CPU Scheduling

- Mechanism
- Policy

Scheduling Policies

- Non-preemptive
  - FCFS
- Preemptive
  - SJF
  - STCF
  - RR

Workload: set of processes running

Job: process

Long vs. Short
- CPU bound
- I/O intensive

Turnaround Time ↓  Response Time ↓
Scheduling

Workload Assumptions

1. Each job runs for the same amount of time.
2. All jobs arrive at the same time.
3. Once started, each job runs to completion.
4. All jobs only use the CPU (i.e., they perform no I/O).
5. The run-time of each job is known.
1. **FCFS (FIFO)**

<table>
<thead>
<tr>
<th>Job</th>
<th>Arrival</th>
<th>( T_{\text{completion}} ) (CPU burst time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>10</td>
</tr>
</tbody>
</table>

Scheduling Metric: **Turnaround Time**

\[
T_{(T.A.)} = T_{\text{completion}} - T_{\text{arrival}}
\]

Average Turnaround Time:

\[
\text{avg. } T_{(T.A.)} = \frac{(10-0) + (20-0) + (30-0)}{3} = \frac{60}{3} = 20
\]
2. SJF

\[ T_{TA} = \frac{10 + 20 + 120}{3} = \frac{150}{3} = 50 \]
3. STCF (PSJF)

\[
\begin{array}{cccccc}
A & B & C & A & \\
0 & 10 & 20 & 30 & 120 \\
\end{array}
\]

\[
\text{ta} & \text{tr} \\
A & 0 & 10 \\
B & 3 & 7 \\
C & 5 & 4 \\
\]

interactive.

CPU:

\[
\begin{array}{cccccccccccc}
& 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 14 & 21 \\
\end{array}
\]

\[
\text{Avg. } T_{T.A.} = \frac{(14-0) + (21-3) + (9-5)}{3} \\
= \frac{14 + 18 + 4}{3} = \frac{36}{3} = 12
\]

Response Time

\[
\frac{T_{\text{firstrun}} - T_{\text{arrived}}}{3}
\]

Response

\[
T_{R}(A) = 0 \\
T_{R}(B) = 14 - 3 = 11 \\
T_{R}(C) = 5 - 5 = 0
\]
4. RR

<table>
<thead>
<tr>
<th>A</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>3</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Time slice quantum: \( q = 1 \)

```
    A  B  A  C  B  A  C
   0  3  4  5  6  ...
```

\[
\text{Avg. } T_R = \frac{0 + 0 + 0}{3} = 0.
\]

```
  0  1  2
A  B  C
```

\( q = \infty \) ?

\( q = 0 \) ?

\( q = nT \) \( \rightarrow \).Timer interrupt (10ms)
MLFQ
Multi Level Feedback Queue

Highest priority Q₂
Q₁
Lowest priority Q₀

CPU: A B A B A B A B

5. Priority Boost ↑
Gaming the scheduler!

yieldC) / sleepC).
Process APIs

1. create?

fork()

P

pid = 2

fork()

P

pid = 3

Shell

> pwd
/home/gerald

> exec()
Basic Rules

1. if Priority(A) > Priority(B), A runs.
2. if Priority(A) = Priority(B), A & B run in RR.

How to change priority?

3. when a job enters the system, it is placed at the highest priority (topmost queue).

4a. If a job uses up an entire time slice while running, its priority is reduced.
4b. If a job gives up the CPU before the time slice is up, it stays at the same priority level.
MLFQ

Basic Rules: 1, 2

How to change priority: 3, 4a, 4b.

Priority Boost

Rule 5: After some time period \( S \), move all the jobs in the system to the topmost queue.

Better Accounting

Rule 4 (new): Once a job uses up its time allotment at a given level, its priority is reduced.