

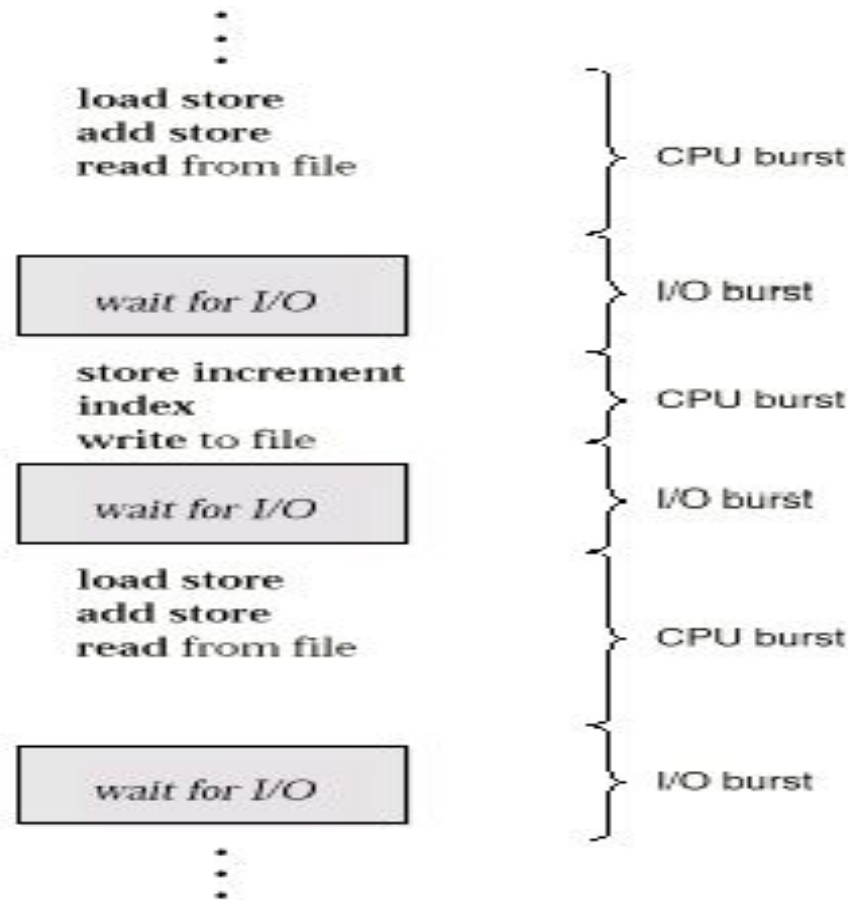
# Operating Systems

## CPU Scheduling Algorithms

-

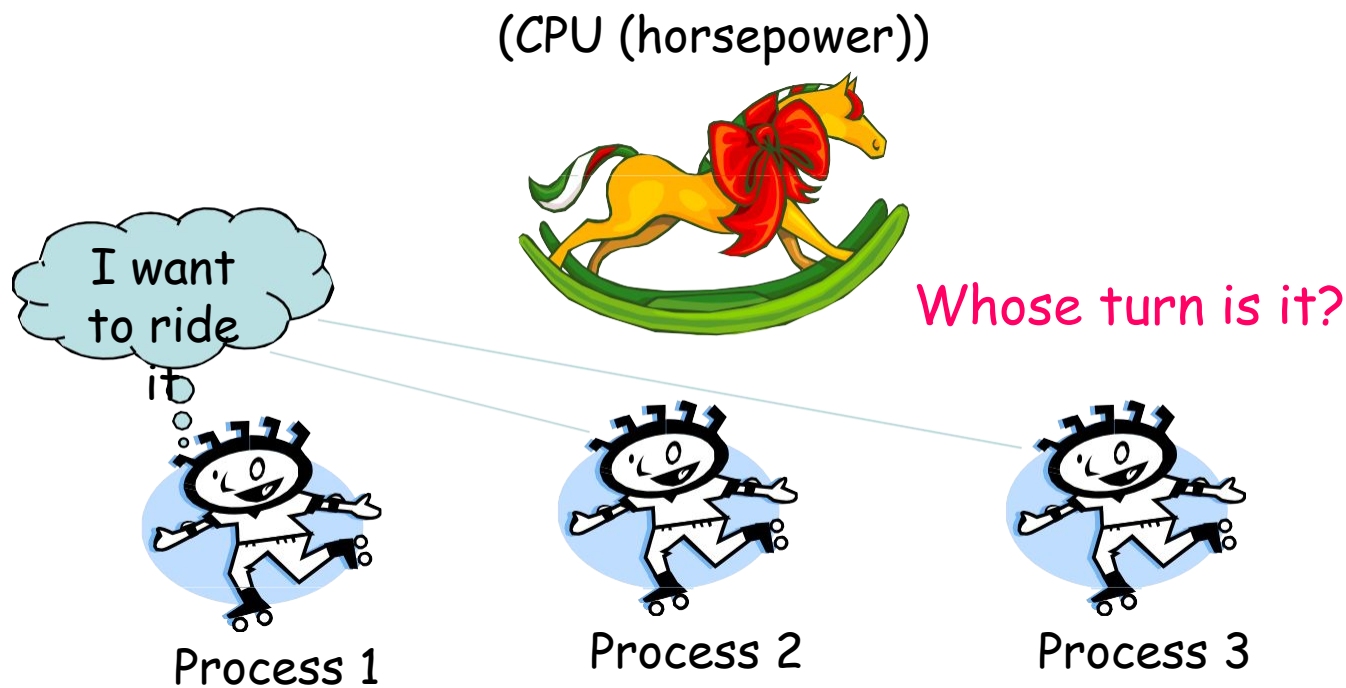
# CPU – I/O Burst Cycle

- Process execution consists of a cycle of CPU execution and I/O wait
  - Processes move back & forth between these two states
- A process execution begins with a CPU burst, followed by an I/O burst and then another CPU burst and so on



# Scheduling

- Deciding which process/thread should occupy a resource (CPU, disk, etc.)



# CPU Scheduler

- When CPU becomes idle
  - OS must select one of the processes from the Ready Queue to be executed.
- **CPU scheduler**
  - is the OS code that implements the CPU scheduling algorithm .
  - selects a process from the processes in memory that are ready to execute (from Ready Queue), and allocates the CPU to one of them.
- CPU scheduling decisions may take place when a process:
  1. Switches **from running to waiting** state
  2. Switches **from running to ready** state (e.g. when time slice of a process expires or an interrupt occurs)
  3. Switches from **waiting to ready** state. (e.g. on completion of I/O)
  4. **Terminates**

# Preemptive vs Non Preemptive Process

## Non-preemptive

### ✓ Process runs until voluntarily relinquish CPU

- process blocks on an event (e.g., I/O)
- process terminates
- process periodically calls the yield() system call to give up the CPU

### ✓ Only suitable for domains where processes can be trusted to voluntarily relinquish the CPU

## Preemptive

- The scheduler actively interrupts and reschedules an executing process
- Special device requires for creating interrupt
- Required when applications cannot be trusted to yield
- Incurs some overhead

# Scheduling Criteria

- **CPU utilization** – percentage of time the CPU is not idle.
- **Throughput** – completed processes per unit of time  
[the amount of material or items passing through a system or process]
- **Waiting time** – time spent in the ready queue
  - For Non preemptive Algos =  $\text{Starting.Time} - \text{Arrival.Time}$
  - For Preemptive Algos =  $\text{Finish.Time} - \text{Arrival.Time} - \text{Burst.Time}$
- **Turnaround time** – amount of time to execute a particular process.
  - $\text{Finish.Time} - \text{Arrival.Time}$
- **Response time** – response latency
  - amount of time it takes from when a request was submitted until the first response is produced, **not** output (for time-sharing environment)

# The Perfect Scheduler

- Max CPU utilization= keep all devices busy
- Max throughput =Maximize jobs/time
- Min turnaround time
- Min waiting time
- Min response time =latency, first response
- Fairness: everyone makes progress, no one starves

# Single CPU–Scheduling Algorithms

- First Come First Serve (FCFS)
- Smallest Next CPU Burst
  - Shortest Job First (SJF)
  - Shortest Remaining Time First (SRTF)
- Round Robin
- Priority Scheduling



# First Come First Serve

- Simplest CPU scheduling algorithm
- Non preemptive
- The process that requests the CPU first is allocated the CPU first
- Implemented with a FIFO queue.
- **Limitations**
  - FCFS **favor long processes as compared to short ones.**  
(Convoy effect)
  - **Average waiting time depends on arrival order**
  - Average waiting time is often quite long
  - FCFS is non-preemptive, so it is trouble some for time sharing systems

# Convoy Effect

“A convoy effect happens when a set of processes need to use a resource for a short time, and one process holds the resource for a long time, blocking all of the other processes. Causes poor utilization of the other resources in the system”

# FCFS – Example

The process that enters the ready queue first is scheduled first, regardless of the size of its next CPU burst

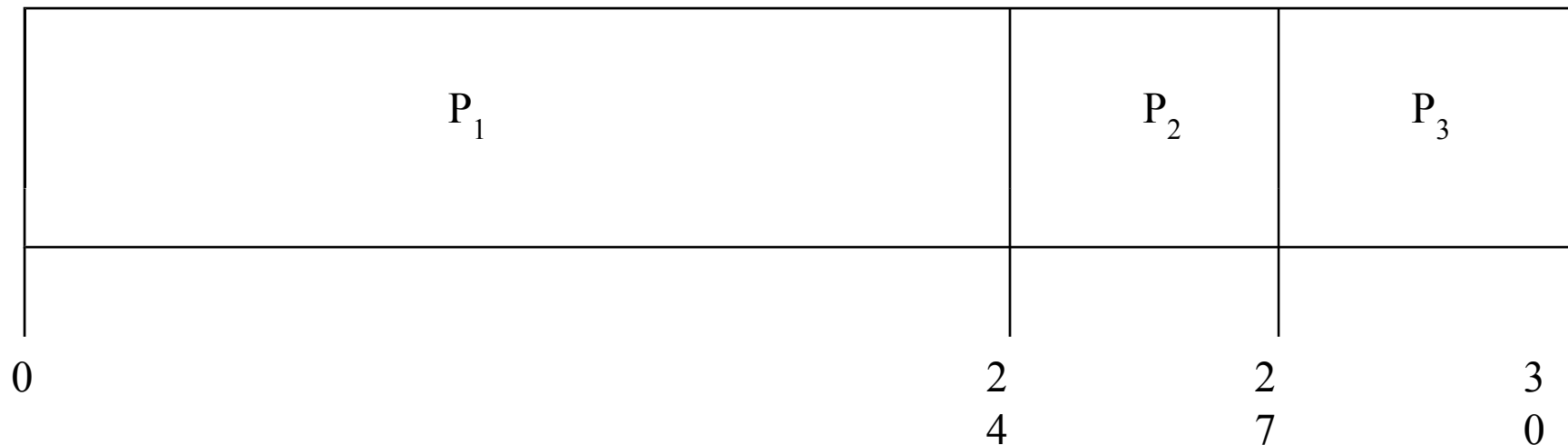
Example	Processes	Burst Time
:		24
	P	3
	2	3
	P	

Suppose that processes arrive into the system in the order: P<sub>1</sub>, P<sub>2</sub>, P<sub>3</sub>

# FCFS – Example

Processes are served in the order: P<sub>1</sub>, P<sub>2</sub>, P<sub>3</sub>

The **Gantt Chart** for the schedule is:

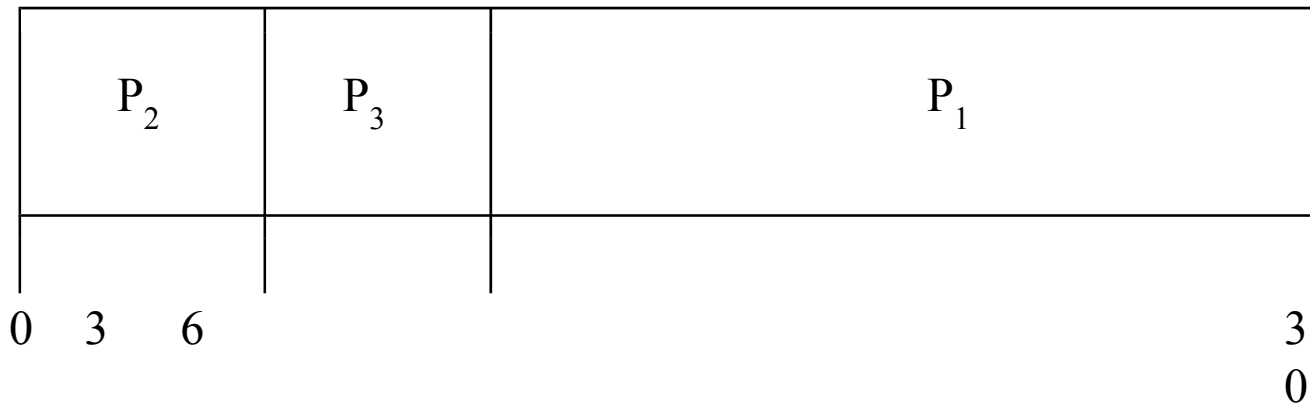


Waiting times P<sub>1</sub> = 0; P<sub>2</sub> = 2; P<sub>3</sub> = 4

Average waiting time:  $(0+2+4)/3 = 2$

## FCFS – Example

Suppose that processes arrive in the order: P<sub>2</sub> , P<sub>3</sub> , P<sub>1</sub> . The Gantt chart for the schedule is:



Waiting time for P<sub>1</sub> = 6; P<sub>2</sub> = 0; P<sub>3</sub> = 3  
Average waiting time:  $(6 + 0 + 3)/3 = 3$

# FCFS – Example

- Draw the graph (Gantt chart) and compute average waiting time for the following processes using **FCFS** Scheduling algorithm.

<u>Process</u>	<u>Arrival time</u>	<u>Burst Time</u>
P1	1	16
P2	5	3
P3	6	4
P4	9	2

# 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 three flavors

- **Shortest Job First (SJF)**
  - It's a non preemptive algorithm.
- **Shortest Remaining Time First (SRTF)**
  - It's a Preemptive algorithm.

# SJF Example

Process	Duration/B.T	Order	Arrival Time
P1	6	1	0
P2	8	2	0
P3	7	3	0
P4	3	4	0



P4 waiting time: 0-0  
P1 waiting time: 3-0  
P3 waiting time: 9-0  
P2 waiting time: 16-0

The total running time is: 28  
The average waiting time (AWT):  
 $(0+3+9+16)/4 = 7$  time units



# SRTF Example

Process	Duration	Order	Arrival Time
P1	10	1	0
P2	2	2	2



P1 waiting time:  $12 - 0 - 10 = 2$   
P2 waiting time:  $4 - 2 - 2 = 0$

The average waiting time (AWT):  
 $(0 + 2) / 2 = 1$

Now run this using SJF!

# SJF & SRTF – Example

Draw the graph (Gantt chart) and compute waiting time and turn around time for the following processes using **SJF** & **SRTF** Scheduling algorithm.

For SJF consider all processes arrive at time 0 in sequence P1, P2, P3,

<u>Process</u>	<u>Arrival Time</u>	<u>Burst Time</u>
P1	0	8
P2	1	4
P3	2	9
P4	3	5

# **SJF & SRTF – Example**

Draw the graph (Gantt chart) and compute waiting time and turn around time for the following processes using **SJF** & **SRTF** Scheduling algorithm.

<u><b>Process</b></u>	<u><b>Arrival Time</b></u>	<u><b>Burst Time</b></u>
P1	0	5
P2	1	2
P3	2	3
P4	3	1

# SJF & SRTF – Example

Draw the graph (Gantt chart) and compute waiting time and turn around time for the following processes using **SJF** & **SRTF** Scheduling algorithm.

<u>Process</u>	<u>Arrival Time</u>	<u>Burst Time</u>
P1	0	9
P2	3	6
P3	6	2
P4	9	1

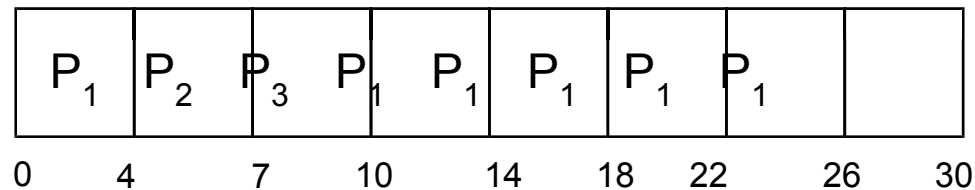
# Round Robin (RR)

- 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

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

- 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

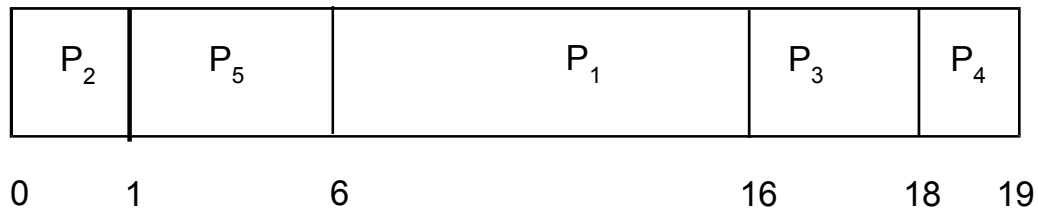
# Priority Scheduling

- 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)
  - Preemptive
  - Nonpreemptive
- SJF is priority scheduling
  - where priority is the inverse of predicted next CPU burst time
  - > Shortest Job, Highest Priority
- Problem  $\equiv$  Starvation – low priority processes may never execute
- Solution  $\equiv$  Aging – as time progresses increase the priority of the process

# Example of Priority Scheduling

<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



- Non-Preemptive Waiting time =  $0 + 1 + 6 + 16 + 18 = 41$
- Average waiting time = 8.2 msec



# Course Materials

## @Galvin-5.1-5.3