OPERATING SYSTEMS

Scheduling Algorithms

Suresh Jamadagni
Department of Computer Science

Slides Credits for all PPTs of this course

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the following resources and persons:

1. Slides of Operating System Concepts, Abraham Silberschatz, Peter Baer Galvin, Greg Gagne - 9th edition 2013 and some slides from 10th edition 2018
2. Some conceptual text and diagram from Operating Systems - Internals and Design Principles, William Stallings, 9th edition 2018
3. Some presentation transcripts from A. Frank – P. Weisberg
4. Some conceptual text from Operating Systems: Three Easy Pieces, Remzi Arpaci-Dusseau, Andrea Arpaci Dusseau

OPERATING SYSTEMS

FCFS and SJF Scheduling

Suresh Jamadagni

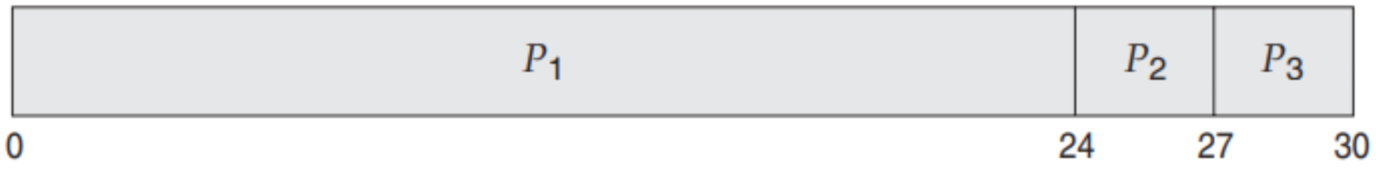
Department of Computer Science

First- Come, First-Served (FCFS) Scheduling

Process	Burst Time
P1	24
P2	3
P3	3

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

  The Gantt Chart for the schedule is:



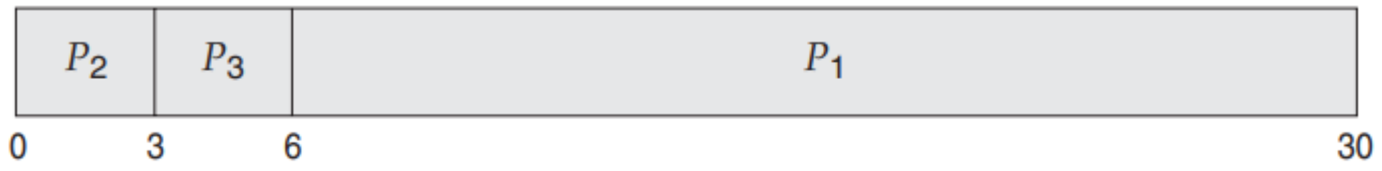
[illegible]

- 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
- ***The average waiting time under an FCFS policy is generally not minimal and may vary substantially if the processes' CPU burst times vary greatly***

FCFS Scheduling (Cont.)

- **Convoy effect** - short process behind long process
  - Consider one CPU-bound and many I/O-bound processes
- FCFS scheduling algorithm is non-preemptive. Once the CPU has been allocated to a process, that process keeps the CPU until it releases the CPU
- FCFS algorithm is thus particularly troublesome for time-sharing systems, where it is important that each process to get a share of the CPU at regular intervals.
- It is not desirable to allow one process to keep the CPU for an extended period

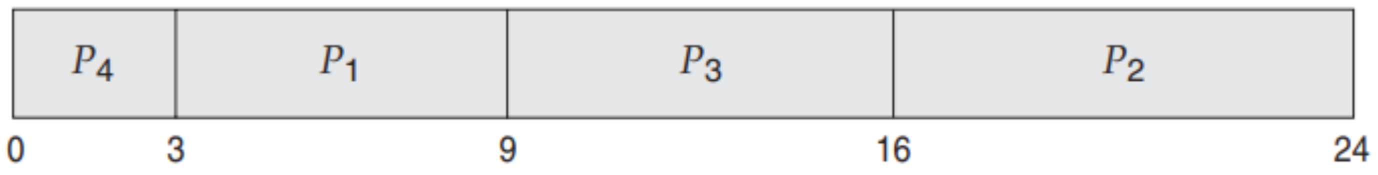
Shortest-Job-First (SJF) Scheduling

- 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
  - Compute an approximation of the length of the next CPU burst

Example of SJF Scheduling

Process	Burst Time
P1	6
P2	8
P3	7
P4	3

- SJF scheduling chart



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

Note: If FCFS scheduling is used, average waiting time = (0 + 6 + 14 + 21) / 4 = 10.25 ms.

Determining Length of Next CPU Burst

- Can be estimated using the length of previous CPU bursts, using exponential averaging

$$\tau_{n+1} = \alpha\, t_n + (1-\alpha)\tau_n$$

where τ_{n+1} is the predicted value for the next CPU burst,

The parameter α controls the relative weight of recent and past history in the prediction .

Commonly, α = 1/2, so recent history and past history are equally weighted

The value of t_n contains most recent information

The value τ_n stores the past history

- Preemptive version is called **shortest-remaining-time-first**

Determining Length of Next CPU Burst

Calculate the exponential averaging with $T_1 = 10$, $\alpha = 0.5$ and the algorithm is SJF with previous runs as 8, 7, 4, 16.

Initially $T_1 = 10$ and $\alpha = 0.5$ and the run times given are 8, 7, 4, 16 as it is shortest job first,

So the possible order in which these processes would serve will be 4, 7, 8, 16 since SJF is a non-preemptive technique.

So, using formula: $T_2 = \alpha * t_1 + (1-\alpha)T_1$

we have,

$T_2 = 0.5*4 + 0.5*10 = 7$, here $t_1 = 4$ and $T_1 = 10$

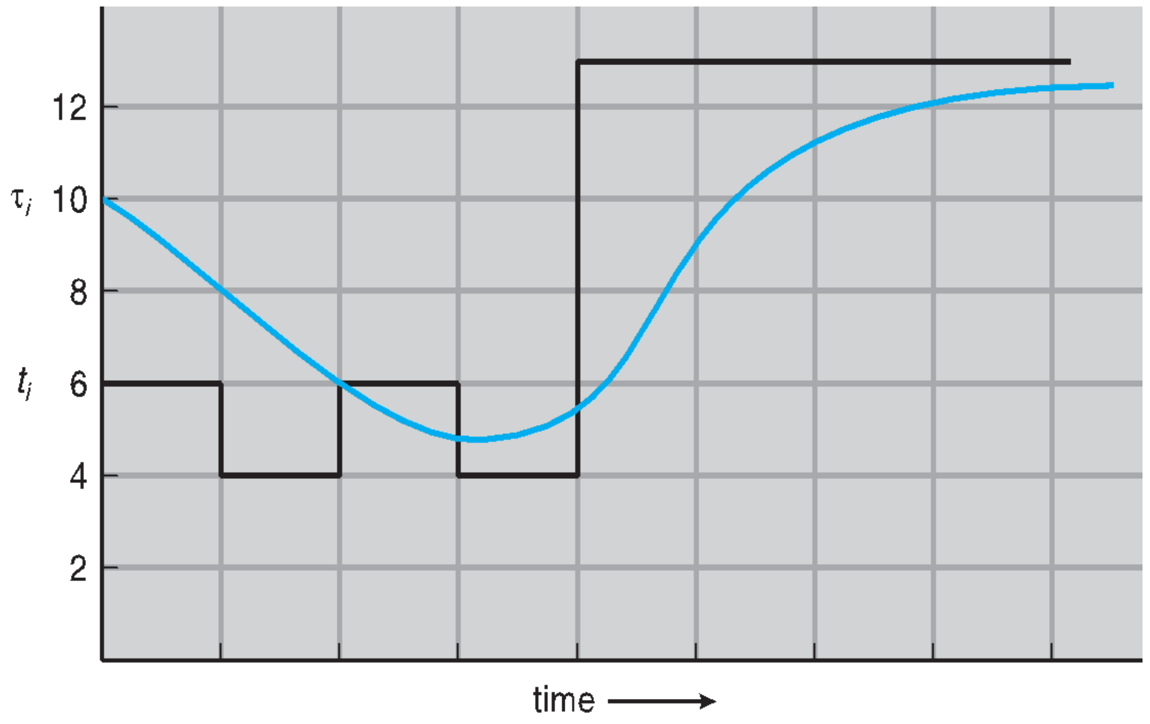
$T_3 = 0.5*7 + 0.5*7 = 7$, here $t_2 = 7$ and $T_2 = 7$

$T_4 = 0.5*8 + 0.5*7 = 7.5$, here $t_3 = 8$ and $T_3 = 7$

$T_5 = 0.5*16 + 0.5*7.5 = 11.8$, here $t_4 = 16$ and $T_4 = 7.5$

So the prediction for 5th burst time will be $T_5 = 11.8$

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 ...

$$\tau_{n+1} = \alpha\, t_n + (1-\alpha)\tau_n$$

Examples of Exponential Averaging

$$\tau_{n+1} = \alpha\, t_n + (1-\alpha)\tau_n$$

- $\alpha = 0$
  - $\tau_{n+1} = \tau_n$
  - Most recent CPU burst does not count
- $\alpha = 1$
  - $\tau_{n+1} = \alpha\, t_n$
  - Only the actual last CPU burst counts
- If we expand the formula, we get:
  - $\tau_{n+1} = \alpha\, t_n + (1 - \alpha)\alpha\, t_{n-1} + \ldots$
  - $+(1 - \alpha)^j \alpha\, t_{n-j} + \ldots$
  - $+(1 - \alpha)^{n+1} \tau_0$
- Since both α and $(1 - \alpha)$ are less than or equal to 1, each successive term has less weight than its predecessor

More examples

Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering the questions, use non-preemptive scheduling, and base all decisions on the information you have at the time the decision must be made.

Process	Burst Time
P_1	8
P_2	4
P_3	1

- What is the average wait time for these processes with the FCFS scheduling algorithm?
  Average wait time = (0 + 8 + 12)/3 = 6.67

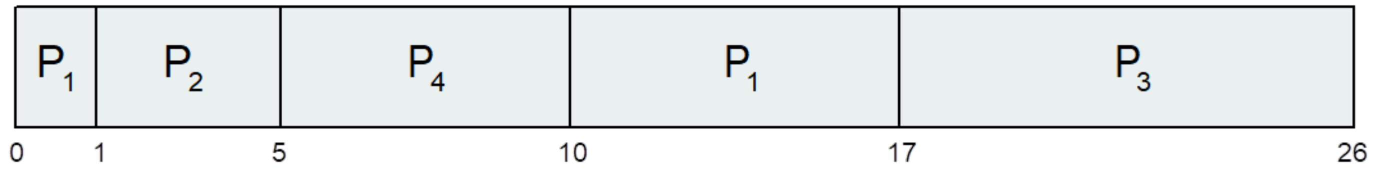
- What is the average wait time for these processes with the SJF scheduling algorithm?
  Average wait time = (5 + 1 + 0)/3 = 2

Example of Shortest-remaining-time-first

- Preemptive SJF Scheduling is sometimes called SRTF
- 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

THANK YOU

Suresh Jamadagni
Department of Computer Science Engineering
sureshjamadagni@pes.edu