Project-Analysis

1. (4 points) Of the four simulated algorithms, which algorithm is the “best” algorithm for CPU-bound processes? Which algorithm is best-suited for I/O-bound processes?

RR is the “best” algorithm for CPU-bound processes. Every process gets an equal share of the CPU. Besides, there is no starvation since RR is cycling in nature.

SRT is best-suited for I/O-bound processes. Processes with short CPU burst time have higher priority, which means I/O bursts are prioritized.

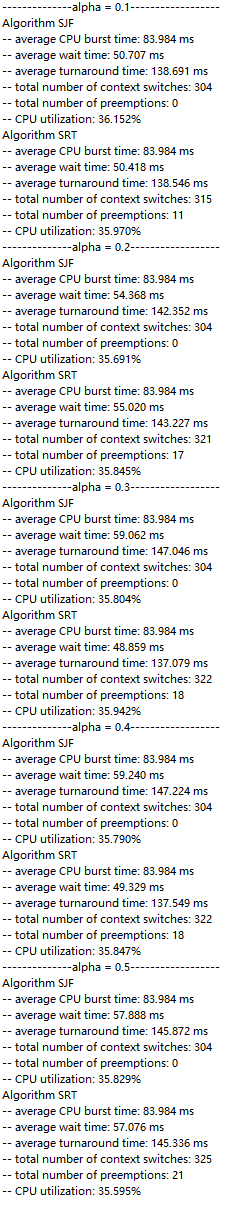
1. (4 points) For the SJF and SRT algorithms, what value of α produced the “best” results?

Argument line: 8 2 0.01 256 4 alpha 128

Text

Description automatically generated with medium confidence

Graph 1



Graph 2

Based on Graph1 and Graph2, both SJF and SRT have the shortest average wait time at alpha = 0.3. Therefore, alpha = 0.3 produces the “best” value.

1. Text

   Description automatically generated(4 points) For the SJF and SRT algorithms, how does changing from a non-preemptive algorithm to a preemptive algorithm impact your results?

Graph 3

Based on the Graph 3 above, we can conclude that the average waiting time of fewer processes is shorter run by SRT. However, SJF may be a little faster than SRT processing more processes like 10 processes.

1. (6 points) Describe at least three limitations of your simulation, in particular how the project specifications could be expanded to better model a real-world operating system.

Limitation 1: We only simulate CPU bursts. I/O bursts are not considered. There could also be some preemptions or something for the I/O bursts Queue.

Limitation 2: Processes with different initial “tau” time are not tested since we can only test one alpha in one test. However, it will be more complex in the real world.

Limitation 3: A seed is set for this random test, so the testing result can be considered as pseudo-randomly. A real-world operating system will be really randomized.

1. (6 points) Describe a priority scheduling algorithm of your own design (i.e., how could you calculate priority?). What are its advantages and disadvantages?

We can create a new I/O burst First algorithm. In this algorithm, we need to sort the ready queue before the process started. Besides, when a new process arrived, we need to check its I/O burst time with the current running process before entering to the ready queue. If it has less I/O burst time than the I/O burst time of current running process, a preemption occurs. Then, the current running process is added back to ready queue.

Advantage: If there are not so many processes, waiting time and turnaround time can be greatly reduced.

Disadvantage: It is not so efficient if there are many processes or the CPU burst time is very long, which is the same as FCFS.