CPU Scheduling

- Schedulers in the OS

- Structure of a CPU Scheduler
  - Scheduling = Selection + Dispatching

- Criteria for scheduling

- Scheduling Algorithms
  - FIFO/FCFS
  - SPF / SRTF
  - Priority-Based

Schedulers

```
start
long-term (admission) scheduler

suspended ready

ready

running

short-term (CPU) scheduler

suspended blocked

medium-term (memory) scheduler

blocked
```
Focus: Short-Term Scheduling

- Recall: Motivation for multiprogramming -- have multiple processes in memory to keep CPU busy.
- Typical execution profile of a process/thread:

  ![CPU burst profile diagram]

  - CPU scheduler is managing the execution of CPU bursts, represented by processes in ready or running state.

Scheduling Decisions

“Who is going to use the CPU next?”

- Scheduling decision points:
  - 1. The running process changes from running to waiting (current CPU burst of that process is over).
  - 2. The running process terminates.
  - 3. A waiting process becomes ready (new CPU burst of that process begins).
  - 4. The current process switches from running to ready.
Intro to Scheduling

Structure of a Scheduler

- ready queue
- PCB

- scheduler
- dispatcher
- CPU

select process
start new process

What Is a Good Scheduler? Criteria

- User oriented:
  - Turnaround time: time interval from submission of job until its completion
  - Waiting time: sum of periods spent waiting in ready queue
  - Response time: time interval from submission of job to first response
  - Normalized turnaround time: ratio of turnaround time to service time

- System oriented:
  - CPU utilization: percentage of time CPU is busy
  - Throughput: number of jobs completed per time unit

- Any good scheduler should:
  - maximize CPU utilization and throughput
  - minimize turnaround time, waiting time, response time

- Huh?
  - maximum/minimum values vs. average values vs. variance
Scheduling Algorithms

- FCFS: First-come-first-served
- SPN: Shortest Process Next
- SRT: Shortest Remaining Time
- priority scheduling
- RR: Round-robin
- MLFQ: Multilevel feedback queue scheduling
- Multiprocessor scheduling

First-Come-First-Served (FCFS/FIFO)

- Advantages:
  - very simple
- Disadvantages:
  - long average and worst-case waiting times
  - poor dynamic behavior (convoy effect)
Shortest Process Next

- Whenever CPU is idle, picks process with shortest next CPU burst.
- Advantages: minimizes average waiting times.
- Problem: How to determine length of next CPU burst?!
- Problem: Starvation of jobs with long CPU bursts.

SJF Minimizes Average Waiting Time

- Provably optimal: Proof: swapping of jobs
- Example:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
| 6 | 12| 8 | 4 | W = 6+18+26 = 50
| 6 | 8 | 12| 4 | W = 6+14+26 = 46
| 6 | 8 | 4 | 12| W = 6+14+18 = 38
| 6 | 4 | 8 | 12| W = 6+10+18 = 34
| 4 | 6 | 8 | 12| W = 4+10+18 = 32

\[ dW = t_{\text{short}} - t_{\text{long}} < 0 \]
(Fixed) Priority Scheduling

- Whenever CPU is idle, picks process with highest priority.
- Priority:
  - process class, urgency, pocket depth.
- Unbounded blocking: Starvation
  - Increase priority over time: aging

Conceptually

Priority Queues

$q = f(p)$

$low \text{ priority} \rightarrow \text{high priority} \rightarrow CPU$

$low \text{ priority} \rightarrow \text{high priority} \rightarrow CPU$
Round-Robin

- FIFO with preemption after time quantum
- Method for time sharing
- Choice of time quantum:
  - large: FCFS
  - small: Processor sharing
- Time quantum also defines context-switching overhead

Multilevel Feedback Queue Scheduling (conceptually)

- FCFS (quantum = infinity)
- quantum = 16 ms
- quantum = 4 ms
- quantum = 2 ms

Selector (compare priorities)

demotion

aging

low priority

high priority