Operating Systems

Isfahan University of Technology
Electrical and Computer Engineering Department
1400-1 semester

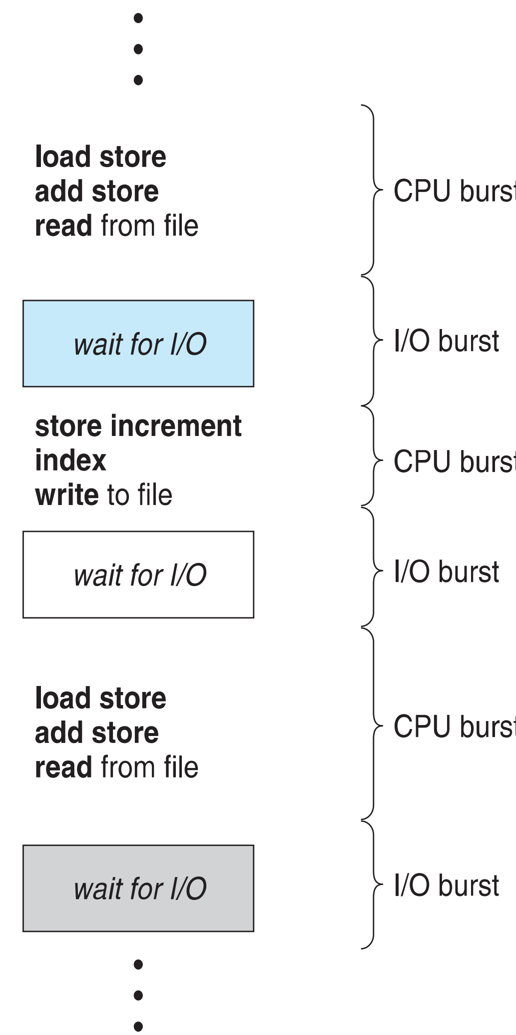
Zeinab Zali

Session 8: CPU Scheduling

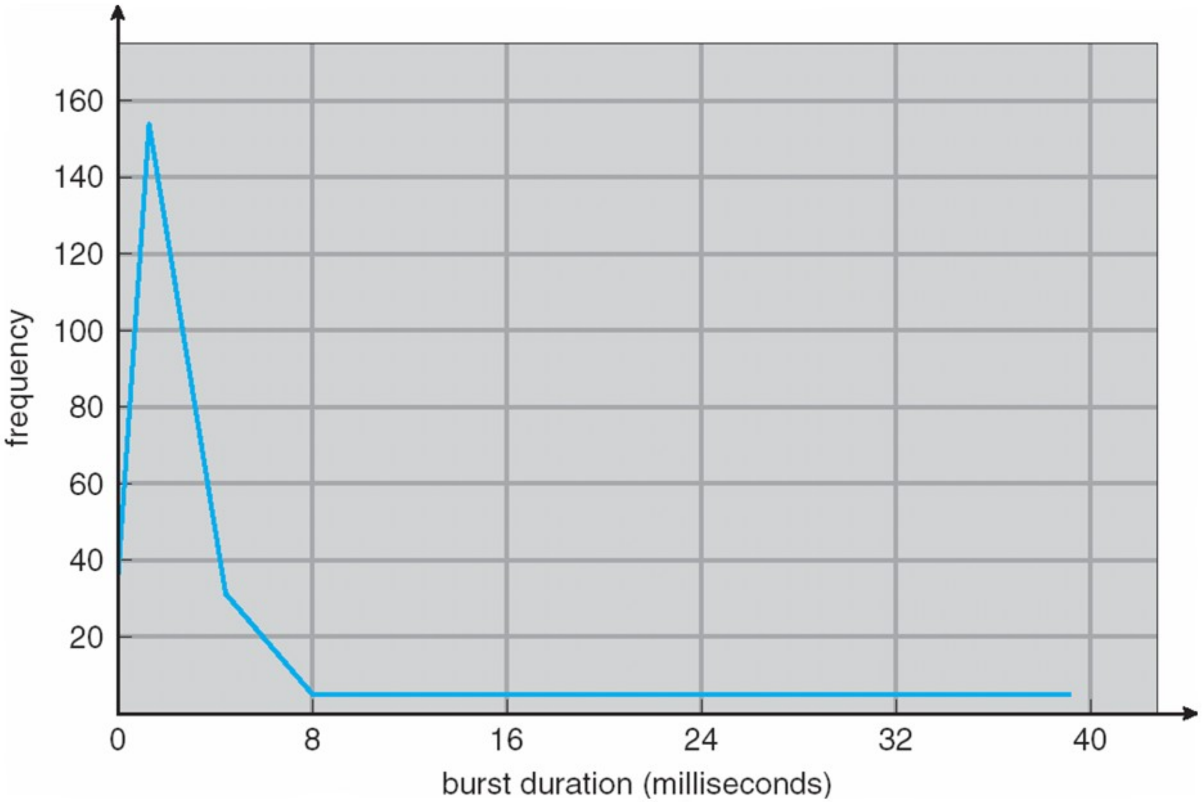
Basic Concepts

- 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

IO-bound and CPU-bound



Histogram of CPU-burst Times



CPU Scheduler

- **Short-term scheduler selects from among the processes in ready queue, and allocates the CPU to one of them**
  - **Queue may be ordered in various ways**
- 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
- Scheduling under 1 and 4 is **nonpreemptive** قبضه نشدنی یا انحصاری
- All other scheduling is **preemptive** قبضه شدنی یا غیرانحصاری
  - Consider access to shared data
  - Consider preemption while in kernel mode
  - Consider interrupts occurring during crucial OS activities

Dispatcher

- **Dispatcher** module gives control of the CPU to the process selected by the short-term scheduler; this involves:
  - switching context
  - switching to user mode
  - jumping to the proper location in the user program to restart that program
- **Dispatch latency** – time it takes for the dispatcher to stop one process and start another running

```
vmstat 1 3
```

```
cat /proc/pid/status
```

Scheduling Criteria

- **CPU utilization :** keep the CPU as busy as possible درصد مصرف پردازنده
- **Throughput:** number of processes that complete their execution per time unit بازده
- **Turnaround time:** amount of time to execute a particular process زمان برگشت
- **Waiting time:** amount of time a process has been waiting in the ready queue زمان انتظار
- **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) زمان پاسخ
- **Fairness:** give each process a fair share of CPU عدالت
- **Overhead:** computation resource required for implementing the scheduling algorithm سربار

Scheduling Algorithm Optimization Criteria

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

First- Come, First-Served (FCFS) Scheduling

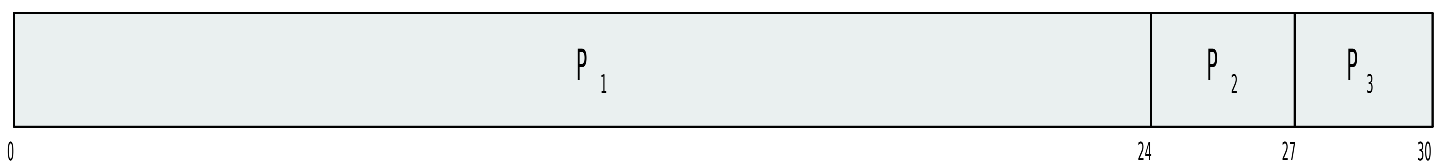
Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3

First- Come, First-Served (FCFS) Scheduling

Process	Burst Time
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.)

Process	Burst Time
P_1	24
P_2	3
P_3	3

Suppose that the processes arrive in the order:

P_2 , P_3 , P_1

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
  - Consider one CPU-bound and many I/O-bound processes

FCFS Scheduling (Example)

پروسس	زمان ورود	زمان پردازش
A	0	3
B	1	3
C	4	3
D	6	2

Waiting_time=(0+2+2+3)/4=7/4

FCFS Properties

- Non-preemptive
- No starvation
- More suitable for short jobs compared to long ones
- The least overhead

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
- **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

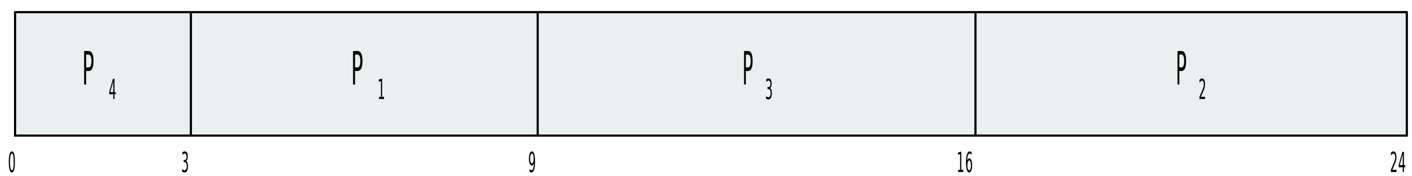
Example of SJF

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

Example of SJF

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

- SJF scheduling chart



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

SJF Properties

- Non preemptive
- Maybe starvation for long processes
- The least average waiting time

Determining Length of Next CPU Burst

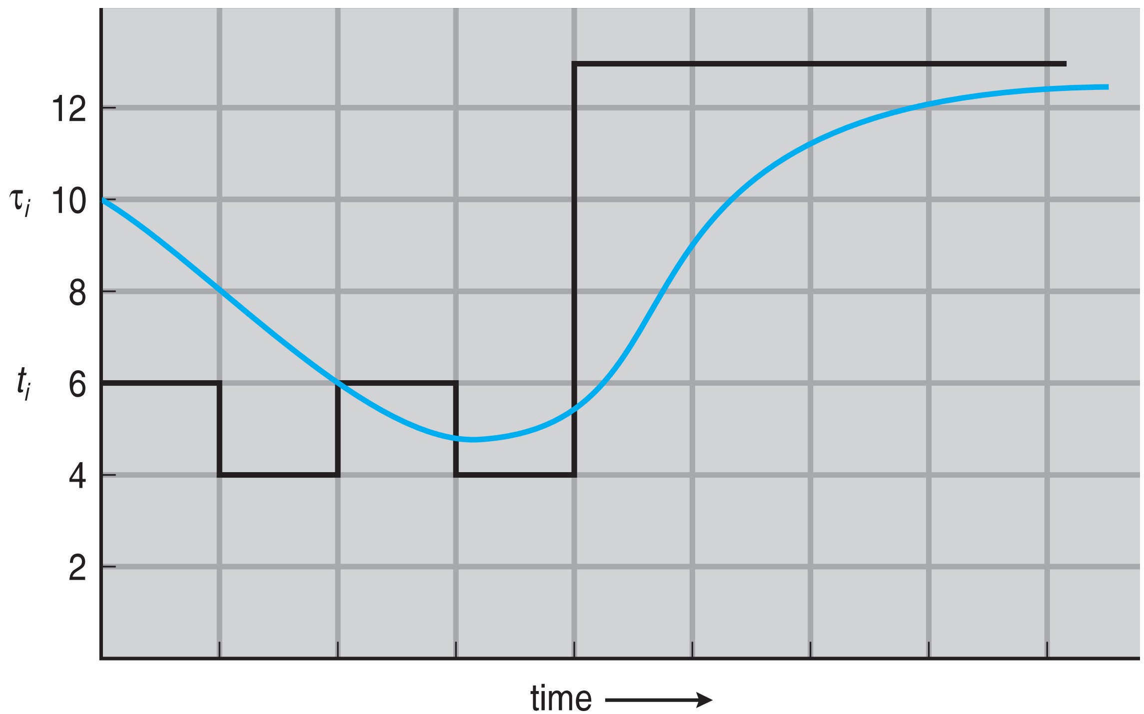
- Can only estimate the length – should be similar to the previous one
  - Then pick process with shortest predicted next CPU burst

- 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. $\tau_{n+1} =$ predicted value for the next CPU burst
  3. $\alpha,\ 0 \le \alpha \le 1$
  4. Define: $\tau_{n+1} = \alpha t_n + (1 - \alpha)\tau_n .$

- Commonly, α set to ½

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	5	9	11	12	...

Example of Shortest-remaining-time-first

- Preemptive version of SJF is called **shortest-remaining-time-first**
- Now we add the concepts of varying arrival times and preemption to the analysis

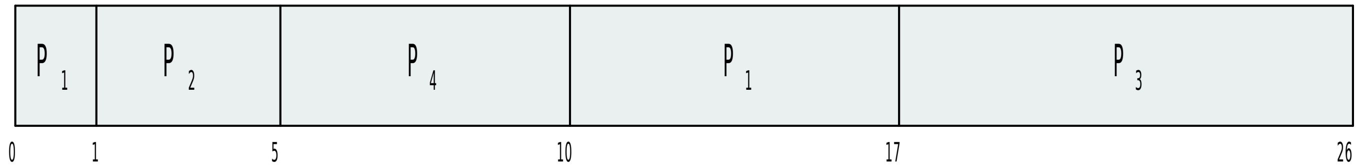
Process	*Arrival* Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

Example of Shortest-remaining-time-first

- Preemptive version of SJF is called **shortest-remaining-time-first**
- Now we add the concepts of varying arrival times and preemption to the analysis

Process	*Arrival* Time	Burst Time
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