
Consider the following set of processes, with the length of the CPU burst given in milliseconds:

<table>
<thead>
<tr>
<th>Process</th>
<th>Burst Time</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>P₂</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P₃</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>P₄</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>P₅</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

The processes are assumed to have arrived in the order {P₁, P₂, P₃, P₄, P₅} all at time 0. Consider the following scheduling algorithms: first-come, first-serve (FCFS), shortest-job first (SJF), nonpreemptive priority (a smaller number implies a higher priority), round-robin (RR) with quantum=1, and RR with quantum=2.

1) For each of the algorithms above:
   • Draw a Gantt chart to illustrate the execution of these processes
   • Calculate individual process turnaround times and the average turnaround time
   • Calculate individual waiting times and average waiting time

2) Considering just the data that you were provided, which is the best scheduling algorithm? Justify your choice by explaining what criteria you used to identify what you called the best.

3) In a general sense, can any of these scheduling algorithms result in process starvation?

4) How realistic is it to assume that the scheduling algorithm will have the length of CPU bursts for the processes that it will need to schedule? How/where can one get information on the length of CPU bursts for a CPU scheduling algorithm to use?

5) How would your Gantt charts and metrics change if we assumed the following arrival times:
   • P₁ arrives at time 0
   • P₂ arrives at time 8
   • P₃ arrives at time 8
   • P₅ arrives at time 9
   • P₄ arrives at time 10

6) How would your Gantt charts and metrics above change for preemptive priority scheduling?