Operating Systems

Lecture 6: CPU Scheduling

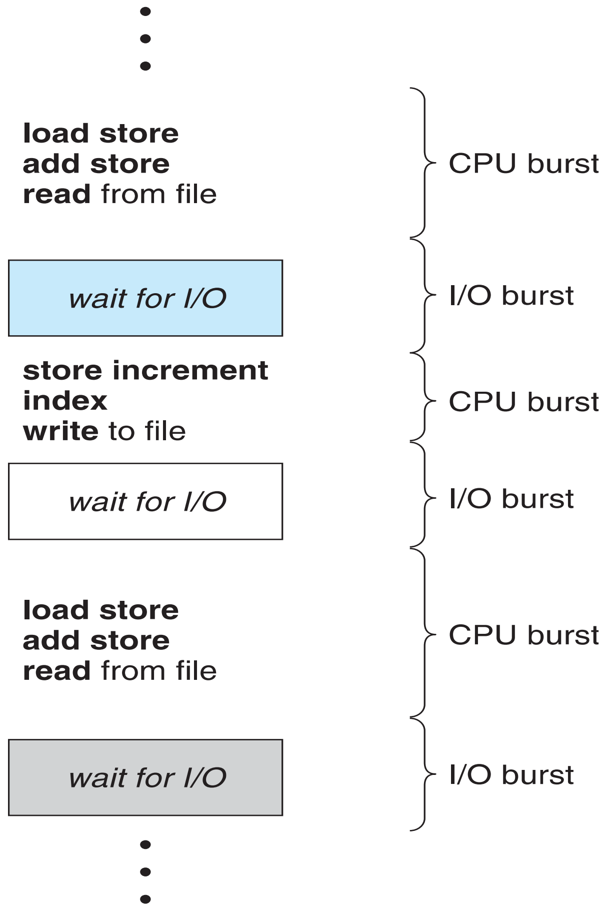
© Fall 2024 – **Dr. Alyaa Hamza – Dr. Ahmed Moawad**
Alyaa.hamza@sut.edu.eg ahmed.moawad@sut.edu.eg

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
  - 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**
- All other scheduling is **preemptive**

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

Scheduling Algorithms

In job scheduling algorithms we follow 3 steps:

- Step 1: draw Gantt chart
- Step 2: calculate Waiting Time

$$\text{WT (Waiting Time)} = \textbf{initial processing time} - \textbf{arrival time}$$

- Step 3: calculate Average Waiting Time (AWT)

$$\text{AWT} = \frac{\mathbf{WT_1 + WT_2 + WT_3 \ldots\ldots + WT_n}}{\textbf{number of processes}}$$

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.)

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

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.

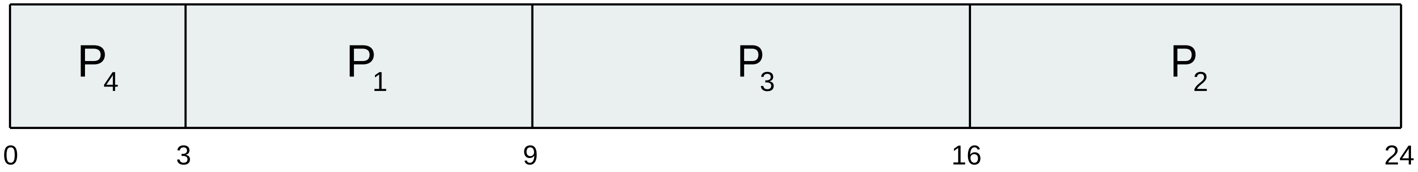
  □ The difficulty is knowing the length of the next CPU request.

  □ Could ask the user.

Example of SJF (non-preemptive)

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

- SJF scheduling chart



- Waiting time for $P_1 = 3$; $P_2 = 16$; $P_3 = 9$; $P_4 = 0$
- Average waiting time = $(3 + 16 + 9 + 0) / 4 = 7$

Example of Shortest-remaining-time-first (preemptive)

- 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

P_1	P_2	P_4	P_1	P_3
0–1	1–5	5–10	10–17	17–26

- Waiting time for P1 = (0-0)+(10-1) = 9 ; P2 = (1-1) = 0
  P3 = (17 – 2) = 15; P4 = (5 – 3) = 2
- Average waiting time = [9 + 0 + 15 + 2]/4 = 26/4 = 6.5 msec

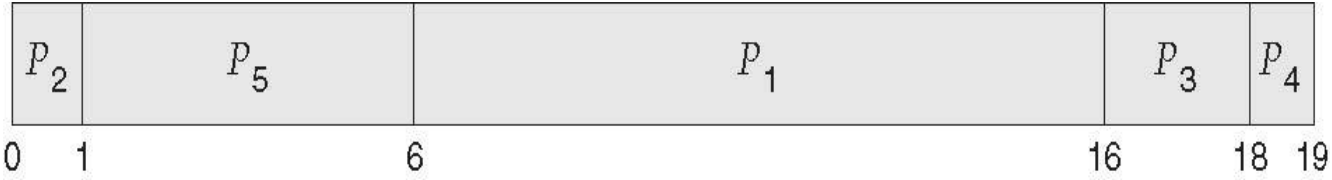
Priority Scheduling

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

Example of Priority Scheduling

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

- Priority scheduling Gantt Chart



- Waiting time for P_1 = 6; P_2 = 0; P_3 = 16; P_4 = 18; P_5 = 1
- Average waiting time = [6 + 0 + 16 + 18 + 1] / 5 = 41 / 5 = 8.2 msec

Round Robin (RR)

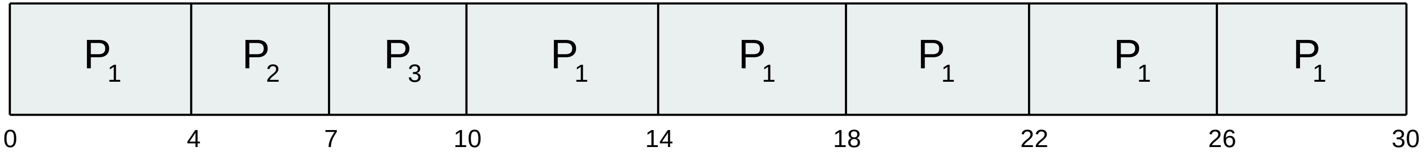
- Each process gets a small unit of CPU time (**time quantum** *q*), usually 10-100 milliseconds.

- After this time has elapsed, the process is preempted and added to the end of the ready queue.

Example of RR with Time Quantum = 4

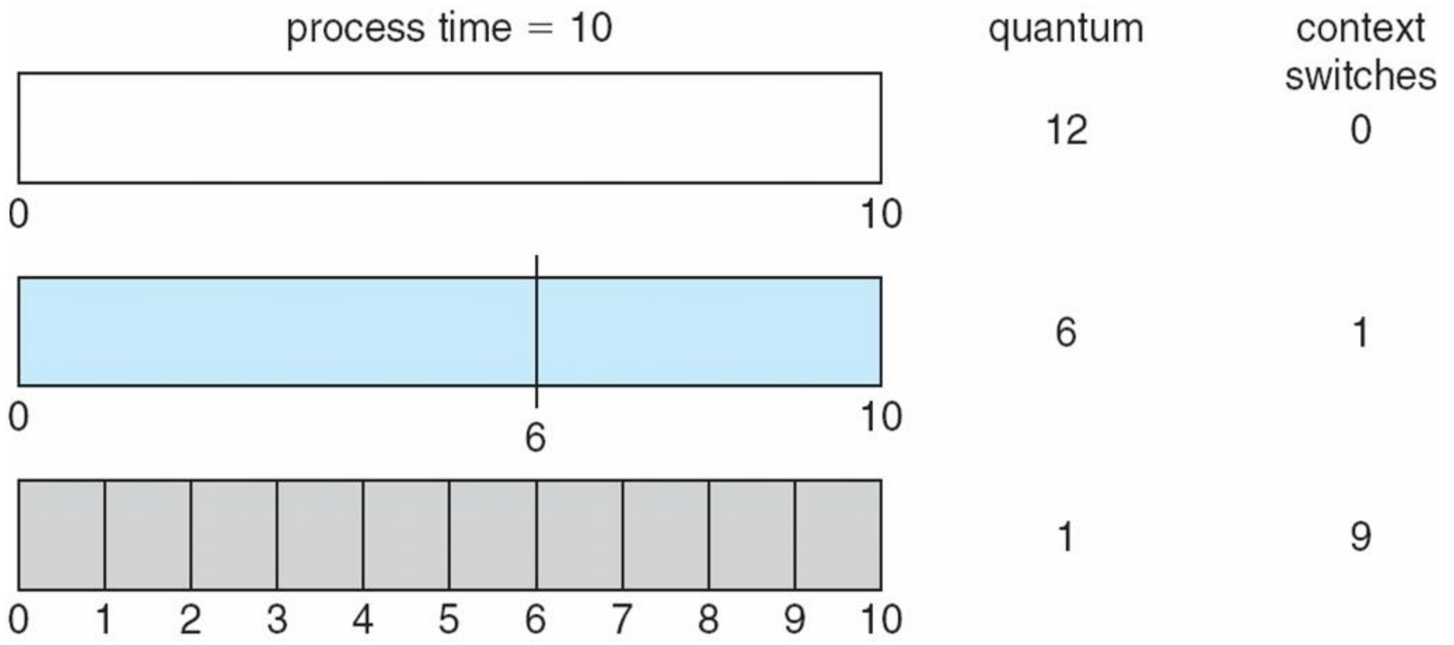
Process	Burst Time
P_1	24
P_2	3
P_3	3

- The Gantt chart is:



- Waiting time for P_1 = (0 – 0) + (10 – 4) + (14 – 14) = 0 + 6 + 0 = 6
  P_2 = (4 – 0) = 4; P_3 = (7 – 0) = 7
- Average waiting time = [6 + 4 + 7] / 3 = 17 / 3 = 5.67 msec

Time Quantum and Context Switch Time



End of Chapter 6