Scheduler Options

- First Come, First Served (FCFS)
- Last In First Out (LIFO)
- Shortest Job First
- May use priorities to determine who runs next
- Round Robin (RR)
- Multilevel Queuing
- Multilevel Queuing with Feedback
- Lottery Scheduling
Shortest-Job-First (SJF) Scheduling

- Estimated CPU burst time associate with each process. Scheduler uses these lengths to schedule the process with the shortest time.

- Approximate CPU-burst duration
  - Based on the durations of the previous bursts
    - The past can be a good predictor of the future

- SJF is optimal – gives minimum average waiting time for a given set of processes
Example of SJF

- Process:
  - $P_1$: 6
  - $P_2$: 8
  - $P_3$: 7
  - $P_4$: 3

- SJF scheduling chart

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

- A priority number (integer) is associated with each process.
- The CPU is allocated to the process with the highest priority.
  - Preemptive
  - Non-preemptive
- Problem: Starvation
  - Low priority processes may never execute.
- Solution: Aging
  - As time progresses increase the priority of the process.
Round Robin (RR)

- Each process gets a small unit of CPU time (time quantum), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.

- If there are \( n \) processes in the ready queue and the time quantum is \( q \), then each process gets at most \( q \) time units at once. No process waits more than \( (n-1)q \) time units.
Round Robin (RR)

- Performance
  - $q$ is too large $\Rightarrow$ FIFO-like behaviour
  - $q$ is too small $\Rightarrow$ $q$ must be large, otherwise overhead is too high
Example of RR with Time Quantum = 4

<table>
<thead>
<tr>
<th>Process</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>24</td>
</tr>
<tr>
<td>$P_2$</td>
<td>3</td>
</tr>
<tr>
<td>$P_3$</td>
<td>3</td>
</tr>
</tbody>
</table>

- The Gantt chart is:

```
+---+---+---+---+---+---+---+---+---+
| $P_1$ | $P_2$ | $P_3$ | $P_1$ | $P_1$ | $P_1$ | $P_1$ | $P_1$ |
+---+---+---+---+---+---+---+---+
| 0  | 4   | 7   | 10  | 14  | 18  | 22  | 26  | 30  |
```

- Typically, higher average turnaround, but better response
- Turnaround time also depends on the size of the time quantum
Multilevel Queue Scheduling

- Today most schedulers use multiple queues
- Essentially the ready queue is really multiple (separate) queues
- The reason is that processes can be classified into different groups
  - Example: foreground (interactive) vs background (batch) processes
Multilevel Queue Scheduling

- Each queue has its own scheduling algorithm e.g.,
  - RR with time quantum of 5
  - RR with time quantum of 8
  - FIFO
Multilevel Queue

- Scheduling must be done between the queues
  - Fixed priority scheduling; (i.e., serve all from foreground then from background).
    - Possibility of starvation.
  - Time slice - each queue gets a certain amount of CPU time which it can schedule amongst its processes
Multilevel Feedback Queue Scheduling

- A process can move between queues
- Separate processes according to the characteristics of the CPU bursts (*feedback*)
  - If a process uses too much CPU time, it will be moved to a lower-priority queue
- In addition, a process that waits too long in a lower-priority queue may be moved to a higher-priority queue
Example: Multilevel Feedback Queues

- Three queues:
  - \( Q_0 \) - (round robin) RR with time quantum 8 milliseconds
  - \( Q_1 \) - RR time quantum 16 milliseconds
  - \( Q_2 \) - FCFS

- The scheduler first executes all processes in \( Q_0 \); it then proceeds to queue \( Q_1 \) followed by queue \( Q_2 \)

- Processes in a queue are served in the order they enter the queue

- Processes entering \( Q_0 \) will preempt a running \( Q_1 \) or \( Q_2 \) processes
Example: Multilevel Feedback Queues

- **Scheduling**
  - A new process is placed on $Q_0$.
  - When it gains CPU, job receives 8 milliseconds. If it does not finish in 8 milliseconds (runs entire time), process is moved to queue $Q_1$.
  - At $Q_1$ job process receives 16 additional milliseconds. If it still does not complete (runs entire time), it is preempted and moved to queue $Q_2$. 
Example: Multilevel Feedback Queues

- What does the algorithm prioritize?
  - CPU bursts 8 milliseconds or less
- Processes that need more than 8 but less than 24 are also served quickly but with lower priority than shorter processes
- Processes that need more than 24 receive the lowest priority
Lottery Scheduling

- Scheduler gives each process some lottery tickets.
- To select the next process to run...
  - The scheduler randomly selects a lottery number
  - The winning process gets to run
- Example
  - Process A gets 50 tickets → 50% of CPU
  - Process B gets 15 tickets → 15% of CPU
  - Process C gets 35 tickets → 35% of CPU
Summary

- Reviewed several scheduling algorithms