**Name: Rituben Patel**

**Scheduling Method: pPH**

**Table of Contents:**

|  |  |
| --- | --- |
| No | Content |
| 1 | Description of Scheduling algorithm |
| 2 | Description of Implementation |
| 3 | Experiments |
| 4 | Conclusion |
| 5 | References |

1. **Description of Scheduling Algorithm**

In preemptive priority high scheduling algorithm, priority is associated with each process, and the CPU is allocated to the process with the highest priority. Higher number means higher priority and lower number means low priority according to arrival time and burst time also given. Preemptive task that means break the execution any time. Here we are going to break execution because it might happen that the process which we are currently running at the same time there is new process coming too and priority of that new process is highest than the currently running process. So we have to stop currently running process and give the priority to highest priority process. We have given quantum value, so we execute all processes based on that quantum value with same scenario.

Example:

|  |  |  |  |
| --- | --- | --- | --- |
| **Process** | **Priority** | **Arrival Time** | **Burst Time** |
| 1 | 2(L) | 0 | 4 |
| 2 | 4 | 1 | 2 |
| 3 | 6 | 2 | 3 |
| 4 | 10 | 3 | 5 |
| 5 | 8 | 4 | 1 |
| 6 | 12(H) | 5 | 4 |
| 7 | 9 | 6 | 6 |

**Gantt Chart**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **P1** | **P3** | **P4** | **P6** | **P4** | **P7** | **P5** | **P3** | **P2** | **P1** |

0 2 4 6 10 13 19 20 21 23 25

At arrival time 0, we have process 1, we will execute this process equal to the amount of quantum value. Suppose quantum is 2 here. After that, according to timer, we will see how many processes we have in ready queue to execute and we will choose one with high priority process 3 here. Same scenario for all processes.

1. **Description of Implementation**
2. Read input.data and scan each value. Store number of processes in numProcess and so on.
3. Initialize PCB for each process with arrival time, burst time, priority and process number.
4. Make array of PCB which stores the each Process.
5. Sort PCB array according to arrival time of processes.
6. Make other array readyQueue of type Object from PCB, which is also sorted according to arrival time of processes.
7. Set timer according to value of quantum.
8. Processes whose arrival time is less or equal to timer, store them into readyQuePri array.
9. Find maximum priority process from readyQuePri array. Execute that process number of times equal to the value of quantum.

* If that process’s cpu\_burst is higher or equal to quantum value than execute that process number of times equal to the value of quantum. Adjust timer to the addition of time and process’s cpu\_burst. Adjust cpu\_burst of that process.
* If that process’s cpu\_burst is less than quantum value than execute that number of times equal to its cpu\_burst . Adjust timer and make cpu\_burst to zero.
* If process’s cpu\_burst is zero than remove that process from the readyQueue and add next higher priority process with accordance of quantum.
* If cpu is idle at any time, adjust value of timer and do nothing.

1. Repeat step 7, 8 until readyQueue become empty.
2. When readyQueue gets empty, Write output in output.data file.
3. **Experiments**

**Experiment # 1:**

1. **INPUT DATA:**

**“input.data”4**

**4**

**1 1**

**0 8 5**

**1 4 1**

**2 9 4**

**3 5 3**

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Arrival Time | Burst Time | Priority |
| P1 | 0 | 8 | 5 |
| P2 | 1 | 4 | 1 |
| P3 | 2 | 9 | 4 |
| P4 | 3 | 5 | 3 |

**b)Gantt Chart:**

|  |  |  |  |
| --- | --- | --- | --- |
| **P1** | **P3** | **P4** | **P2** |

0 8 17 22 26

**c)Average Waiting Time:**

**=[** P1(0) + P2(22-1)+P3(8-2)+P4(17-3)**]/**4

**=[** 0 + 21 + 6 + 14**]/**4

**=10.25**

**Average Response Time:**

**=[** P1(0) + P2(22-1) + P3(8-2) + P4(17-3) **]/**4

**=[** 0+21+6+14**]/**4

**=10.25**

**d) SHOULD-BE Output e) MY Output (produced by program):**

**“output.data” My “output.data”**

**0 8 1**

**8 17 3**

**17 22 4**

**22 26 2**

**0 8 1**

**8 17 3**

**17 22 4**

**22 26 2**

**Experiment # 2:**

1. **INPUT DATA:**

**“input.data”**

**4**

**1 3**

**0 5 2**

**4 8 1**

**6 2 4**

**8 6 3**

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Arrival Time | Burst Time | Priority |
| P1 | 0 | 5 | 2 |
| P2 | 4 | 8 | 1 |
| P3 | 6 | 2 | 4 |
| P4 | 8 | 6 | 3 |

Quantum = 3

**b) Gantt Chart:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P1** | **P2** | **P3** | **P4** | **P2** |

**0 5 8 10 16 21**

**c) Average Waiting Time:**

**=[** P1(16-5) + P2(5+18-5-4) + P3(8-6) +P4(10-8) **]/**4

**=[**11+14+2+2**]/**4

**=7.25**

**Average Response Time:**

**=[**P1(0) + P2(5-4) + P3(8-6) + P4(10-8)**]/**4

**=[**0+1+2+2**]/**4

**=1.25**

**d) SHOULD-BE Output e) MY Output (produced by program):**

**“output.data” My “output.data”**

**0 5 1**

**5 8 2**

**8 10 3**

**10 16 4**

**16 21 2**

**0 5 1**

**5 8 2**

**8 10 3**

**10 16 4**

**16 21 2**

**Experiment # 3:**

1. **INPUT DATA:**

**“input.data”**

**4**

**1 2**

**0 7 4**

**3 1 2**

**6 2 1**

**8 6 3**

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Arrival Time | Burst Time | Priority |
| P1 | 0 | 3 | 2 |
| P2 | 4 | 1 | 1 |
| P3 | 6 | 2 | 4 |
| P4 | 8 | 6 | 3 |

**Gantt Chart:**

|  |  |  |  |
| --- | --- | --- | --- |
| **P1** | **P2** | **P4** | **P3** |

**0 7 8 14 16**

**Average Waiting Time:**

**=[** P1(0) + P2(7-4) + P3(14-6) +P4(8-8) **]/**4

**=[**0+3+8+0**]/**4

**=2.75**

**Average Response Time:**

**=[P1(0) + P2(7-4) + P3(14-6) + P4(8-8)]/4**

**=**[0+3+8+0]/4

**=2.75**

**d) SHOULD-BE Output e) MY Output (produced by program):**

**“output.data” My “output.data”**

**“output.data”**

**0 3 1**

**4 5 2**

**6 8 3**

**8 14 4**

**“output.data”**

**0 3 1**

**4 5 2**

**6 8 3**

**8 14 4**

1. **Conclusion**

* **It minimizes the average waiting time of the processes and maximizes CPU utilization.**
* **If a high priority process frequently arrives in the ready queue, low priority process may starve.**

1. **References**

* <https://www.techopedia.com/definition/21478/priority-scheduling>
* <http://ijcsmc.com/docs/papers/September2015/V4I9201563.pdf>