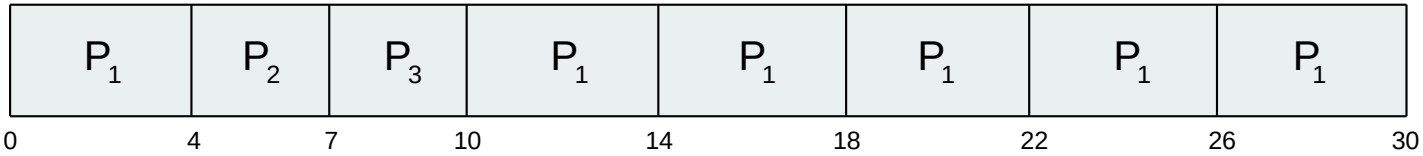
- Each process gets a small unit of CPU time (**time quantum** *q*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.
- If there are *n* processes in the ready queue and the time quantum is *q*, then each process gets 1/*n* of the CPU time in chunks of at most *q* time units at once. No process waits more than (*n*-1)*q* time units.
- Timer interrupts every quantum to schedule next process
- 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 = 4

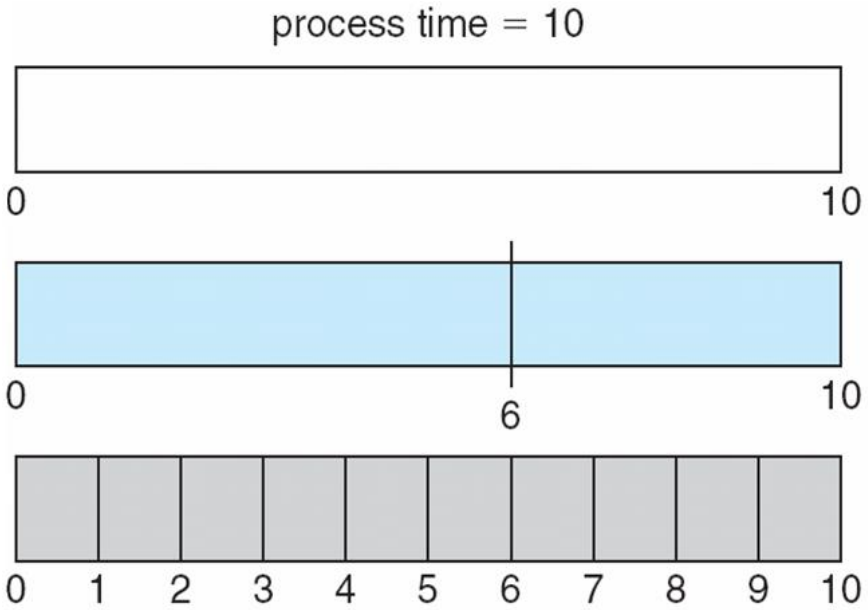
Process	Burst Time
P_1	24
P_2	3
P_3	3

- The Gantt chart is:



- Typically, higher average turnaround than SJF, but better ***response***
- q should be large compared to context switch time
- q usually 10ms to 100ms, context switch < 10 usec

Time Quantum and Context Switch Time



quantum	context switches
12	0
6	1
1	9

Round Robin Examples

Example 1: Consider the following 6 processes: **P1, P2, P3, P4, P5, and P6** with their arrival time and burst time as given below:

Question. What are the average waiting and turnaround times for the round-robin scheduling algorithm (RR) with a time quantum of **4 units**?

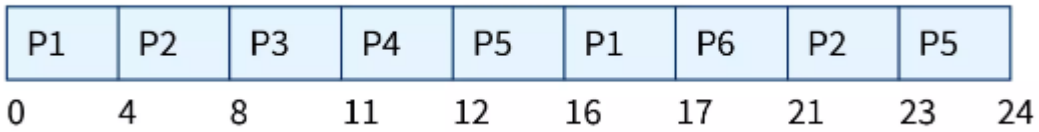
Process ID	Arrival Time	Burst Time
P1	0	5
P2	1	6
P3	2	3
P4	3	1
P5	4	5
P6	6	4

Round Robin Examples

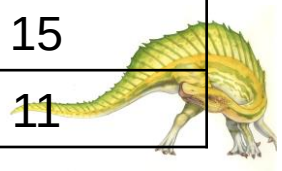
Example 1: Solution

Round-robin scheduling algorithm (RR) with a time quantum of **4 units**.

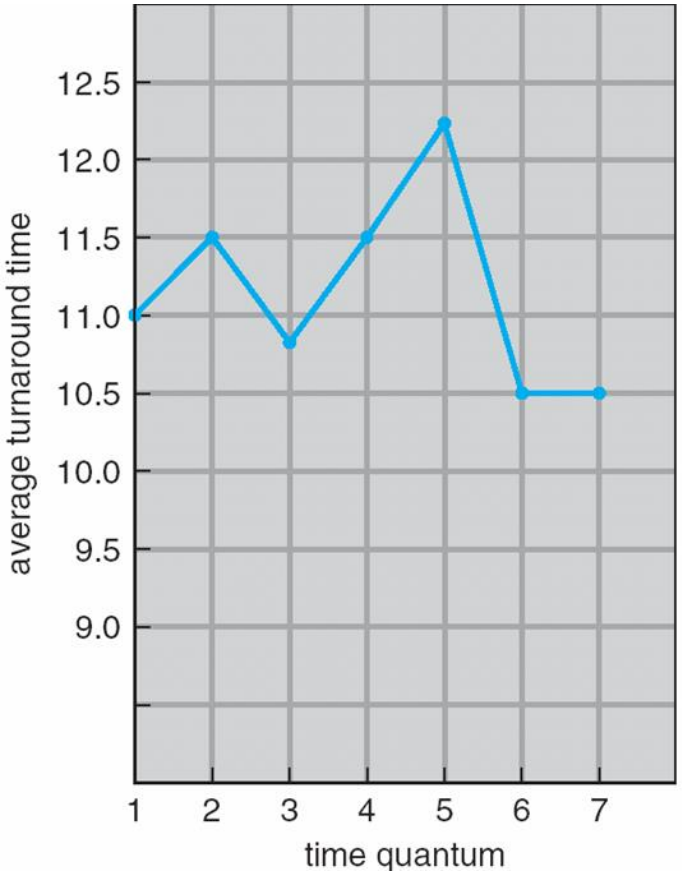
Ready Queue:	P1	P2	P3	P4	P5	P1	P6	P2	P5



Processes	Arrival Time (AT)	Burst Time (BT)	Turn Around Time (TAT)	Waiting Time (WT)
P1	0	5	17	12
P2	1	6	22	16
P3	2	3	9	6
P4	3	1	9	8
P5	4	5	20	15
P6	6	4	15	11



Turnaround Time Varies With The Time Quantum



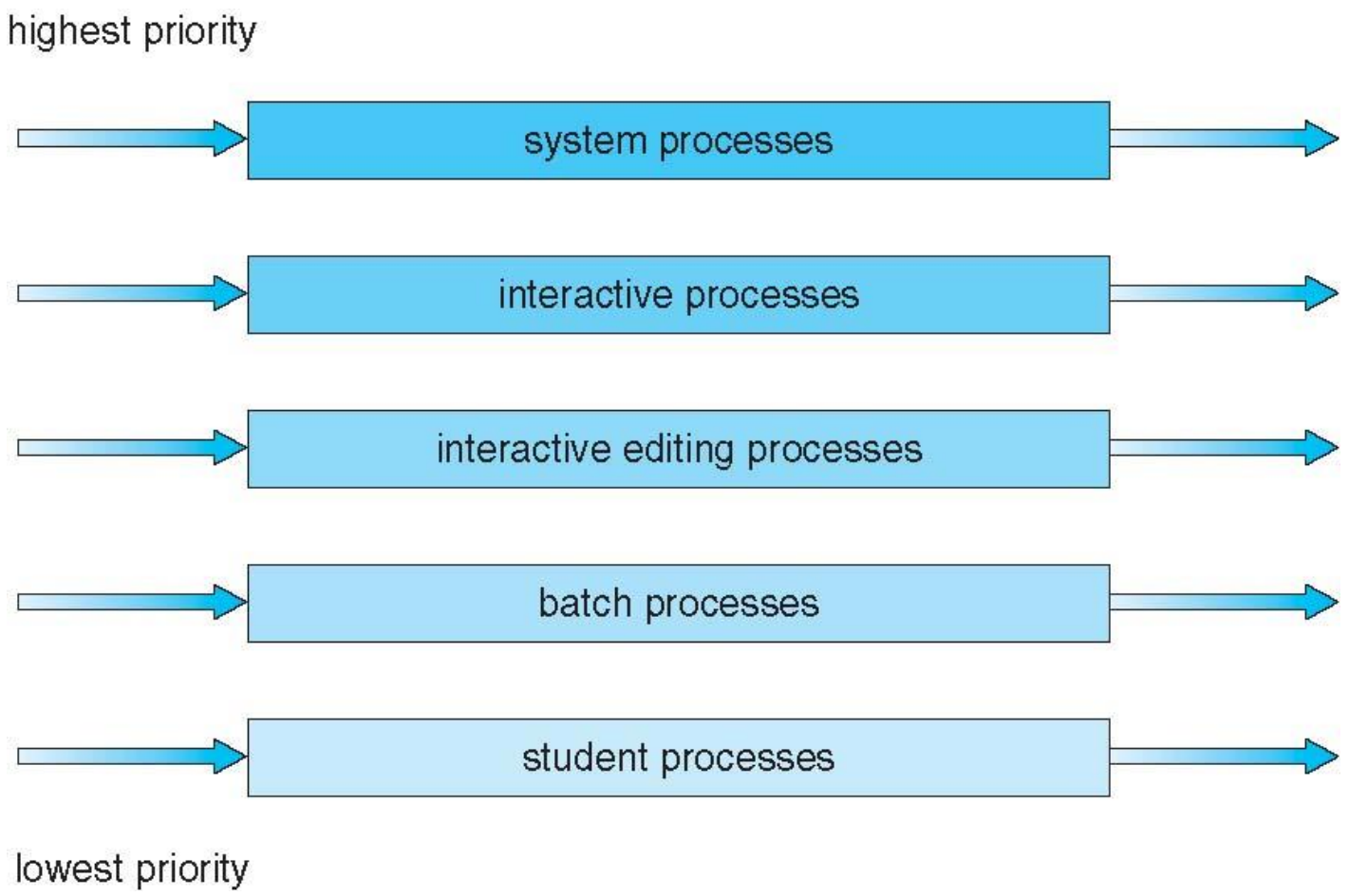
process	time
P_1	6
P_2	3
P_3	1
P_4	7

80% of CPU bursts should be shorter than q

Multilevel Queue

- Ready queue is partitioned into separate queues, eg:
  - **foreground** (interactive)
  - **background** (batch)
- Process permanently in a given queue
- 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

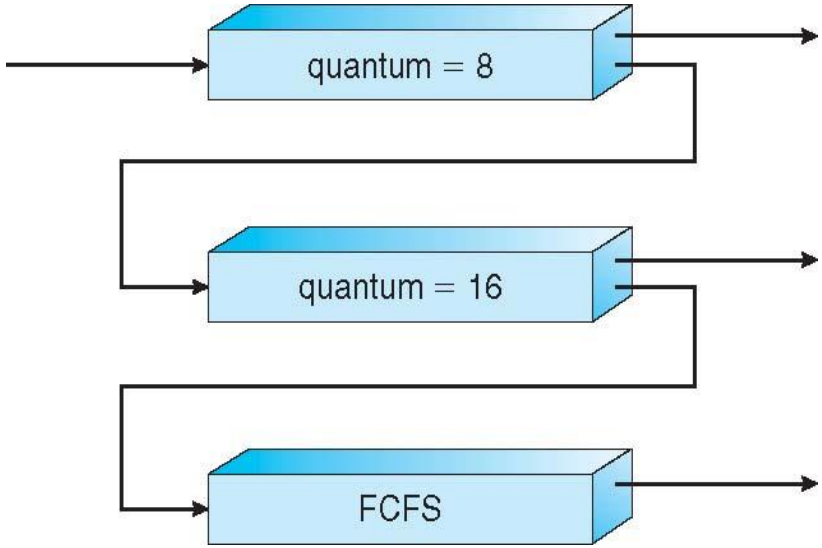


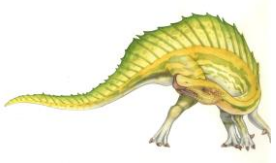
Multilevel Feedback Queue

- 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 – RR with time quantum 8 milliseconds
  - Q_1 – RR 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





End of Chapter 6