William Giddins Operating Systems

12-19-18

HW 3: CPU Scheduling

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Avg Turnaround | Avg Wait Time | Longest Wait  Time | Total Run | Total Context  Switches |
| FCFS | 3971 | 3681 | 6742 | 7524 | 13 |
| SJN | 2983 | 2844 | 5942 | 6464 | 13 |
| Priority | 3253 | 3114 | 6194 | 6664 | 23 |
| RR FCFS | 3545 | 3299 | 5942 | 6371 | 64 |
| RR SRTN | 3442 | 3107 | 5942 | 6464 | 13 |
| RR Priority | 3253 | 3114 | 6194 | 6664 | 23 |

Out of all the CPU scheduling algorithms my implementation found that the one with the longest run time was FCFS and the shortest was RR FCFS. The most responsive was SJN and the least was FCFS. The one with the longest avg wait time was FCFS and the least was SJN. The algorithm with the longest total wait time was FCFS and the shortest was SJN. The reason is because in FCFS the last process to get into the queue must wait a considerable amount of time before it even gets to run even once. The best SPU utilization was SRTN and the worst was RR FCFS.

I came across many issues throughout the implementation of my scheduler. In the end I was unable to obtain both pre-emption and non-pre-emption correctly. It seems that every time I solved a problem that would allow me to run the scheduler with non-preemption it would ruin my execution of pre-emption algorithms and vice-versa. It was mainly due to how I implemented the processes arriving into the ready queue and how I dealt with the scheduler only having the first process in the queue. As a result, some of the algorithms have the same if not similar results to their non- preemptive or pre-emptive counterparts. Another issue I was unable to solve was determining when a context switch was occurring within the loop I created. As it is now every pass is assumed to be a context switch which is incorrect and I had to subtract error from the results.