Operating Systems

Lab 07

Content

CPU Scheduling
Basic Concepts
Scheduling Algorithms

Basic Concepts

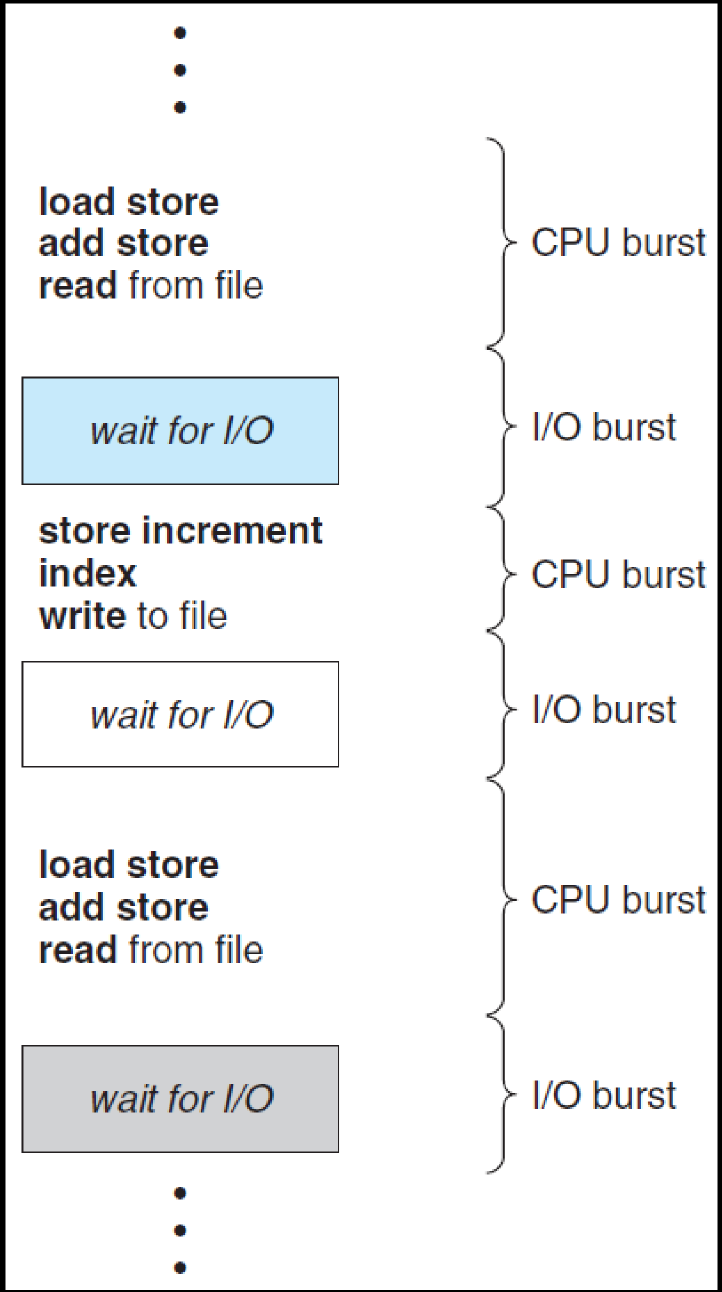
- CPU scheduling is the basis of multiprogrammed operating systems.
    - The operating system can make the computer more productive.

- The objective of multiprogramming is to have some process **running at all times**, to maximize CPU utilization.

- A process is executed until it must wait, typically for the completion of some I/O request.
    - When one process has to wait, the operating system takes the CPU away from that process and gives the CPU to another process.

Basic Concepts

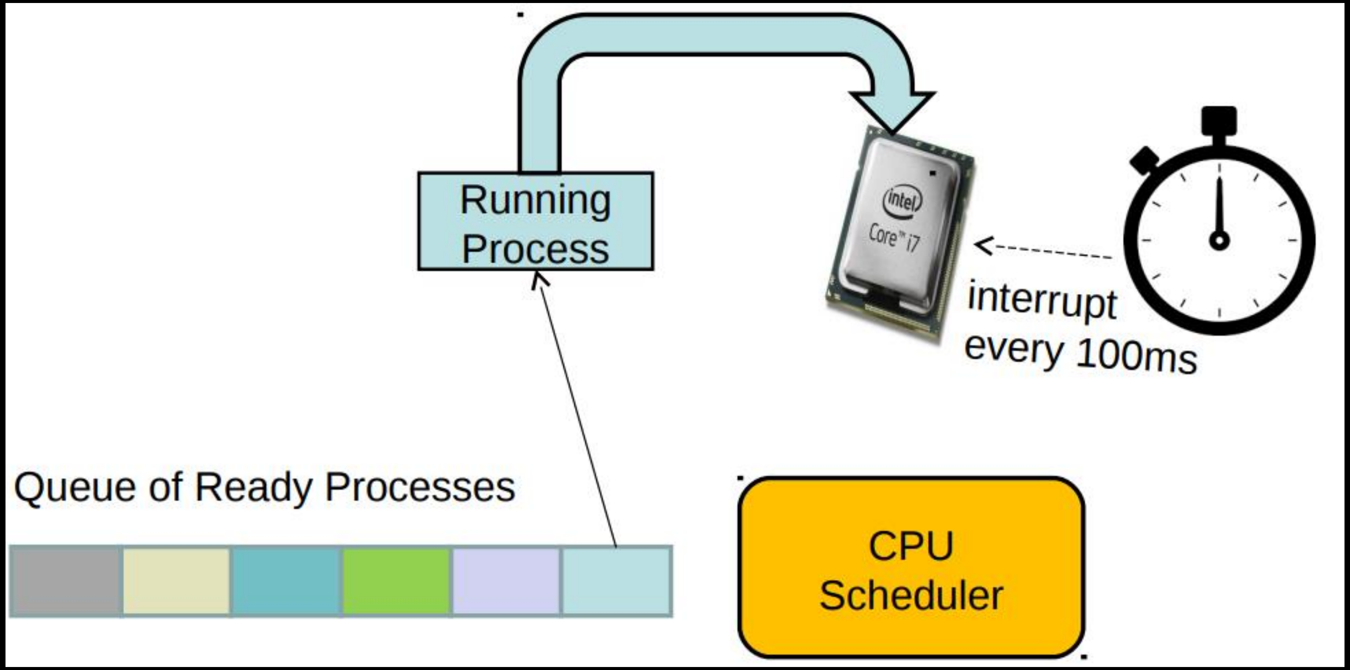
- Process execution consists of a **cycle** of CPU execution and I/O wait.
- A CPU burst of performing calculations.
- An I/O burst, waiting for data transfer in or out of the system.



Basic Concepts

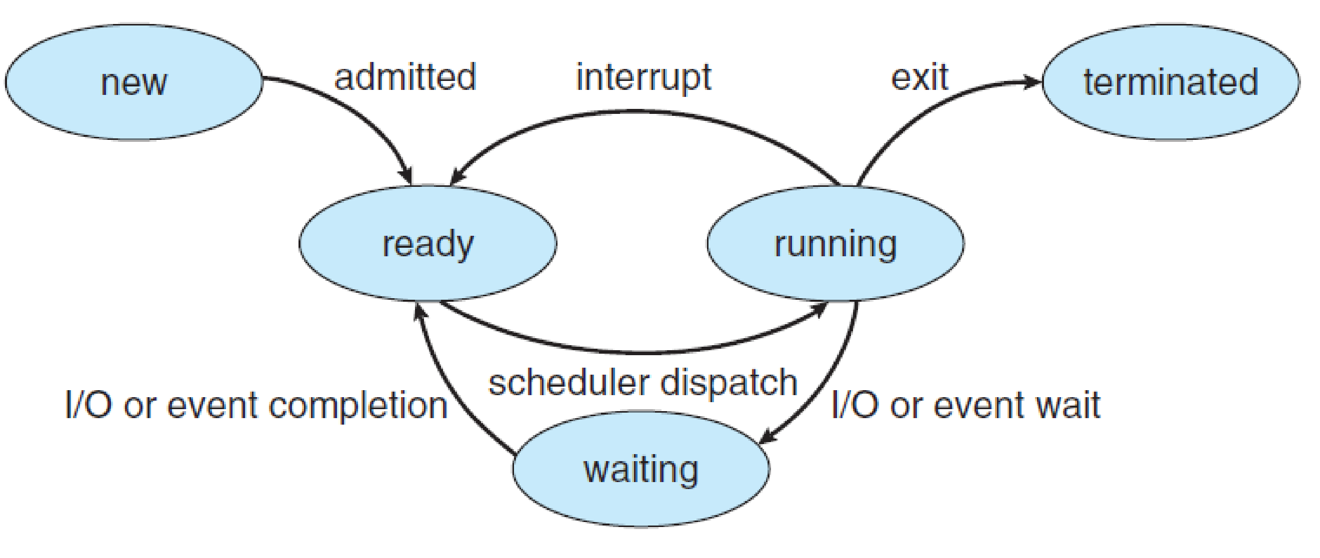
CPU Scheduler

- Whenever the CPU becomes idle, it is the job of the CPU Scheduler to select another process from the ready queue to run next.



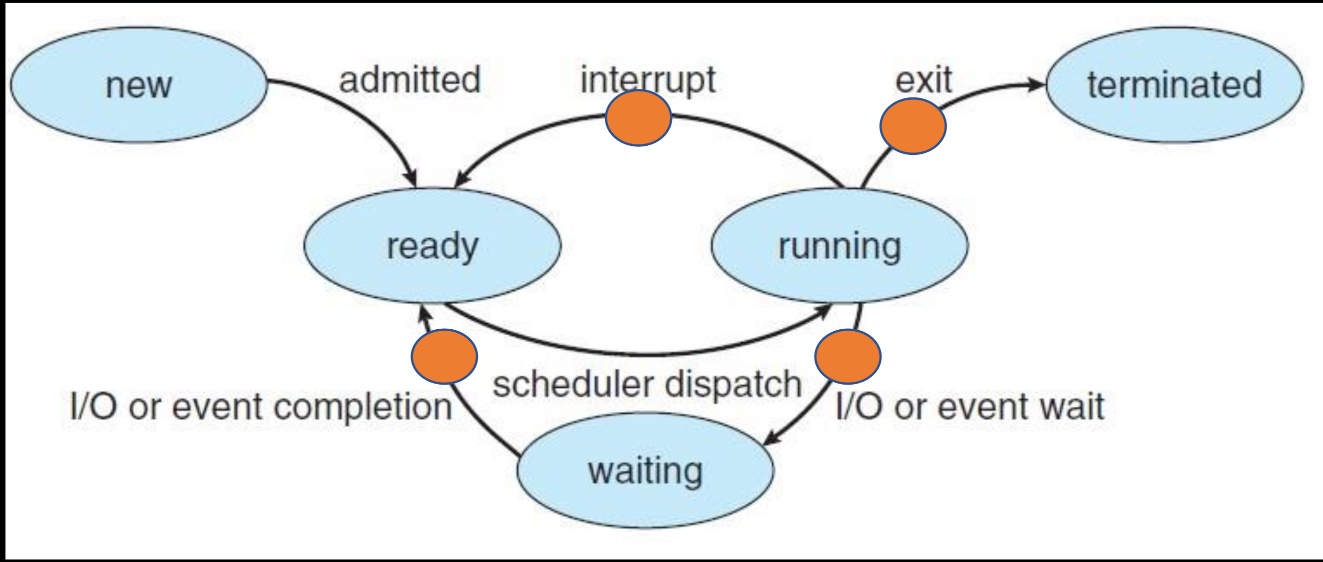
Basic Concepts

- Process state
  - o **New**: The process is being created.
  - o **Running**: Instructions are being executed.
  - o **Waiting**: The process is waiting for some event to occur (such as an I/O completion).
  - o **Ready**: The process is waiting to be assigned to a processor.
  - o **Terminated**: The process has finished execution.



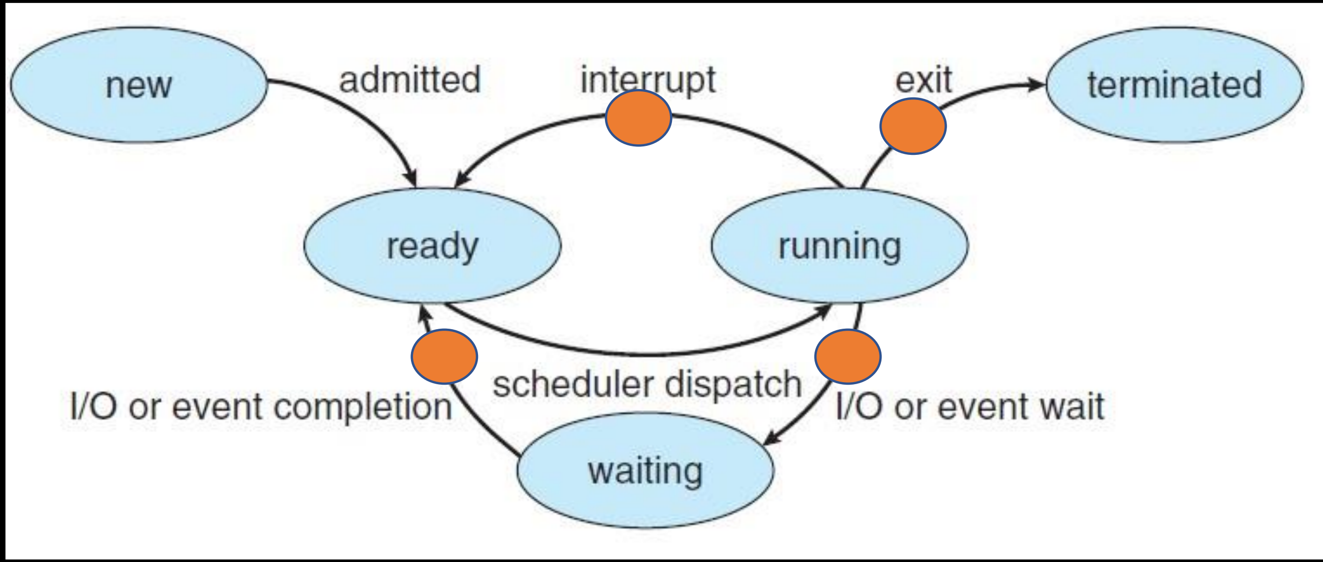
Basic Concepts

- CPU scheduling decisions take place under one of four conditions:
  - When a process switches from the running state to the waiting state.
  - When a process switches from the running state to the ready state.
  - When a process switches from the waiting state to the ready state.
  - When a process terminates.



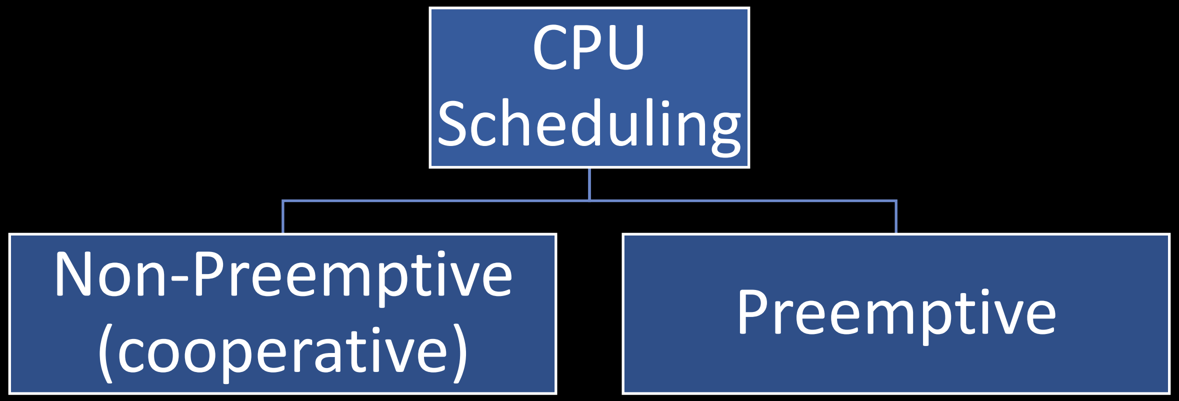
Basic Concepts

- CPU scheduling decisions take place under one of four conditions:
  - When a process switches from the running state to the waiting state.
  - When a process switches from the running state to the ready state.
  - When a process switches from the waiting state to the ready state.
  - When a process terminates.



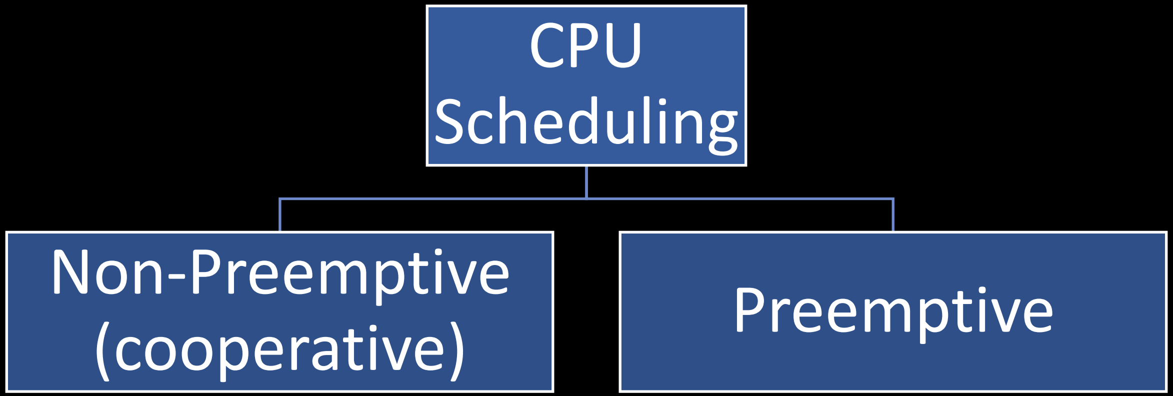
Basic Concepts

- Scheduling types



Basic Concepts

- Scheduling types



- The processor has no choice but selecting a process to execute.
- Occurs when:
  - When a process switches from the running state to the waiting state.
  - When a process terminates

Basic Concepts

- Scheduling types

CPU Scheduling

Non-Preemptive (cooperative)

Preemptive

Non-Preemptive (cooperative)

- The processor has no choice but selecting a process to execute.
- Occurs when:
  - When a process switches from the running state to the waiting state.
  - When a process terminates.

Preemptive

- The processor chooses either continue running the current process, or select a different one.
- Occurs when:
  - When a process switches from the running state to the ready state.
  - When a process switches from the waiting state to the ready state.

Basic Concepts

- Scheduling Criteria
  - **CPU utilization** - We want to keep the CPU as busy as possible (100% of the time).

Basic Concepts

- Scheduling Criteria
  - **CPU utilization** - We want to keep the CPU as busy as possible (100% of the time).
  - **Throughput** - Number of processes completed per unit time.

Basic Concepts

- Scheduling Criteria
  - **CPU utilization** - We want to keep the CPU as busy as possible (100% of the time).
  - **Throughput** - Number of processes completed per unit time.
  - **Turnaround time** - Time required for a particular process to complete, from submission time to completion. It is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.

Basic Concepts

- Scheduling Criteria
  - **CPU utilization** - We want to keep the CPU as busy as possible (100% of the time).
  - **Throughput** - Number of processes completed per unit time.
  - **Turnaround time** - Time required for a particular process to complete, from submission time to completion. It is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.
  - **Waiting time** - How much time processes spend in the ready queue waiting their turn to get on the CPU.

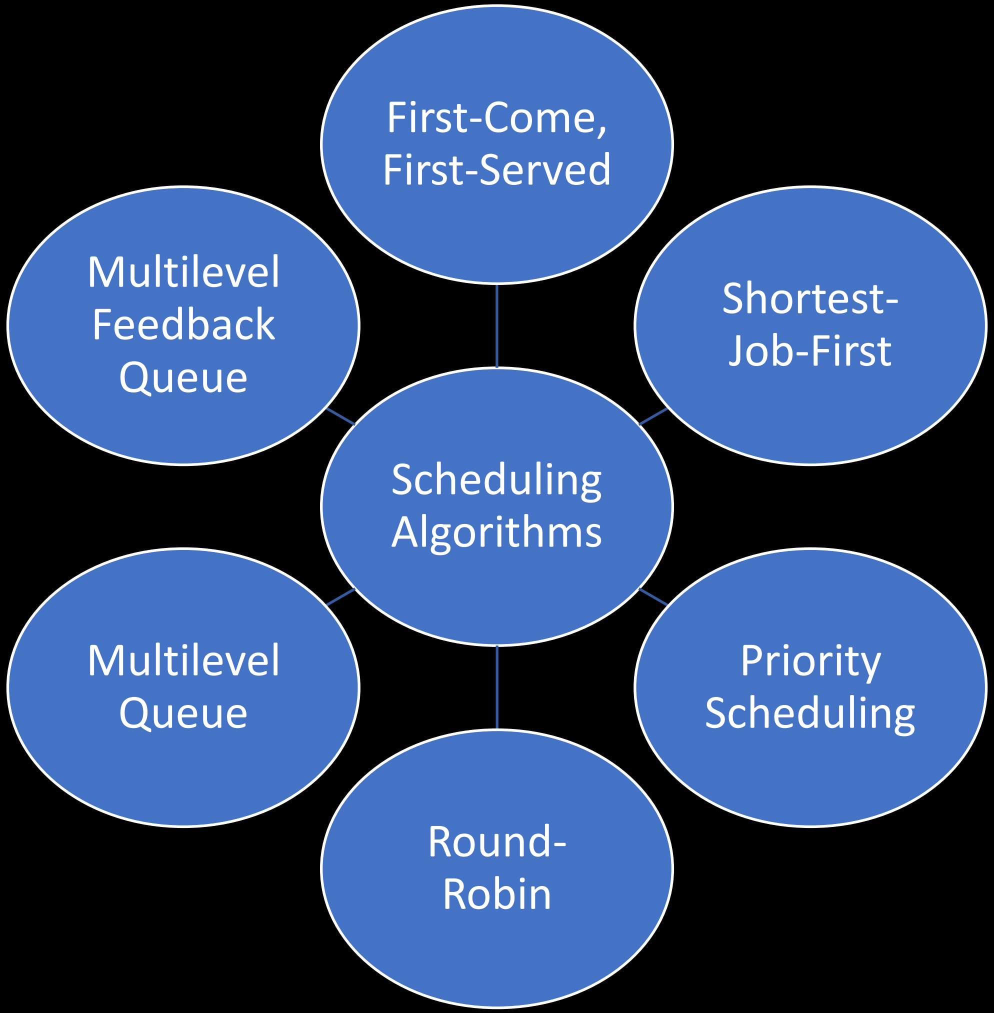
Basic Concepts

- Scheduling Criteria
  - **CPU utilization** - We want to keep the CPU as busy as possible (100% of the time).
  - **Throughput** - Number of processes completed per unit time.
  - **Turnaround time** - Time required for a particular process to complete, from submission time to completion. It is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.
  - **Waiting time** - How much time processes spend in the ready queue waiting their turn to get on the CPU.
  - **Response time** - The time from the submission of a request until the first response is produced.

Content

CPU Scheduling
Basic Concepts
Scheduling Algorithms

Scheduling Algorithms



First-Come, First-Served Scheduling, FCFS

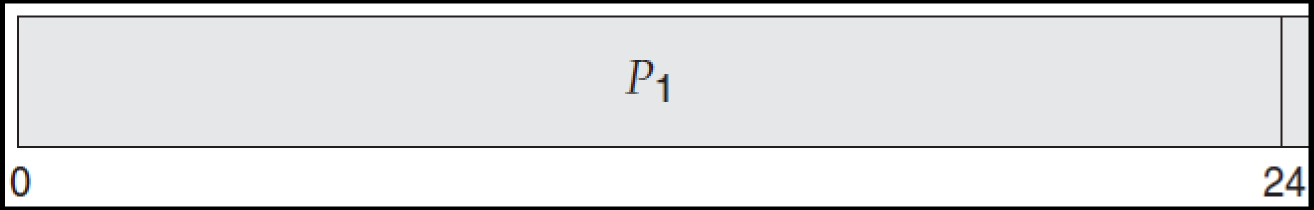
- The process that requests the CPU first is allocated the CPU first.
  - The implementation is based on a FIFO queue.
  - Like customers waiting in line at the bank.
- Example:

Process	Burst Time
P1	24
P2	3
P3	3

First-Come, First-Served Scheduling, FCFS

- The process that requests the CPU first is allocated the CPU first.
  - The implementation is based on a FIFO queue.
  - Like customers waiting in line at the bank.
- Example:

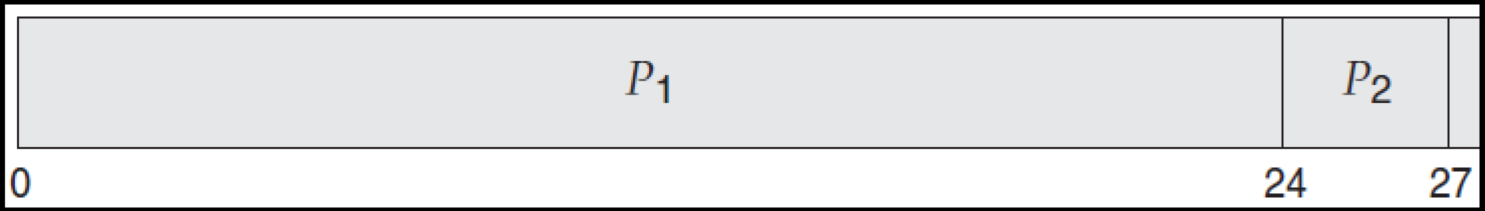
Process	Burst Time
P1	24
P2	3
P3	3



First-Come, First-Served Scheduling, FCFS

- The process that requests the CPU first is allocated the CPU first.
  - The implementation is based on a FIFO queue.
  - Like customers waiting in line at the bank.
- Example:

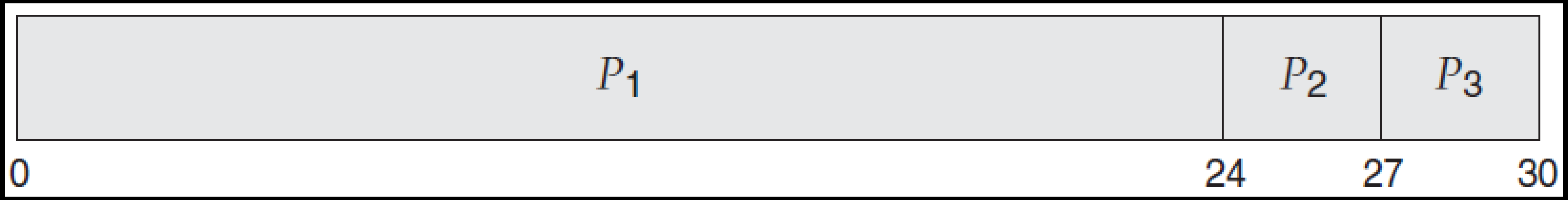
Process	Burst Time
P1	24
P2	3
P3	3



First-Come, First-Served Scheduling, FCFS

- The process that requests the CPU first is allocated the CPU first.
  - The implementation is based on a FIFO queue.
  - Like customers waiting in line at the bank.
- Example:

Process	Burst Time
P1	24
P2	3
P3	3

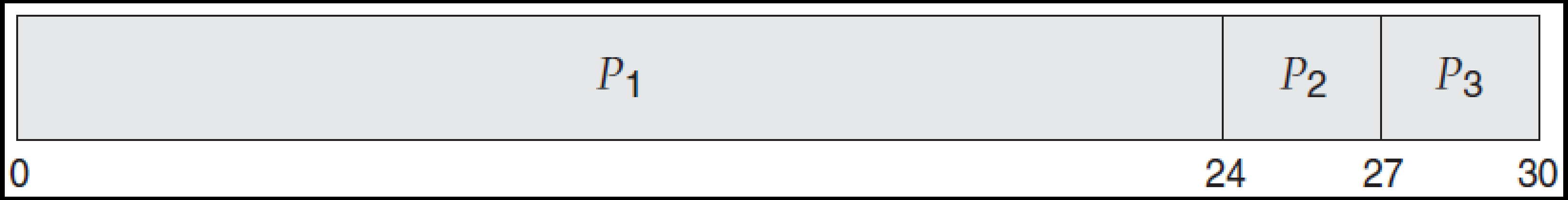


First-Come, First-Served Scheduling, FCFS

- The process that requests the CPU first is allocated the CPU first.
  - The implementation is based on a FIFO queue.
  - Like customers waiting in line at the bank.
- Example:

Process	Burst Time	Waiting Time
P1	24	0
P2	3	24
P3	3	27

Average waiting time = $\frac{0+24+27}{3} = 17\ ms$



First-Come, First-Served Scheduling, FCFS

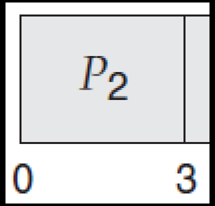
- If the processes arrive in order: P_2, P_3, P_1, it becomes

Process	Burst Time
P2	3
P3	3
P1	24

First-Come, First-Served Scheduling, FCFS

- If the processes arrive in order: P_2, P_3, P_1, it becomes

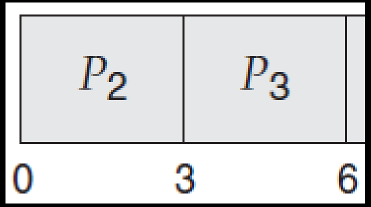
Process	Burst Time
P2	3
P3	3
P1	24



First-Come, First-Served Scheduling, FCFS

- If the processes arrive in order: P_2, P_3, P_1, it becomes

Process	Burst Time
P2	3
P3	3
P1	24



First-Come, First-Served Scheduling, FCFS

- If the processes arrive in order: P_2, P_1, P_3, it becomes

Process	Burst Time
P2	3
P3	3
P1	24

First-Come, First-Served Scheduling, FCFS

- If the processes arrive in order: P_2, P_3, P_1, it becomes

Process	Burst Time	Waiting Time
P2	3	0
P3	3	3
P1	24	6

Average waiting time = $\frac{0+3+6}{3} = 3\ ms$

Shortest-Job-First Scheduling, SJF

- This algorithm associates with each process the length of the process's next CPU burst.
- When the CPU is available, it is assigned to the process that has the smallest next CPU burst.
- If the next CPU bursts of two processes are the same, FCFS scheduling is used to break the tie.

shortest-next- CPU-burst algorithm

Shortest-Job-First Scheduling, SJF

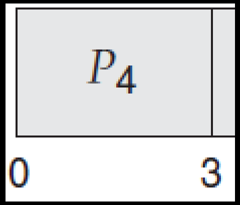
- Example: *suppose that the processes came at the same time.*

Process	Burst Time
P1	6
P2	8
P3	7
P4	3

Shortest-Job-First Scheduling, SJF

- Example: *suppose that the processes came at the same time.*

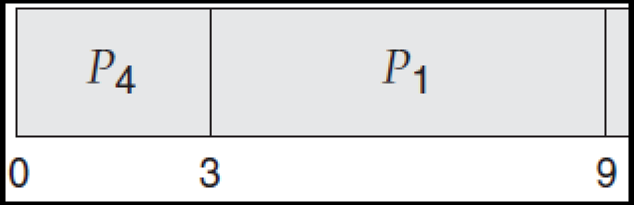
Process	Burst Time
P1	6
P2	8
P3	7
P4	3



Shortest-Job-First Scheduling, SJF

- Example: *suppose that the processes came at the same time.*

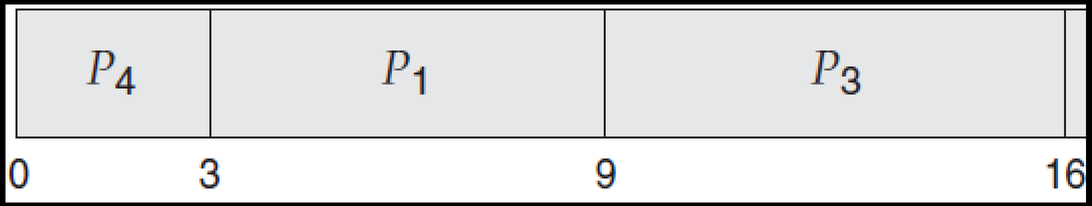
Process	Burst Time
P1	6
P2	8
P3	7
P4	3



Shortest-Job-First Scheduling, SJF

- Example: *suppose that the processes came at the same time.*

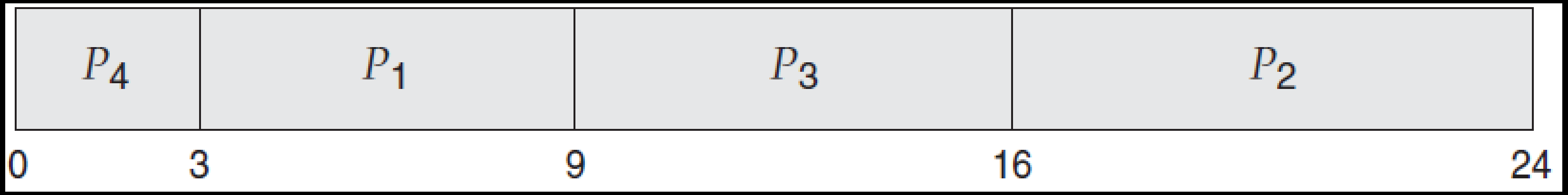
Process	Burst Time
P1	6
P2	8
P3	7
P4	3



Shortest-Job-First Scheduling, SJF

- Example: *suppose that the processes came at the same time.*

Process	Burst Time
P1	6
P2	8
P3	7
P4	3

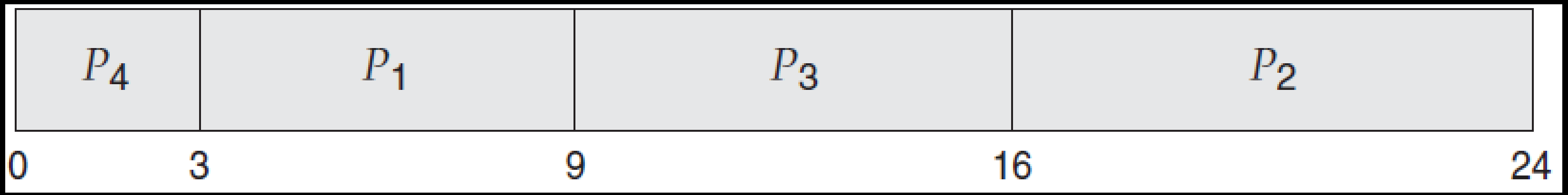


Shortest-Job-First Scheduling, SJF

- Example: *suppose that the processes came at the same time.*

Process	Burst Time	Waiting Time
P1	6	3
P2	8	16
P3	7	9
P4	3	0

Average waiting time = $\frac{3+16+9+0}{4} = 7\ ms$

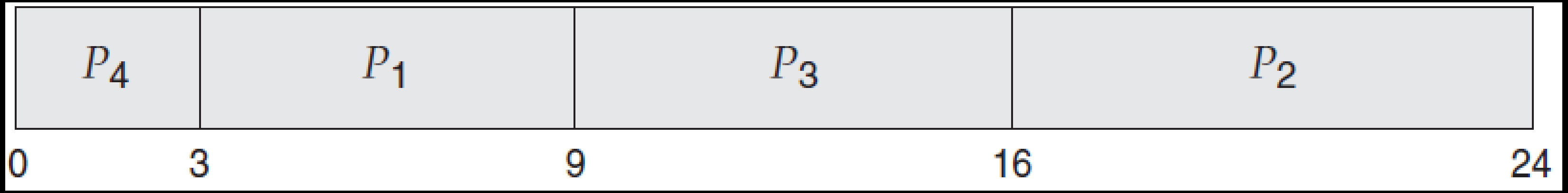


Shortest-Job-First Scheduling, SJF

- Example: *suppose that the processes came at the same time.*

Process	Burst Time	Waiting Time
P1	6	3
P2	8	16
P3	7	9
P4	3	0

Average waiting time = $\frac{3+16+9+0}{4} = 7\ ms$



This is non-preemptive scheduling,
since the process never leaves the CPU until it finishes

Shortest-Job-First Scheduling, SJF

- SJF can be preemptive.
- When a new process arrives in the ready queue that has a burst time shorter than the process is currently on the CPU, the CPU switches the currently executing process with the shorter one.
  - Preemptive SJF is referred to as ***shortest remaining time first scheduling***.

Shortest-Job-First Scheduling, SJF

At time 0 which the shortest remaining process?

- Example

Process	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	9
p4	3	5

Shortest-Job-First Scheduling, SJF

At time 0 which the shortest remaining process?

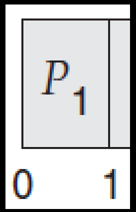
- Example

Process	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	9
p4	3	5

Shortest-Job-First Scheduling, SJF

- Example

Process	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	9
p4	3	5

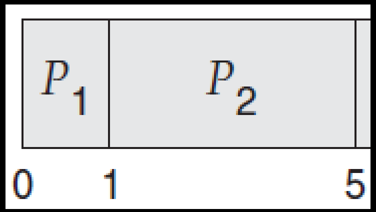


Shortest-Job-First Scheduling, SJF

- Example

At time 1, P2 come with shorter burst time, so it is switched with P1.

Process	Arrival Time	Burst Time	Remaining Time
P1	0	8	7
P2	1	4	
P3	2	9	
p4	3	5	

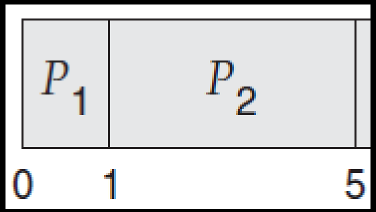


Shortest-Job-First Scheduling, SJF

P2 is finished. We are at time 5.
what is the shortest remaining process?

- Example

Process	Arrival Time	Burst Time	Remaining Time
P1	0	8	7
P2	1	4	0
P3	2	9	9
p4	3	5	5

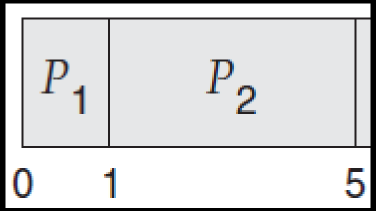


Shortest-Job-First Scheduling, SJF

- Example

P2 is finished. We are at time 5.
what is the shortest remaining process?

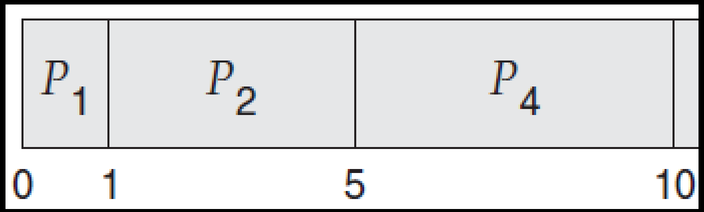
Process	Arrival Time	Burst Time	Remaining Time
P1	0	8	7
P2	1	4	0
P3	2	9	9
p4	3	5	5



Shortest-Job-First Scheduling, SJF

- Example

Process	Arrival Time	Burst Time	Remaining Time
P1	0	8	7
P2	1	4	0
P3	2	9	9
p4	3	5	5

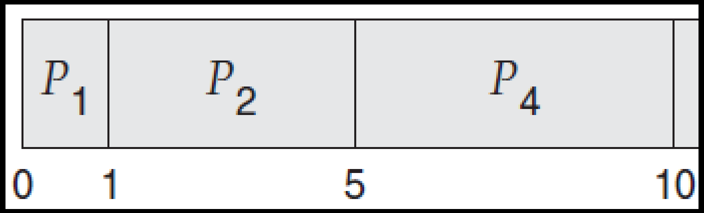


Shortest-Job-First Scheduling, SJF

- Example

P4 is finished. We are at time 10.
what is the shortest remaining process?

Process	Arrival Time	Burst Time	Remaining Time
P1	0	8	7
P2	1	4	0
P3	2	9	9
p4	3	5	0

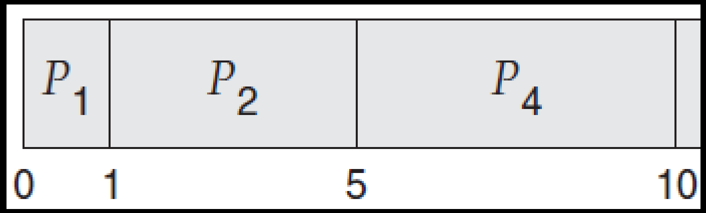


Shortest-Job-First Scheduling, SJF

- Example

P4 is finished. We are at time 10.
what is the shortest remaining process?

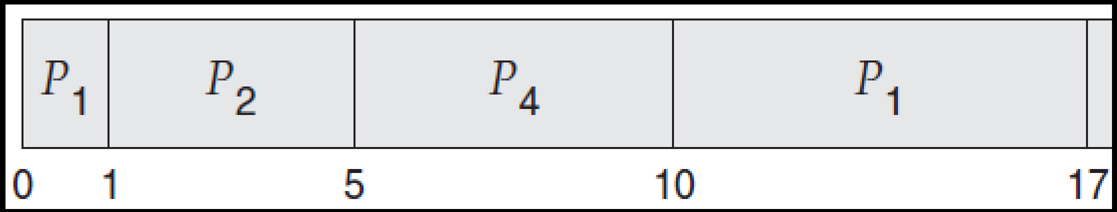
Process	Arrival Time	Burst Time	Remaining Time
P1	0	8	7
P2	1	4	0
P3	2	9	9
p4	3	5	0



Shortest-Job-First Scheduling, SJF

- Example

Process	Arrival Time	Burst Time	Remaining Time
P1	0	8	7
P2	1	4	0
P3	2	9	9
p4	3	5	0

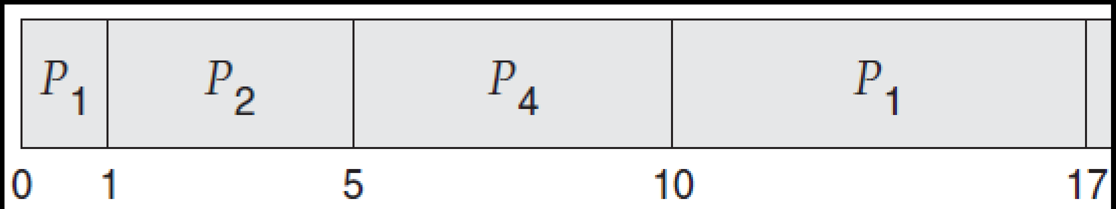


Shortest-Job-First Scheduling, SJF

P1 is finished. We are at time 17.
what is the shortest remaining process?

- Example

Process	Arrival Time	Burst Time	Remaining Time
P1	0	8	0
P2	1	4	0
P3	2	9	9
p4	3	5	0

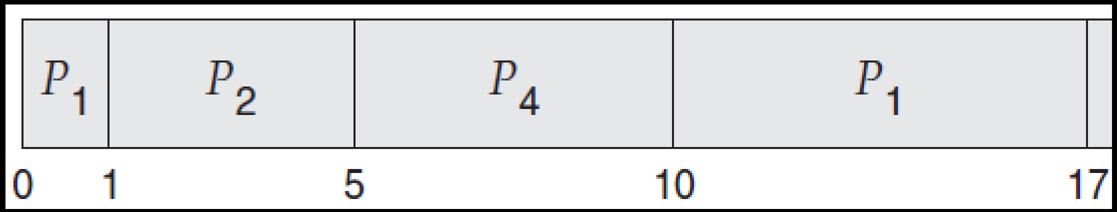


Shortest-Job-First Scheduling, SJF

P1 is finished. We are at time 17.
what is the shortest remaining process?

- Example

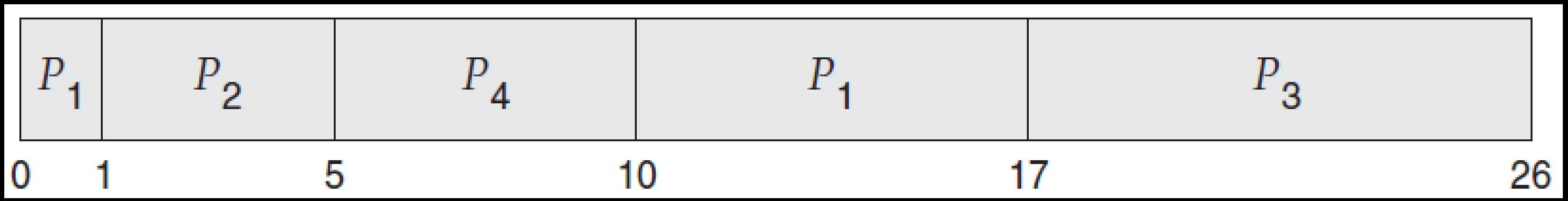
Process	Arrival Time	Burst Time	Remaining Time
P1	0	8	0
P2	1	4	0
P3	2	9	9
p4	3	5	0



Shortest-Job-First Scheduling, SJF

- Example

Process	Arrival Time	Burst Time	Remaining Time
P1	0	8	0
P2	1	4	0
P3	2	9	9
p4	3	5	0



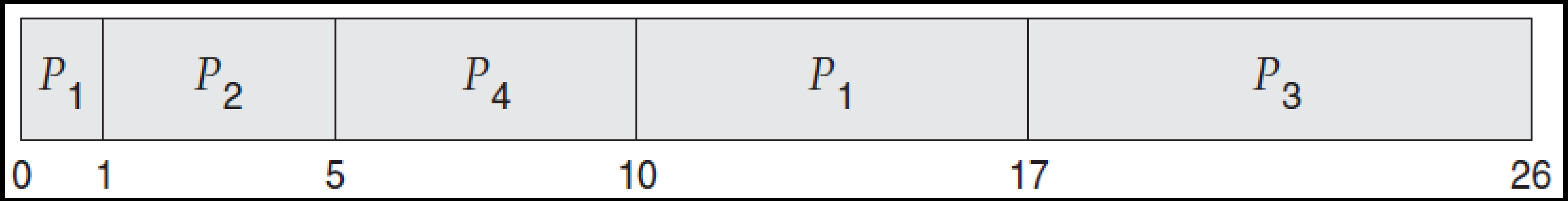
Shortest-Job-First Scheduling, SJF

All the processes are done!

- Example

Process	Arrival Time	Burst Time	Remaining Time	Waiting Time
P1	0	8	0	10 – 1
P2	1	4	0	1 – 1
P3	2	9	0	17 – 2
p4	3	5	0	5 - 3

Waiting time = burst time – arrival time



$$\text{Average waiting time} = \frac{9+0+15+2}{4} = 6.5\ ms$$

Shortest-Job-First Scheduling, SJF

- To avoid confusion, use this trick to solve the preemptive SJF.
  - Create a chart, with the x-axis is the time, and the y-axis the process.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
p1																												
p2																												
p3																												
p4																												

Shortest-Job-First Scheduling, SJF

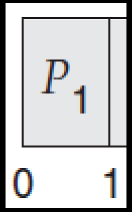
- To avoid confusion, use this trick to solve the preemptive SJF.
  - List each process from its arrival time until it finishes.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
p1	*	*	*	*	*	*	*	*	*																			
p2		*	*	*	*	*																						
p3			*	*	*	*	*	*	*	*	*	*																
p4				*	*	*	*	*	*																			

Shortest-Job-First Scheduling, SJF

- To avoid confusion, use this trick to solve the preemptive SJF.
  - Select each time slot with respect to the shortest time.

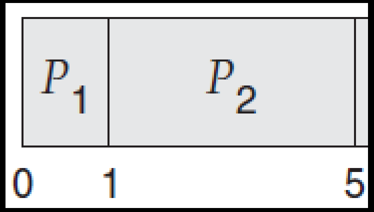
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
p1	*	*	*	*	*	*	*	*	*																			
p2		*	*	*	*	*																						
p3			*	*	*	*	*	*	*	*	*	*																
p4				*	*	*	*	*	*																			



Shortest-Job-First Scheduling, SJF

- To avoid confusion, use this trick to solve the preemptive SJF.
  - Select each time slot with respect to the shortest time.

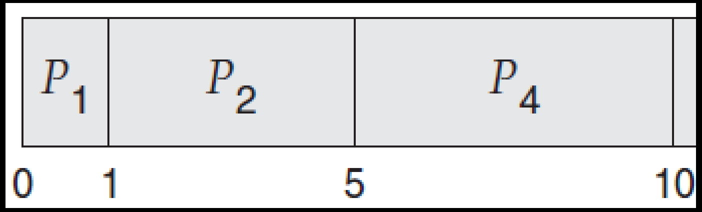
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
p1	*	*	*	*	*	*	*	*	*																			
p2		*	*	*	*	*																						
p3			*	*	*	*	*	*	*	*	*	*																
p4				*	*	*	*	*	*																			



Shortest-Job-First Scheduling, SJF

- To avoid confusion, use this trick to solve the preemptive SJF.
  - Select each time slot with respect to the shortest time.

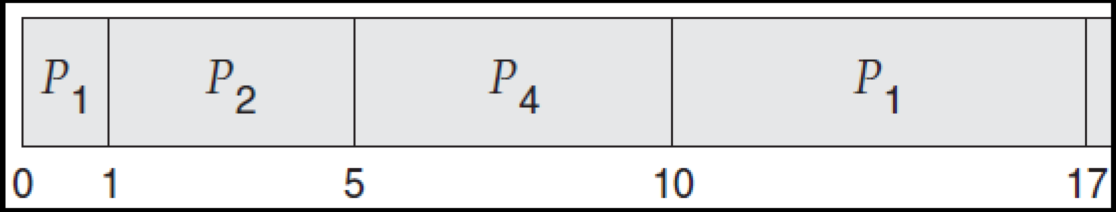
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
p1	*	*	*	*	*	*	*	*	*																			
p2		*	*	*	*	*																						
p3			*	*	*	*	*	*	*	*	*	*																
p4				*	*	*	*	*	*																			



Shortest-Job-First Scheduling, SJF

- To avoid confusion, use this trick to solve the preemptive SJF.
  - Select each time slot with respect to the shortest time.

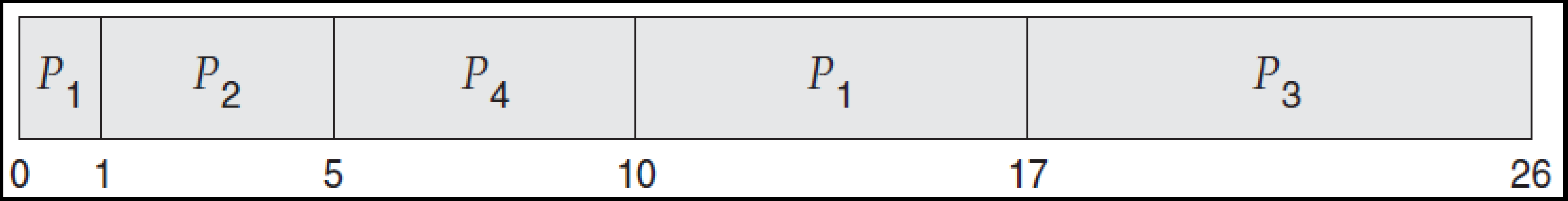
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
p1	*	*	*	*	*	*	*	*	*																			
p2		*	*	*	*	*																						
p3			*	*	*	*	*	*	*	*	*	*																
p4				*	*	*	*	*	*																			



Shortest-Job-First Scheduling, SJF

- To avoid confusion, use this trick to solve the preemptive SJF.
  - Select each time slot with respect to the shortest time.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
p1	*	*	*	*	*	*	*	*	*																			
p2		*	*	*	*	*																						
p3			*	*	*	*	*	*	*	*	*	*																
p4				*	*	*	*	*	*																			



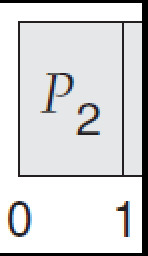
Priority Scheduling

- The CPU selects the process with the highest priority.
  - A priority is associated with each process.
  - Equal-priority processes are scheduled in FCFS order.
  - The SJF algorithm is a special case of the general **priority-scheduling** algorithm.
    - The larger the CPU burst, the lower the priority, and vice versa.
- The higher the priority, the lower its numerical value.
  - 0 is higher than 10.
  - 5 is less than 2.

Priority Scheduling

- Example: *suppose the processes came at the same time.*

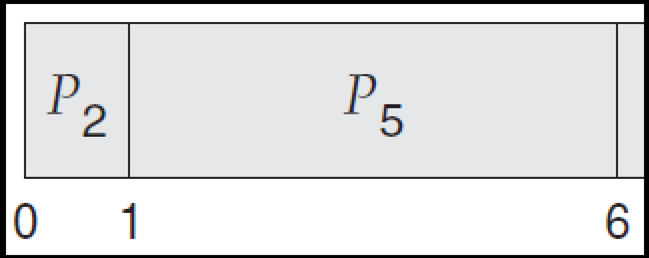
Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2



Priority Scheduling

- Example: *suppose the processes came at the same time.*

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



Priority Scheduling

- Example: *suppose the processes came at the same time.*

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

Priority Scheduling

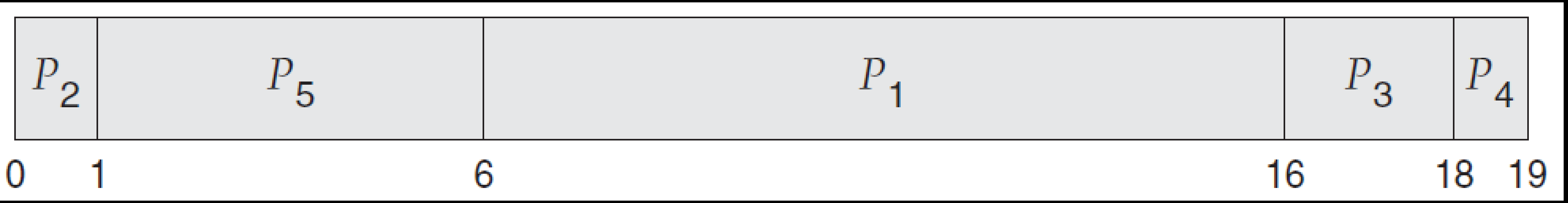
- Example: *suppose the processes came at the same time.*

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

Priority Scheduling

- Example: *suppose the processes came at the same time.*

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

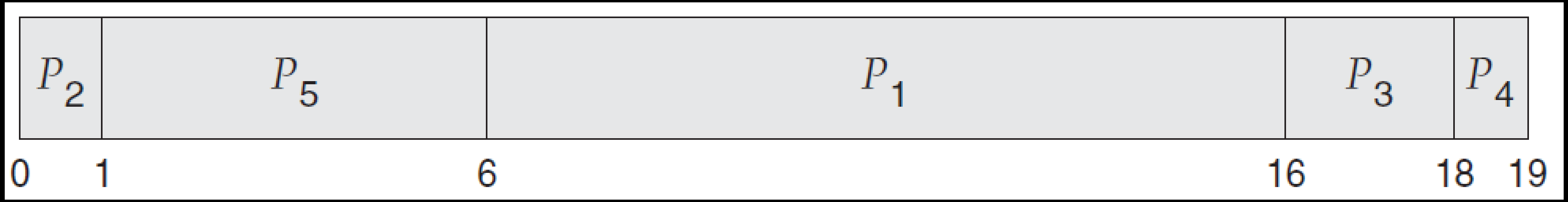


Priority Scheduling

- Example: *suppose the processes came at the same time.*

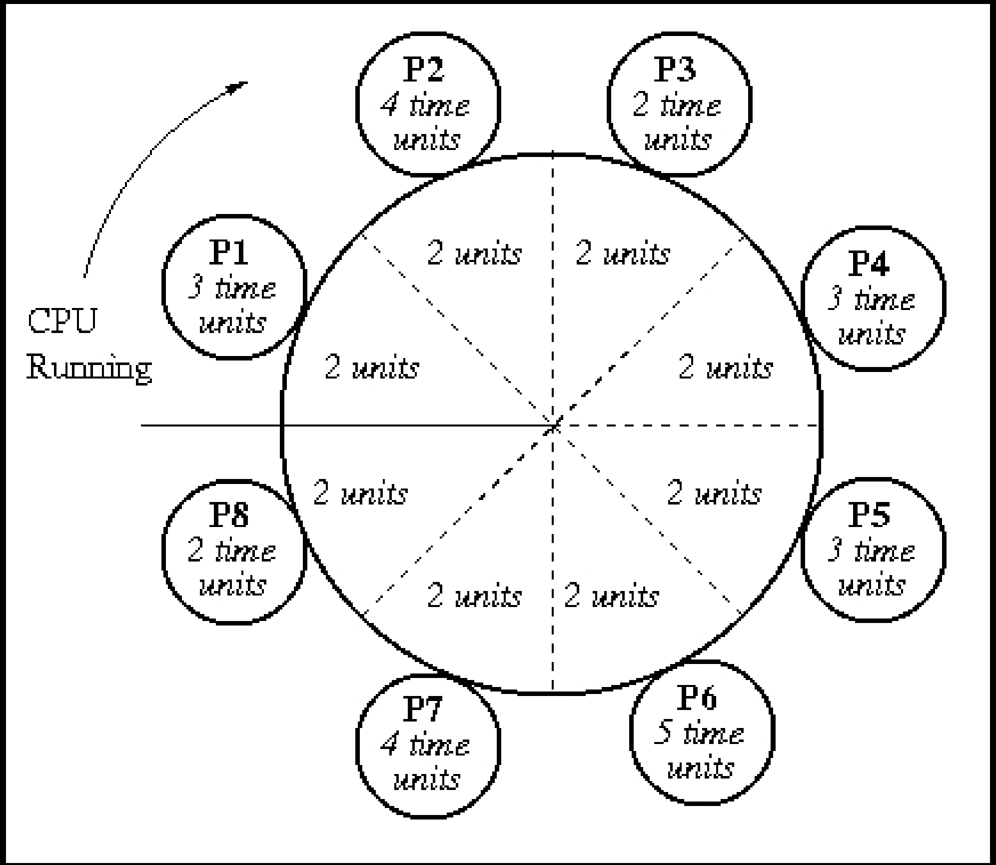
Process	Burst Time	Priority	Waiting Time
P1	10	3	6
P2	1	1	0
P3	2	4	16
P4	1	5	18
P5	5	2	1

Avg waiting time = $\frac{6+0+16+18+1}{5} = 8.2\ ms$



Round-Robin Scheduling

- Each process is assigned a fixed time slot in a cyclic way.
  - The unit of time, called time quantum or time slice.
- The ready queue is treated as a circular queue.
  - The CPU goes around the ready queue, allocating the CPU to each process for a time interval.



Round-Robin Scheduling

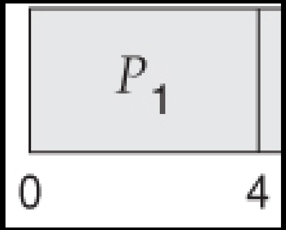
- Example: suppose that the processes came at the same time, and the time quantum is 4 ms.

Process	Burst Time
P1	24
P2	3
P3	3

Round-Robin Scheduling

- Example: suppose that the processes came at the same time, and the time quantum is 4 ms.

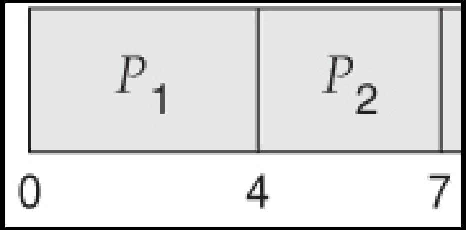
Process	Burst Time	Remaining Time
P1	24	20
P2	3	
P3	3	



Round-Robin Scheduling

- Example: suppose that the processes came at the same time, and the time quantum is 4 ms.

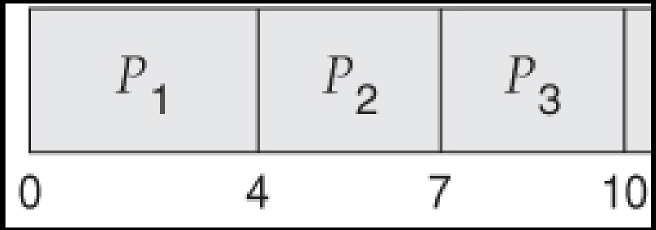
Process	Burst Time	Remaining Time
P1	24	20
P2	3	0
P3	3	



Round-Robin Scheduling

- Example: suppose that the processes came at the same time, and the time quantum is 4 ms.

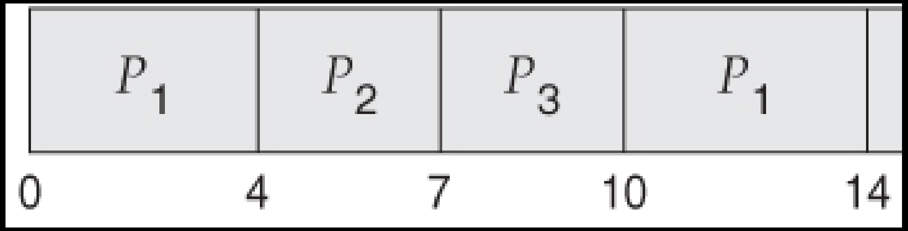
Process	Burst Time	Remaining Time
P1	24	20
P2	3	0
P3	3	0



Round-Robin Scheduling

- Example: suppose that the processes came at the same time, and the time quantum is 4 ms.

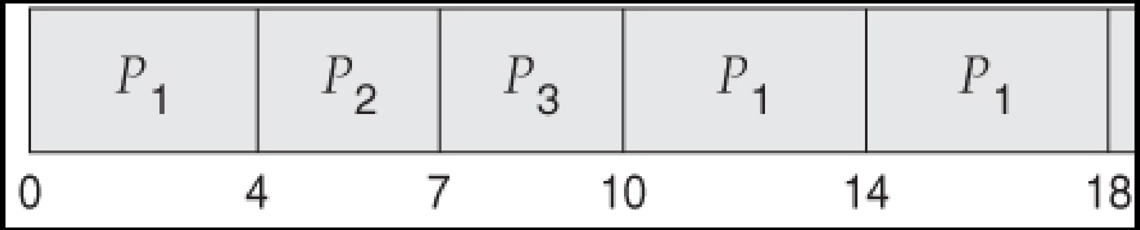
Process	Burst Time	Remaining Time
P1	24	16
P2	3	0
P3	3	0



Round-Robin Scheduling

- Example: suppose that the processes came at the same time, and the time quantum is 4 ms.

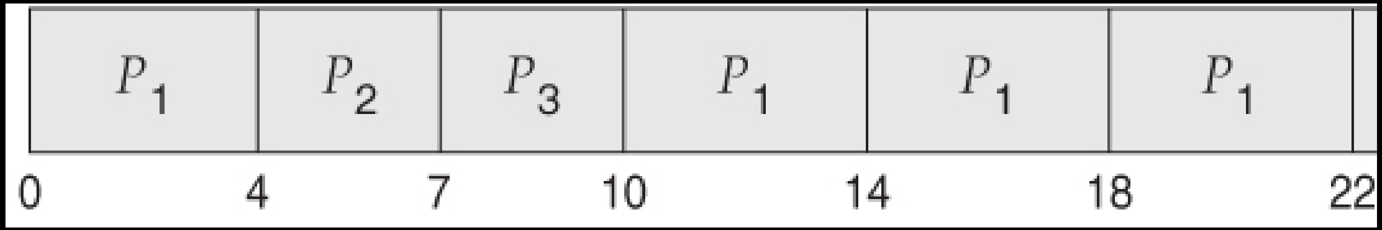
Process	Burst Time	Remaining Time
P1	24	12
P2	3	0
P3	3	0



Round-Robin Scheduling

- Example: suppose that the processes came at the same time, and the time quantum is 4 ms.

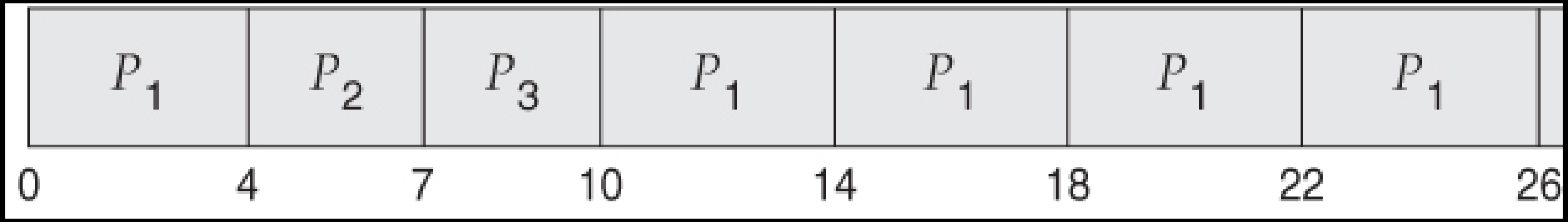
Process	Burst Time	Remaining Time
P1	24	8
P2	3	0
P3	3	0



Round-Robin Scheduling

- Example: suppose that the processes came at the same time, and the time quantum is 4 ms.

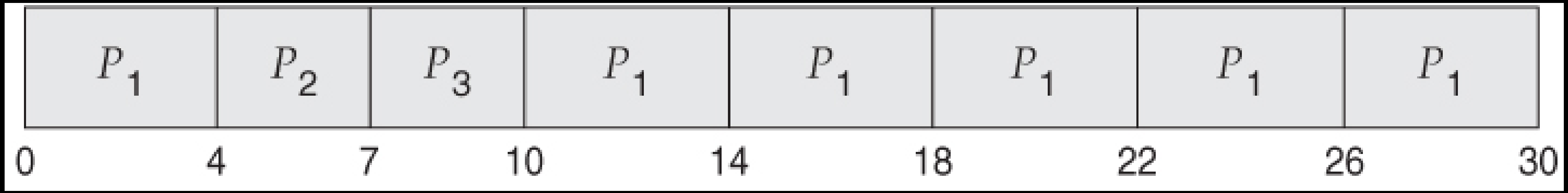
Process	Burst Time	Remaining Time
P1	24	4
P2	3	0
P3	3	0



Round-Robin Scheduling

- Example: suppose that the processes came at the same time, and the time quantum is 4 ms.

Process	Burst Time	Remaining Time
P1	24	0
P2	3	0
P3	3	0

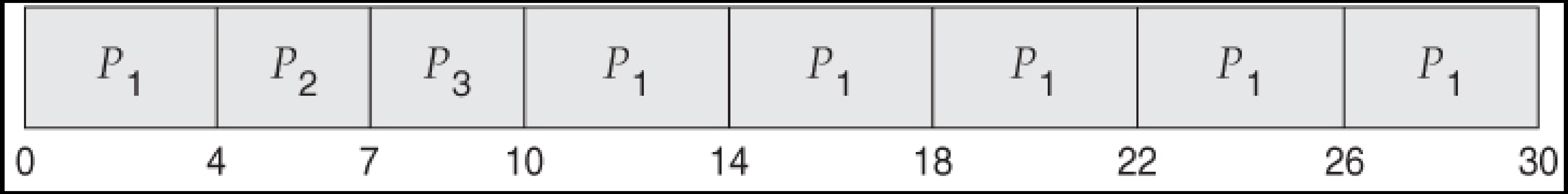


Round-Robin Scheduling

- Example: suppose that the processes came at the same time, and the time quantum is 4 ms.

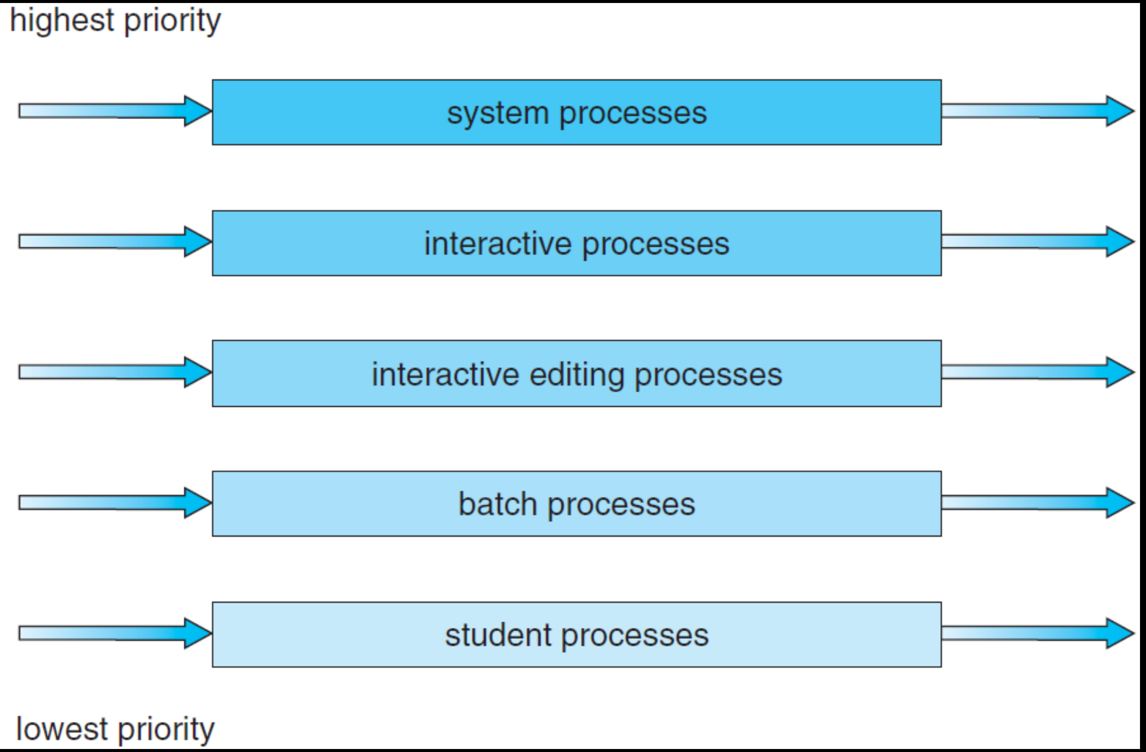
Process	Burst Time	Remaining Time	Waiting Time
P1	24	0	10-4
P2	3	0	4
P3	3	0	7

Avg waiting time = $\frac{10-4+4+7}{3} = 5.66\ ms$



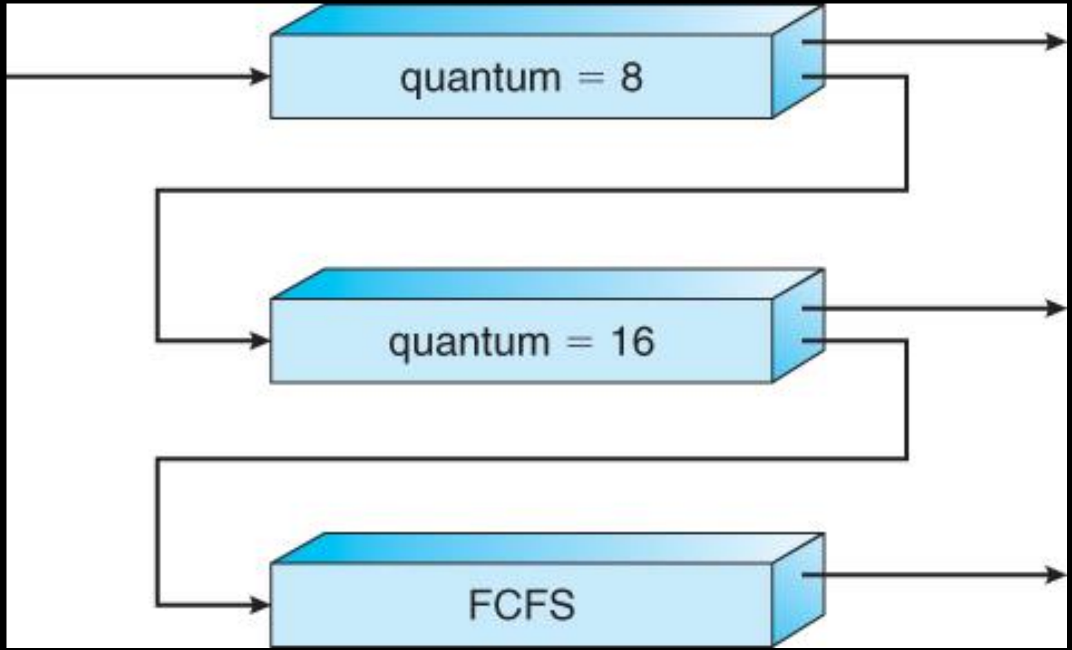
Multilevel Queue Scheduling

- Partitions the ready queue into several separate queues.
- The processes are permanently assigned to one queue.
  - Based on the property of the process (memory size, process priority, or process type).
- Each queue has its own scheduling algorithm.
- No process in the batch queue could run unless the queues for system processes, interactive processes, and interactive editing processes were all empty



Multilevel Feedback Queue Scheduling

- Like the ordinary multilevel queue scheduling, except jobs may be moved from one queue to another.
- the most general CPU-scheduling algorithm.
  - It can be configured to match a specific system under design.



TASK

6.16 Consider the following set of processes, with the length of the CPU burst given in milliseconds:

Process	Burst Time	Priority
P_1	2	2
P_2	1	1
P_3	8	4
P_4	4	2
P_5	5	3

The processes are assumed to have arrived in the order P_1, P_2, P_3, P_4, P_5, all at time 0.

a. Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, nonpreemptive priority (a larger priority number implies a higher priority), and RR (quantum = 2).

b. What is the turnaround time of each process for each of the scheduling algorithms in part a?

c. What is the waiting time of each process for each of these scheduling algorithms?

d. Which of the algorithms results in the minimum average waiting time (over all processes)?

Summary

- Scheduling algorithms are considered:
  - Preemptive
  - Non-preemptive