Stephen Sallas
04/18/21
COMP-3500

Solution Description

1. I separated the scheduling mechanism from scheduling policies by implementing an “if” statement in my main function. This “if” statement uses the policy typed in on the command line to choose a policy.

2. I created three separate functions that can each be called when needed.

3. I calculated waiting times by creating a vector called waitingTimes. Then each time a process would finish, I would calculate this process’s waiting time using turnaround time minus burst time. The value would be added to the vector, and then after all processes finish I averaged all values in the vector.

4. I calculated response times by creating vectors called startTimes and responseTimes. Then each time a process would start, I would add the current time to startTimes. Then when a process finished, I would calculate this process’s waiting time using start time minus arrival time. The value would be added to the responseTimes vector, and then after all processes finish I averaged all values in the vector.

5. I calculated waiting times by creating a vector called turnaroundTimes. Then each time a process would finish, I would calculate this process’s turnaround time using current time minus arrival time. The value would be added to the vector, and then after all processes finish I averaged all values in the vector.

6. I used the commonly known parameter “int argc” and “char* argv” to read in command line arguments. I then used a series of “if statements to verify the arguments were correctly formatted.

7. My solution is moderately general. It only assumes the task input file is ordered by arrival times and the PID’s are in order starting at 1.

8. It would be very easy. You would only need to modify the “if” statement in main.

9. Yes, this program checks for input errors with command line arguments and the file input.

10. I did not use any other than class notes.