Chris Bechter
02/06/2019
Prof. Fiore
CIS-3702

Readme for Discrete Event Simulator

The discrete event simulator is just that, a model to simulate and event handler in C to better understand how an operating system processes and handles the simultaneous events of each job it receives, except in a single thread oppose to multithread. Commonly though, if a true definition of an discrete event simulator is sought after, Wikipedia provides a very good overview of exactly how one is described (https://en.wikipedia.org/wiki/Discrete_event_simulation). It defines the different STATES that a function may possess to accurately deal with each event job through the cycle of arriving till it exits the CPU or DISKs.

Firstly, we start by setting up some global variables, each of the config file variables, the statistic variables, various libraries, and most importantly, the structs (structures) that will hold not only our Events, but a second that will hold equally important, our Linked List, which we will be preparing our queues out of. The two linked list queues we then establish are the FIFO which is first in first out queue, and a Priority queue, which sorts the queue by the event jobs time. These linked list queues are very common in data structures and are good because the simulator can generate hundreds to million jobs that an array couldn't handle without being initialized and a circular array would be way more work to code. After we create these queues, we then create some simple functions, such as creating the Event, which will utilize the event struct, and make each event. A random number generator for the times of job in each of the various processes. After we set all the previous variables, STATES, linked list nodes, and statistics to zero and NULL, we are ready to get into the CPU and DISK processes.

Once the program is loaded, it loads in the config file. Sets the mins, maxes, seed, everything for the CPU and both DISK. In the while loop, a switch is used to cycle through the different queues and processes, until simulation ends and prints out the statistics. Now, when the event jobs begin to generate, they are put into the first queue, a priority queue, which sorts them by their job time, after which they travel metaphorically to the CPU. It is here in the CPU, they wait in the CPU queue, the first FIFO queue, until the CPU is in an IDLE state, triggering the CPU to pop the first FIFO in the queue, processes for a random amount of time set by the config as MIN/MAX randomization, then moves to the QUIT_PROB if statement, where it is randomly assigned a percent value, then compared to see if it is less than the QUIT_PROB percent value, if so, it exits the CPU and simulation, otherwise travels to the DISK process.

Now that the process has moved to the DISK portion of the simulation, the program has some decisions to make itself. Here the program has two more FIFO queues, one for the DISK1 and one for the DISK2. Now both of these queues start at zero nodes, but as the CPU pushes event jobs to the DISK process, the jobs then have to face an IF statement, which selects the appropriate DISK queue to store the job in, generally the DISK FIFO queue with the least amount of nodes, unless both are equal, then the job has a fifty percent chance at either queues. After the DISK process decides which DISK FIFO to send the event job too, it waits in the queue until the DISK is IDLE, at which point now the event job enters. Stays for a random amount of time that depends on the MIN/MAX DISK times, then is spit out and immediately returned to the CPU queue to wait for another cycle through the enter event simulator process, hoping to hit the QUIT_PROB and exit the simulation.

While this enter sequence of processes is happening, the three FIFO, first CPU, and second and third DISK1 and DISK2, the entire process is consistently creating more event jobs to put into the event priority queue. This results in tens to thousands of jobs depending on the finish time of the simulation which can be set in the config. While not all jobs will exit the simulation, this is a key program that shows how the event jobs are processed, moved through the program, recycled, and ultimately either exiting or the simulation ending. Through the entire process, a lot of debugging print statements were put in and removed, which helped create the print statistics method and find all the averages and queue lengths. Adjusting the config showed a lot of variances when changing times of MIN and MAX and even the finish time of the simulation.