LSN 15
Processor Scheduling

ECT362 Operating Systems
LSN 15 – Processor Scheduling

Diagram:

1. New
   - Long-term scheduling
2. Ready/Suspend
   - Medium-term scheduling
3. Ready
   - Long-term scheduling
   - Short-term scheduling
4. Running
5. Exit
6. Blocked/Suspend
   - Medium-term scheduling
LSN 15 – FCFS/FIFO Scheduling

• Each process joins the Ready queue
• When the current process ceases to execute, the oldest process in the Ready queue is selected
• A short process may have to wait a very long time before it can execute
• Favors CPU-bound processes
  – I/O processes have to wait until CPU-bound process completes
### LSN 15 – FCFS/FIFO Scheduling

<table>
<thead>
<tr>
<th>Process</th>
<th>$\tau(p_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
</tr>
<tr>
<td>2</td>
<td>475</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
</tr>
<tr>
<td>4</td>
<td>75</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time Slot</th>
<th>$p_0$</th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
<th>$p_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>350</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>475</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>950</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1275</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- $T_{TRnd}(p_0) = \tau(p_0) = 350$
- $T_{TRnd}(p_1) = (\tau(p_1) + T_{TRnd}(p_0)) = 125 + 350 = 475$
- $T_{TRnd}(p_2) = (\tau(p_2) + T_{TRnd}(p_1)) = 475 + 475 = 950$
- $T_{TRnd}(p_3) = (\tau(p_3) + T_{TRnd}(p_2)) = 250 + 950 = 1200$
- $T_{TRnd}(p_4) = (\tau(p_4) + T_{TRnd}(p_3)) = 75 + 1200 = 1275$
- $T_{TRnd}(AVG) = \frac{\sum T_{TRnd}}{n} = 850$
LSN 15 – Priority Based Scheduling

- Scheduler will always choose a process of higher priority over one of lower priority
- Have multiple ready queues to represent each level of priority
LSN 15 – Priority Based Scheduling

<table>
<thead>
<tr>
<th>Process</th>
<th>$\tau(p_i)$</th>
<th>Pri</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>475</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>75</td>
<td>4</td>
</tr>
</tbody>
</table>

$T_{TRnd}(p_0) = \tau(p_0) + \tau(p_4) + \tau(p_2) + \tau(p_1) = 350 + 75 + 475 + 125 + 250 = 1275$

$T_{TRnd}(p_1) = \tau(p_1) + \tau(p_3) = 125 + 250 = 375$

$T_{TRnd}(p_2) = \tau(p_2) + \tau(p_1) + \tau(p_3) = 475 + 125 + 250 = 850$

$T_{TRnd}(p_3) = \tau(p_3) = 250$

$T_{TRnd}(p_4) = \tau(p_4) + \tau(p_2) + \tau(p_1) + \tau(p_3) = 75 + 475 + 125 + 250 = 925$

$T_{TRnd}(AVG) = \frac{1275 + 375 + 850 + 250 + 925}{5} = 735$
LSN 15 – Round Robin Scheduling

- Uses preemption based on a clock
- Clock interrupt is generated at periodic intervals
- When an interrupt occurs, the currently running process is placed in the read queue
  - Next ready job is selected
- Known as time slicing
LSN 15 – Round Robin Scheduling

<table>
<thead>
<tr>
<th>Process</th>
<th>( \tau(p_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
</tr>
<tr>
<td>2</td>
<td>475</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
</tr>
<tr>
<td>4</td>
<td>75</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
T_{TRnd}(p_0) & = 1100 \\
T_{TRnd}(p_1) & = 550 \\
T_{TRnd}(p_2) & = 1275 \\
T_{TRnd}(p_3) & = 950 \\
T_{TRnd}(p_4) & = 475 \\
T_{TRnd}(AVG) & = 870
\end{align*}
\]
LSN 15 – Round Robin Scheduling

<table>
<thead>
<tr>
<th>Process</th>
<th>$\tau(p_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
</tr>
<tr>
<td>2</td>
<td>475</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
</tr>
<tr>
<td>4</td>
<td>75</td>
</tr>
</tbody>
</table>

$T_{TRnd}(p_0) = 1320$
$T_{TRnd}(p_1) = 660$
$T_{TRnd}(p_2) = 1535$
$T_{TRnd}(p_3) = 1140$
$T_{TRnd}(p_4) = 565$
$T_{TRnd}(AVG) = 1044$
LSN 15 – Shortest Process Next Scheduling

- Nonpreemptive policy
- Process with shortest expected processing time is selected next
- Short process jumps ahead of longer processes in ready queue
- Predictability of longer processes is reduced
- If estimated time for process not correct, the operating system may abort it
### LSN 15 – Shortest Process Next Scheduling

<table>
<thead>
<tr>
<th>Process</th>
<th>( \tau(p_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
</tr>
<tr>
<td>2</td>
<td>475</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
</tr>
<tr>
<td>4</td>
<td>75</td>
</tr>
</tbody>
</table>

\[
T_{TRnd}(p_0) = \tau(p_0) + \tau(p_3) + \tau(p_1) + \tau(p_4) = 350 + 250 + 125 + 75 = 800
\]
\[
T_{TRnd}(p_1) = \tau(p_1) + \tau(p_4) = 125 + 75 = 200
\]
\[
T_{TRnd}(p_2) = \tau(p_2) + \tau(p_0) + \tau(p_3) + \tau(p_1) + \tau(p_4) = 475 + 350 + 250 + 125 + 75 = 1275
\]
\[
T_{TRnd}(p_3) = \tau(p_3) + \tau(p_1) + \tau(p_4) = 250 + 125 + 75 = 450
\]
\[
T_{TRnd}(p_4) = \tau(p_4) = 75
\]
\[
T_{TRnd}(AVG) = \frac{800 + 200 + 1275 + 450 + 75}{5} = 560
\]
LSN 15 – Implementing The Scheduler

From Other States

Ready Process

Process Descriptor

Enqueuer

Ready List

Dispatcher

Context Switcher

CPU

Running Process
LSN 15 – Implementing The Scheduler

- Voluntary CPU sharing
  - Each process/thread will explicitly invoke the scheduler periodically
  - Some hardware uses a `yield()` instruction to allow a process to release the CPU
  - Nonpreemptive scheduling

- Involuntary CPU sharing
  - Timer interrupt
  - Preemptive scheduling
LSN 15 – Windows Thread Scheduling

- Multi-level feedback
  - Tries to provide very high levels of service to threads that need very rapid responses
- Uses programmable interval times (20 → 200 ms)
- Supports 32 different scheduling levels
  - 16 highest priority queues → real-time level queues
  - Next 15 priority queues → variable level queues
  - Lowest level queue → system level queue
- Fully preemptive scheduling
- All threads that are ready to run are placed into their appropriate priority level queues
• Dispatcher is the kernel function `schedule()`
  – Called by system functions after every system call and normal interrupt
  – Inspects the set of tasks in TASK_RUNNING state

• Scheduling policy is a variant of round robin scheduling
  – Uses a conventional time slicing mechanism
  – Uses a computed dynamic priority
  – Dispatcher selects process with maximum counter value
LSN 15 – Homework

• Reading
  – Chapter 9.1 – 9.2