Preemptive Scheduling

CS 241
Oct. 7, 2013
Preemptive Algorithms

• A scheduling algorithm is non-preemptive if and only if a task is never removed from the CPU once it has started.

• Otherwise, the scheduling algorithm is preemptive.
  – **Cost:** Every time a process is moved off a CPU, a context switch is required (expensive).
PSJF: Preemptive Shortest Job First
aka SRT: Shortest Remaining Time

<table>
<thead>
<tr>
<th>Process</th>
<th>Duration</th>
<th>Priority</th>
<th>Arrival</th>
<th>Waiting Time</th>
<th>Turnaround Time</th>
<th>Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>8</td>
<td>6</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Schedule:

Time: 0
**PPRI: Preemptive Fixed Priority**

- **PPRI:** *(Low number ➔ Higher priority)*

<table>
<thead>
<tr>
<th>Process</th>
<th>Duration</th>
<th>Priority</th>
<th>Arrival</th>
<th>Waiting Time</th>
<th>Turnaround Time</th>
<th>Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>7</td>
<td>3</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>3</td>
<td>5</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Schedule:**

Time: 0
Round Robin (RR)

• In RR:
  – The scheduler will maintain a queue of jobs
  – A time quantum will be defined (eg: RR4 ➔ quantum == 4 time units)

• Running RR:
  – When a job arrives, it is placed on the end of the queue.
  – The job on the front of the queue will run for up to a quantum of time.
  – When a quantum of time has passed, the running job will be placed on the back of the queue.
RR2: Round Robin (q=2)

<table>
<thead>
<tr>
<th>Process</th>
<th>Duration</th>
<th>Priority</th>
<th>Arrival</th>
<th>Waiting Time</th>
<th>Turnaround Time</th>
<th>Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>8</td>
<td>6</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>7</td>
<td>3</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>3</td>
<td>5</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Schedule:

Time: 0
Round Robin Quantum

• Optimizing the time quantum is an important aspects of a RR algorithm.

• Too large of a quantum:

• Too small of a quantum:
Real-time Scheduling

• Some programs have a “real-time constraint”, where a task must be complete by some deadline defined in terms of “real-time” (referred to as the deadline).
  – Ex: P1’s deadline is 4 seconds from now

• Applications:
  – Media applications (software DVD players)
  – Embedded systems
Other Algorithms

• Cooperative Scheduling

• Earliest Deadline First (EDF, RT only)

• Rate-Monotonic (RM, RT only)

• Critical Section Preemptive Scheduling

• Graph-based Scheduling
Algorithms Comparisons

- Average Response Time:

- Average Waiting Time:
What does Linux use?

• A **task** (thread or process) can be assigned one of five different scheduling queues:
  – Fixed priority scheduling:
    • **SCHED_FIFO**
    • **SCHED_RR**
  – Dynamic priority scheduling (using “nice”)
    • **SCHED_BATCH**: Used to notify the kernel that a task is non-interactive and CPU-intensive
    • **SCHED_IDLE**: Used to notify the kernel that a task is a background task and should only run when the CPU would be otherwise idle
    • **SCHED_OTHER**: Default scheduler.
nice

• Every **SCHED_OTHER** task has a "nice" value that determines its relative priority.
  – Default: 0
  – Positive: Less priority
  – Negative: Greater priority

```
nice -n 19 ./mytask
```
SCHED_OTHER

- The dynamic priority is based on the nice value and increased for each time quantum the thread is ready to run, but denied to run by the scheduler.
  - This ensures fair progress among all SCHED_OTHER threads.