Operating Systems Principles

CPU Management (Processes)

Slides are adapted from Prof. Bhuvan Urgaonkar and Siberschatz, Galvin and Gagne, Operating System Concepts (9th edition)

Choosing the Right Scheduling Algorithm/Scheduling Criteria

- Performance
  - Waiting time or response time
    - Response time – amount of time it takes from when a request was submitted until the first response is produced
    - Worth reducing in time-sharing environments
  - Throughput – # of processes that complete their execution per time unit
    - Worth increasing in batch-processing environments
- Fairness
  - E.g., proportional sharing
  - E.g., reservations

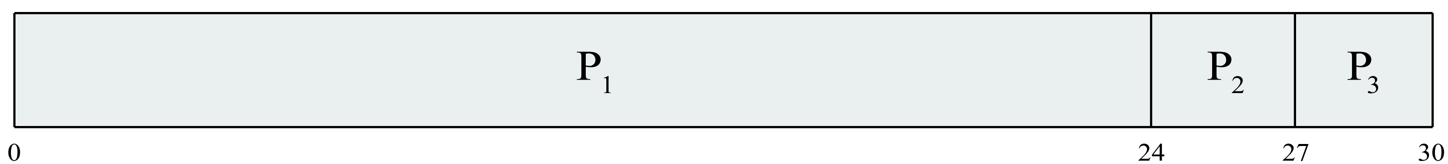
What's Wrong with FCFS?

- First-Come-First-Served
  - A natural and easy to understand/implement policy
  - What's wrong with it?
- Lets consider some better options
  - Also pay attention to data structures that the scheduler might want to use for each
  - Efficiency of the scheduler itself – time/space complexity

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

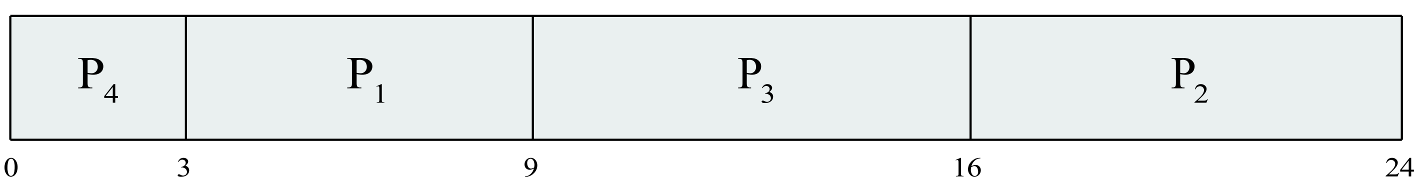
Shortest-Job-First (SJF)

- 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 for avg. waiting time – gives minimum average waiting time for a given set of processes
  - In class: Compute average waiting time for an example with SJF and compare w/ FCFS
  - Exercise: Prove the optimality claimed above

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

Why Pre-emption is Necessary

- To improve CPU utilization
  - Most processes are not *ready* at all times during their lifetimes
  - E.g., think of a text editor waiting for input from the keyboard
  - Also improves I/O utilization
- To improve responsiveness
  - Many processes would prefer "slow but steady progress" over "long wait followed by fast process"
- Most modern CPU schedulers are pre-emptive

SJF: Variations on the theme

- **Non-preemptive**: once CPU given to the process it cannot be preempted until completes its CPU burst - the SJF we already saw

- **Preemptive**: if a new process arrives with CPU length less than remaining time of current executing process, preempt. This scheme is know as *Shortest-Remaining-Time-First (SRTF)*
  - Also called *Shortest Remaining Processing Time (SRPT)*

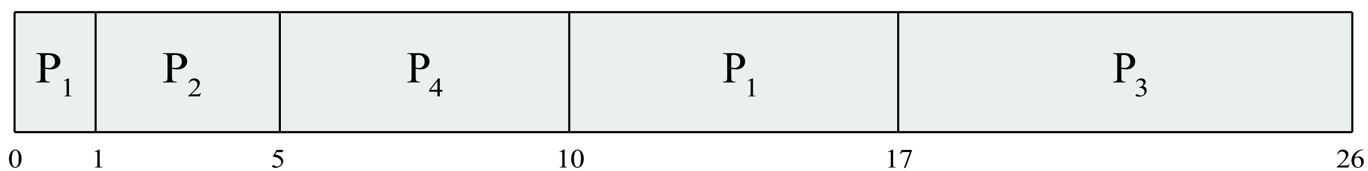
- Why SJF/SRTF may not be practical
  - CPU requirement of a process rarely known in advance

Example of 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