CPU Scheduling Simulation

Dennis Chen <chend12> Sebastian Martinez <martis10> Ethan Tuttle <tuttle2>

1. The best algorithm for CPU-bound processes is the Shortest Job First (SJF) algorithm. This is because it has the shortest wait times and turn around times among all four of the algorithms. It also has the most CPU utilization of the 4 algorithms. The algorithm that is best suited for I/O-bound processes is once again the Shortest Job First algorithm. The reason why again is the fact that it has the shortest wait time among all the 4 algorithms. The results from one simulation run are shown in **Figure 1**. There you can see that the SJF algorithm has lower average times , fewer context switches, and more CPU utilization.

Figure 1

Student simout.txt

```
Algorithm FCFS
-- average CPU burst time: 84.304 ms
-- average wait time: 215.423 ms
-- average turnaround time: 303.726 ms
-- total number of context switches: 537
-- total number of preemptions: 0
-- CPU utilization: 57.779%
Algorithm SJF
-- average CPU burst time: 84.304 ms
-- average wait time: 199.345 ms
-- average turnaround time: 287.648 ms
-- total number of context switches: 537
-- total number of preemptions: 0
-- CPU utilization: 59.366%
Algorithm SRT
-- average CPU burst time: 84.304 ms
-- average wait time: 216.981 ms
-- average turnaround time: 305.851 ms
-- total number of context switches: 613
-- total number of preemptions: 76
-- CPU utilization: 58.679%
Algorithm RR
-- average CPU burst time: 84.304 ms
-- average wait time: 229.361 ms
-- average turnaround time: 320.108 ms
-- total number of context switches: 865
-- total number of preemptions: 328
-- CPU utilization: 58.355%
```

2. There are different values of α that produces the best results for the SJF and SRT algorithms. For the SJF algorithm, the best values of alpha would either be towards an extreme end (alpha being 0.1 or 0.9) or towards the middle (alpha being 0.5) as seen in **Figure 2**. When alpha is any of these values for SJF, they produce the results with higher CPU utilization and better overall times. The best alphas for the SRT algorithm would be either 0.7, 0.3 as those alphas give the best times and CPU utilization as seen in **Figure 3**.

Figure 2

Algorithm SJF (alpha 0.1)
-- average CPU burst time: 84.304 ms
-- average wait time: 199.091 ms
-- average turnaround time: 287.395 ms
-- total number of context switches: 537
-- total number of preemptions: 0
-- CPU utilization: 60.097%
-- Overall Time: 75330

Algorithm SJF (alpha 0.3)
-- average CPU burst time: 84.304 ms
-- average wait time: 199.993 ms
-- average turnaround time: 288.296 ms
-- total number of context switches: 537
-- total number of preemptions: 0
-- CPU utilization: 58.537%
-- Overall Time: 77338

Figure 3

Algorithm SRT (alpha 0.3)
-- average CPU burst time: 84.304 ms
-- average wait time: 216.233 ms
-- average turnaround time: 304.991 ms
-- total number of context switches: 598
-- total number of preemptions: 61
-- CPU utilization: 60.859%
-- Overall Time: 74387

Algorithm SRT (alpha 0.5)
-- average CPU burst time: 84.304 ms
-- average wait time: 230.251 ms
-- average turnaround time: 318.965 ms
-- total number of context switches: 592
-- total number of preemptions: 55
-- CPU utilization: 58.832%
-- Overall Time: 76950

3. With regard to the SJF and SRT algorithms, the changing from a non-preemptive algorithm to a preemptive algorithm impacts the results by adding a lot of additional wait time, context switches, and turn around times. This is because of how each algorithm is structured; the SJF algorithm just places processes in the ready queue based on expected burst time and grabs the next shortest one when a process finishes its burst. Meanwhile the SRT algorithm will also preempt processes in the CPU if a new process has a shorter expected runtime. The time from those context switches can add up and make the SRT algorithm slower. The comparison of these algorithms can be seen in **Figure 4**.

Figure 4

```
Algorithm SJF
-- average CPU burst time: 903.431 ms
-- average wait time: 507.518 ms
-- average turnaround time: 1414.949 ms
-- total number of context switches: 473
-- total number of preemptions: 0
-- CPU utilization: 42.831%
Algorithm SRT
-- average CPU burst time: 903.431 ms
-- average wait time: 544.928 ms
-- average turnaround time: 1452.638 ms
-- total number of context switches: 506
-- total number of preemptions: 33
-- CPU utilization: 42.502%
```

4. The first limitation of our simulation is the total number of algorithms we are doing. We have tested a lot of the main algorithms, but there are some others that can be implemented in order to better represent the job an operating system has to take on. Some other notable algorithms to test the efficacy of CPU scheduling are the Multilevel Queue Scheduling and the Priority Scheduling algorithms. The second limitation of our simulation is that there is no priority variable. Real operating systems have priorities they must follow to ensure the most important processes are executed first, and not having this factor limits our simulation. The third limitation of our simulation is not working with an I/O subsystem. Without working with a tangible I/O subsystem, many of the wait times will be predetermined based on the seed as opposed to being determined based on different processes.

5. One priority scheduling algorithm of our design can be an algorithm in which a process has a predetermined importance level attached to it and the algorithm bases priority on total completion time and level of importance. The one with a higher priority will go first and the tie breaker will be based on which process is longer. The advantages of it are that more important processes will be done first and that the longer processes will not be CPU starved. However the disadvantages of it are that there may be many context switches depending on if a process is of higher priority. Also, if there is a longer process to be done then the results have the potential to be inconsistent.