PRINCIPLES OF OPERATING SYSTEMS

LECTURE 9
Principles of Operating Systems

CPU SCHEDULING ALGORITHMS
(FCFS AND SJF)

Scheduling Policies

- First-Come First-Serve (FCFS)
- Shortest Job First (SJF)
  - Non-preemptive
  - Pre-emptive

First Come First Serve (FCFS) Scheduling

- Policy: Process that requests the CPU *FIRST* is allocated the CPU *FIRST.*
  - FCFS is a non-preemptive algorithm.
- Implementation - using FIFO queues
  - incoming process is added to the tail of the queue.
  - Process selected for execution is taken from head of queue.
- Performance metric - Average waiting time in queue.
- Gantt Charts are used to visualize schedules.

First-Come, First-Served(FCFS) Scheduling

- Example

Process	Burst Time
P1	24
P2	3
P3	3

Gantt Chart for Schedule

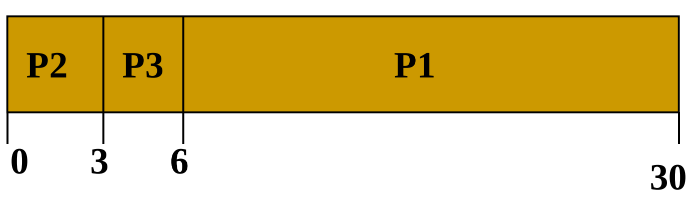
- Suppose the arrival order for the processes is
  - P1, P2, P3
- Waiting time
  - P1 = 0;
  - P2 = 24;
  - P3 = 27;
- Average waiting time
  - (0+24+27)/3 = 17
- Average completion time
  - (24+27+30)/3 = 27

FCFS Scheduling (cont.)

- Example

Process	Burst Time
P1	24
P2	3
P3	3

Gantt Chart for Schedule



- Suppose the arrival order for the processes is
  - P2, P3, P1
- Waiting time
  - P1 = 6; P2 = 0; P3 = 3;
- Average waiting time
  - (6+0+3)/3 = 3 , better..
- Average waiting time
  - (3+6+30)/3 = 13 , better..
- *Convoy Effect*:
  - short process behind long process, e.g. 1 CPU bound process, many I/O bound processes.

Shortest-Job-First(SJF) Scheduling

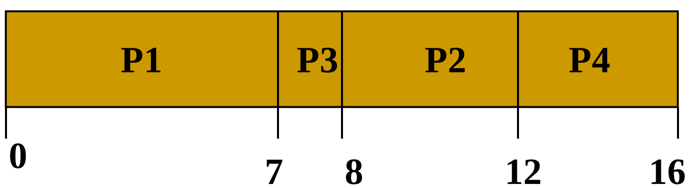
- Associate with each process the length of its next CPU burst.
- Use these lengths to schedule the process with the shortest time.
- Two Schemes:
  - Scheme 1: Non-preemptive
    - Once CPU is given to the process it cannot be preempted until it completes its CPU burst.
  - Scheme 2: Preemptive
    - If a new CPU process arrives with CPU burst length less than remaining time of current executing process, preempt. ***Also called Shortest-Remaining-Time-First (SRTF).***.

SJF and SRTF (Example)

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	1
P4	5	4

Non-Preemptive SJF Scheduling

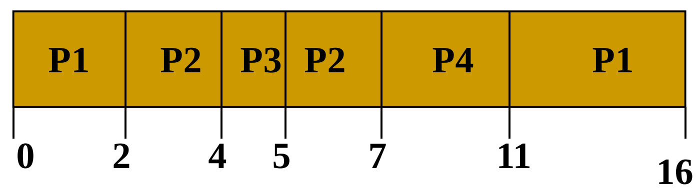
Gantt Chart for Schedule



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

Preemptive SJF Scheduling

Gantt Chart for Schedule



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

SJF/SRTF Discussion

- SJF/SRTF are the best you can do at minimizing average response time
  - Provably optimal (SJF among non-preemptive, SRTF among preemptive)
  - Since SRTF is always at least as good as SJF, focus on SRTF
- Comparison of SRTF with FCFS and RR
  - What if all jobs the same length?
    - SRTF becomes the same as FCFS (i.e. FCFS is best can do if all jobs the same length)
  - What if jobs have varying length?
    - SRTF (and RR): short jobs not stuck behind long ones
- Starvation
  - SRTF can lead to starvation if many small jobs!
  - Large jobs never get to run

SRTF Further discussion

- Somehow need to predict future
  - How can we do this?
  - Some systems ask the user
    - When you submit a job, have to say how long it will take
    - To stop cheating, system kills job if takes too long
  - But: Even non-malicious users have trouble predicting runtime of their jobs
- Bottom line, can't really know how long job will take
  - However, can use SRTF as a yardstick for measuring other policies
  - Optimal, so can't do any better
- SRTF Pros & Cons
  - Optimal (average response time) (+)
  - Hard to predict future (-)
  - Unfair (-)

Determining Length of Next CPU Burst

- One can only estimate the length of burst.
- Use the length of previous CPU bursts and perform exponential averaging.
  - t_n = actual length of nth burst
  - τ_{n+1} =predicted value for the next CPU burst
  - $\alpha = 0, \ 0 \leq \alpha \leq 1$
  - Define
    - $\tau_{n+1} = \alpha \, t_n + (1 - \alpha) \, \tau_n$

Exponential Averaging(cont.)

- $\alpha = 0$
  - $\tau_{n+1} = \tau_n$; Recent history does not count
- $\alpha = 1$
  - $\tau_{n+1} = t_n$; Only the actual last CPU burst counts.
- Similarly, expanding the formula:
  - $\tau_{n+1} = \alpha t_n + (1-\alpha)\, \alpha t_{n-1} + \ldots +$ j

    $(1-\alpha)$^j $\alpha t_{n-j} + \ldots$

    $(1-\alpha)$^$(n+1)\ \tau_0$
    - Each successive term has less weight than its predecessor.