



# **SNS COLLEGE OF TECHNOLOGY**



**Coimbatore-35.**

**An Autonomous Institution**

**Accredited by NBA – AICTE and Accredited by NAAC – UGC with ‘A+’ Grade  
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai**

**COURSE NAME : 19CSB201 – OPERATING SYSTEMS**

**II YEAR/ IV SEMESTER**

**UNIT – II Process Scheduling And Synchronization**

**Topic: CPU Scheduling : Scheduling Algorithms**

Mr.N.Selvakumar

Assistant Professor

Department of Computer Science and Engineering



# CPU Scheduling Algorithms

There are many CPU scheduling algorithms for deciding which of the process in the ready queue is to be allocated to the CPU.

## Types

- First Come First Serve Scheduling
- Shortest Job First Scheduling
- Priority Scheduling
- Round Robin Scheduling



# First Come First Serve Scheduling

- It is the simplest CPU Scheduling algorithm
- In this, the process that requests the CPU first is allocated to the CPU first.
- The Implementation of FCFS is easily managed with a FIFO queue.



## FIFO Queue



- The process enters the ready queue onto the tail of the queue
- When the CPU is free the process at the head of the queue will be allocated



Consider the following set of processes Arrives at time 0, with the length of the CPU burst time given in milliseconds

<u>Process</u>	<u>Burst Time</u>
$P_1$	24
$P_2$	3
$P_3$	3

- i. Draw gantt chart illustrating process execution for all scheduling
- ii. Find Turn around time
- iii. Find Waiting time



## Gantt Chart

It is a bar chart that illustrates the process schedule, including the start and finish times of each participating processes.

## Waiting Time

Waiting Time is the sum of time period spent waiting in the ready queue.

## Turnaround Time

Turnaround time is the sum of period spent waiting to get into memory, waiting in the ready queue, executing on the CPU and doing I/O



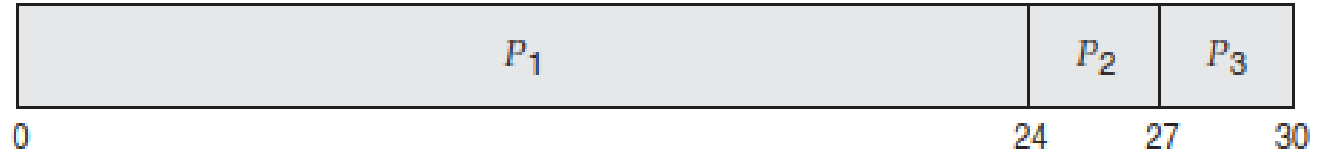
# FCFS Scheduling

## Example 1



<u>Process</u>	<u>Burst Time</u>
$P_1$	24
$P_2$	3
$P_3$	3

### Gantt Chart



Arrives at time 0

Process	Burst Time	Waiting Time	Turnaround Time
P1	24	0	24
P2	3	24	27
P3	3	27	30
Average Waiting Time/ Turnaround Time		$(51/3) = 17$	$(81/3) = 27$



- **Average Waiting Time(Wt)** = ( Wt P1 + Wt P2 + Wt P3 ) / 3  
= ( 0+24+27 ) / 3  
= 51 / 3 = 17
- **Average Turnaround Time(Tt)** = ( Tt P1 + Tt P2 + Tt P3 ) / 3  
= ( 24+27+30 ) / 3  
= 81 / 3 = 27





# FCFS Scheduling



<u>Process</u>	<u>Burst Time</u>
$P_1$	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:  $(0 + 24 + 27)/3 = 17$
- **Convoy effect** - short process behind long process



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



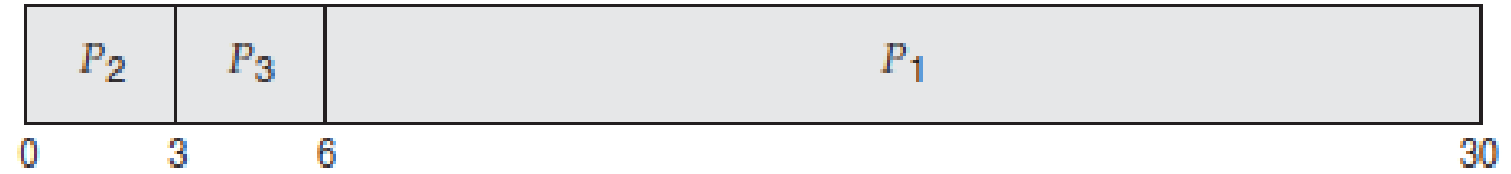
# FCFS Scheduling

## Example 2



<u>Process</u>	<u>Burst Time</u>	<u>Arrival Time</u>
P1	24	3
P2	3	0
P3	3	1

### Gantt Chart



Process	Burst Time	Waiting Time	Turnaround Time
P1	24	$6 - 3 = 3$	$3 + 24 = 27$
P2	3	$0 - 0 = 0$	$0 + 3 = 3$
P3	3	$3 - 1 = 2$	$2 + 3 = 5$
Average Waiting Time/ Turnaround Time		$(5/3) = 1.67$	$(35/3) = 11.67$



# Formulae

## Waiting Time

$$\text{Waiting Time} = \text{Starting Time} - \text{Arrival Time}$$

## Turnaround Time

$$\text{Turnaround time} = \text{Waiting Time} + \text{Burst Time}$$



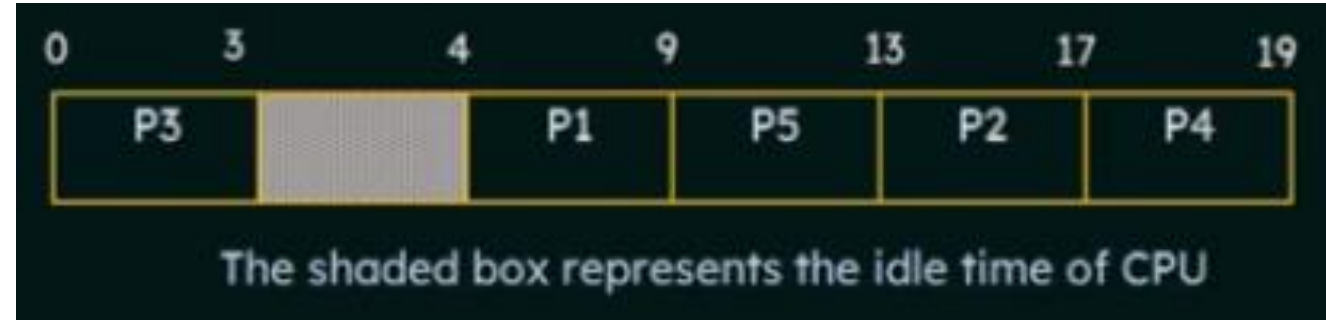
# FCFS Scheduling

## Example 3



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

### Gantt Chart



Process ID	Completion Time	Turnaround Time	Waiting Time
P1	9	$9 - 4 = 5$	$5 - 5 = 0$
P2	17	$17 - 6 = 11$	$11 - 4 = 7$
P3	3	$3 - 0 = 3$	$3 - 3 = 0$
P4	19	$19 - 6 = 13$	$13 - 2 = 11$
P5	13	$13 - 5 = 8$	$8 - 4 = 4$



# Formulae

## Turnaround Time

Turnaround Time = Completion Time – Arrival Time

## Waiting Time

Waiting Time = Turnaround Time – Burst Time



$$\begin{aligned}\text{Average Turn Around time} &= (5 + 11 + 3 + 13 + 8) / 5 \\ &= 40 / 5 \\ &= 8 \text{ units}\end{aligned}$$

$$\begin{aligned}\text{Average waiting time} &= (0 + 7 + 0 + 11 + 4) / 5 \\ &= 22 / 5 \\ &= 4.4 \text{ units}\end{aligned}$$



# Shortest Job First Scheduling

- This algorithm 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



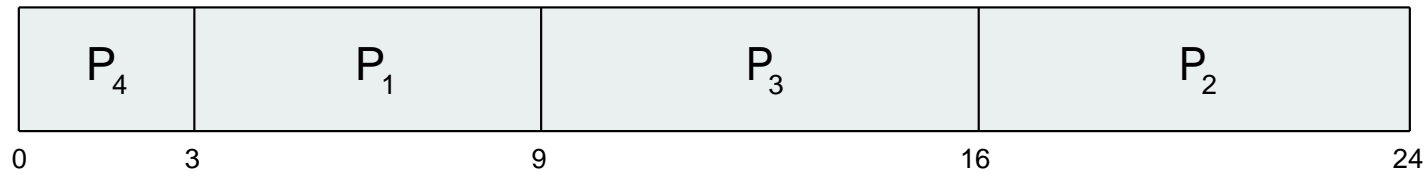


# SJF Scheduling



<u>Process</u>	<u>Burst Time</u>
$P_1$	6
$P_2$	8
$P_3$	7
$P_4$	3

- SJF scheduling chart



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

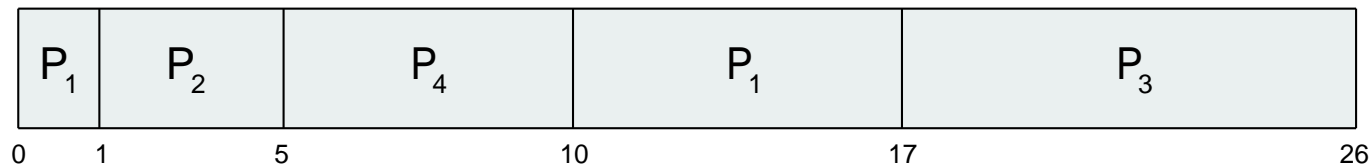


## Preemptive SJF

Preemptive version called **shortest-remaining-time-first**

<u>Process</u>	<u>Arrival Time</u>	<u>Burst Time</u>
$P_1$	0	8
$P_2$	1	4
$P_3$	2	9
$P_4$	3	5

- *Preemptive* SJF Gantt Chart



- Average waiting time =  $[(10-1)+(1-1)+(17-2)+5-3])/4 = 26/4 = 6.5$  msec



# Priority Scheduling

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer  $\equiv$  highest priority)

## Types

- Preemptive
- Nonpreemptive



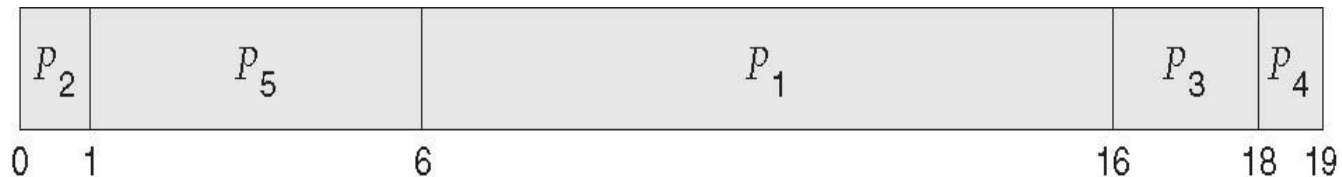
- SJF is priority scheduling where priority is the inverse of predicted next CPU burst time
- Problem  $\equiv$  **Starvation** – low priority processes may never execute
- Solution  $\equiv$  **Aging** – as time progresses increase the priority of the process



## Non-Preemptive

<u>Process</u>	<u>Burst Time</u>	<u>Priority</u>
$P_1$	10	3
$P_2$	1	1
$P_3$	2	4
$P_4$	1	5
$P_5$	5	2

- Priority scheduling Gantt Chart



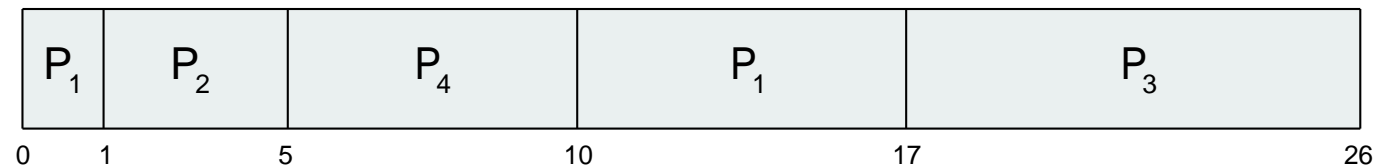
- Average waiting time = 8.2 msec



## Preemptive

<u>Process</u>	<u>Arrival Time</u>	<u>Burst Time</u>	<u>Priority</u>
$P_1$	0	8	3
$P_2$	1	4	1
$P_3$	2	9	4
$P_4$	3	5	2

- Priority scheduling Gantt Chart



- Average waiting time = 6.5 msec



# Round Robin Scheduling

- It is similar to FCFS Scheduling, but preemption is added to enable switch between processes.
- A small unit of time called **time quantum** or time slice is defined.
- After the execution of time quantum period, the running process will be placed at back of ready queue & allows next process to occupy the CPU.



- 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





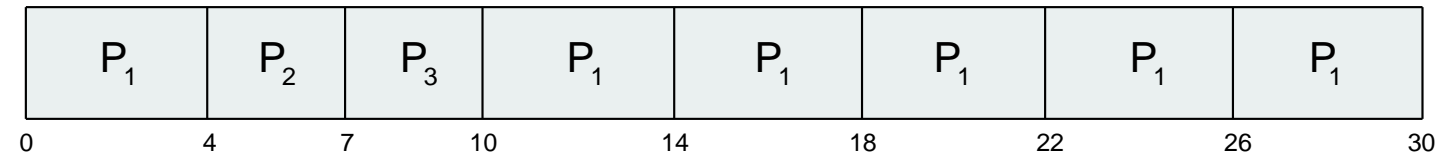
# Round Robin Scheduling



<u>Process</u>	<u>Burst Time</u>
$P_1$	24
$P_2$	3
$P_3$	3

If Time Quantum is **4 milliseconds**

- The Gantt chart is:



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



# REFERENCES

## TEXT BOOKS:

- T1 Silberschatz, Galvin, and Gagne, “Operating System Concepts”, Ninth Edition, Wiley India Pvt Ltd, 2009.)
- T2. Andrew S. Tanenbaum, “Modern Operating Systems”, Fourth Edition, Pearson Education, 2010

## REFERENCES:

- R1 Gary Nutt, “Operating Systems”, Third Edition, Pearson Education, 2004.
- R2 Harvey M. Deitel, “Operating Systems”, Third Edition, Pearson Education, 2004.
- R3 Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, “Operating System Concepts”, 9th Edition, John Wiley and Sons Inc., 2012.
- R4. William Stallings, “Operating Systems – Internals and Design Principles”, 7th Edition, Prentice Hall, 2011

