Project Report #5

Scheduling Algorithms

Sujeet Ojha
sxo160430@utdallas.edu
April 23, 2020

I. Problem Statement

The assigned task was to implement, explore and analyze the distinct process scheduling algorithms. There was leeway of choosing between first-fit, best-fit, next-fit, and worst-fit in order to perform memory scheduling. Four algorithms: first come first serve (FCFS), shortest process next (SPN), shortest remaining time (SRT), and round robin with quantum of one (RRQ1) were required to be analyzed. Furthermore, it was required to run these algorithms a thousand times for a total of a thousand tasks each iteration in order to conclude the average turn-around time and the average relative turn-around time for every algorithm.

II. Importance of Scheduling Algorithms

Process scheduling is used to ensure that an operating system executes processes in an efficient manner with reduced wait times. The goal, in the short term, is to optimize system behavior. An efficient scheduling algorithm focuses on enhancing system behavior, along with enforcing user and system criteria. Although these criteria vary based on system-to-system needs, a common goal is to maximize response time and ensure throughput is increased.

III. Algorithm Usage

The four different algorithms described in this report are very different in the way they proceed to schedule the processes. The FCFS algorithm follows a queue-based ordering where the process with the minimal arrival time is scheduled for execution first. This non-preemptive algorithm ensures that once a task enters a running state, it is carried out until completion. The SPN algorithm is similar to the FCFS algorithm in the way that it is non-preemptive. However, this algorithm chooses to schedule the process with the lowest duration first. The SRT algorithm is a pre-emptive algorithm that allocates the process that is closest to competition. However, a process can be preempted by a new process with shorter competition times. The final algorithm is RRQ1, which provides each process a fixed quantum of execution time. Each process is executed in bursts for a certain amount of time and context switching is used.

This report will utilize the first-fit algorithm each time a new task needs to be placed in memory in order to ensure that processes are being placed in memory the same way for all four algorithms. The first-fit algorithm was chosen due to its simplicity, along with its reliability. Since the goal of this report is used analyze the differences between the scheduling algorithms, having a simple memory scheduling algorithm allowed for simpler coding techniques to be utilized.

This report will run each algorithm an equal amount of times with an equal number of tasks each time in order to ensure results are unbiased. The report will also state the average turn-around time and the average relative turn-around time for each scheduling algorithm.

III. Approach

The programming language selected for this project is C. The approach taken was to first generate a thousand random task streams consisting of the size needed for allocation and duration. Keeping the limited units of memory available into account, the first fit algorithm was used to place the tasks in memory. Compaction sequence was not utilized in order to avoid the issue of continuous compaction. The tasks in memory were scheduled using the process scheduling algorithms. This sequence of tasks was repeated until no processes were left in memory and all processes had been scheduled. Each algorithm scheduled a thousand processes, a thousand times in order to ensure accurate data regarding the average turn-around and the average relative turn-around time was collected.

IV. Solution

Since the program dealt with physical memory, and not virtual memory, the examples provided in the book could not be followed for every algorithm. The true limitation of this program comes from the limited memory size. The program first creates an array of a thousand structures and fills the size and duration with randomized values from 1-16. Once the task stream is created, the program creates an array that represents the 56 units of memory. The program then processes the task stream by using the first-fit algorithm. If multiple processes are able to be placed one after the other, their arrival time is the same. This is because the time it would take to place processes in memory without having to create space for them is marginal.

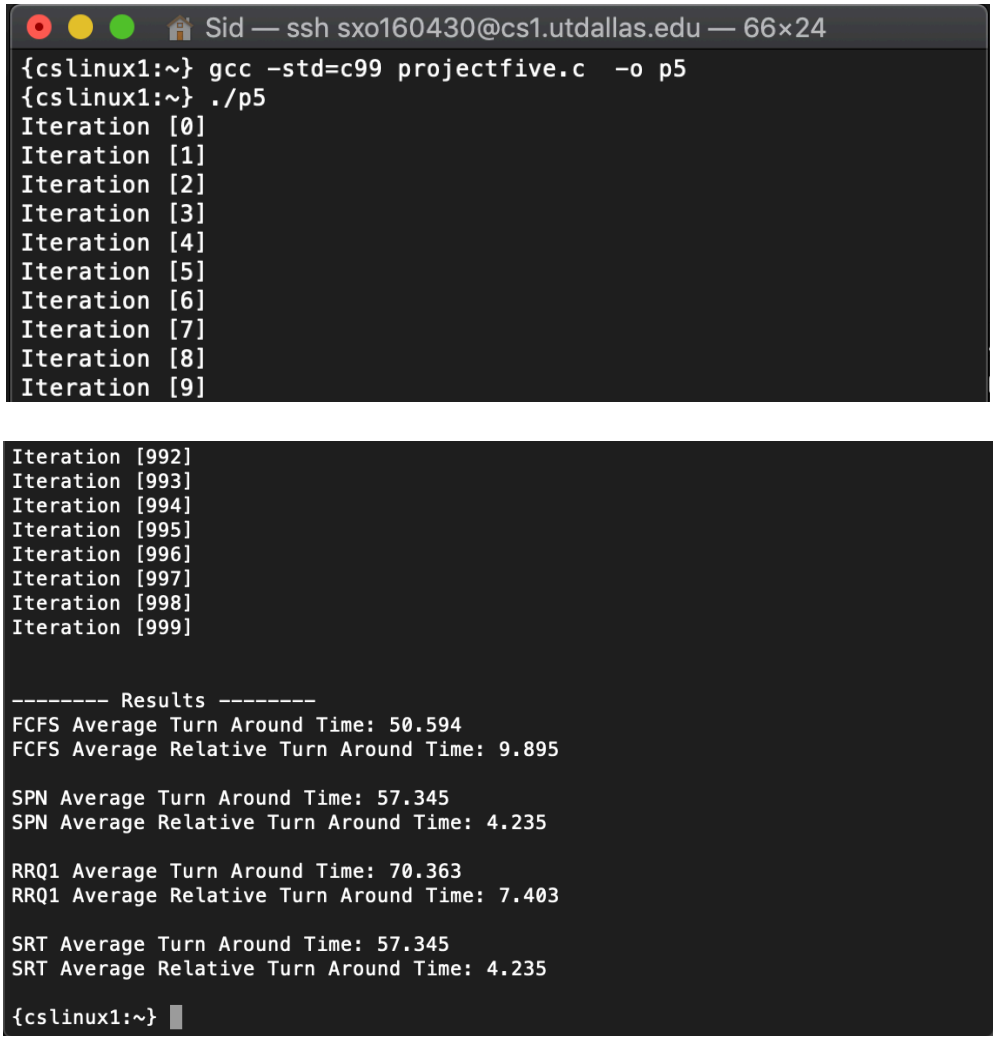
If a process could not be placed in memory, a process scheduling algorithm was used in order to free up space in memory. Each time a process was removed from memory, the turnaround time, and the relative turnaround time was calculated using data collected previously.

The FCFS algorithm and the SPN algorithm, being non-preemptive algorithms, provided the expected results. RRQ1 and SRT, on the other hand, were affected by the limitation of space in memory. As a result, the SRT algorithm performed a lot like the SPN algorithm. Similarly, process bursts for the RRQ1 algorithm didn't follow the exact same technique as it would have in infinite memory.

The information regarding the average turn-around time and the average relative turn-around time of time was then used to analyze each algorithm and make conclusions. Furthermore, in order to explore the algorithms in greater depth, the algorithms were run with a different number of iterations and task stream sizes. A plot was then created using the various amount of data gathered in order to visualize the results.

Figures:

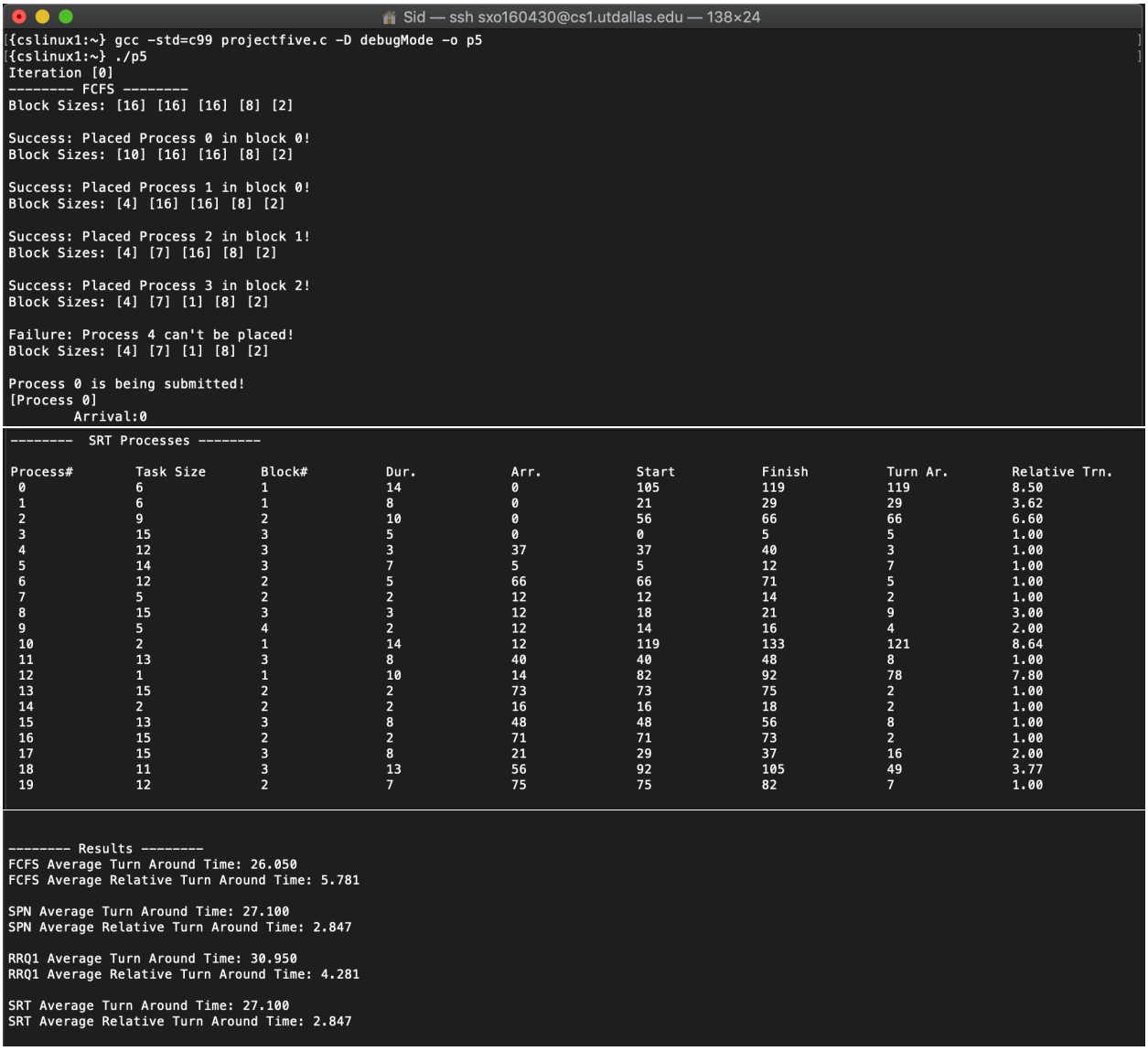
- *Plot (1):* Plot (1) depicts the build and execution of the program in the CS1 Linux server.
- *Plot (2):* Plot (2) depicts the build and execution of the program in debug mode in the CS1 Linux server.
- *Plot (3):* Plot (3) depicts a plot of the turnaround time results with distinct task stream sizes.
- *Plot (4):* Plot (4) depicts a plot of the relative turnaround time results with distinct task stream sizes.



Plot (1): Build and Execution of the program

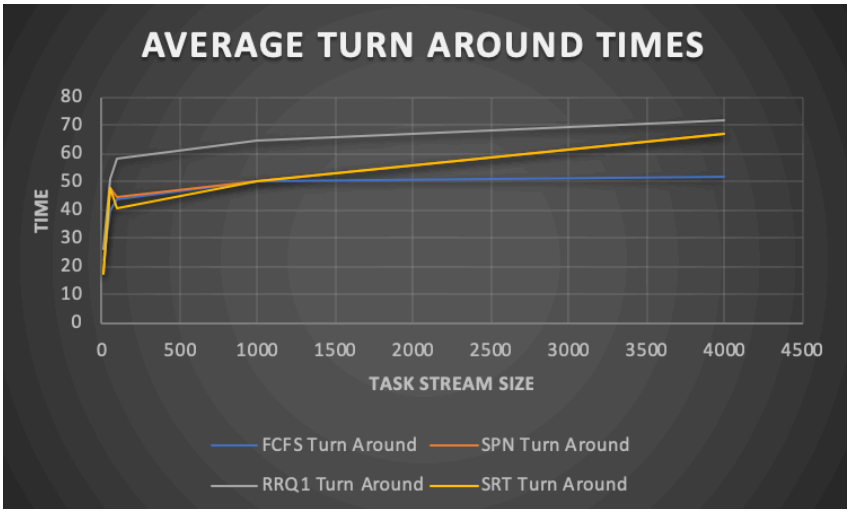
The CS1 Linux server was used to build and execute the code. The program was traditionally compiled without added statements. The program displays the final results that make it easy for the user to analyze the average turn-around time, and the relative turn-around time for each algorithm.

Note: The program can also be executed on the CS2 Linux server.



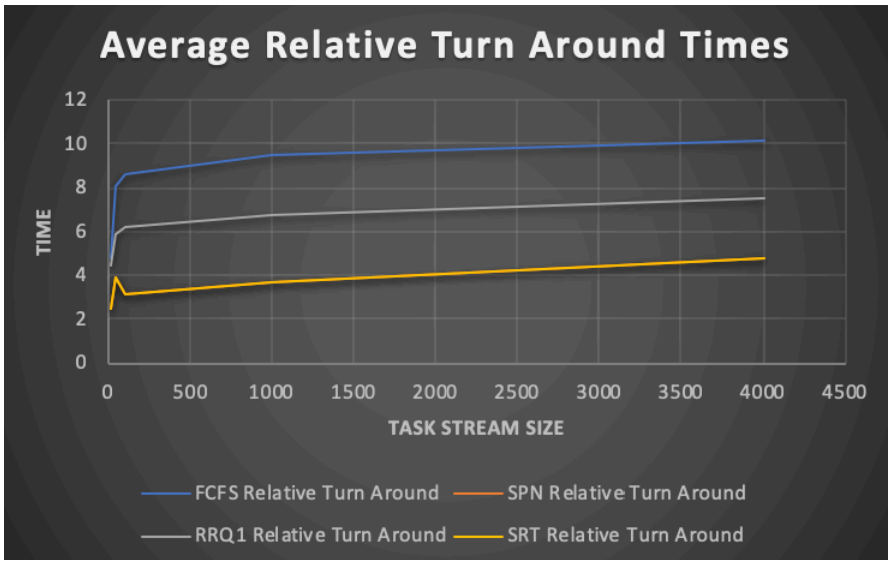
Plot (2): Build and Execution in debug mode

The CS1 Linux server was used to build and execute the code in debug mode. In order to execute the code in debug mode, "-D debugMode" should be added. In debug mode, the program performs only one iteration for a set of 20 tasks. Important steps in the algorithm are displayed for the user, along with a visual description of the process' contents.



Plot (3): Average turn around times

The plot above depicts the turn around times for various task stream sizes. The turn around time for all algorithms increase as the task size increases. There is a ceiling to this growth as task sizes grow exponentially. For the smaller task streams, the turn around times of all algorithms are very close together. However, as the number of tasks increases, the differnces are made more apparent.



Plot (4): Average relative turn around times

The plot above depicts the relative turn around times for various task stream sizes. The behavior is fairly different than the behavior presented for turn around times. Relative turn around times also seem to grow marginally as the number of tasks increase. However, the behavior regarding which algorithm results in the lowest relative turn around time follows the same pattern.

V. Observations

Analyzing the results, a few notable observations can be made.

The SPN scheduling algorithm has the same results as the SRT scheduling algorithm. This is because the program depicts and works with physical memory, not infinite virtual memory. Since the scheduling algorithm works with processes that are already in memory, and a process cannot arrive and be placed in memory until another process is scheduled to make space, there is no distinction between the two algorithms. As a result, the pre-emptive capabilities of SRT cannot be depicted accurately.

The RRQ1 algorithm also has the highest turn around times, even for smaller task streams. This is because the quantum of one drags out the execution process and prevents processes from finishing shorly after execution. As a result of the processes being sent out in bursts, the finish time increases and that in turn increases the turnaround time.

The relative turn around time and the turn around time are very different for all algorithms. Before implementing a process scheduling algorithm, one must carefully examine system specifications and make a decision between which of the two units is to be considered more important.