Cassidy Brown – cmb195
EECS 338
12/6/2016

# Final Project Progress Report (Beta)

## *Project Description*

My project simulates a CPU scheduler. I plan to compare first come, first served (FCFS), shortest job first (SJF), shortest time remaining first (STRF), and round robin (RR) approaches by examining their CPU utilization, throughput, turnaround time, waiting time, and response time.

I use a loop with a counter to simulate the clock and have a queue of "processes," structs that have metadata and an array of burst times to represent how much CPU time they get in one shot. The processes are rotated through, given "CPU time," while collecting data about how long it takes, according to the loop clock.

## *Implementation*

*Interesting Code*

simulator.c — contains the simulation code. A set of "processes" is loaded and

schedules.c — contains the comparison functions which embody the various scheduling approaches. Since the scheduling approach determines how tasks are prioritized and prioritization is manifested in how the processes are sorted in the heap, each comparison function corresponds to an approach.

*Data Structures* (I'm not expected to write these myself, right?)

heap.c — Generic heap implementation for queuing jobs for CPU and IO.
Source: Jason Marcell (jasmarc) on GitHub

queue.c — Generic queue for queuing processes before entering the heaps and after execution is completed. Implementation coming soon!

*Other*

expgen.c — Exponential distribution generator to produce burst times. Will be modified to return single values and/or arrays rather than writing to a file.
Source: Ken Christensen at University of South Florida

## *Sample Output*

The simulator will write meta data to files, which I will analyze and graph in a different program. The files will contain the information about CPU utilization (UT), throughput (THR), turnaround time (TRN), waiting time (WT), and response time (RSP). The simulator will run multiple (10-20) times for each approach, and each file will store the information for one approach, with each line of the file representing an execution of the simulator. Here's an example of an output file, with completely fake data:

```
UT   THR  TRN  WT   RSP
90   6.8  116  22   5
87   7.3  140  27   8
92   6.7  122  20   4
86   7.1  136  26   10
..   ...  ...  ..   ..
```

***Current Progress***

| **File** | **Done** | **To do** |
|---|---|---|
| simulator | - Process struct designed<br>- Main loop partially implemented and otherwise pseudocoded/wireframed | - Finish coding simulator<br>- Writing to the output file |
| schedules | - FCFS and SJF comparison functions written, but those were the easy ones | - STRF and RR are preemptive approaches, so they will take more consideration<br>- Make the function parameters void pointers, as the heap requires. Alternately, change the heap so that it is not generic |
| heap | - Acquired | -- |
| queue | -- | - Find an implementation |
| expgen | - Acquired | - Adapt to not write to file, but simply return a single value or array |