USER GUIDE

Overview

This project was created to simulate the loading and executing of files in an operating system. The files that are loaded simulate real software applications. Each class of the source code represents the role and function of different hardware components.

Terminal

When the terminal is launched, it waits for input from the user. The user can type one of the following commands:

- **PROC -** typing this command will show all unfinished processes in the system as well as their information.
- **MEM –** shows the current usage of memory space
- **LOAD –** this command followed by the name of a job file will load the job file, and then load the list of program files contained within the job file.
- **EXE –** after the user enters the load command and all of the program files, the terminal will wait for "exe" to be typed. Once "exe" is typed, the simulation will begin running. During execution it will continually print to the console the different attributes of the process, such as the different times associated with it, the memory required, and which state it is currently in. When it is done running, it will display total stats about each process that ran.
- **RESET –** resets the terminal
- **EXIT –** terminates the simulator
- **SPEED** – accepts one argument and changes the CPU cycle speed

Job and Program Files

For this project, job files are text files that contain a list program file names (which should also be text files) as well as the arrival time for each program. Each program file contains numerous sets of each operations (CALCULATE, I/O, YEILD, OUT) as well as a critical section. Critical sections are marked by CRIT_ON and CRIT_OFF. Each critical section is handled with a variation of mutex lock, with an int variable called 'lock'. If lock = -1, that means there is no process within the critical section. If lock is anything other than -1, lock gets set equal to the process ID.

Scheduler

The scheduling algorithm used for this simulator is First-Come-First-Served. It was made to simulate 4 threads.

Paging

The total memory is 4096MB which is represented by an array with 4096 elements. When processes are loaded, they take up the first available element in the array. After termination,

they free up the elements they were in. This is implemented similar to the mutex lock variation, where an empty element is set equal to -1 and a filled element contains the process ID.