Sherdil Khawaja, Chris Maka, Shahir Abdul Satar

CMSC 312 Operating System Simulator Documentation

To run this simulator please double click on OS.jar in the main folder.

**Features:**

* PROC – prints the list of processes that have been loaded or are currently running
* LOAD – uploads the process into the simulator, and later can use PROC to see it
* MEM – shows the usage of memory currently being used by processes
* EXE – when typed into GUI it executes the given job files, and are prompted to show how many cycles are being run. E.g. EXE 10 – meaning it runs for 10 cycles.
* RESET – when typed into the GUI it deletes all processes and starts over.
* EXIT – exits the GUI

**Job File Operations:**

When the job file is loaded into the simulator and then executed, four different commands are read, “CALC”, “I/O”, “YIELD” and “OUTPUT.”

It’s the job of the EXE command to read through job file and read and react depending on the commands of the job files, in the given number of cycles.

* “CALC” will determine the number of cycles for the process.
* “I/O” basically serves as an interrupt for the processer so when initiated a random number will be generated and the process will be put in a WAIT state.
* “YIELD” is a form of random interrupter and gives priority to another process that is currently running
* “OUT” prints out which program is running and its output

**Process:**

We created a class for the process states, NEW, READY, RUN, WAIT, EXIT. Each are identified as stated. Then we created the object for the processes, so the Process Control Block. Which is a data structure that contains all the information of a process, so the process state, process id, name, arrival, wait and CPU time; All of which were the parameters. All the parameters were information for the given processes that were plugged into the simulator when LOAD function was used.

**Event Control Block and Event Queue:**

The event control block was created to represent the processes that will be entering the event queue, the two parameters for this object where the time it checked in and the process itself. Basically, it will be any event in the simulator that is awaiting.

**Interrupts**

When the job files read I/O then an interrupt was issued, in which the process was paused and with the IOBurst it was given a random time to wait before it could start up again. The scheduler was used to determine which process was first executed or interrupted.