Preview

- Process Scheduler
  - Short Term Scheduler
  - Long Term Scheduler

- Process Scheduling Algorithms for Batch System
  - First Come First Serve
  - Shortest Job First
  - Shortest Remaining Job First

- Process Scheduling Algorithms for Interactive System
  - Round Robin
  - Priority Queue
Scheduler

- Chooses the next process (for execution), based on the rule (scheduling algorithm).
- Process scheduler is designed for making efficient use of the CPU.
- **Long-Term Scheduler** – selects a process from the pool of jobs and loads into memory for execution
- **Short-term scheduler** – selects a process from the ready queue and allocates to the CPU.
- **Memory Scheduler** – decides which processes should be kept in memory and which ones kept on disk when the number of processes is too large.
Three Level Scheduling

Arriving job
Input queue
Long-Term Scheduler (Admission Scheduler)
Main Memory
Short-Term Scheduler (CPU Scheduler)
CPU
Disk
Memory scheduler
Effective CPU Scheduler is essential

The scheduler should make efficient use of the CPU because process switching is very expensive.

When switching from user mode to kernel mode:
1. All information for the blocked (become ready state or block state) must be saved (state, register, stack pointer, PC, ..)
2. The memory map (information regarding memory management) must be saved.
3. A new process is selected by the scheduler.
4. Information for the new process must be loaded.
5. Start running the new process.
Process behavior

- **CPU-bounded process** – a process spends most of its time computing (using CPU)
- **I/O-bounded process** – a process spends most of its time waiting for I/O.
When to make scheduling decision?

1. A process switching from “running” state to the “blocked” state
2. A process switching from “running” state to “ready” state
3. Terminating a process
4. A process switching from “blocked” state to “ready” state
**Scheduling Algorithm Goals**

Different system environments have different goals, and different scheduling algorithms are needed.

- **Batch System** –
  - Throughput – maximize jobs per hour
  - Turnaround time – minimize time between submission and termination
  - CPU utilization – keep the CPU busy all the time

- **Interactive System** –
  - Response time – response to request quickly

- **Real Time System** –
  - Meeting deadlines – avoid losing data
First Come First Serve (Batch)

- Non-preemptive
- Simplest of all CPU-scheduling algorithms
- Implementation of the FCFS is easily managed with FIFO queue.
- When a process enters the ready queue, its is put on the end of the queue.
- When the CPU is free, scheduler chooses a process from the head of ready queue.
- The average turnaround time might be long with FCFS.
First Come First Serve (Batch)

3 processes in the job queue in the order of $P_1, P_2, P_3$

<table>
<thead>
<tr>
<th>Process</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>24</td>
</tr>
<tr>
<td>$P_2$</td>
<td>3</td>
</tr>
<tr>
<td>$P_3$</td>
<td>3</td>
</tr>
</tbody>
</table>

Average turnaround time = $\frac{24+27+30}{3}=27$

Average turnaround time = $\frac{3+6+30}{3}=13$
Shortest Job First (Batch)

- Non-Preemptive
- Just pick the shortest job first

<table>
<thead>
<tr>
<th>Process</th>
<th>CPU run time needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>6(msec)</td>
</tr>
<tr>
<td>$P_2$</td>
<td>8(msec)</td>
</tr>
<tr>
<td>$P_3$</td>
<td>7(msec)</td>
</tr>
<tr>
<td>$P_4$</td>
<td>3(msec)</td>
</tr>
</tbody>
</table>
Shortest Job First (Batch)

Assume that the arriving times are in the order of $P_1 > P_2 > P_3 > P_4$

<table>
<thead>
<tr>
<th>Process</th>
<th>CPU run time needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>6(msec)</td>
</tr>
<tr>
<td>$P_2$</td>
<td>8(msec)</td>
</tr>
<tr>
<td>$P_3$</td>
<td>7(msec)</td>
</tr>
<tr>
<td>$P_4$</td>
<td>3(msec)</td>
</tr>
</tbody>
</table>

Average turnaround time $= \frac{6+14+21+24}{4} = 16.25$ for FCFS

Average turnaround time $= \frac{3+9+16+24}{4} = 13$ for SJF
Shortest Job First (Batch)

- When all the jobs are available simultaneously, Shortest Job First algorithm is optimal in that it gives the minimal average turnaround time for a given set of processes.
Shortest Remaining Time (Batch)

- Preemptive version of the SJF algorithm is Shortest Remaining Time Next algorithm.
- When a new process arrives at the ready queue while previous process is executing.
- The newly arrived process might be shorter than what is left of the current executing process $\Rightarrow$ preemption
Shortest Remaining Time (Batch)

<table>
<thead>
<tr>
<th>Processes</th>
<th>Arrival Time</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>P₂</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>P₃</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>P₄</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

Average Turnaround time = \(\frac{(17 - 0) + (5-1) + (26-2) + (10-3))}{4} = 13\)
The Round-Robin Scheduling algorithm is designed especially for the timesharing systems.

A small unit of time called time quantum is defined (between 20 to 50 millisecond).

Ready queue is treated as a circular queue.
Round-Robin (Interactive)

![Diagram]

Current process

Next process

(a)

Current process

(b)
Round-Robin (Interactive)

The main issue with round-robin algorithm is the length of the quantum.

- If quantum is too short, CPU time are wasted for process switching – low CPU utilization.
- If quantum is too long, slow response time.
Priority Queue Scheduling

- A priority is assigned to each process
- CPU is allocated to a process based on the priority.
- Equal priority processes are scheduled using FCFS or Round-Robin.
- Example:
  - School: faculty members have higher priority than students.
  - Company: Customers who pay more money can get higher priority.
Priority Queue Scheduling

Queue headers

Priority 4
Priority 3
Priority 2
Priority 1

Runnable processes

(Highest priority)

(Lowest priority)
Priority Queue Scheduling

- A major problem with priority scheduling algorithm is the **starvation** of lower priority processes.
- A solution to the problem of starvation is **aging**.
- **Aging** is a technique of gradually increasing the priorities of processes that wait in the system for a long time.