Uniprocessor scheduling

Scheduling of processes
Processor scheduling

- Schedule processes on the processor to meet system objectives
- System objectives:
  - Assigned processes to be executed by the processor
  - Response time
  - Throughput
  - Processor efficiency
Processor scheduling

- Scheduling involves managing queues to minimize queuing delay
- Four types:
  - Long-term
  - Medium-term
  - Short-term
  - I/O scheduling (not this chapter)
Figure 9.1  Scheduling and Process State Transitions
Long-Term Scheduling

- Determines which programs are admitted to the system for processing
- Controls the degree of multiprogramming
  - Add a new process if there is a need to increase degree of multiprogramming
- More processes, smaller percentage of time each process is executed
Medium-Term Scheduling

- Part of the swapping function
- Based on the need to manage the degree of multiprogramming
  - If too many processes, will swap a process to disk (suspension).
Short-Term Scheduling

- Scheduler known as the dispatcher
- Executes most frequently
- Invoked when an event occurs
  - Clock interrupts
  - I/O interrupts
  - Operating system calls
  - Signals (from semaphores)
  - New process enters the system
Short-Term Scheduling Criteria

- **User-oriented**
  - **Response time (interactive process)**
    - Elapsed time between the submission of a request until there is output
  - **Turnaround time (batch process)**
    - Elapsed time between submission of a job and its completion

- **System-oriented**
  - **Effective and efficient utilization of the processor**
    - E.g. throughput,
    - processor utilization
Short-Term Scheduling Criteria

- Performance related
  - Quantitative
  - Measurable
    - E.g. response time and throughput
- Not performance related
  - Qualitative
  - Can’t measure easily
    - E.g. predictability, fairness
Scheduling criteria are interrelated

- The scheduling criteria are interdependent
  - E.g. good response time may require frequent process switching
  - This would mean a lot of overhead
  - Hence lower throughput
Priority-base scheduling policies

- Scheduler will always choose a process of higher priority over one of lower priority.
- Have multiple ready queues to represent each level of priority.
- Lower-priority processes may suffer starvation.
  - allow a process to change its priority based on its age or execution history.
Figure 9.4  Priority Queuing
Alternative scheduling policies

- Require a selection function
  - Determines which among ready processes to select
  - Function may be based on resource management or execution characteristics of processes
- three quantities are important
  - $w$ – time spent in system, waiting and executing so far
  - $e$ – time spent in execution so far
  - $s$ – total service time required by the process
    - the time that the process actually runs (includes $e$)
Alternative scheduling policies

- Use **decision mode** to specify the instants in time at which the selection function is exercised
  - **Nonpreemptive**
    - Once a process is in the running state, it will continue until it terminates or blocks itself for I/O
    - Selection function used only when a process blocks or terminates
  - **Preemptive**
    - Running process may be interrupted and moved to ready queue
    - Allows for better service since any one process cannot monopolize the processor for very long
    - More switching overhead
    - Selection function is used
FCFS/FIFO policy

- Each process joins the ready queue
- When the current process ceases to execute, the oldest process in the ready queue is selected
  - nonpreemptive
- A short process may have to wait a very long time before it can execute
## FCFS example

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival time</th>
<th>Burst length</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0</td>
<td>24</td>
</tr>
<tr>
<td>P2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>P3</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Calculate turnaround time for each process and the average turnaround time

Turnaround time = total time process spends in the system (wait + service)
FCFS

- Favors CPU-bound processes
  - I/O processes have to wait until CPU-bound process completes
- Rarely used on its own
Process Scheduling Example

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Service Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>E</td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>
Round robin scheduling policy

- Uses preemptive decision mode
  - Preempt a process using a clock interrupt
- A process is allowed to run for a time-slice or quantum
- The next process in the ready queue is selected to run following the preemption (FIFO)
- Prevents large processes from monopolizing the processor
- Known as time slicing
Round robin scheduling policy

- Design issue
  - length of quantum
- Short quantum
  - short processes will finish quickly
  - lots of process switches
- Choose such that typical processes can complete without being interrupted
Round robin scheduling policy

- Unfair to I/O-bound processes
  - They typically do not use a complete quantum
  - CPU-bound processes do
- Use virtual round robin
  - Use an auxiliary queue to hold processes that have I/O requests serviced
  - Process in auxiliary queue gets preference over those in the main ready queue
  - Process from the auxiliary queue runs for remaining time from last time-slice
Figure 9.7  Queuing Diagram for Virtual Round-Robin Scheduler
Shortest-Process-Next (SPN)

- Uses nonpreemptive decision mode
- Selects process with shortest expected processing time next
- Reduces the response time for shorter processes
- Less predictable response time for longer processes
- Must estimate the running time for each process
Shortest Remaining Time (SRT)

- Preemptive version of SPN
- Check when a new process enters the system and when the current process voluntarily releases the processor
- Better response time as short jobs given immediate preference
- Fewer process switches than Round robin
- Must keep track of elapsed time
Highest Response Ratio Next (HRRN)

Response ratio = \frac{\text{wait} + s}{s}

wait – time spent waiting for the processor
s – expected service time
HRRN scheduling policy

- Uses nonpreemptive decision mode
- Select the process that has the highest response ratio
- Short process (small “s”) given preference
- Long process (large “s”) that has been waiting a long time (large “wait”) is given preference
- Accounts for age of process
- Must estimate “s”
Multi-level Feedback scheduling

- Uses preemptive decision mode (time quantum)
- Maintain many ready queues of varying priorities
  - Each queue uses FIFO
  - A process that is preempted is moved to a lower priority ready queue
  - Thus longer processes are penalized
- Don’t have to estimate running time
- Note: If there are no processes waiting, don’t preempt current process
Figure 9.10  Feedback Scheduling
Multi-level Feedback (quantum = $2^i$)

- With quantum = 1
  - long processes will have very large turnaround time
- In this case
  - a process in queue RQi will run for quantum = $2^i$
Fair-Share Scheduling

- Treat multiple processes and threads of a user as a collection
- The performance of the collection of threads is optimized not just that of any one thread or process
- Fair share scheduling ensures that users are given a fair share of the processor time
Fair-Share Scheduling

- Divide user community into groups
- Each user process’ priority is based on the process’ execution time and the execution time of processes within the same group