

Program 5

1. Problem Statement

The goal of this project was to analyze four scheduling algorithms to determine which algorithm had the best average turn around time and best average relative turn around time. The four algorithms are First Come First Serve (FCFS), Round Robin (RRq1), Shortest Process Next (SPN), and Shortest Remaining Time (SRT). The project requires us to generate a task stream of 1000 tasks, the tasks then are to be partitioned in memory using the memory portioning algorithm of our choice. The tasks are then scheduled using different scheduling algorithms and their results of their turn around time are presented.

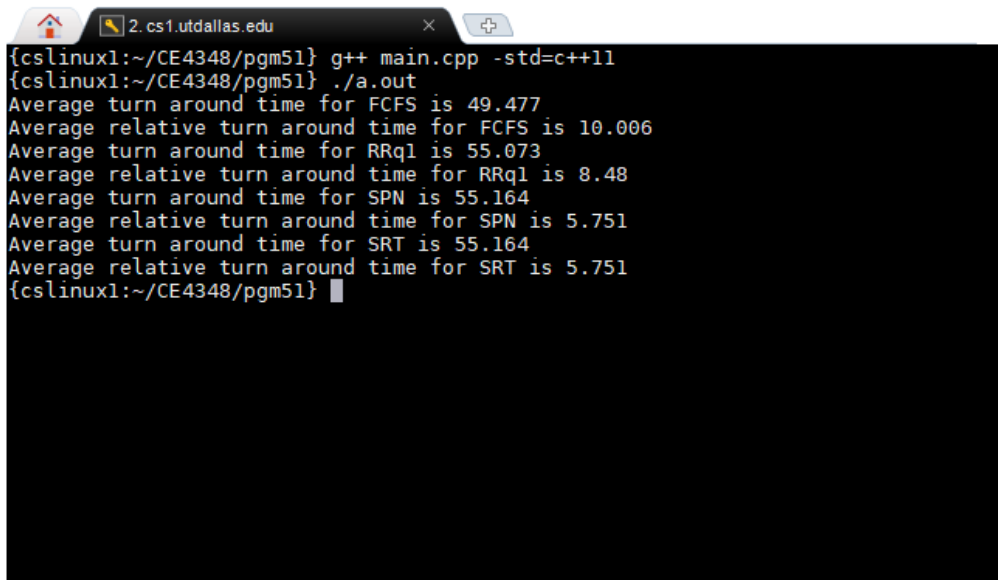
2. Approach to Solution

A struct was used to hold the information of each task in the info task stream. The struct held information such as size, duration, and additional elements that helped keep track of each task's information. An array was used to represent 56 units of available memory. The memory was separated into blocks of size 16 units. The setTasks() function was used to randomize the information of the tasks. The FCFS function performed its scheduling by processing each task as it arrives, and since this method is non-preemptive each task is processed to its completion. The RRq1 function will give each task a quantum of one or a time unit of one, and the tasks are processed are executed in bursts. SPN is also non-preemptive where the shortest process available next is processed. Similarly, SRT processes tasks that have the shortest remaining time next, this method is used to avoid starvation that usually occurs in SPN. The main function performs 1000 experiments of each algorithm and determines the average turnaround time and the average relative turnaround time. This program was developed in C++, and Clion was the IDE used.

3. Solution Description

The solution to this program is as expected, with SPN and SRT having similar results since SRT is an augmented version of SPN that accounts for starvation. RRq1 resulted in the longest turnaround time and relative turnaround time due to the quantum of 1. If the quantum was higher the results of this algorithm would be far lower. The FCFS resulted in the lowest time which was expected.

To run the program in cslinux1 use the following commands:

A terminal window titled '2.cs1.utdallas.edu' showing the execution of a C++ program. The user runs 'g++ main.cpp -std=c++11' and then './a.out'. The output displays performance metrics for four scheduling algorithms: FCFS, RRq1, SPN, and SRT. For each algorithm, both 'Average turn around time' and 'Average relative turn around time' are shown. The results indicate that SRT has the lowest values for both metrics, followed by SPN, RRq1, and FCFS.

```
{cslinux1:~/CE4348/pgm51} g++ main.cpp -std=c++11
{cslinux1:~/CE4348/pgm51} ./a.out
Average turn around time for FCFS is 49.477
Average relative turn around time for FCFS is 10.006
Average turn around time for RRq1 is 55.073
Average relative turn around time for RRq1 is 8.48
Average turn around time for SPN is 55.164
Average relative turn around time for SPN is 5.751
Average turn around time for SRT is 55.164
Average relative turn around time for SRT is 5.751
{cslinux1:~/CE4348/pgm51} █
```

Figure: Example of code being run in cslinux1

The program functions as expected on the cslinux1 server. However `-std=c++11` was required for the program to be executed. The program will take a few moments to complete.

The results lead me to choose SRT to be the best scheduling algorithm since it has a good turnaround time and has the best relative turn around time. SRT is a better scheduling algorithm as opposed to SPN is due to it being preemptive. FCFS has a good turnaround time but a pretty bad relative turn around time. RR would have presented better results if the quantum was higher than one.