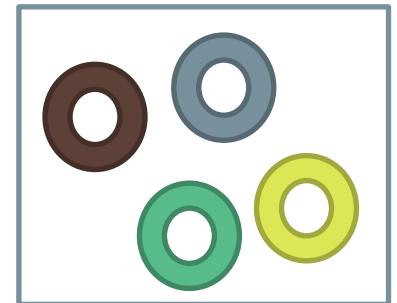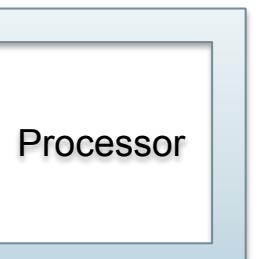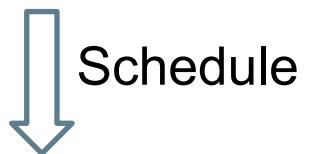OPERATING SYSTEMS
# CPU Scheduling

# Basic Concepts

- Maximum CPU utilization obtained with multiprogramming

- Continuous Cycle :
  - one process has to wait (I/O)
  - Operating system takes the CPU away
  - Give CPU to another process
  - This pattern continues

- CPU–I/O Burst Cycle : Process execution consists of a *cycle* of CPU execution and I/O wait

Processes in
Ready Queue

Schedule

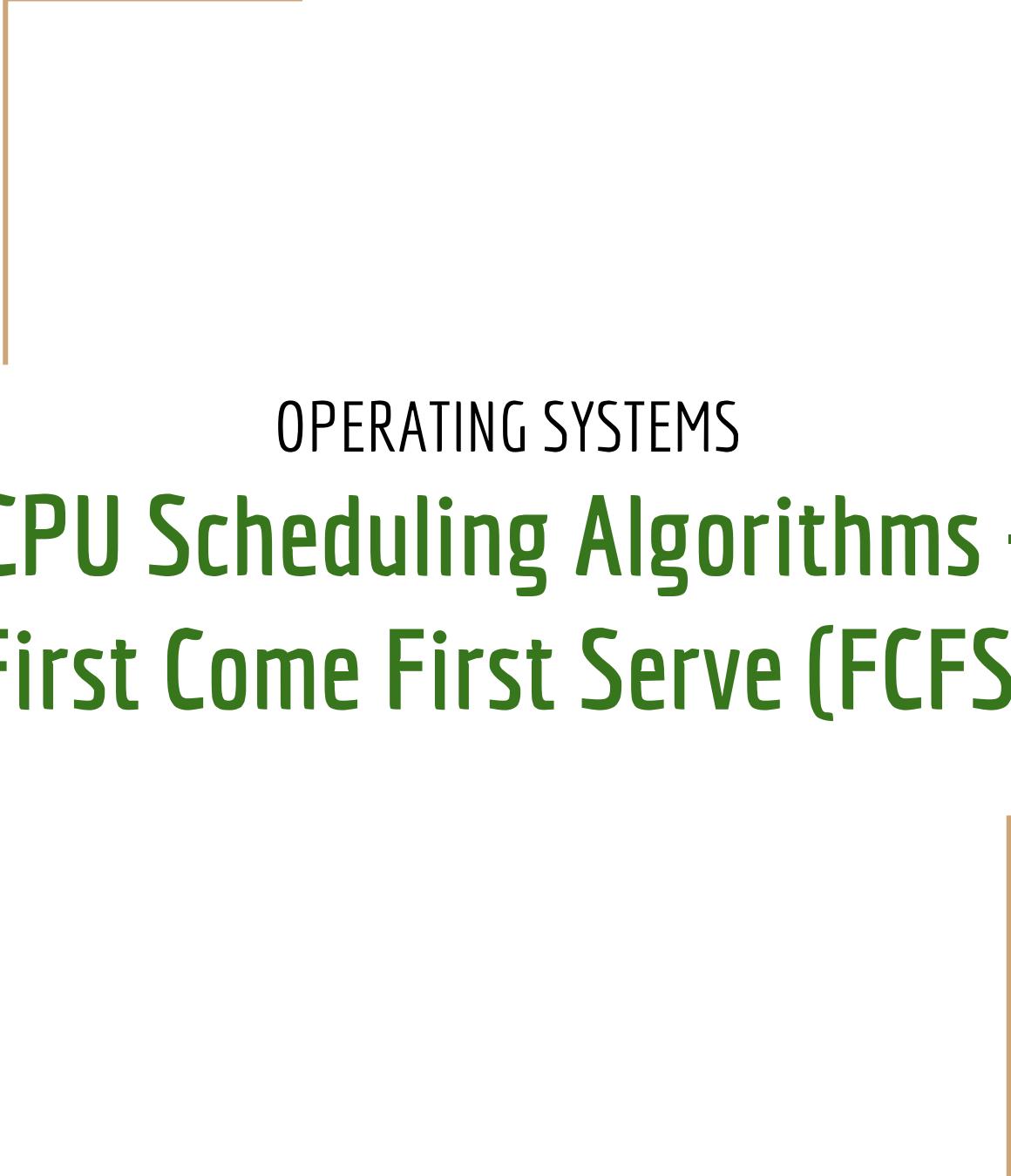Processor

# CPU Scheduler

- **Selects from among the processes in ready queue, and allocates the CPU to one of them**
  - FIFO queue
  - Priority queue
  - Tree
  - Unordered linked-list
- **CPU scheduling decisions may take place when a process:**
  1. Switches from running to waiting state (I/O request)
  2. Switches from running to ready state (e.g. when interrupt occurs)
  3. Switches from waiting to ready (e.g. at completion of I/O)
  4. Terminates
- Scheduling under 1 and 4 is **_nonpreemptive_**
- All other scheduling is **_preemptive_**
  - Consider access to shared data
  - Consider preemption while in kernel mode
  - Consider interrupts occurring during crucial OS activities

# 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
  -- the interval from the time of submission of a process to the time of the completion.
  -- sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, doing I/O

- **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)

# Scheduling Algorithm Optimization Criteria

- Max CPU utilization

- Max throughput

- Min turnaround time
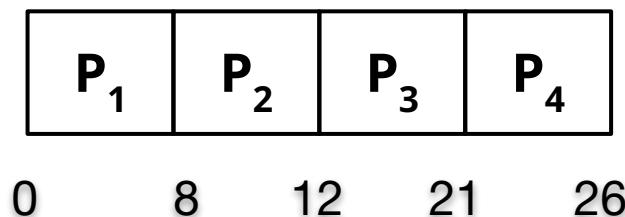
- Min waiting time

- Min response time

OPERATING SYSTEMS

# CPU Scheduling Algorithms - First Come First Serve (FCFS)
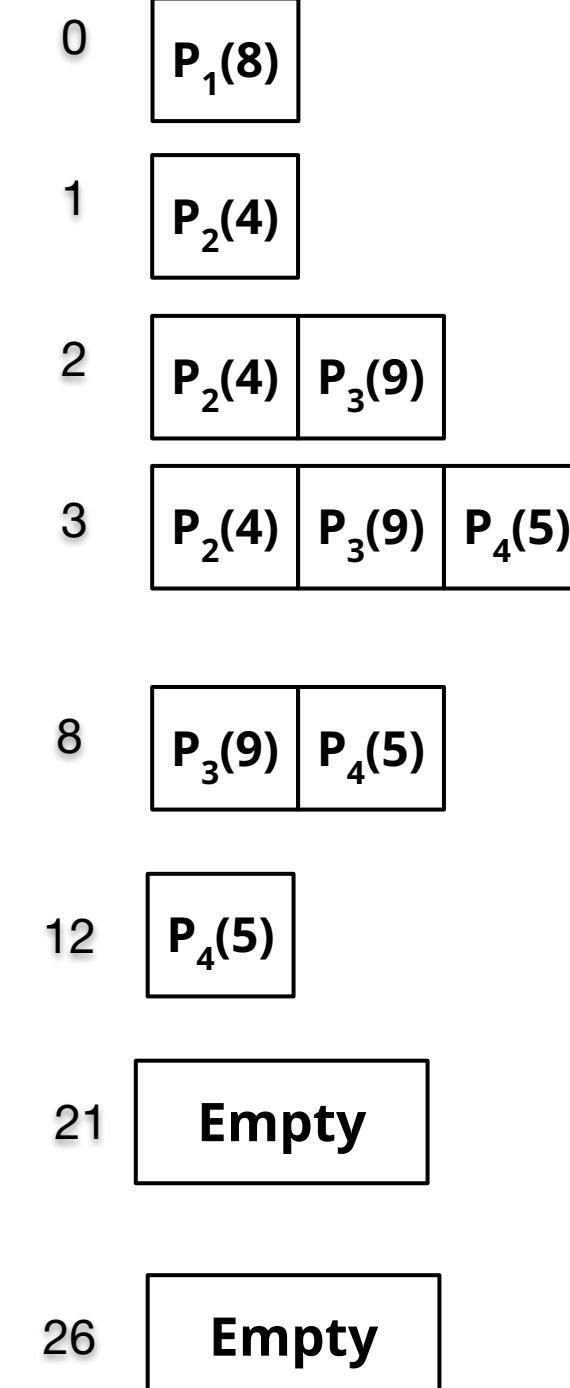
# First-Come, First-Served (FCFS) Scheduling

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

The Gantt Chart for the schedule is:

| $P_1$ | $P_2$ | $P_3$ | $P_4$ |
|-------|-------|-------|-------|

0     8    12    21    26

## Ready Queue

0   $P_1(8)$

1   $P_2(4)$

2   $P_2(4)$   $P_3(9)$

3   $P_2(4)$   $P_3(9)$   $P_4(5)$

8   $P_3(9)$   $P_4(5)$

12   $P_4(5)$

21   **Empty**

26   **Empty**

# CPU Scheduling Algorithms - Shortest Job First (SJF)

# 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:

    - **Non-preemptive** – 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 known as the Shortest-Remaining-Time-First (SRTF)

- SJF is optimal – gives minimum average waiting time for a given set of processes

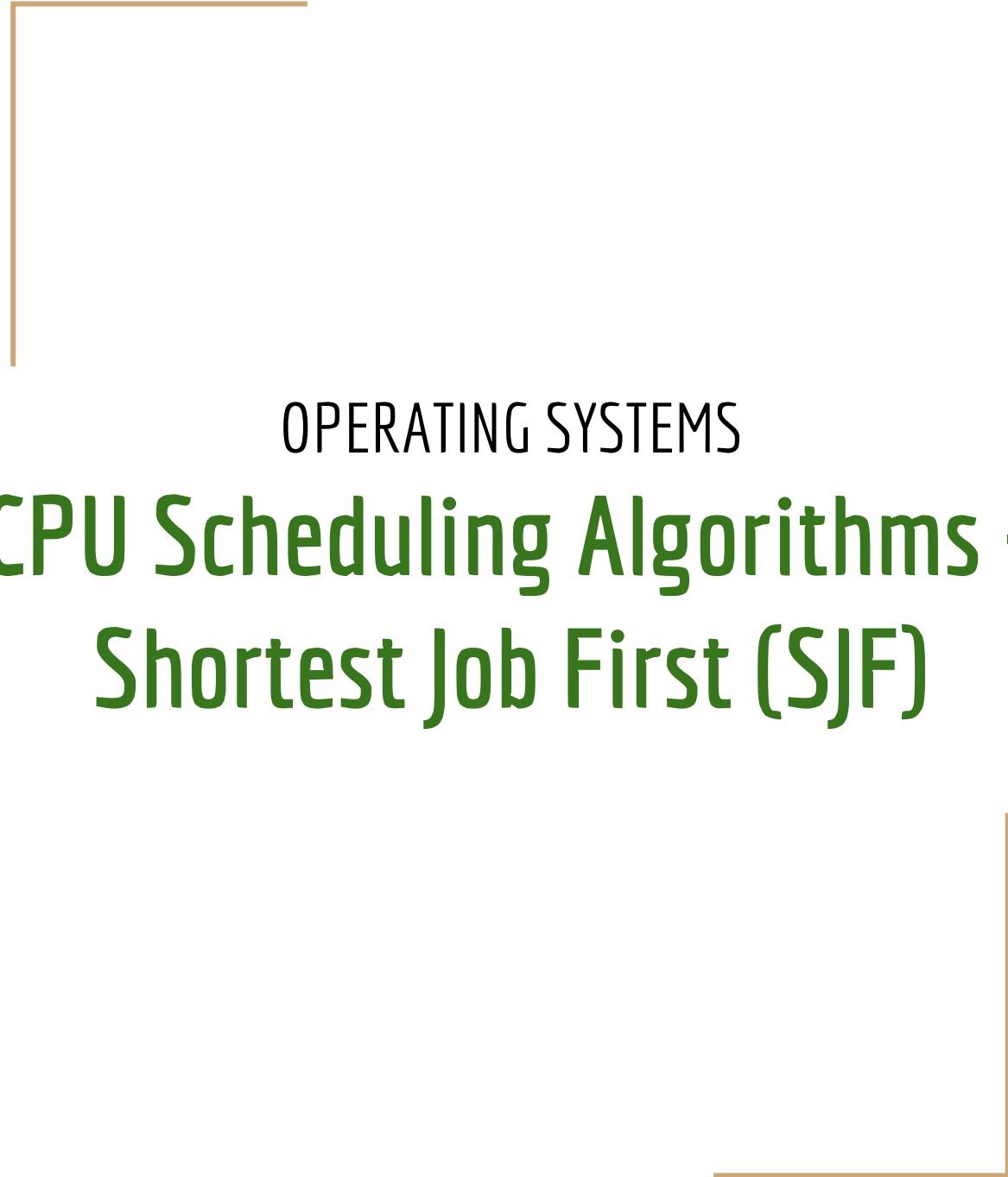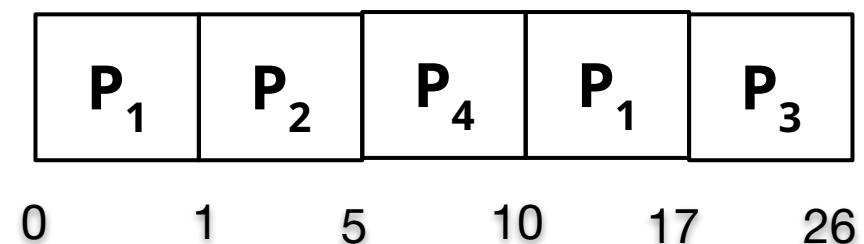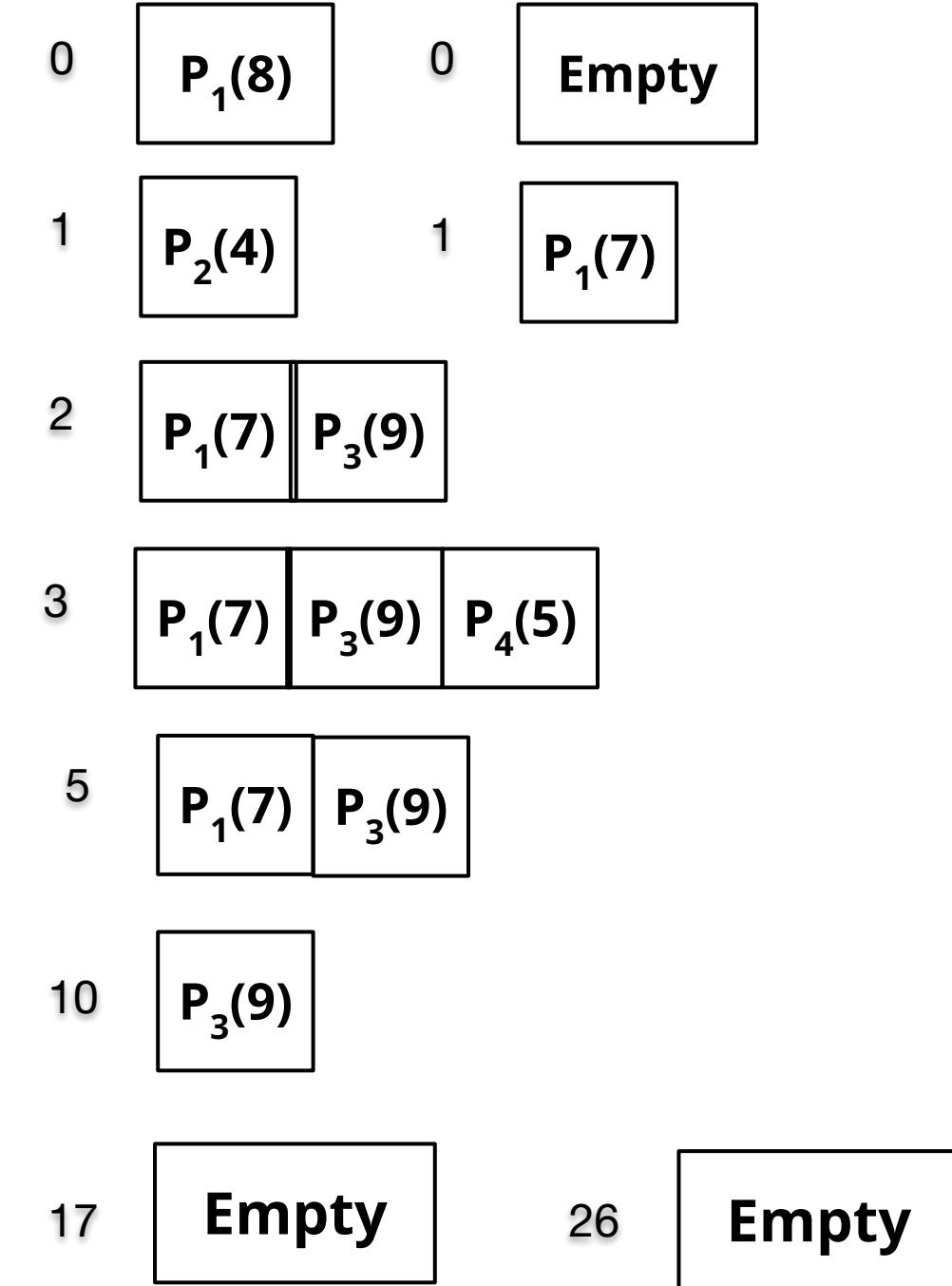# Example of Shortest-remaining-time-first
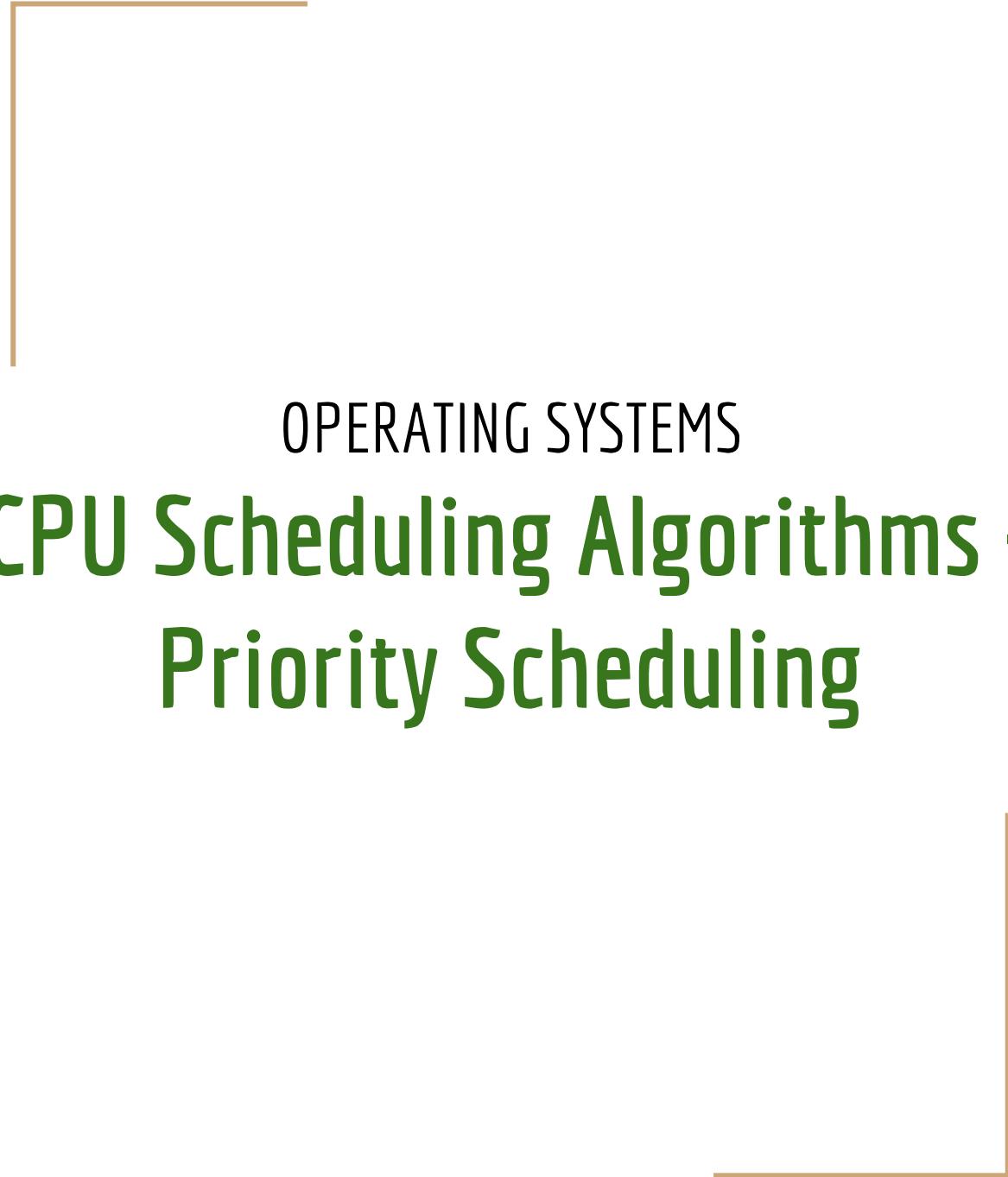
| Process | Arrival Time | Burst Time |
|---------|--------------|------------|
| $P_1$ | 0 | 8 |
| $P_2$ | 1 | 4 |
| $P_3$ | 2 | 9 |
| $P_4$ | 3 | 5 |

## Gantt Chart:

| $P_1$ | $P_2$ | $P_4$ | $P_1$ | $P_3$ |
|-------|-------|-------|-------|-------|

0    1    5    10    17    26

## Ready Queue

0  | $P_1(8)$ |     0  | Empty |

1  | $P_2(4)$ |     1  | $P_1(7)$ |

2  | $P_1(7)$ | $P_3(9)$ |

3  | $P_1(7)$ | $P_3(9)$ | $P_4(5)$ |

5  | $P_1(7)$ | $P_3(9)$ |

10  | $P_3(9)$ |

17  | Empty |     26  | Empty |

OPERATING SYSTEMS

# CPU Scheduling Algorithms - Priority Scheduling

# 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

- Priority can be defined either internally or externally.
    - Factors for internal priority assignment:
        - Time limit, memory requirements, the number or open files etc.
    - Factors for external priority assignment:
        - Importance of the process, the type and amount of funds being paid for computer use, department sponsoring works etc.

# Example of Priority Scheduling (Preemptive)
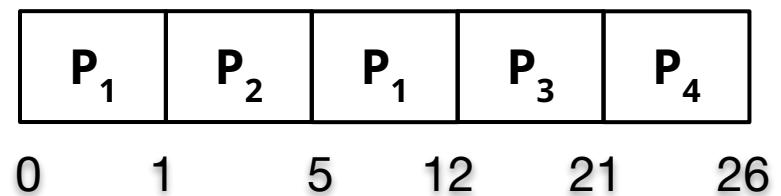
| Process | Arrival Time | Burst Time | Priority |
|---------|--------------|------------|----------|
| $P_1$ | 0 | 8 | 3 |
| $P_2$ | 1 | 4 | 1 |
| $P_3$ | 2 | 9 | 4 |
| $P_4$ | 3 | 5 | 5 |
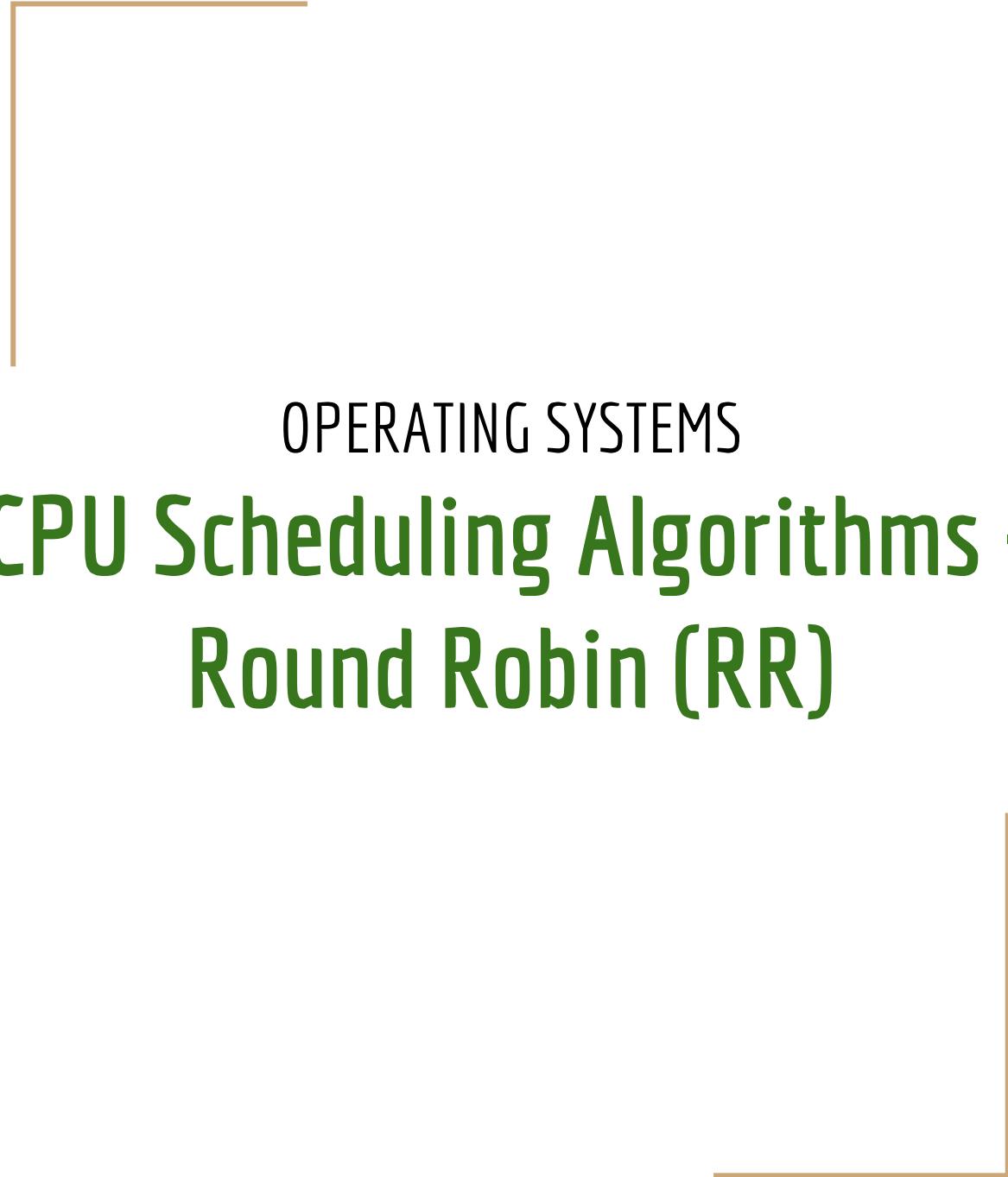
## Priority scheduling Gantt Chart

| $P_1$ | $P_2$ | $P_1$ | $P_3$ | $P_4$ |
|-------|-------|-------|-------|-------|

0   1   5   12   21   26

- Problem ≡ **Starvation** – low priority processes may never execute

- Solution ≡ **Aging** – as time progresses increase the priority of the process

## Ready Queue

0 | $P_1$(8)     0 | Empty

1 | $P_2$(4)     1 | $P_1$(7)

2 | $P_1$(7) | $P_3$(9)

3 | $P_1$(7) | $P_3$(9) | $P_4$(5)

5 | $P_3$(9) | $P_4$(5)

12 | $P_4$(5)

21 | Empty

26 | Empty

OPERATING SYSTEMS

# CPU Scheduling Algorithms - Round Robin (RR)
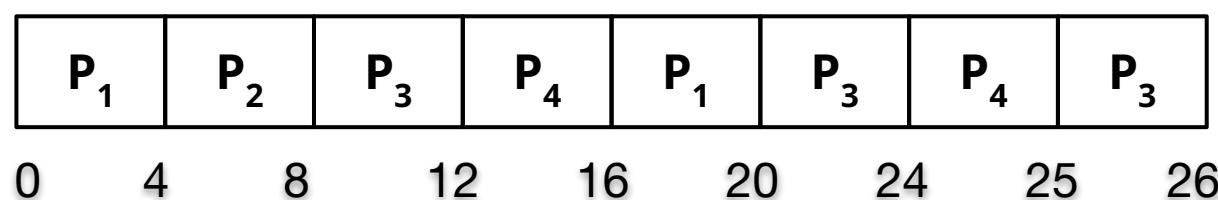
# Round Robin (RR)

- 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 $(n\text{-}1)q$ 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 | Arrival Time | Burst Time |
|---------|--------------|------------|
| $P_1$ | 0 | 8 |
| $P_2$ | 1 | 4 |
| $P_3$ | 2 | 9 |
| $P_4$ | 3 | 5 |

The Gantt chart is:

| $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_1$ | $P_3$ | $P_4$ | $P_3$ |
|-------|-------|-------|-------|-------|-------|-------|-------|

0    4    8    12    16    20    24    25    26
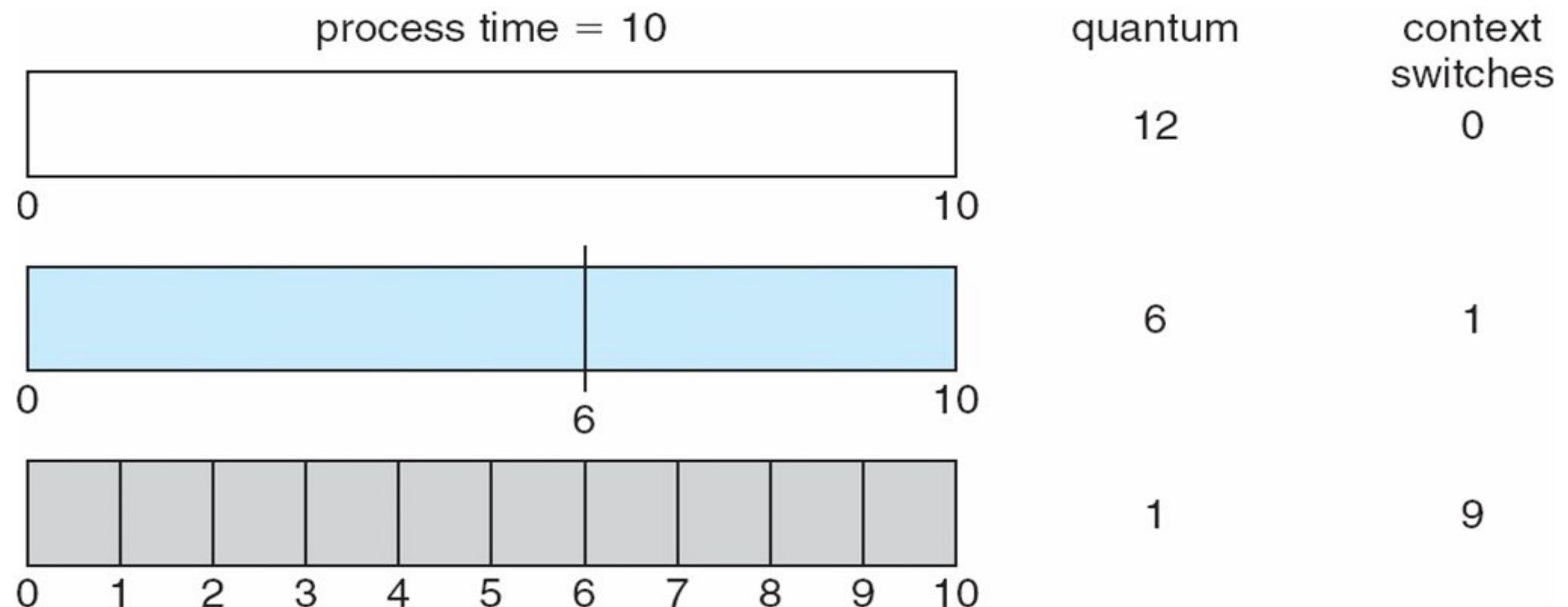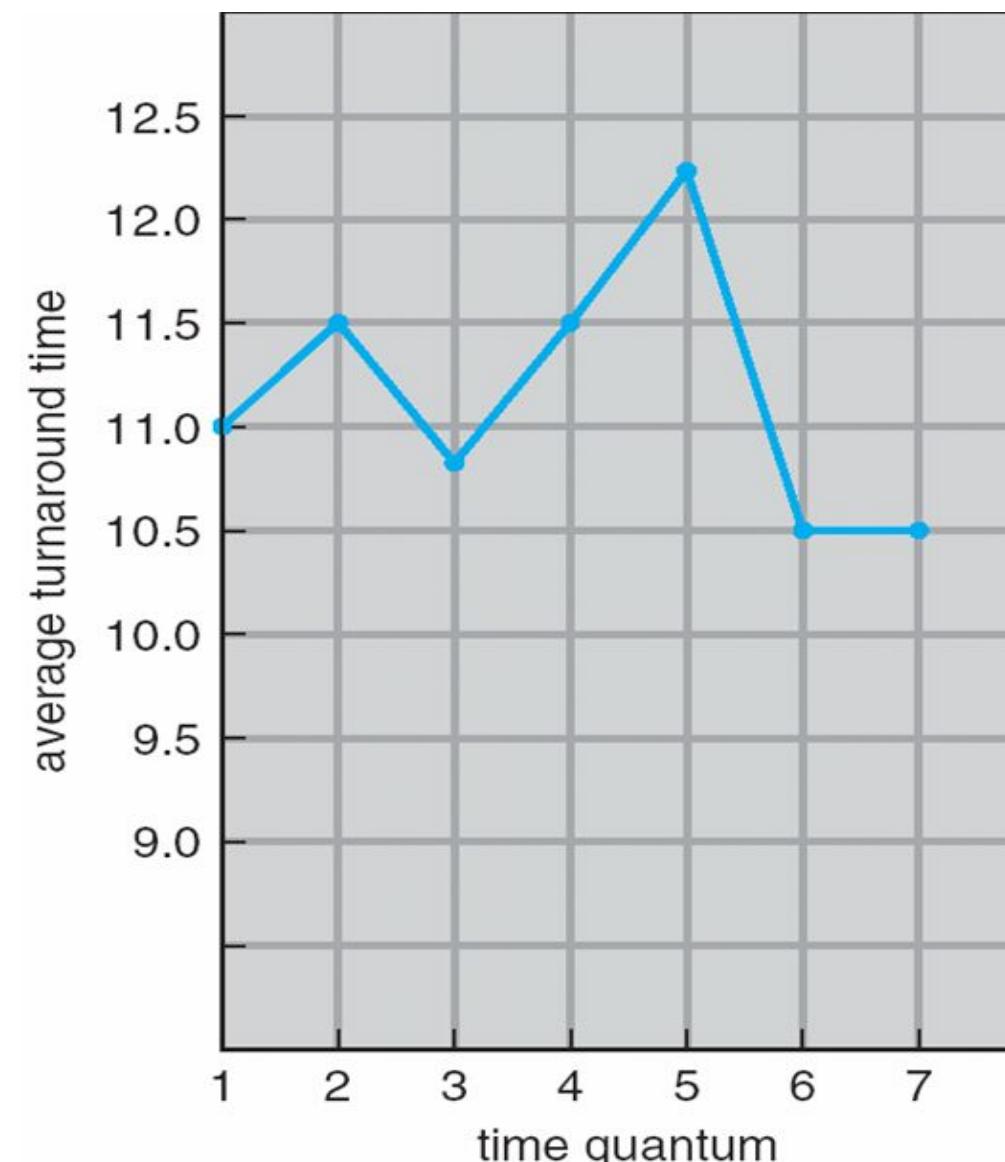
Ready Queue:

0    $P_1(8)$

1    $P_2(4)$

2    $P_2(4)$  $P_3(9)$

3    $P_2(4)$  $P_3(9)$  $P_4(5)$

4    $P_3(9)$  $P_4(5)$  $P_1(4)$

8    $P_4(5)$  $P_1(4)$

12    $P_1(4)$  $P_3(5)$

16    $P_3(5)$  $P_4(1)$

20    $P_4(1)$

24    $P_3(1)$

25    **Empty**    26    **Empty**

# 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 |

80% of CPU bursts should be shorter than quantum

OPERATING SYSTEMS

# CPU Scheduling – Multilevel Queue, Multilevel Feedback Queue

# Multilevel Queue

- Another class of scheduling algorithm needs- in which processes are classified into different groups, e.g.:
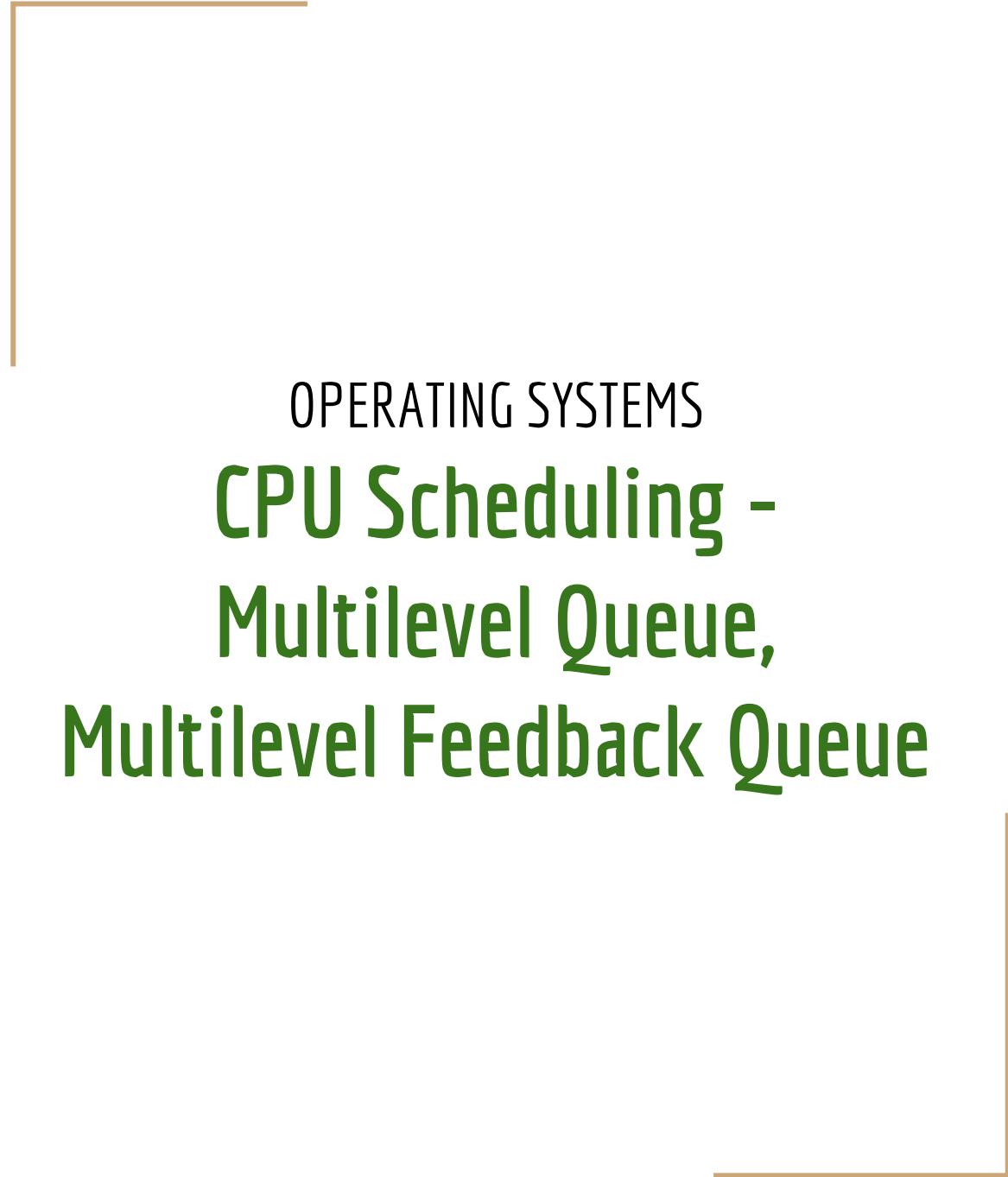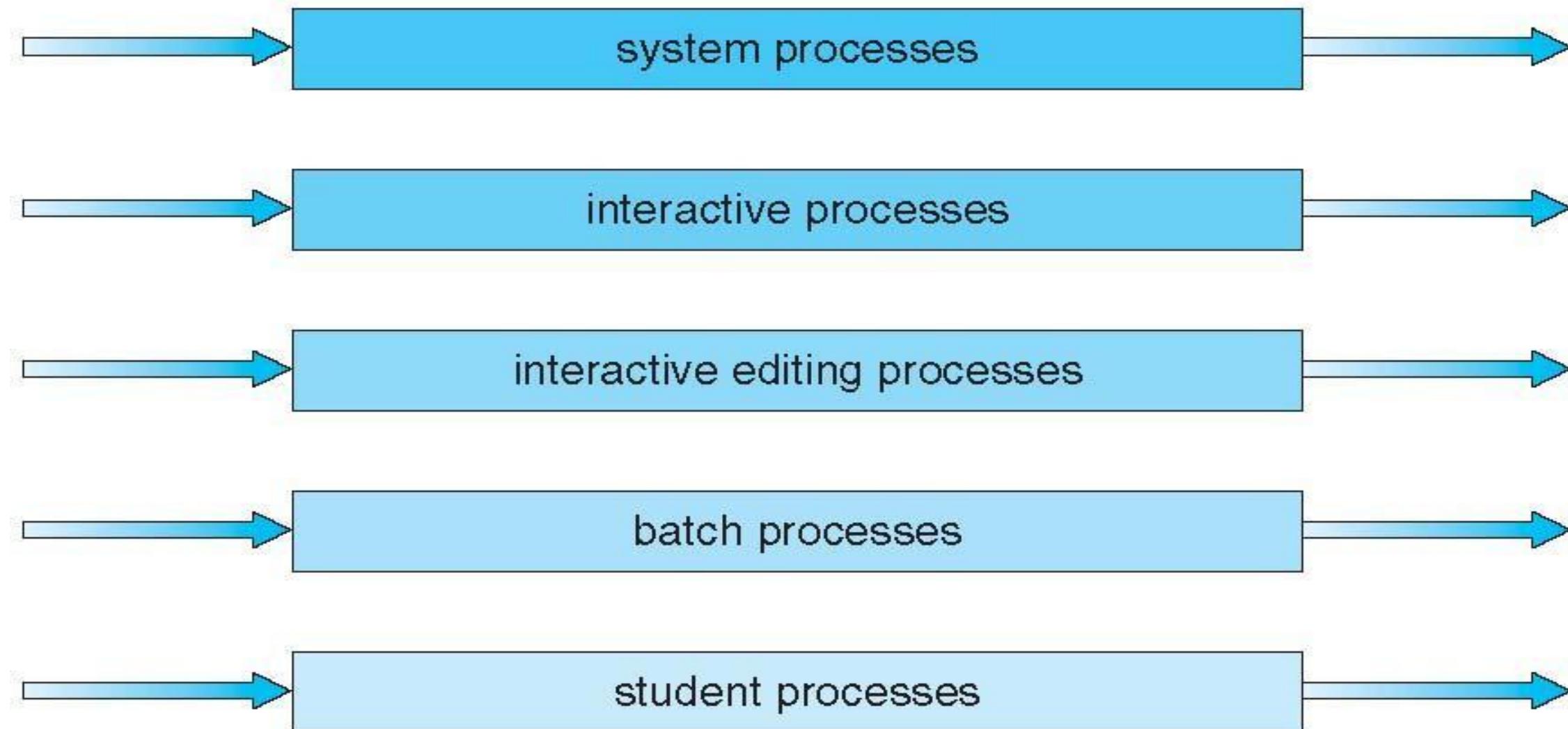    - foreground (interactive) processes
    - background (batch) processes
- They have different response time requirements-so different scheduling needs.
- Foreground processes may have priority over background processes.
- A multilevel queue-scheduling algorithm partitions the ready queue into several separate queues-we can see it in the figure of next slide:-

- Each queue has its own scheduling algorithm:
    - Foreground queue scheduled by – RR algorithm
    - Background queue scheduled by – FCFS algorithm

- Scheduling must be done between the queues:
    - Fixed priority preemptive 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., foreground queue can be given 80% of the CPU time for RR-scheduling among its processes, while 20% to background in FCFS manner.

# Multilevel Queue Scheduling

highest priority

system processes

interactive processes

interactive editing processes

batch processes

student processes

lowest priority

# Multilevel Feedback Queue scheduling

- Multilevel Feedback Queue scheduling, allows a process to move between queues.
- If a process uses too much CPU time, it will be moved to a lower priority queue.
- Similarly, a process that waits too long in a lower-priority queue may me moved to a higher-priority queue.

- 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
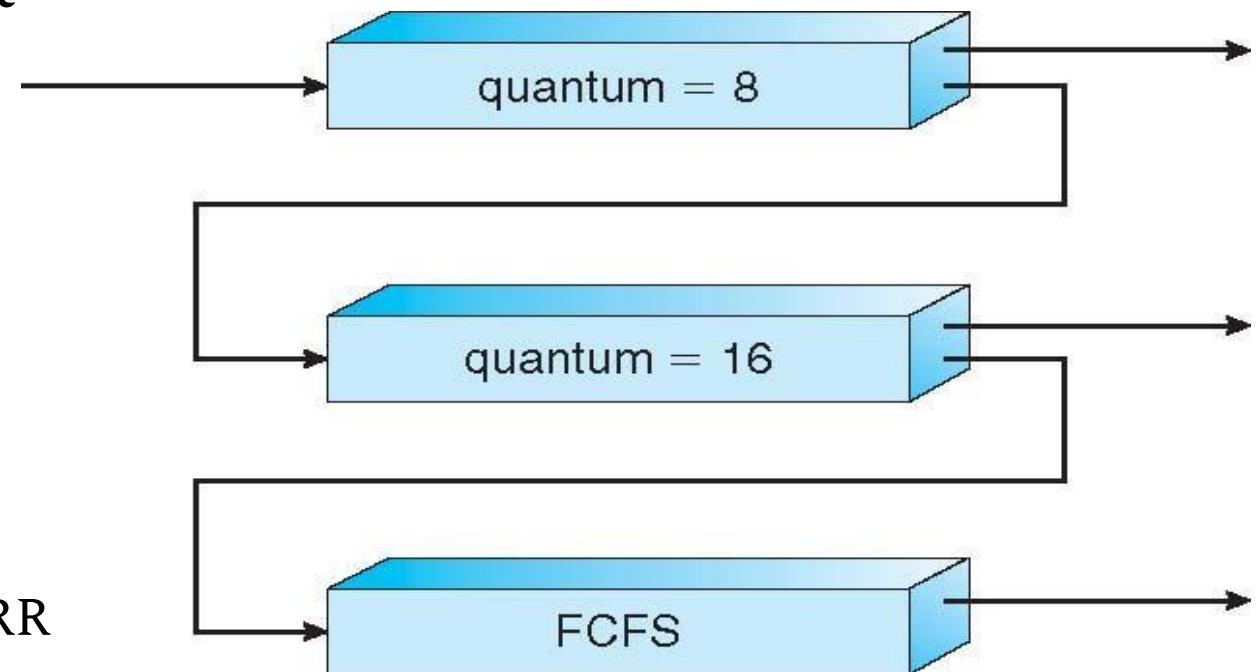
# Example of Multilevel Feedback Queue

- Three queues: (can see the figure in next slide)
  - $Q_0$ – 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 for RR
    - 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 RR and receives 16 additional milliseconds
    - If it still does not complete, it is preempted and moved to queue $Q_2$



quantum = 8

quantum = 16

FCFS

# Example of Multilevel Feedback Queue
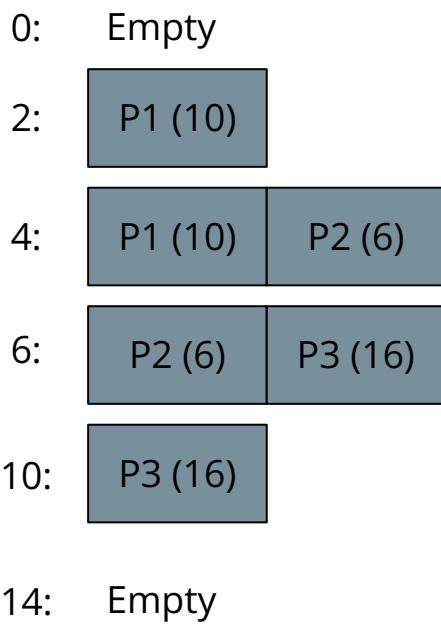
## Problem 1:

| Processes | Burst Time |
|-----------|------------|
| P1 | 12 |
| P2 | 8 |
| P3 | 18 |

Queue 0 (Priority 0): Round-robin (quantum=2)
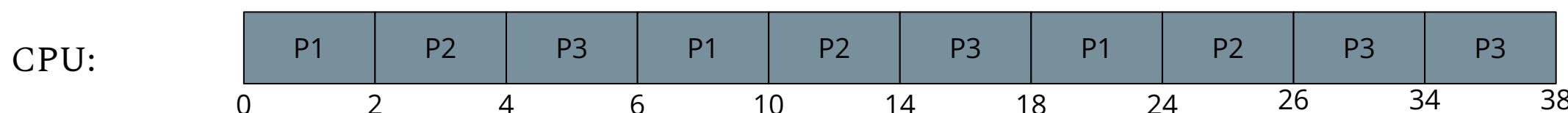
0: | P1 (12) | P2 (8) | P3 (18) |

2: | P2 (8) | P3 (18) |

4: | P3 (18) |

4: Empty

Queue 1 (Priority 1): Round-robin (quantum=4)

0: Empty

2: | P1 (10) |

4: | P1 (10) | P2 (6) |

6: | P2 (6) | P3 (16) |

10: | P3 (16) |

14: Empty

Queue 2 (Priority 2): Round-robin (quantum=8)

0: Empty

10: | P1 (6) |

14: | P1 (6) | P2 (2) |

18: | P2 (2) | P3 (12) |

24: | P3 (12) |

34: | P3 (4) |

38: Empty

Gantt Chart:

CPU: | P1 | P2 | P3 | P1 | P2 | P3 | P1 | P2 | P3 | P3 |

0   2   4   6   10   14   18   24   26   34   38

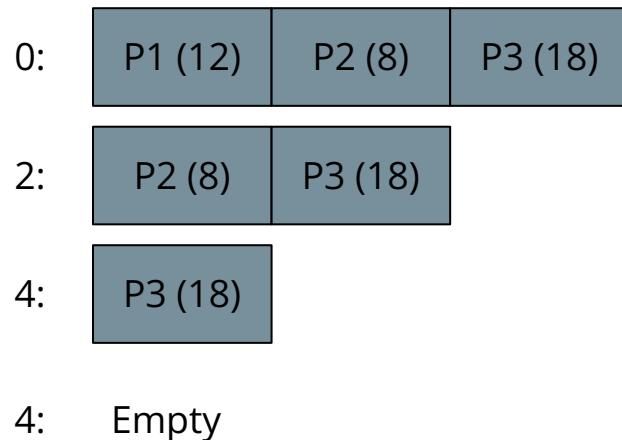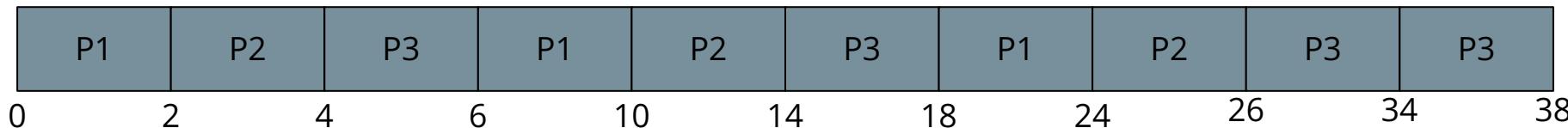# Example of Multilevel Feedback Queue

## Problem 1:

| Processes | Burst Time |
|-----------|------------|
| P1        | 12         |
| P2        | 8          |
| P3        | 18         |

Gantt Chart:

CPU:

| P1 | P2 | P3 | P1 | P2 | P3 | P1 | P2 | P3 | P3 |
|----|----|----|----|----|----|----|----|----|----|

0   2   4   6   10   14   18   24   26   34   38

**Waiting Time:**

P1=(0-0)+(6-2)+(18-10)=12

P2=(2-0)+(10-4)+(24-14)=18

P3=(4-0)+(14-6)+(26-18)+(34-34)=20

Avg=(12+18+20)/3=16.67

**Response Time:**

P1=(0-0)=0

P2=(2-0)=2

P3=(4-0)=4

Avg=(0+2+4)/3=2

**Turnaround Time:**

P1=(24-0)=24

P2=(26-0)=26

P3=(38-0)=38

Avg=(24+26+38)/3=29.33

# Example of Multilevel Feedback Queue

## Problem 2:

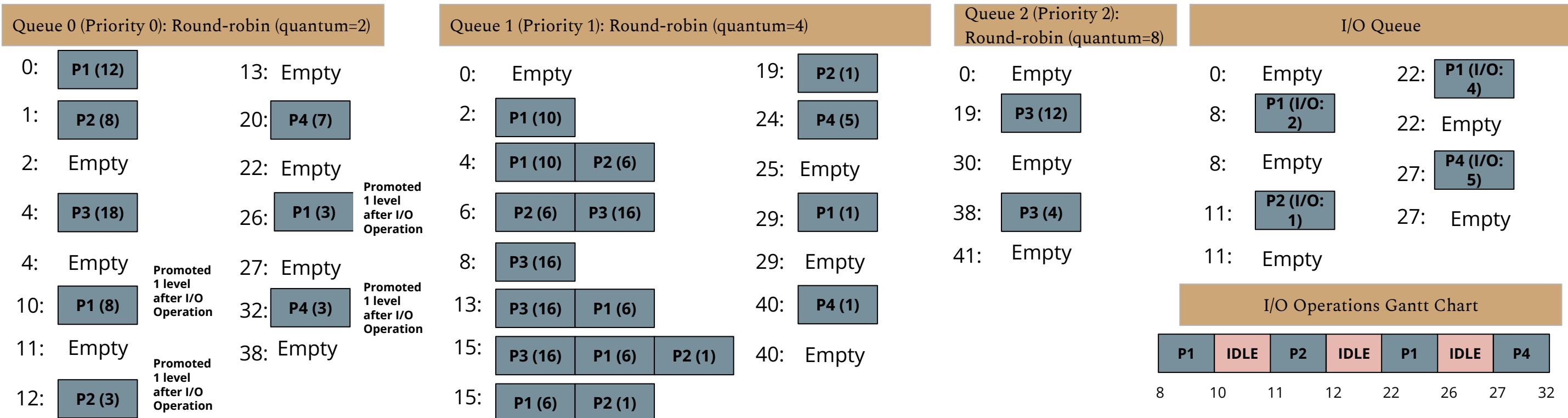| Processes | Burst Time | Arrival Time | I/O Time |
|---|---|---|---|
| P1 | 12 | 0 | 6 <br> [2s I/O operation after total 4s of CPU allocation & 4s I/O operation after total 9s of CPU allocation] |
| P2 | 8 | 1 | 1 <br> [1s I/O operation after total 5s of CPU allocation] |
| P3 | 18 | 4 | N/A |
| P4 | 7 | 20 | 5 <br> [5s I/O operation after total 4s of CPU allocation] |

### Queue 0 (Priority 0): Round-robin (quantum=2)

| | | | |
|---|---|---|---|
| 0: | P1 (12) | 13: | Empty |
| 1: | P2 (8) | 20: | P4 (7) |
| 2: | Empty | 22: | Empty |
| 4: | P3 (18) | 26: | P1 (3) — Promoted 1 level after I/O Operation |
| 4: | Empty — Promoted 1 level after I/O Operation | 27: | Empty |
| 10: | P1 (8) | 32: | P4 (3) — Promoted 1 level after I/O Operation |
| 11: | Empty | 38: | Empty |
| 12: | P2 (3) — Promoted 1 level after I/O Operation | | |

### Queue 1 (Priority 1): Round-robin (quantum=4)

| | | | |
|---|---|---|---|
| 0: | Empty | 19: | P2 (1) |
| 2: | P1 (10) | 24: | P4 (5) |
| 4: | P1 (10)  P2 (6) | 25: | Empty |
| 6: | P2 (6)  P3 (16) | 29: | P1 (1) |
| 8: | P3 (16) | 29: | Empty |
| 13: | P3 (16)  P1 (6) | 40: | P4 (1) |
| 15: | P3 (16)  P1 (6)  P2 (1) | 40: | Empty |
| 15: | P1 (6)  P2 (1) | | |

### Queue 2 (Priority 2): Round-robin (quantum=8)

| | |
|---|---|
| 0: | Empty |
| 19: | P3 (12) |
| 30: | Empty |
| 38: | P3 (4) |
| 41: | Empty |

### I/O Queue

| | | | |
|---|---|---|---|
| 0: | Empty | 22: | P1 (I/O: 4) |
| 8: | P1 (I/O: 2) | 22: | Empty |
| 8: | Empty | 27: | P4 (I/O: 5) |
| 11: | P2 (I/O: 1) | 27: | Empty |
| 11: | Empty | | |

### I/O Operations Gantt Chart

| P1 | IDLE | P2 | IDLE | P1 | IDLE | P4 |
|---|---|---|---|---|---|---|
| 8 | 10 | 11 | 12 | 22 | 26 | 27 ... 32 |

## Gantt Chart:

I/O blocks: I/O   I/O (at ~7-11)    I/O (at ~22)    I/O (at ~27)

CPU: P1 | P2 | P3 | P1 | P2 | P1 | P2 | P3 | P1 | P4 | P2 | P4 | P1 | P1 | P3 | P4 | P4 | P3

Times: 0  2  4  6  8  11  13  15  19  22  24  25  27  29  30  38  40  41  45

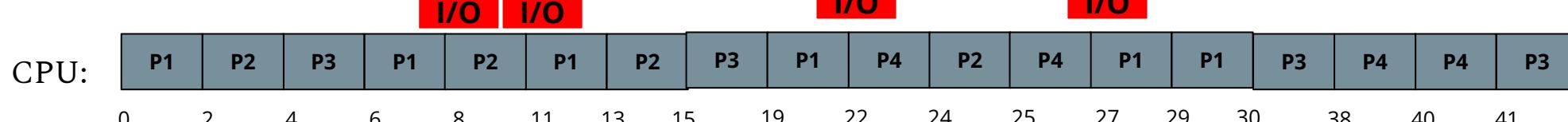# Example of Multilevel Feedback Queue

## Problem 2:

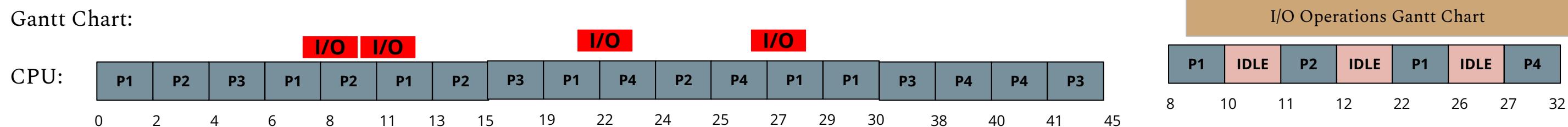| Processes | Burst Time | Arrival Time | I/O Time |
|-----------|------------|--------------|----------|
| P1 | 12 | 0 | 6<br>[2s I/O operation after total 4s of CPU allocation & 4s I/O operation after total 9s of CPU allocation] |
| P2 | 8 | 1 | 1<br>[1s I/O operation after total 5s of CPU allocation] |
| P3 | 18 | 4 | N/A |
| P4 | 7 | 20 | 5<br>[5s I/O operation after total 4s of CPU allocation] |

Gantt Chart:



I/O Operations Gantt Chart

Waiting Time:

P1=(0-0)+(6-2)+(11-10)+(19-13)+(27-26)+(29-29)
    =12

P2=(2-1)+(8-4)+(13-12)+(24-15)=15

P3=(4-4)+(15-6)+(30-19)+(41-38)=23

P4=(22-20)+(25-24)+(38-32)+(40-40)=9

Avg=(12+15+23+9)/4=14.75

Response Time:

P1=(0-0)=0

P2=(2-1)=1

P3=(4-4)=0

P4=(22-20)=2

Avg=(0+1+0+2)/4=0.75

Turnaround Time:

P1=(30-0)=30

P2=(25-1)=24

P3=(45-4)=41

P4=(41-20)=21

Avg=(30+24+41+21)/4=29