Multiprogramming

- **Without multiprogramming**

<table>
<thead>
<tr>
<th>CPU</th>
<th>I/O</th>
<th>CPU</th>
<th>I/O</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

  time = 10 tu  
adaverage execution time = 7.5 tu

  throughput = 0.2 proc/tu  
  CPU usage = 50%

- **With multiprogramming**

<table>
<thead>
<tr>
<th>CPU</th>
<th>I/O</th>
<th>CPU</th>
<th>I/O</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

  time = 6 tu  
adaverage execution time = 5.5 tu

  throughput = 0.33 proc/tu  
  CPU usage = 100%
Process Scheduling Structures

- Ready and I/O queues

- System queue diagram
Process Schedulers

- Short term (CPU)
  - Selects processes from the ready queue
  - Runs very often and therefore must be very efficient

- Long term (jobs)
  - Selects processes that will be allowed in the system
  - Tries to balance I/O-bound and CPU-bound processes
Process Schedulers

- **Medium-term (swapper)**
  - Temporarily suspends processes
  - To keep the balance between I/O and CPU usage
  - Due to memory depletion

Diagram:

- `swap in` → `swap disk` → `swap out`
- `ready queue` → `CPU`
- `I/O` → `I/O queue` → `I/O request` → `time` → `wait`
- `event` → `event queue`
A process must be chosen to occupy the CPU whenever a process
1 - Changes state from running to waiting
2 - Changes state from running to ready
3 - Changes state from waiting to ready
4 - Finishes

- Preemptive scheduling: 1, 2, 3 and 4
- Non-preemptive scheduling: 1 and 4
Process Scheduling Criteria

- Maximize CPU utilization
- Maximize system throughput (jobs/time)
- Minimize turnaround time (total time)
- Minimize waiting time (time waiting to run)
- Minimize and stabilize (user) response time
Process Scheduling Policy

- First Come First Served
- Shortest Job First
- Static Priority
- Dynamic Priority
- Round-Robin
- Multilevel Queue
- And thousands of derivations thereof
First Come First Served (FCFS)

- **Policy**
  - Ready queue under FIFO policy
  - New processes are inserted at the end
  - Non-preemptive

- **Performance**
  - Extremely poor when a CPU-bound process blocks an I/O-bound process

- **Example**

<table>
<thead>
<tr>
<th>Process</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU time</td>
<td>9</td>
<td>4</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Arrival time</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>9</th>
<th>13</th>
<th>21</th>
<th>26</th>
</tr>
</thead>
</table>

\[ TA = \left(9 + 13 + 21 + 26\right) / 4 = 17.25 \text{ tu} \]
\[ WT = \left(0 + 9 + 13 + 21\right) / 4 = 10.75 \text{ tu} \]
Shortest Job First (SJF)

- **Policy**
  - Process that will need the shortest CPU time is scheduled first
  - Preemptive or non-preemptive

- **Performance**
  - Optimal algorithm in terms of TA and WT

\[
\begin{align*}
\text{TA} &= \frac{a + (a+b) + (a+b+c) + (a+b+c+d)}{4} = \frac{4a + 3b + 2c + d}{4} \text{ tu} \\
\text{WT} &= \frac{0 + a + (a+b) + (a+b+c)}{4} = \frac{3a + 2b + c}{4} \text{ tu}
\end{align*}
\]

- Useful for processes for which the maximum execution time is known
Shortest Job First (SJF)

Example

<table>
<thead>
<tr>
<th>Process</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU time</td>
<td>9</td>
<td>4</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Arrival time</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

\[
TA = \frac{9 + 12 + 24 + 15}{4} = 15 \text{ tu}
\]

\[
WT = \frac{0 + 8 + 16 + 10}{4} = 8.5 \text{ tu}
\]

\[
TA = \frac{18 + 4 + 24 + 7}{4} = 13.25 \text{ tu}
\]

\[
WT = \frac{9 + 0 + 16 + 2}{4} = 6.75 \text{ tu}
\]
SJF Approximation

■ Policy
  ● Future estimation based on recent past
  ● Process that has been having the shortest CPU cycles is scheduled first

■ Formula
\[ \pi_{n+1} = \alpha t_n + (1 - \alpha) \pi_n \]

■ Example (\(\alpha = 1/2\))

\[
\begin{align*}
TA &= (22 + 30 + 36) / 3 = 29.3 \text{ tu} \\
WT &= (10 + 18 + 24) / 3 = 17.3 \text{ tu}
\end{align*}
\]

\[
\begin{array}{cccccc}
\text{Process} & \pi_0 & t_0 & \pi_1 & t_1 & \pi_2 & t_2 & \pi_3 \\
A & 1 & 2 & 1 & 4 & 2 & 6 & 4 \\
B & 1 & 4 & 2 & 4 & 3 & 4 & 3 \\
C & 1 & 6 & 3 & 4 & 3 & 2 & 2 \\
\end{array}
\]

\[
\begin{array}{ccccccc}
\text{Process} & \text{A} & \text{A} & \text{B} & \text{B} & \text{B} & \text{C} & \text{C} \\
0 & 2 & 6 & 10 & 16 & 22 & 26 & 30 \\
\end{array}
\]
Priority

• Policy
  • Process with highest priority is scheduled first
  • Priorities can be assigned to processes either statically or dynamically
  • Preemptive or non-preemptive

• Processes might wait indefinitely
  • Low-priority processes only run when high-priority processes are waiting

• Typical of real-time systems
Static Priority

**Example**

<table>
<thead>
<tr>
<th>Process</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU time</td>
<td>2</td>
<td>8</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Priority</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Arrival time</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
TA &= (2 + 9 + 11 + 15) / 4 = 9.25 \text{ tu} \\
WT &= (0 + 1 + 8 + 10) / 4 = 4.75 \text{ tu}
\end{align*}
\]

\[
\begin{align*}
TA &= (13 + 8 + 10 + 15) / 4 = 11.5 \text{ tu} \\
WT &= (11 + 0 + 7 + 10) / 4 = 7 \text{ tu}
\end{align*}
\]
Round-Robin

- **Policy**
  - Processes are rescheduled periodically based on a time *quantum*
  - FIFO circular queue
  - Preemptive

- **Formula**
  - For a given set of processes with *n* elements and a time quantum of *q*:
    - Each process gets $1/n$ of CPU time in cycles that are no longer than *q* time units
    - Maximum waiting time = $(n - 1) q$

- **Typical of time-sharing systems**
Round-Robin

- Example \( (q = 5 \text{ tu}) \)

<table>
<thead>
<tr>
<th>Process</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU time</td>
<td>8</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Arrival time</td>
<td>0</td>
<td>4</td>
<td>9</td>
<td>14</td>
</tr>
</tbody>
</table>

\[ TA = \frac{(13 + 6 + 15 + 12)}{4} = 11.5 \text{ tu} \]

\[ Wt = \frac{(5 + 1 + 9 + 5)}{4} = 5 \text{ tu} \]
Multilevel Queue

Policy
- Processes are grouped
  - E.g. system, interactive, batch
- Each group has its own queue under a specific policy
- Processes might be allowed to change groups