CPU Scheduling

RJ Straiton

April 27, 2025

1 Abstract

This is the final report document for the CPU Scheduling Simulation project for my Operating Systems class at Kennesaw State University. This project had students write and measure scheduling algorithms to reinforce learning and see how the algorithms compare.

2 Introduction

The CPU Scheduling Simulation aimed at ensuring Operating Systems students learned about the different types of CPU Scheduling algorithms and how they work. Students were instructed to implement two different scheduling algorithms and measure their performance. The performance metrics are average wait time (AWT), average turnaround time (ATT), CPU utilization, and throughput.

Students were given the choice of whether they would start from scratch, or add to a pre-existing program. I wanted to add to the pre-existing program, but unfortunately I ran into several issues when attempting to get the code running on my virtual machine. For the scheduling algorithms, I decided to implement Shortest Remaining Time First (SRTF) and Multi-Level Feedback Queue (MLFQ).

3 Implementation Details

For the shortest remaining time first algorithm, there are two separate methods, the only difference being the parameters that the method takes in. One method is allowing the user to input the arrival time and burst time for each of the processes that they chose to run. The other method is used for the test values I created and it takes in an array of MyProcess, which is a simple class I created for both the SRTF and MLFQ algorithms to use. The SRTF algorithm will take an array of processes, sort the array by arrival time, and enter a while loop which will end when the number of completed processes is equal to the total number of processes. Then, we check to see if the remaining time of the current process is less than the remaining time of another process. When a process completes, we set the completion time to the current time and calculate both turnaround time and waiting time. At the end of the algorithm, we calculate total wait time and total turnaround time by going through each process in the array with a foreach loop and print it to the console. The algorithm was developed based on some Java code I found on a GeeksForGeeks article which I will of course credit in the references section in this document.

As with the SRTF algorithm, there are two separate methods for the multilevel feedback queue algorithm. The differences are the same as before, with one method being used for users, and the other for testing the algorithm. The MLFQ algorithm will take in an array of MyProcess, create an array of four queues, and put the array of processes into the first queue in the array, that being the highest priority one. This is because new processes are assigned to the highest priority. The first process in the queue will be executed and, if it does not finish, it will be dequeued and added to the next queue. This would continue until a process goes into the lowest queue, and eventually finishes executing. When a process finishes, I made it so that it would be dequeued from its current queue and added to a new queue called completedProcesses. This is so that, when all queues in the array are empty, I can easily calculate the performance metrics of the algorithm. This algorithm was inspired by a similar Python algorithm I found on how.dev, which will of course be credited later in this document.

4 Testing and Results

In order to compare the SRTF and MLFQ algorithms, I gave them both the same array of 5 processes. The details for each process can be found in the table below, as well as the results of each algorithm.

Process	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	10
P4	5	3
P5	8	22

Table 1: Process information

5 Analysis and Discussion

In Table 4, we can see that Shortest Remaining Time First has a far lower average waiting time, as well as a lower average turnaround time when compared to Multi-Level Feedback Queue. The difference can be seen when looking at Table 2, which shows that two processes have a wait time of zero under SRTF. MLFQ has consistently big waiting times as we can see in Table 3.

Process	Completion Time	Wait Time	Turnaround Time
P1	15	7	15
P2	5	0	4
P3	25	13	23
P4	8	0	3
P5	47	17	39

Table 2: Test results for SRTF algorithm

Process	Completion Time	Wait Time	Turnaround Time
P1	23	23	23
P2	8	6	7
P3	33	29	31
P4	15	5	10
P5	47	31	39

Table 3: Test results for MLFQ algorithm

Algorithm	Avg. Wait Time	Avg. Turnaround Time	Throughput
SRTF	7.4	16.8	0.1064
MLFQ	18.8	22	0.1064

Table 4: SRTF vs. MLFQ final results

The two algorithms surprisingly have the exact same throughput. If we look at Table 2 and Table 3, Process 5 has a completion time of 47 in both algorithms. Process 5 also has the same turnaround time in both algorithms. I was surprised to see this similarity as I thought all of the data across the tables would be noticeably different. I believe that this similarity is due to Process 5's long burst time, which seems to be affecting both of the algorithms.

Ultimately, I believe that when dealing with a smaller number of processes, Shortest Remaining Time First is the better choice. SRTF had a lower average turnaround time, and a far lower average waiting time when compared to Multi-Level Feedback Queue.

6 Challenges and Lessons Learned

As mentioned earlier, I wanted to add onto the pre-existing code because I thought it would be good practice for working on legacy code, which I very likely will do in the future. However, I kept getting errors saying that I did not have the necessary dependencies to run the application through Visual Studio. I could manually run the exe and open the application, but it would crash after taking in my inputs. So I unfortunately had to start from scratch.

Unfortunately, I did not have time to implement a way of tracking the CPU utilization, which was a required metric for measuring the algorithms. My project for Software Project Management, as well as several other class projects, has taken up a large portion of my time recently, so I currently only have average wait time, average turnaround time, and throughput implemented.

7 Conclusion

I think that this project really helped to deepen my understanding of the CPU Scheduling algorithms, as well as their strengths and weaknesses. Learning in class is one thing, but I always learn best by actually programming what we learn. Not many classes in the Software Engineering program give coding projects, so I always appreciate having more opportunities to code and further my skills as a developer. Measuring the two algorithms was a very interesting way to see how they perform and how their individual performance compares to each other.

8 References

"Shortest Remaining Time First (Preemptive SJF) Scheduling Algorithm." Shortest Remaining Time First, GeeksforGeeks, 3 Feb. 2025, www.geeksforgeeks.org/shortest-remaining-time-first-preemptive-sjf-scheduling-algorithm/.

"What Is Multilevel Feedback Queue Scheduling?" HowDev, how.dev/answers/what-is-multilevel-feedback-queue-scheduling. Accessed 25 Apr. 2025.