Allie Wolf

Brian Steele

Cole Walsh

**Project 2:  CPU Scheduler Simulator Report**

The CPU scheduler simulator was designed and programmed in group or paired programming.  These group meetings occurred three to five times per week.

The initial design document was made in group meetings in collaboration and updated only after the code was changed or updated to accommodate improved implementation.

Group coding sessions were attended by all group members 3-5 times a week.

Cole gathered resources and wrote preliminary skeleton classes very early on, focusing on implementation and logic of the main scheduling algorithms.  New classes were written by Allie and group coding sessions were held to add methods and test, debug and improve program utilization.  Allie deserves the bulk of the credit for writing the initial version of the program.  Brian spear-headed the Java documentation and a JFileChooser for choosing scenario files.  He also composed the Log implementation.  The report was curated by Cole.

During these group coding sessions all team members contributed to the continuous testing and debugging of the program.

**Conclusions**

As can be observed in our Excel data file and accompanying documents, we ran a series of tests incorporating several different data scenarios.  We created a ‘typical’ data scenario (similar to what was shown in class), a data scenario with only one process, a data scenario with long burst times, and a data scenario with long CPU bursts as opposed to short IO bursts.  Each algorithm was run in each of the data scenarios--Round Robin twice with two different quantum times for comparison.

In general, when running the program, our output reinforced the concepts discussed in class and in our class resources.  For instance, as expected, with long CPU bursts or both long CPU bursts and IO bursts, the SJF algorithm proved to be the shortest and Round-Robin the longest when it came to average wait time.

In the data scenario which included only one process, we observed several things: average wait time of zero, and CPU utilization went down.  None of the scheduling algorithms made any demonstrative difference--the results were identical.

Our scenario with a balance of processes with CPU and IO bursts that were comparable in length had a high CPU utilization.

Our scenario with high CPU bursts as compared to a typical IO nearly maxed out the CPU utilization.

Our scenario with long bursts of both CPU and IO had CPU utilization in the mid-70s range.  We would have thought CPU utilization would have been higher.

Our scenario with the highest CPU bursts had appropriately long turnaround times as did the scenario where all of the bursts were intentionally made long.

Please refer to log and console output files for further details.

**Resources**

<https://www.geeksforgeeks.org/program-for-fcfs-cpu-scheduling-set-1/>

<https://emunix.emich.edu/~mevett/OS/Assignments/Scheduler/SimpleScheduler_v2.htm>

<http://jimweller.com/jim-weller/jim/java_proc_sched/index.html>

<http://pages.cs.wisc.edu/~mattmcc/cs537/archives/projects-sum01/project2.html>

<https://javahungry.blogspot.com/2013/09/round-robin-scheduling-algorithm-with-example-java-program-code.html>

<https://www.thejavaprogrammer.com/java-program-first-come-first-serve-fcfs-scheduling-algorithm/>

<https://www.thejavaprogrammer.com/java-program-shortest-job-first-sjf-scheduling/>

<https://www.youtube.com/watch?v=R8LYxTT9wmY&ab_channel=CoreySchafer>

<https://www.youtube.com/watch?v=R8LYxTT9wmY&ab_channel=CoreySchafer>

<https://www.youtube.com/watch?v=uFYJh22NQb4&ab_channel=freeCodeCamp.org>

<https://github.com/mtuts/process-scheduling-algorithms>

<https://github.com/Wituz/java-task-queue>

<https://alvinalexander.com/java/jwarehouse/eclipse/org.eclipse.core.jobs/src/org/eclipse/core/internal/jobs/JobQueue.java.shtml>

<https://github.com/rubencal99/Queue.java>

<https://github.com/KyleBenson/CPUSchedulingSimulator/blob/master/src/com/jimweller/cpuscheduler/CPUScheduler.java>

<https://www.youtube.com/watch?v=4Bpg5i4tUFg>

https://stackoverflow.com/questions/2533227/how-can-i-disable-the-default-console-handler-while-using-the-java-logging-api

https://www.codejava.net/java-se/swing/show-simple-open-file-dialog-using-jfilechooser