The CPU Scheduling Problem

We have multiple processes/threads, but only one CPU

How much time does each process/thread get on CPU?

Possibilities
- Keep it till done
- Each uses it a bit and passes it on
- Each gets proportional to what they pay

Which is the best policy?
Classifying Schedulers

Preemptive vs. non-preemptive

Real-time: non vs. soft vs. hard

Interactive vs. non-interactive (batch)

Mixture

Scheduling: An Example

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Service Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

What order minimizes average turnaround time?

Turnaround time: time between arrival and departure

- arrive, wait for CPU (waiting time)
- use CPU (service time), depart

Longest First vs. Shortest First

Longest First

P1
P2
P3

Shortest First

P1
P2
P3

Shortest First Is Provably Optimal

- Given n processes with service times $S_1, S_2, S_3, \ldots, S_n$

- Average Turnaround Time (ATT)

  $\frac{[S_1 + (S_1 + S_2) + (S_1 + S_2 + S_3) + \ldots + (S_1 + \ldots + S_n)]}{n}$

  $\frac{[(n * S_1) + ((n-1) * S_2) + ((n-2) * S_3) + \ldots + S_n]}{n}$

- $S_1$ has maximum weight (n), minimize it

- $S_2$ has next-highest weight (n-1), minimize it after $S_1$

- In general: order by shortest to longest
Consider Different Arrival Times

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Service Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

• Optimal for non-preemptive, must know exec times

First-Come First-Served

Allocate CPU in the order that processes arrive

<table>
<thead>
<tr>
<th>Process</th>
<th>Departure Time</th>
<th>Turnaround Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>6</td>
</tr>
</tbody>
</table>

- Average Turnaround Time = (5 + 7 + 6)/3 = 6.0
- Simple, non-preemptive, poor for short processes

Round Robin

Time-slice CPU: give each processes a quantum in turn

Shortest Process Next

Select process with shortest execution time

<table>
<thead>
<tr>
<th>Process</th>
<th>Departure Time</th>
<th>Turnaround Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

- Average Turnaround Time = (5 + 8 + 3)/3 = 5.3
- Optimal for non-preemptive, must know exec times
Shortest Remaining Time

Select process with shortest remaining time

<table>
<thead>
<tr>
<th>Process</th>
<th>Departure Time</th>
<th>Turnaround Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>P2</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>P3</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

- Average Turnaround Time = \((9 + 4 + 1)/3 = 4.7\)
- Optimal, must know execution times

Multi-Level Feedback Queues

Multiple ready queues 0, 1, ..., n
Always select process in lowest-numbered queue
Run selected process for \(2^i\) quants (for queue i)
If process doesn't block, place in next higher queue (except last)

Example using Feedback Queues

Select process in lowest-numbered queue
- preemptive: if a new process arrives, current process goes back to queue it came from

<table>
<thead>
<tr>
<th>Process</th>
<th>Departure Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>9</td>
</tr>
<tr>
<td>P2</td>
<td>5</td>
</tr>
<tr>
<td>P3</td>
<td>4</td>
</tr>
</tbody>
</table>

- Average Turnaround Time = \((9 + 5 + 1)/3 = 5.0\)
- Favors shorter processes over longer, dynamic

Priority Scheduling

Select process with highest priority

Example: P1 = medium, P2 = high, P3 = low

- Allows scheduling based on arbitrary criteria
  - External, e.g., based on who's willing to pay most
  - Internal, e.g., past CPU usage (dynamic)
**Fair Share (Proportional Share)**

Processes get predetermined fraction of CPU time
- Example: P1 gets 25%, P2 gets 50%, P3 gets 25%

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>100%</td>
<td>50%</td>
<td>33%</td>
<td>25%</td>
<td>40%</td>
<td>33%</td>
<td>43%</td>
<td>50%</td>
<td>56%</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td></td>
<td>100%</td>
<td>100%</td>
<td>67%</td>
<td>50%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Compute ratios of fraction of time used
- at time 2, P2/P1 = 100/50 = 2:1: *fair share*

---

**Computing Fraction of Time**

Compute fraction $F_n$ of CPU time used up to time $n$

$$F_n = \alpha L + (1 - \alpha) F_{n-1}$$

- $L$: did process run during last interval? 1=yes, 0=no
- $\alpha$: number between 0 and 1
  - indicates importance of recent CPU usage relative to past
  - large $\alpha$: give more weight to recent usage, forget quickly
  - small $\alpha$: give more weight to past usage, forget slowly

---

**Examples: $F_n = \alpha L + (1 - \alpha) F_{n-1}$**

- $\alpha = 1$ means ignore the past
  - $F_n = L$
- $\alpha = 0.5$, recent usage and past equally important
  - $F_n = 0.5 L + 0.5 F_{n-1}$
  - this is an *exponential average* (typically $\alpha < 0.2$)
- $\alpha = 1/n$, simple average over all intervals
  - $F_n = (1/n) L + (1 - 1/n) F_{n-1}$
  - $F_n = (L + F_1 + F_2 + F_3 + ... + F_{n-1})/n$

---

**Fair Share using Simple Average**

$\alpha = 1/n : F_n = (L + F_1 + F_2 + F_3 + ... + F_{n-1})/n$

Example: P1 gets 25%, P2 gets 50%, P3 gets 25%

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>100%</td>
<td>50%</td>
<td>33%</td>
<td>25%</td>
<td>40%</td>
<td>33%</td>
<td>43%</td>
<td>50%</td>
<td>56%</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td></td>
<td>100%</td>
<td>100%</td>
<td>67%</td>
<td>50%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: $n$ is time intervals since process started
Fair Share using Exponential Average

α = 0.5 : \( F_n = 0.5 L + 0.5 F_{n-1} \)

Example: P1 gets 25%, P2 gets 50%, P3 gets 25%