Round-Robin Scheduling
Overview

- Scheduler overview
- Scheduling goals
  - Batch systems
  - Interactive systems
- Scheduling policies
  - Batch scheduling policies
    - First-Come, First Served (FIFO)
    - Shortest Job First (SJF)
    - Shortest Remaining Time (SRT)
  - Interactive scheduling policies
    - Round-Robin
    - Static priority scheduling
    - Dynamic priority scheduling
Interactive Systems

- Short running, IO-bound programs
- Interactive users, response time constraints
- The scheduler in these systems aims to
  - Provide low response time
    - Reduce time between receiving request and producing response
  - Good throughput as well
    - Maximize number of tasks that complete per unit time
Problems with Batch Policies

- Have long response time
  - FIFO and SJF run jobs to completion
    - A job needs to wait for all, or shorter previous jobs to complete
  - SRT suffers similar problem

- SJF and SRT
  - Can starve long running jobs
  - Require estimate of processing time
    - Hard to do for short, IO-bound jobs
Round-Robin Scheduling

- Preemptive version of FIFO
- Run: Select threads in the order they arrive (FIFO)
  - Enqueue new threads at the back of the ready queue
  - Run thread at the front of the ready queue
- Stop: Preempt thread when time slice expires
  - Each thread runs for at most a time slice before another thread runs
Round-robin preempts current thread when time slice ends, so requires timer interrupts.
How should the time slice be chosen?

- Response time $\approx$ time slice * number of READY threads
  - Smaller time slice is better
- Context switch overhead = $\frac{CS}{(TS + CS)}$
  - Larger time slice is better, typically $TS \gg CS$

- Typical $CS \approx 10\mu s$, $TS = 1\text{ ms}$, overhead $\approx 1\%$
Does Round-Robin Fix Batch Problems?

- Long response time?
- Starvation?
- Estimation of processing time?
## Round-Robin Example

- Select threads in the order they arrive
- Run thread to completion or until time slice expires
  - Time slice is 1 unit

<table>
<thead>
<tr>
<th>Thread</th>
<th>Arrival Time</th>
<th>Running Time</th>
<th>Time</th>
<th>Current Thread</th>
<th>Ready Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>16</td>
<td>3+</td>
<td>2 2+2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6</td>
<td>16</td>
<td>2*</td>
<td>2+2 5</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>4</td>
<td>12</td>
<td>2*</td>
<td>2 2 5+</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>5</td>
<td>18</td>
<td>2*</td>
<td>2 2 5+3</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>2</td>
<td>10</td>
<td>2*</td>
<td>2 2 3</td>
</tr>
</tbody>
</table>

![Round-Robin Timeline](image-url)

- Time slice is 1 unit
Round-Robin Metrics

- Average waiting time: \( (1 + 10 + 9 + 9 + 5)/5 = 6.8 \)
- Waiting time is higher than batch scheduling policies
- Number of preemptions: \( (1 + 5 + 3 + 3 + 1) = 13 \)

<table>
<thead>
<tr>
<th>Thread</th>
<th>Arrival Time</th>
<th>Running Time</th>
<th>Turnaround Time</th>
<th>Waiting Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>4</td>
<td>13</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>5</td>
<td>14</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>2</td>
<td>7</td>
<td>5</td>
</tr>
</tbody>
</table>

![Round-Robin Scheduler Diagram]
Interactive systems are designed for short-running, IO bound programs
  o Scheduling policies need to provide good response time

Round-robin scheduling
  o Preemptive FIFO
  o Response time depends on time slice

Next lecture: priority-based scheduling policies
Think Time

- Use the “man sched” command to find out the default time slice period of the round-robin scheduler in Linux