
MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥
Dr. Vishwanath Karad
MIT WORLD PEACE
UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

# Scheduling

# Module 2

**Process Management**

- **Process:** Concept of a Process, Process States, Process Control - creation, new program execution, termination. Interposes communication(IPC). Examples of IPC.

- **Threads:** Differences between Threads and Processes. Concept of Threads, Concurrency. Multi- threading, Types of Threads. POSIX Threads functions.

- **Scheduling:** Concept of Scheduler, Scheduling Algorithms: FCFS, SJF, SRTN, Priority, Round Robin.

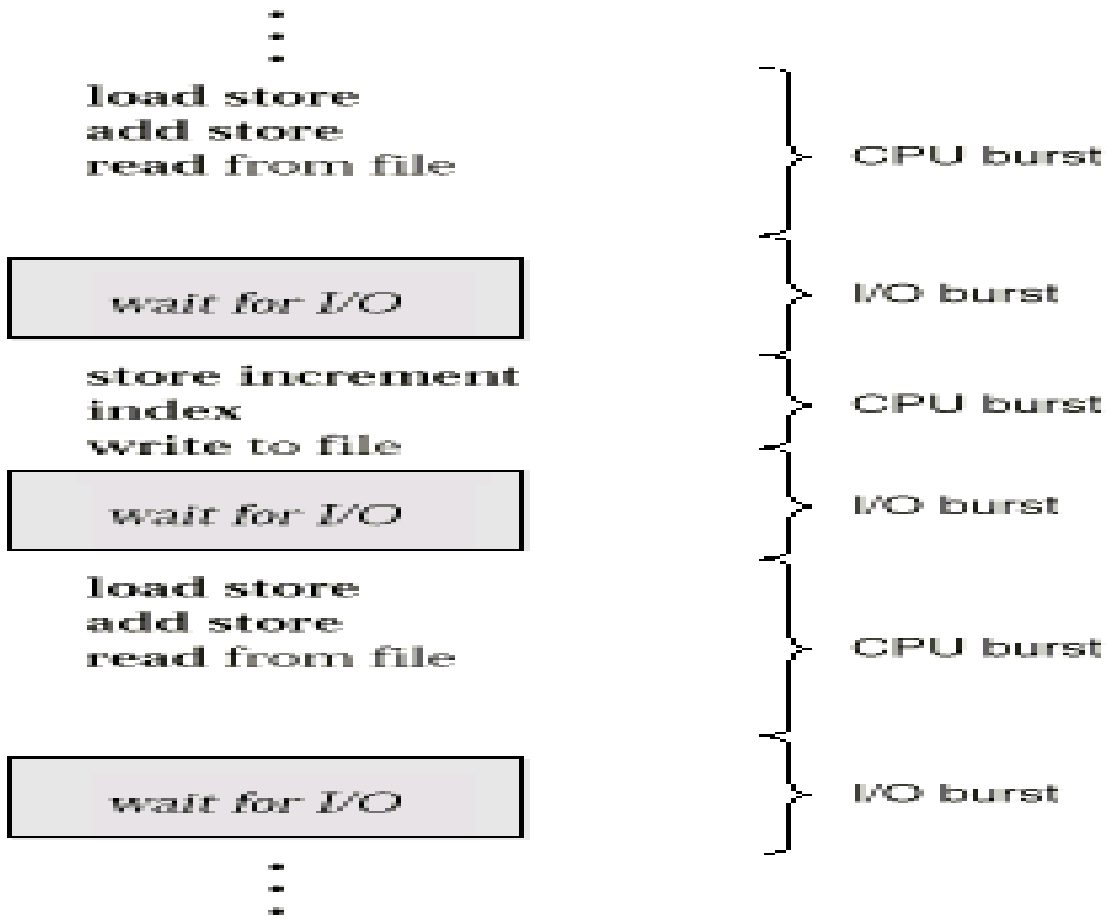
# Scheduling

- **Scheduling:** Concept of Scheduler, Scheduling Algorithms: FCFS, SJF, SRTN, Priority, Round Robin.
-

# Scheduling

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

# Alternating Sequence of CPU And I/O Bursts



# CPU / Process Scheduler

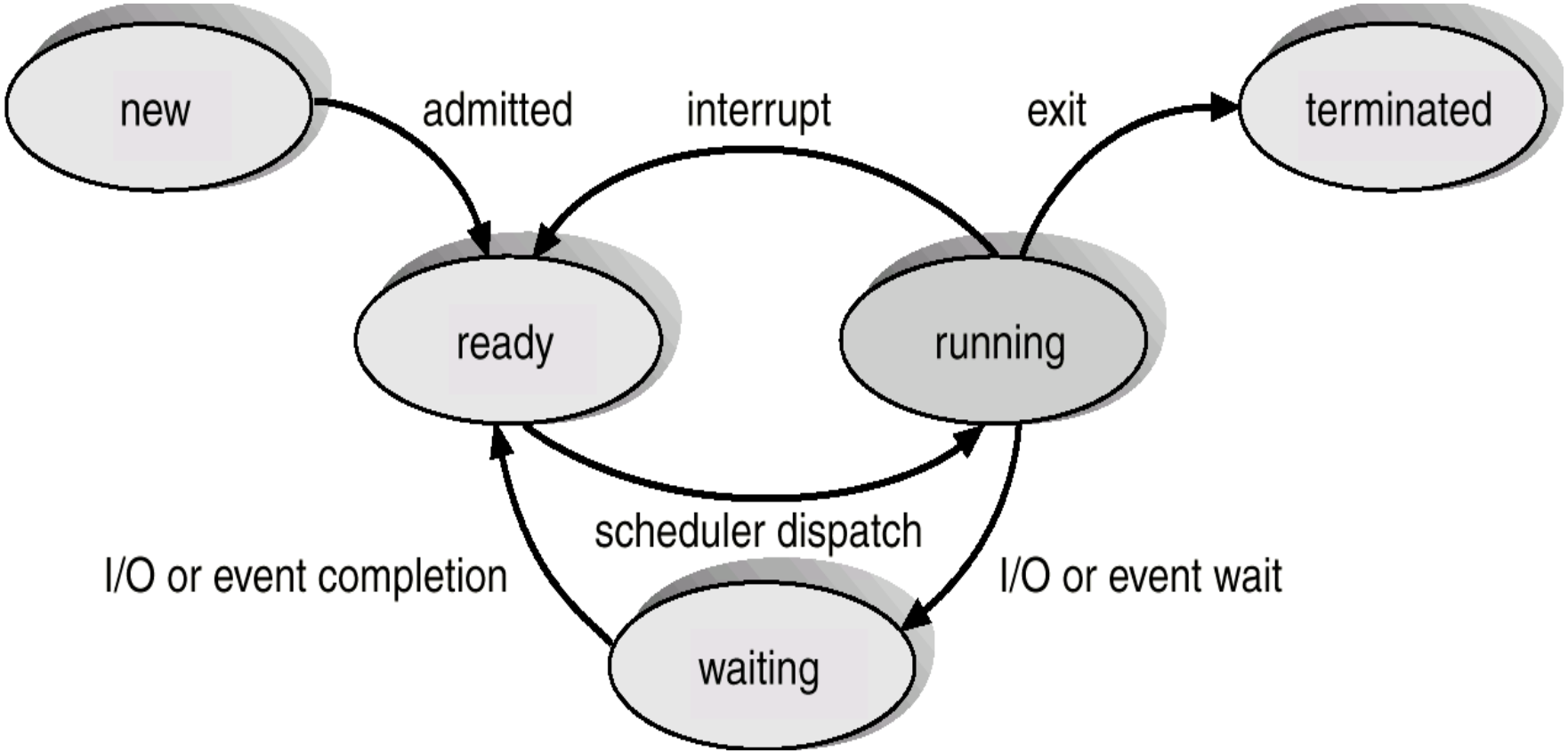
- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them.

- Scheduling may be *preemptive or non-preemptive*
- ***Non-preemptive*** – If CPU allocated to a process, it keeps the CPU until it releases it by terminating or switching to waiting state
- ***Preemptive*** - If CPU allocated to a process, it may be released if high priority process needs the CPU

# Scheduler

- 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 state
  4. Terminates
- Scheduling under 1 and 4 is *non-preemptive*
- All other scheduling is *preemptive*

# Diagram for Process States



# Scheduling Algorithms

They deal with the problem of deciding which of the processes in ready queue is to be allocated the CPU

Algorithm compared based on following criteria

- **CPU utilization** – keep the CPU as busy as possible
- **Throughput** – number of processes completed or amount of work done per unit time
- **Turnaround time** – time of submission of a process to the time of completion
- **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

# Optimization Criteria

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

# Types of process schedulers / Scheduling categories

Scheduling is broken down into three categories:

**1. Long term scheduling:**

- Is performed when a new process is created.
- Long-term scheduler (or job scheduler) – selects which processes should be brought into the ready queue.
- Long-term scheduler is invoked very infrequently (seconds, minutes) ⇒ (may be slow)
- The long-term scheduler controls the *degree of multiprogramming*.

•

# Types of process schedulers / Scheduling categories

**2. Medium term scheduling:**

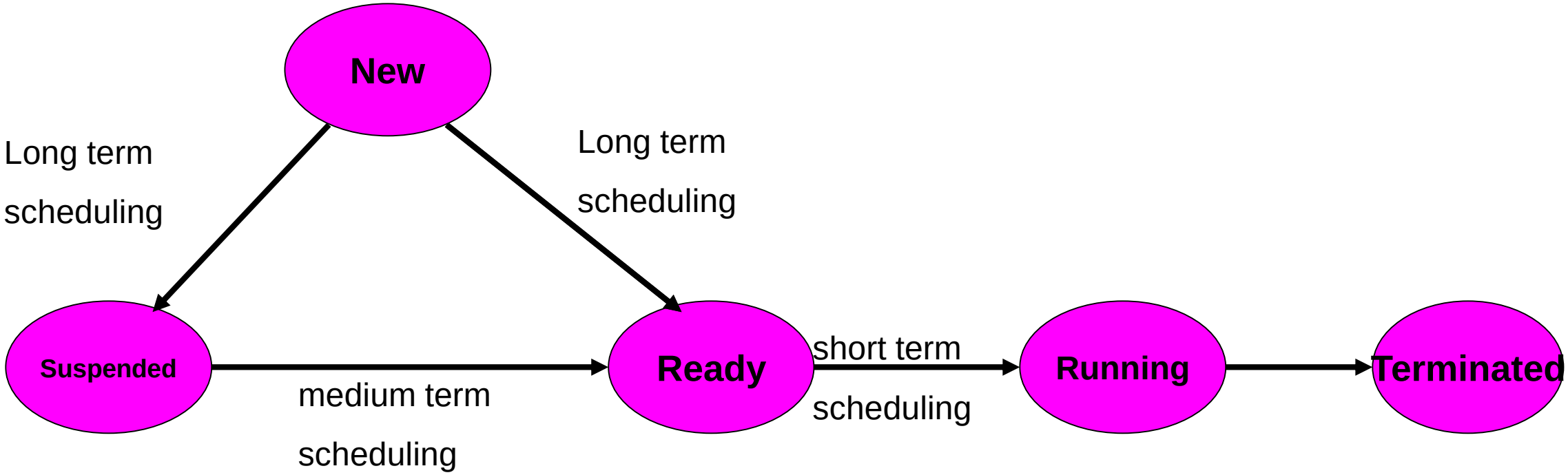
- Part of the swapping function
- Swapping-in decisions are taken by medium term scheduler
- Based on the need to manage the degree of multiprogramming

# Types of process schedulers / Scheduling categories

**3.Short term scheduling:**

- Determines which ready process will be assigned the CPU when it next becomes available and actually assign the CPU to this process **.**
- Short-term scheduler (or CPU scheduler) – selects which process should be executed next and allocates CPU.
- Short-term scheduler is invoked very frequently (milliseconds)⇒ (must be fast).

# Types of process schedulers / Scheduling categories



# Scheduling Algorithms

- FCFS (First Come First Serve)
- SJF (Shortest Job First)
- Priority scheduling
- Round Robin scheduling

# FCFS Scheduling: Characteristics

**Selection Function:** max(w), selects the process which is waiting in the ready queue for maximum time.

**Decision Mode :** Non_preemptive

**Throughput:** Not emphasized

**Response Time:** May be high, especially if there is a large variance in the process execution times.

**Overhead:** Minimum

**Effect on Processes:** Penalizes short processes

**Starvation:** No

- **Completion Time**

  Time at which process completes its execution.
- **Turn Around Time**

  Time Difference between completion time and arrival time.

  Turn Around Time = Completion Time – Arrival Time
- **Waiting Time(W.T)**

  Time Difference between turn around time and burst time.

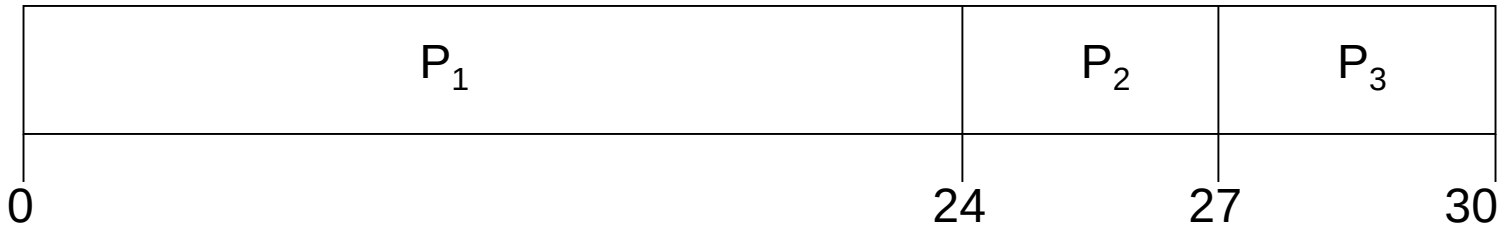
  Waiting Time = Turn Around Time – Burst Time

| Process | Burst Time |
|---|---|
| *P1* | 24 |
| *P2* | 3 |
| *P3* | 3 |

- Suppose that the processes arrive in the order: *P1* , *P2* , *P3*

# The Gantt Chart for the schedule is:

# The Gantt Chart for the schedule is:

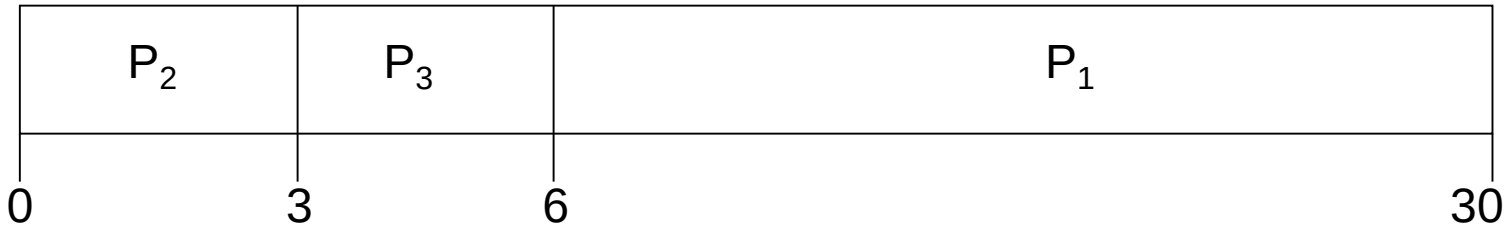



Waiting time for *P1* = 0; *P2* = 24; *P3* = 27
Average waiting time: (0 + 24 + 27)/3 = 17

Turnaround time for *P1* = 24; *P2* = 27; *P3* = 30
Average turnaround time : (24 + 27 + 30)/3 = 27

# FCFS Scheduling (Cont.)

- Suppose that the processes arrive in the order : *P2* , *P3* , *P1*



# FCFS Scheduling (Cont.)

- Suppose that the processes arrive in the order : *P2* , *P3* , *P1*


Waiting time for *P1* = 6; *P2* = 0; *P3* = 3
Average waiting time: (6 + 0 + 3)/3 = 3

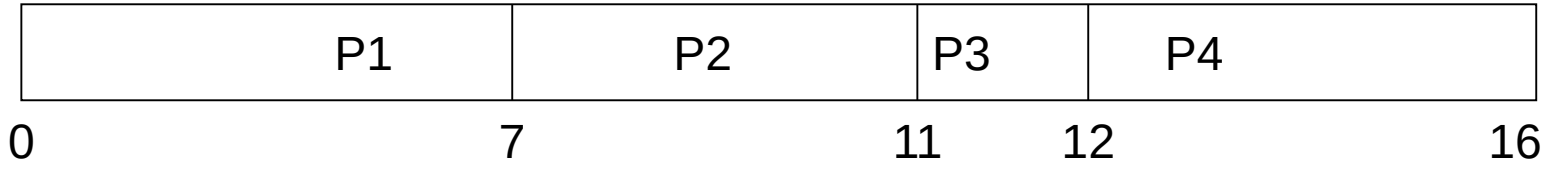
Turnaround time for *P1* = 30; *P2* = 3; *P3* = 6
Average turnaround time : (30 + 3 + 6)/3 = 13

# Example-2 of FCFS

| Process | Arrival Time | Burst Time |
|---|---|---|
| *P1* | 0.0 | 3 |
| *P2* | 2.0 | 6 |
| *P3* | 4.0 | 4 |
| *P4* | 6.0 | 5 |
| P5 | 8.0 | 2 |

# Example-2 of FCFS

| Process | Arrival Time | Burst Time |
|---|---|---|
| *P1* | 0.0 | 7 |
| *P2* | 2.0 | 4 |
| *P3* | 4.0 | 1 |
| *P4* | 5.0 | 4 |



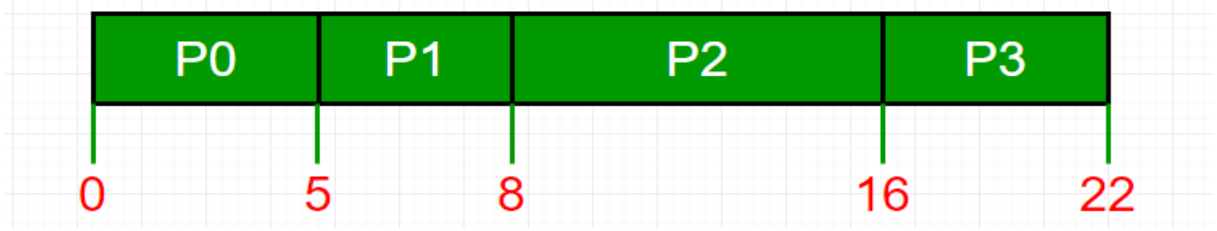

**Average waiting time = (0 + 5 + 7 + 7)/4 = 4.75**
**Average Turnaround Time = (7+9+8+11)/4=8.75**

# First Come First Serve (FCFS) ( Non Pre-emptive)

| Processes | Burst time | Arrival Time | Service Time |
|---|---|---|---|
| P0 | 5 | 0 | 0 |
| P1 | 3 | 1 | 5 |
| P2 | 8 | 2 | 8 |
| P3 | 6 | 3 | 16 |

- **Service Time :** Service time means amount of time after which a process can start execution.

It is summation of burst time of previous processes (Processes that came before)



# First Come First Serve (FCFS) ( Non Pre-emptive)

| Processes | Burst time | Arrival Time | Service Time |
|---|---|---|---|
| P0 | 5 | 0 | 0 |
| P1 | 3 | 1 | 5 |
| P2 | 8 | 2 | 8 |
| P3 | 6 | 3 | 16 |

- **To find waiting time:**

  Time taken by all processes before the current process to be started

  (i.e. burst time of all previous processes) – arrival time of current process

  **wait_time[i] = (bt[0] + bt[1] +…… bt[i-1] ) – arrival_time[i]**

| Process | Wait Time : Service Time - Arrival Time |
|---|---|
| P0 | 0 - 0 = 0 |
| P1 | 5 - 1 = 4 |
| P2 | 8 - 2 = 6 |
| P3 | 16 - 3 = 13 |

Average Wait Time: (0 + 4 + 6 + 13) / 4 = 5.75

# First Come First Serve (FCFS) ( Non Pre-emptive)

**Implementation**

1) Input the processes along with their burst time (bt) and arrival time (at).
2) Find **waiting time (wt)** for all processes. i.e. for a given process i:

   wt[i] = (bt[0] + bt[1] + . . . . . + bt[i-1]) - at[i] .

3) Now find **turnaround time** = waiting_time + burst_time for all processes.
4) Find **average waiting time** = total_waiting_time / no_of_processes.
5) Similarly, find **average turnaround time** = total_turn_around_time / no_of_processes.

# Shortest-Job-First (SJF) Scheduling

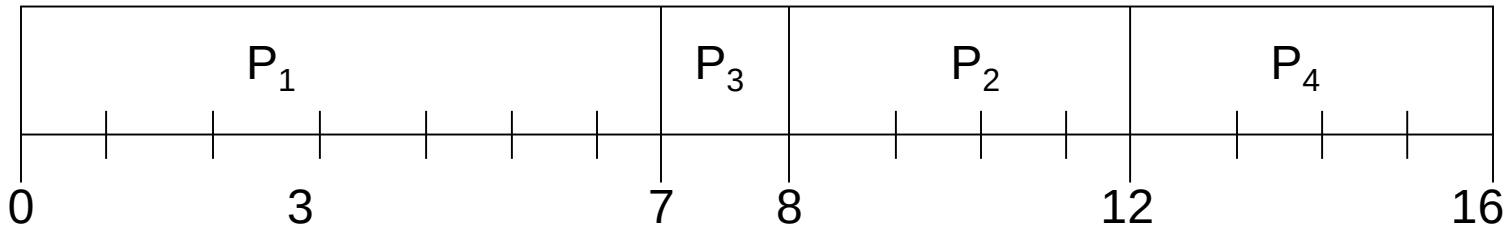
- Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time
- Two schemes:
  - **Nonpreemptive** – once CPU given to the process it cannot be preempted until completes its CPU burst
  - **Preemptive** – If a new process arrives with 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 |
|---|---|---|
| *P1* | 0.0 | 7 |
| *P2* | 2.0 | 4 |
| *P3* | 4.0 | 1 |
| *P4* | 5.0 | 4 |

# Example of Non-Preemptive SJF

| Process | Arrival Time | Burst Time |
|---|---|---|
| *P1* | 0.0 | 7 |
| *P2* | 2.0 | 4 |
| *P3* | 4.0 | 1 |
| *P4* | 5.0 | 4 |




- **Average waiting time = (0 + 6 + 3 + 7)/4 = 4**
- **Average Turnaround Time = (7+10+4+11)/4=8**

# Shortest Job First Preemptive or Shortest Remaining Time

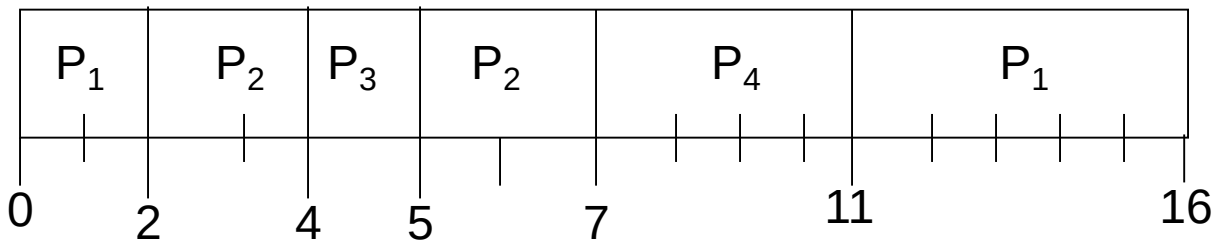
- It is a preemptive version of SJF. In this policy, scheduler always chooses the process that has the **shortest expected remaining processing time.**
- When a new process arrives in the ready queue, it may in fact have a shorter remaining time than the currently running process.
- Accordingly, the scheduler may preempt whenever a new process becomes ready.
- Scheduler must have an estimate of processing time to perform the selection function.

# Shortest Job First Preemptive or Shortest Remaining Time: characteristics

- **Selection Function**: minimum total service time required by the process, minus time spent in execution so far.
- **Decision Mode :** Preemptive ( At arrival time)
- **Throughput:** High
- **Response Time:** Provides good response time
- **Overhead:** Can be high
- **Effect on Processes:** Penalizes long processes.
- **Starvation:** Possible

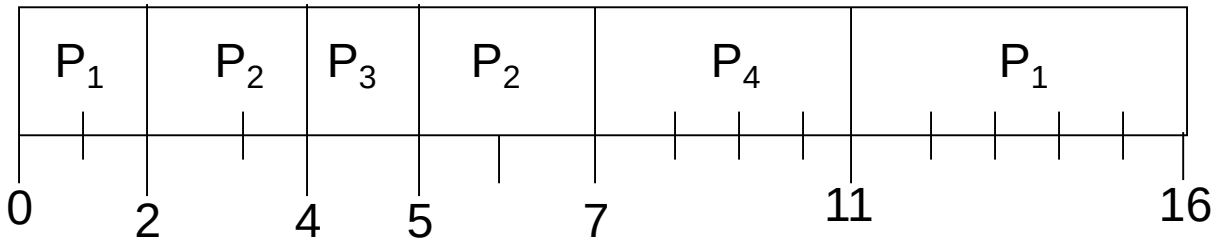
# Example of Preemptive SJF

| 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 |



# Example 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 |

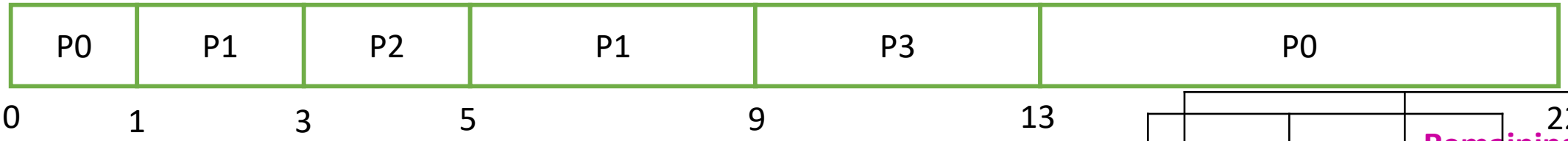



- Average waiting time = (9 + 1 + 0 +2)/4 = 3
- Average turnaround time= (16+5+1+6)/4=7

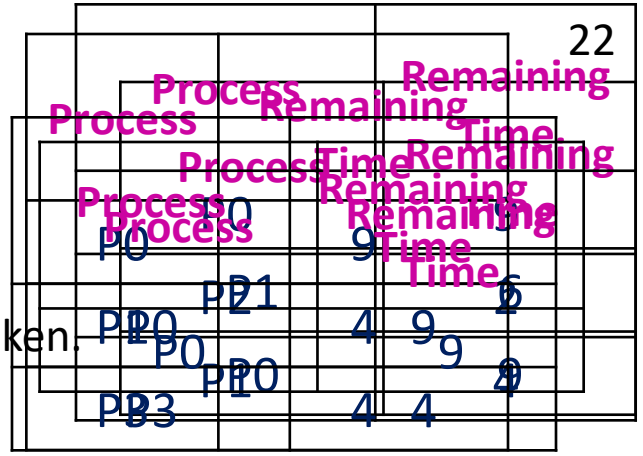
# Shortest Remaining Time Next (SRTN) ( Pre-emptive)_Example

| Process | Arrival Time (T0) | CPU Burst Time (in milliseconds) (Time required for completion ΔT) |
|---|---|---|
| P0 | 0 | 10 |
| P1 | 1 | 6 |
| P2 | 3 | 2 |
| P3 | 5 | 4 |

**Gantt Chart**




- Initially only process P0 is present and it is allowed to run
- But when process P1 comes, it has shortest remaining run time.
- So, P0 is pre-empted and P1 is allowed to run.
- Whenever new process comes or current process blocks, such type of decision is taken.
- This procedure is repeated till all processes complete their execution.

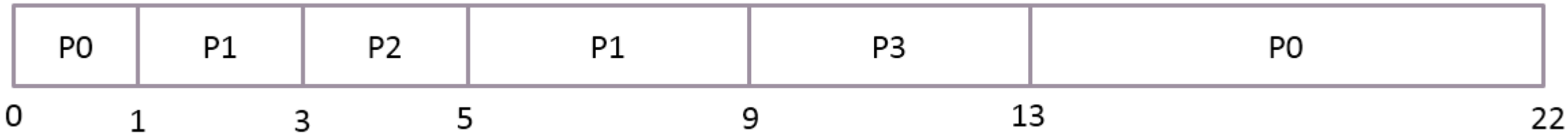


# Shortest Remaining Time Next (SRTN) ( Pre-emptive)_Example

## Output

| Process | Arrival Time (T0) | Burst Time (ΔT) | Finish Time (T1) |
|---|---|---|---|
| P0 | 0 | 10 | 22 |
| P1 | 1 | 6 | 9 |
| P2 | 3 | 2 | 5 |
| P3 | 5 | 4 | 13 |

## Gantt Chart

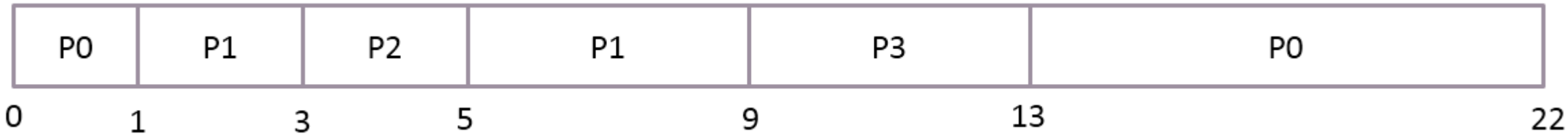


# Shortest Remaining Time Next (SRTN) ( Pre-emptive)_Example

## Output

| Process | Arrival Time (T0) | Burst Time (ΔT) | Finish Time (T1) | Turnaround Time (TAT = T1 - T0) |
|---|---|---|---|---|
| P0 | 0 | 10 | 22 | 22 |
| P1 | 1 | 6 | 9 | 8 |
| P2 | 3 | 2 | 5 | 2 |
| P3 | 5 | 4 | 13 | 8 |

## Gantt Chart



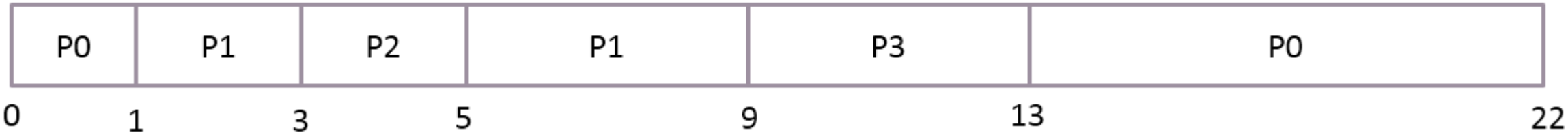

- Average Turnaround Time= (22+8+2+8) / 4 = 10 milliseconds

# Shortest Remaining Time Next (SRTN) ( Pre-emptive)_Example

## Output

| Process | Arrival Time (T0) | Burst Time (ΔT) | Finish Time (T1) | Turnaround Time (TAT = T1 - T0) | Waiting Time (WT = TAT - ΔT) |
|---|---|---|---|---|---|
| P0 | 0 | 10 | 22 | 22 | 12 |
| P1 | 1 | 6 | 9 | 8 | 2 |
| P2 | 3 | 2 | 5 | 2 | 0 |
| P3 | 5 | 4 | 13 | 8 | 4 |

## Gantt Chart




- Average Turnaround Time= (22+8+2+8) / 4 = 10 milliseconds
- Average Waiting Time = (12+2+0+4) / 4= 4.5 milliseconds

# 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

MIT-WPU
॥ विश्वशान्तिर्भुवं ध्रुवा ॥

## Example

Consider the set of processes with arrival time (in milliseconds), CPU burst time (in milliseconds) , and priority (0 is the highest priority) shown below. None of the processes have I/O burst time.

| 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 |

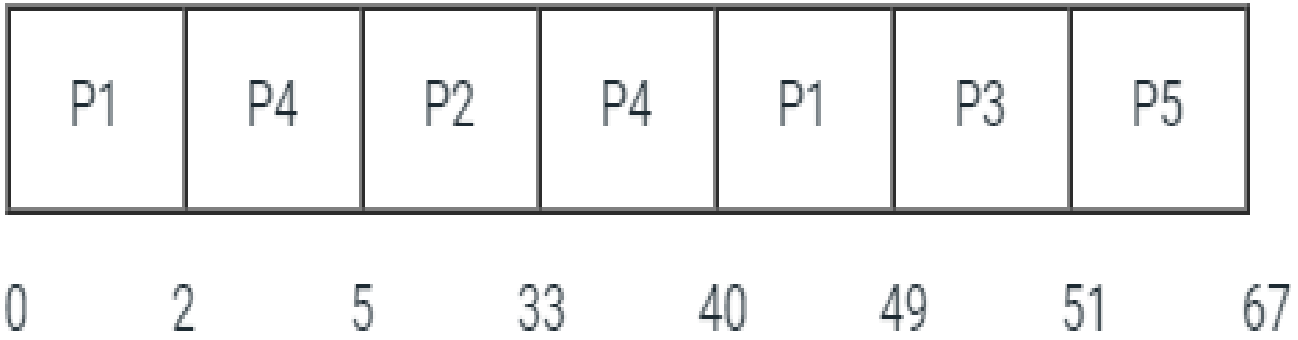
The average waiting time (in milliseconds) of all the processes using preemptive priority scheduling algorithm is _____.

MIT-WPU

।। विश्वशान्तिर्भुवं ध्रुवा ।।

| Process | Arrival Time | Burst Time | Priority |
|---|---|---|---|
| $P_1$ | 0 | 11 | 2 |
| $P_2$ | 5 | 28 | 0 |
| $P_3$ | 12 | 2 | 3 |
| $P_4$ | 2 | 10 | 1 |
| $P_5$ | 9 | 16 | 4 |

**Gantt chart:**



**Process Table:**

| Process | Arrival Time | Burst Time | Priority | Completion time (CT) | Turnaround time (TAT)<br>TAT= CT – AT | Waiting time (WT)<br>WT = TAT-BT |
|---|---|---|---|---|---|---|
| $P_1$ | 0 | 11 | 2 | 49 | 49 | 38 |
| $P_2$ | 5 | 28 | 0 | 33 | 28 | 0 |
| $P_3$ | 12 | 2 | 3 | 51 | 39 | 37 |
| $P_4$ | 2 | 10 | 1 | 40 | 38 | 28 |
| $P_5$ | 9 | 16 | 4 | 67 | 58 | 42 |

MIT-WPU

॥ विश्वशान्तिर्भुवं ध्रुवा ॥

## Example

Consider the set of processes with arrival time (in milliseconds), CPU burst time (in milliseconds) , and priority (0 is the highest priority) shown below. None of the processes have I/O burst time.

| 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 |

The average waiting time (in milliseconds) of all the processes using preemptive priority scheduling algorithm is _____.

$$\text{Average waiting time} = \frac{\sum waiting\ time\ of\ all\ the\ processes}{number\ of\ processes}$$

$$= \frac{38+0+37+28+42}{5} = 29 \text{ milliseconds}$$

Answer =29

# Round Robin (RR)

- Each process gets a small unit of CPU time (*time quantum*)
- After 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.

# RR: characteristics

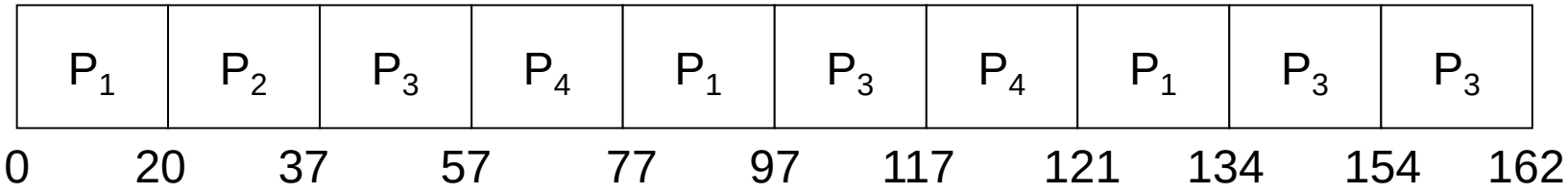
- **Selection Function**: constant
- **Decision Mode :** Preemptive ( At time quantum)
- **Throughput:** May be low if time quantum is too small
- **Response Time:** Provides good response time for short processes
- **Overhead:** Minimum
- **Effect on Processes:** Fair treatment
- **Starvation**: No

# Example of RR with Time Quantum = 20

| Process | Burst Time |
|---|---|
| *P1* | 53 |
| *P2* | 17 |
| *P3* | 68 |
| *P4* | 24 |

# Example of RR with Time Quantum = 20

| Process | Burst Time |
|---|---|
| *P1* | 53 |
| *P2* | 17 |
| *P3* | 68 |
| *P4* | 24 |




- Average waiting time = (81+ 20 + 94+ 97)/4 = 73
- Average turnaround time= (134+37+162+121)/4=113.5

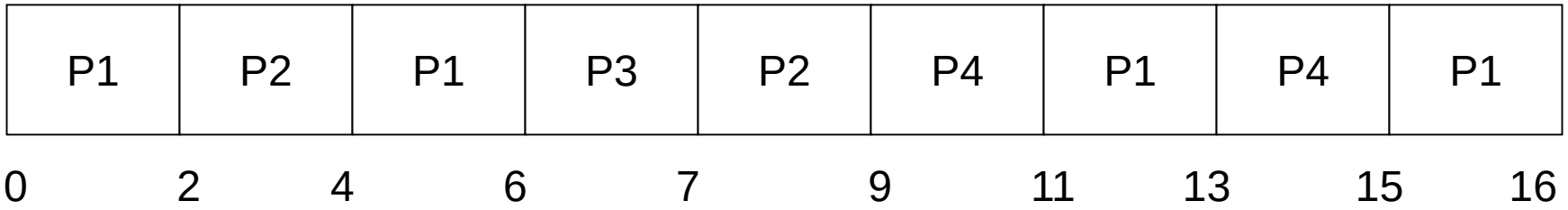
- Typically, higher average turnaround than SJF, but better *response*

# Example: Round Robin (By Default preemptive: Time Quantum 2)

| 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 |

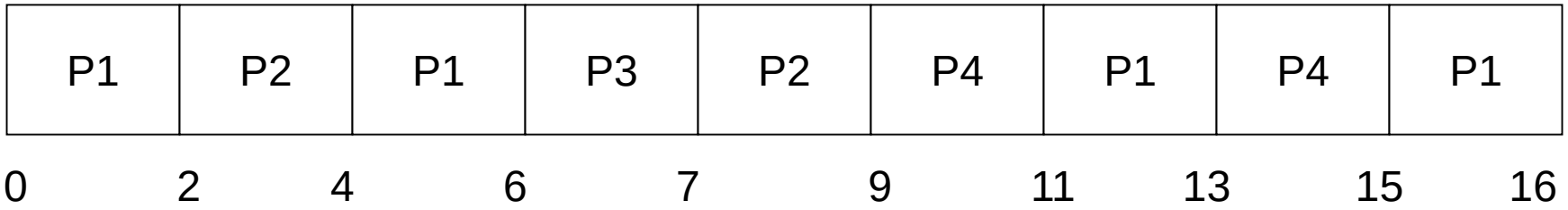
# Example: Round Robin (By Default preemptive: Time Quantum 2)

| 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 |



# Example: Round Robin (By Default preemptive: Time Quantum 2)

| 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 |




Average waiting time = (9 + 3 + 2 +6)/4 =5
Average turnaround time =(16+7+3+10)/4=9

# References

1. William Stallings, Operating System: Internals and Design Principles, Prentice Hall, ISBN-10: 0-13-380591-3, ISBN-13: 978-0-13-380591-8, 8th Edition
2. Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, Operating System Concepts, WILEY,ISBN 978-1-118-06333-0, 9th Edition