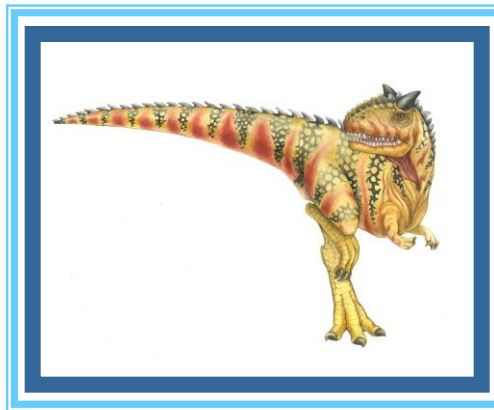
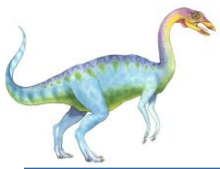


Chapter 6: CPU Scheduling





Chapter 6: CPU Scheduling

- ❑ Basic Concepts
- ❑ Scheduling Criteria
- ❑ Scheduling Algorithms
- ❑ Algorithm Evaluation





Objectives

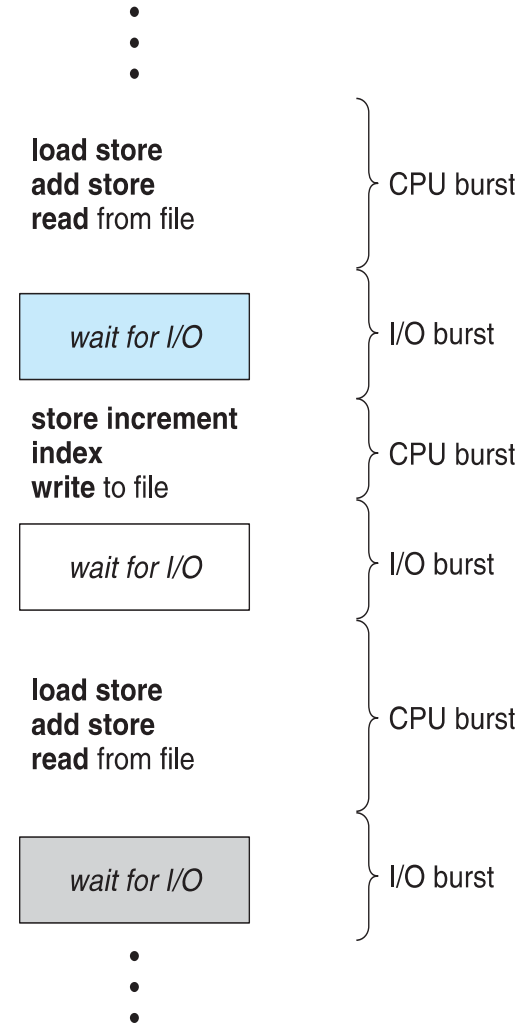
- To introduce CPU scheduling, which is the basis for multiprogrammed operating systems
- To describe various CPU-scheduling algorithms
- To discuss evaluation criteria for selecting a CPU-scheduling algorithm for a particular system





Basic Concepts

- ❑ Maximum CPU utilization obtained with multiprogramming
- ❑ CPU–I/O Burst Cycle – Process execution consists of a **cycle** of CPU execution and I/O wait
- ❑ **CPU burst** followed by **I/O burst**
- ❑ CPU burst distribution is of main concern





CPU Scheduler

- **Short-term scheduler** selects from among the processes in 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
 3. Switches from waiting to ready
 4. Terminates
- The scheduling algorithms can be divided into two categories with respect to how they deal with clock interrupts:
 - **Non-preemptive**:: A scheduling discipline is non-preemptive if, once a process has been given the CPU, the CPU cannot be taken away from that process
 - **Preemptive**: A scheduling discipline is preemptive if, once a process has been given the CPU, the CPU can be taken away from that process.

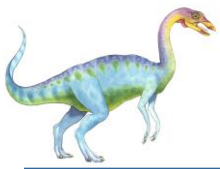




Scheduling Criteria

- ❑ **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
- ❑ **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

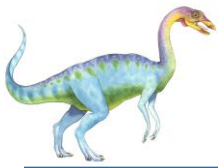




CPU Scheduling Algorithms

- CPU scheduling deals with the problem of deciding which of the processes in the ready queue is to be allocated the CPU. There are many different CPU scheduling algorithms. This section describes several of them.
 - First-Come, First-Served (FCFS)
 - Shortest-Job-First (SJF)
 - Priority Scheduling
 - Round Robin (RR)





First- Come, First-Served (FCFS) Scheduling

- ❑ Other names of this algorithm are:
 - ❑ First-In-First-Out (FIFO)
 - ❑ Run-to-Completion
- ❑ With this scheme, the process that requests the CPU first is allocated the CPU first. The implementation of the FCFS policy is easily managed with FIFO queue. When a process enters the ready queue, its PCB is linked onto the tail of the queue. When the CPU is free, it is allocated to the process at the head of the queue.
- ❑ FCFS algorithm is the simplest scheduling algorithm.
- ❑ Being a non-preemptive discipline, once a process has a CPU, it runs to completion.





FCFS Scheduling Example

<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: $(0 + 24 + 27)/3 = 17$





FCFS Scheduling (Cont.)

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
- **Convoy effect** - short process behind long process.

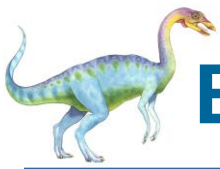




Shortest-Job-First (SJF) Scheduling

- 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
- The difficulty is knowing the length of the next CPU request

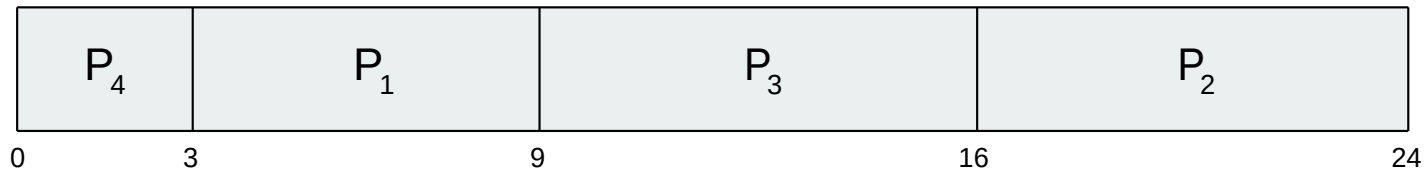




Example-1 of Non-Preemptive SJF

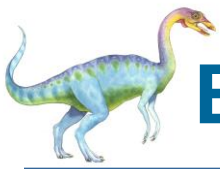
<u>Process</u>	<u>Arrival Time</u>	<u>Burst Time</u>
<i>P1</i>	0	6
<i>P2</i>	0	8
<i>P3</i>	0	7
<i>P4</i>	0	3

□ SJF scheduling chart



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





Example-2 of Non-Preemptive SJF

<u>Process</u>	<u>Arrival Time</u>	<u>Burst Time</u>
<i>P1</i>	0	7
<i>P2</i>	2	4
<i>P3</i>	4	1
<i>P4</i>	5	4

□ SJF scheduling chart



□ Average waiting time = $(0 + 6 + 3 + 7) / 4 = 4$



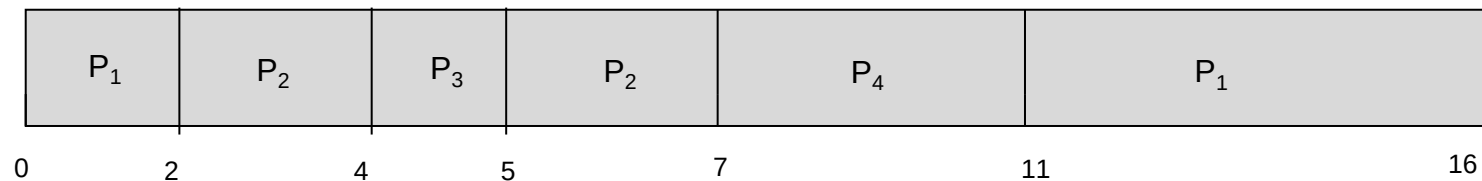


Example-3 of Preemptive SJF

- shortest-remaining-time-first (SRTF)

<u>Process</u>	<u>Arrival Time</u>	<u>Burst Time</u>
<i>P1</i>	0	7
<i>P2</i>	2	4
<i>P3</i>	4	1
<i>P4</i>	5	4

□ SJF scheduling chart



□ Average waiting time = $(9 + 1 + 0 + 2) / 4 = 3$



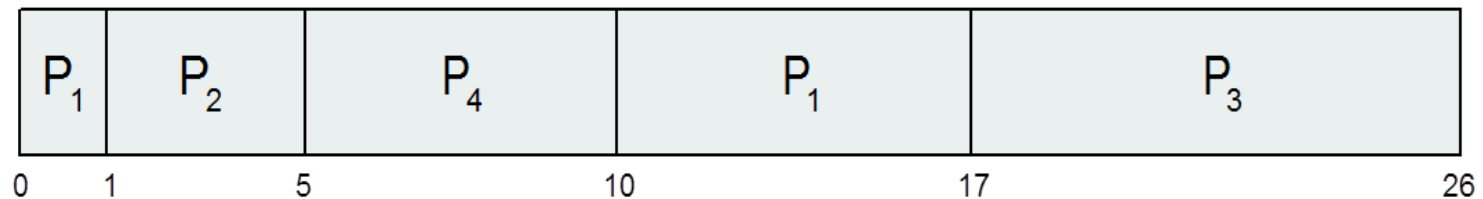


Example-4 of Preemptive SJF

- shortest-remaining-time-first (SRTF)

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

- SJF scheduling chart



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





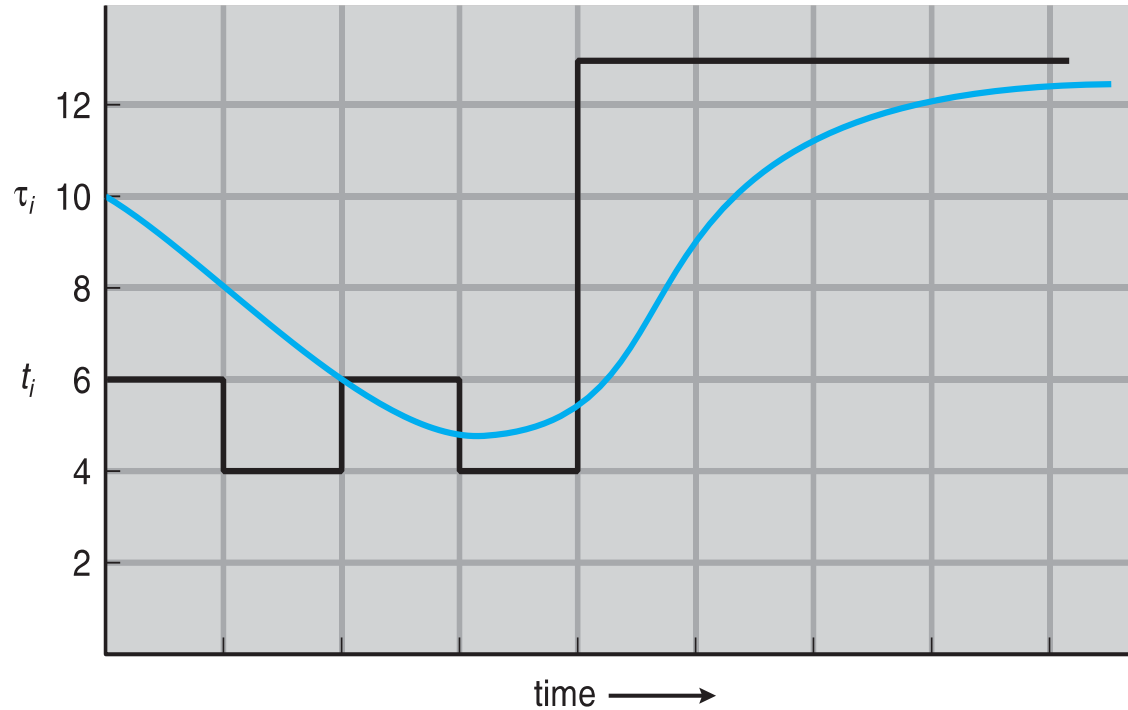
Determining Length of Next CPU Burst

- Can only estimate the length – should be similar to the previous one
 - Then pick process with shortest predicted next CPU burst
- Can be done by using the length of previous CPU bursts, using exponential averaging
 1. t_n = actual length of n^{th} CPU burst
 2. τ_{n+1} = predicted value for the next CPU burst
 3. $\alpha, 0 \leq \alpha \leq 1$
 4. Define : $\tau_{n+1} = \alpha t_n + (1 - \alpha)\tau_n$.
- Commonly, α set to $\frac{1}{2}$
- Preemptive version called **shortest-remaining-time-first**





Prediction of the Length of the Next CPU Burst



CPU burst (t_i)	6	4	6	4	13	13	13	...
"guess" (τ_i)	10	8	6	6	9	11	12	...





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
- Problem $\equiv$ **Starvation** – low priority processes may never execute
- Solution $\equiv$ **Aging** – as time progresses increase the priority of the process

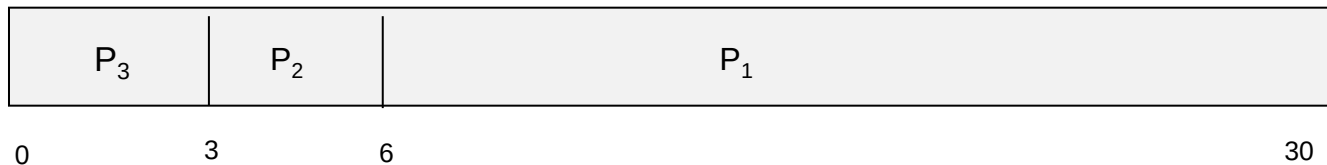




Example-1 of Priority Scheduling

<u>Process</u>	<u>Burst Time</u>	<u>Priority</u>
P_1	24	4
P_2	3	2
P_3	3	1

□ Priority scheduling Gantt Chart



□ Average waiting time = **3** msec

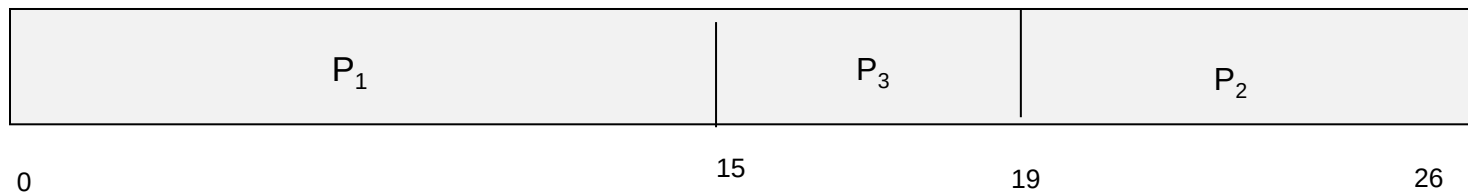




Example-2 (A) Non-preemptive Priority

<u>Process</u>	<u>Arrival Time</u>	<u>Burst Time</u>	<u>Priority</u>
P_1	0	15	3
P_2	0	7	4
P_3	2	4	1

□ Priority scheduling Gantt Chart



□ Average waiting time = **10.67** msec

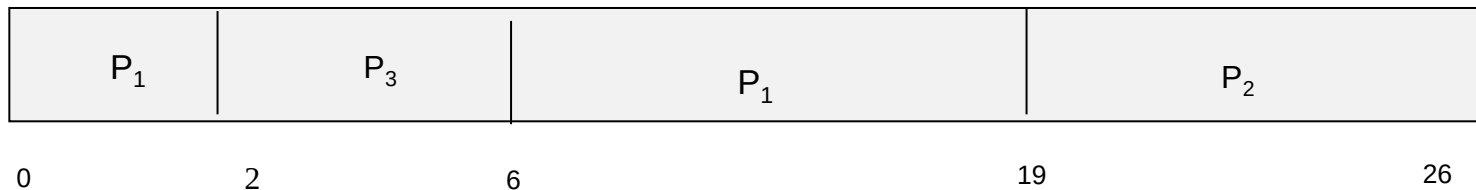




Example-2 (B) Preemptive Priority

<u>Process</u>	<u>Arrival Time</u>	<u>Burst Time</u>	<u>Priority</u>
P_1	0	15	3
P_2	0	7	4
P_3	2	4	1

□ Priority scheduling Gantt Chart



□ Average waiting time = **7.67** msec

□ Much better than previous case (A. non-preemptive)





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
- ❑ Typically, higher average turnaround than SJF, but better **response**
- ❑ Performance
 - ❑ q large $\Rightarrow$ FIFO
 - ❑ q small $\Rightarrow q$ must be large with respect to context switch, otherwise overhead is too high

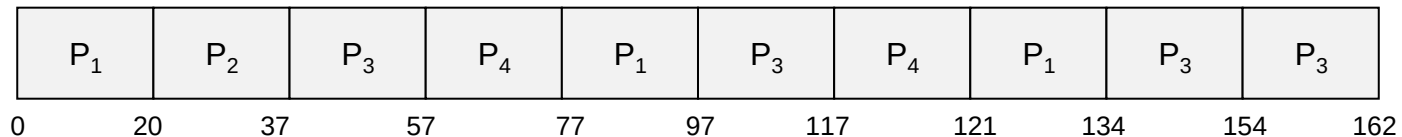




Example of RR with Time Quantum = 20

<u>Process</u>	<u>Burst Time</u>
P_1	53
P_2	17
P_3	68
P_4	24

□ The Gantt chart is:

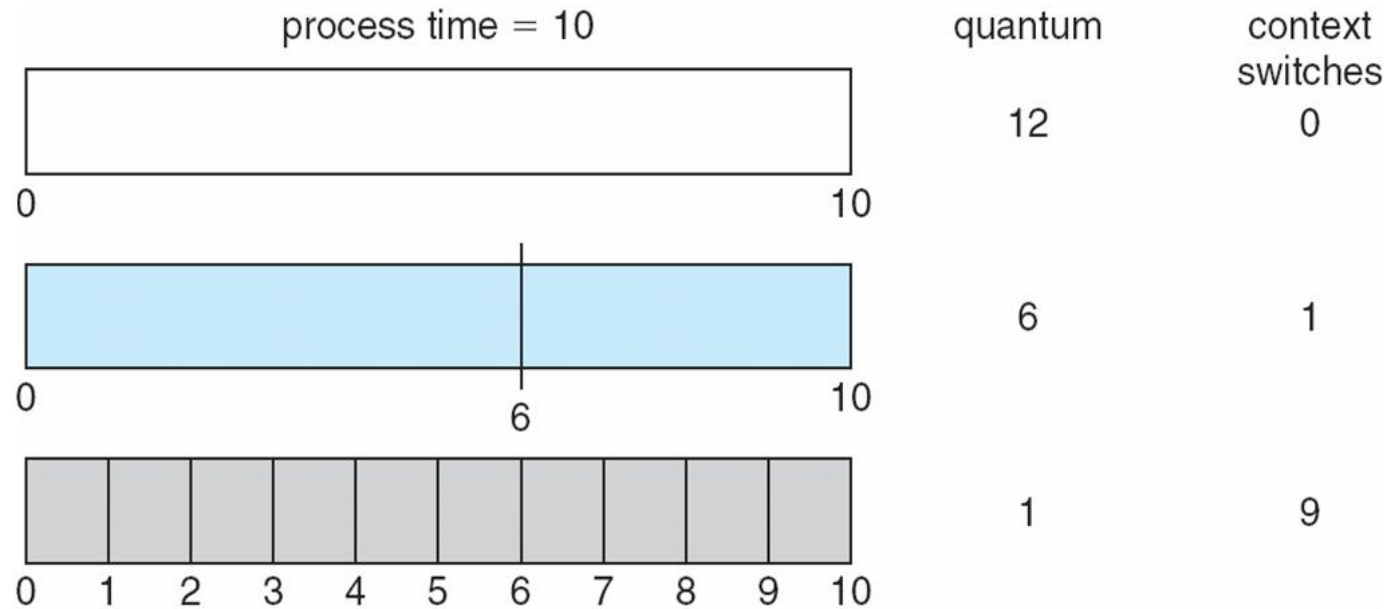


	Response Time	Waiting Time	Turnaround Time
P1	0	81	134
P2	20	20	37
P3	37	94	162
P4	57	97	121
Average	28.5	73	97.75





Time Quantum and Context Switch Time





Algorithm Evaluation

- How to select CPU-scheduling algorithm for an OS?
- Determine criteria, then evaluate algorithms
- **Deterministic modeling**
 - Type of **analytic evaluation**
 - Takes a particular predetermined workload and defines the performance of each algorithm for that workload
- Consider 5 processes arriving at time 0:

<u>Process</u>	<u>Burst Time</u>
P_1	10
P_2	29
P_3	3
P_4	7
P_5	12

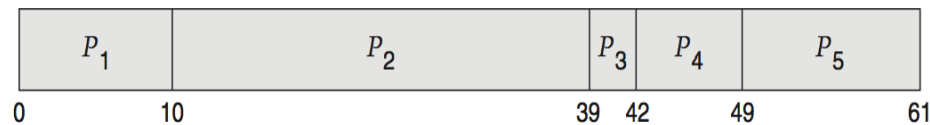




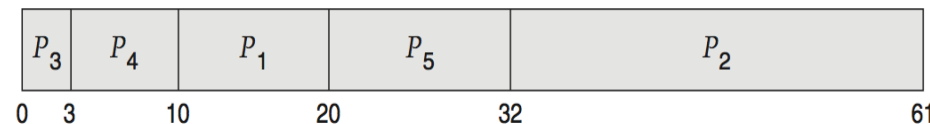
Deterministic Evaluation

- For each algorithm, calculate minimum average waiting time
- Simple and fast, but requires exact numbers for input, applies only to those inputs

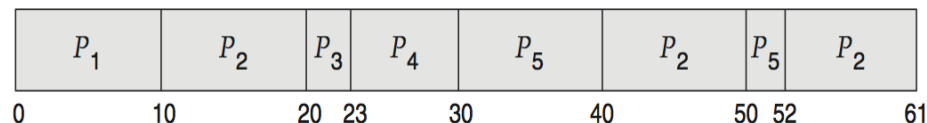
□ FCS is 28ms:



□ Non-preemptive SFJ is 13ms:



□ RR is 23ms:



End of Chapter 6

