Justin Hoyle
V00894837
CMSC 312
Project Part 3

GitHub Link: https://github.com/JustinHoyle/CMSC312Project_JustinHoyle

Part 3 Important Notes

Minimal changes for this part had to be made as essentially everything required was already implemented since part 1. All of the processes already run parallel with multithreading with the QThread class creation. Qthread also dynamically allocates each process to a cpu making sure that each core has as close to equal amounts of processes as possible. Documentation can be found here: https://doc.qt.io/qt-6/qthread.html. Additionally, since the user defines the number of processes, any number of processes can be defined from 1 onwards, not just a static 4.

Part 2 Important Notes

The memory taken up by the GUI is not included in the 512mb limit since it is a separate thread that is unimportant to the process calculation, but the combined value is displayed on the GUI for system resources used. Process memory used is calculated at the start of the new state and given a random value for its memory size. Due to the processes being run by QThreads, the pages and memory spaces are allocated upon object creation.

Overview

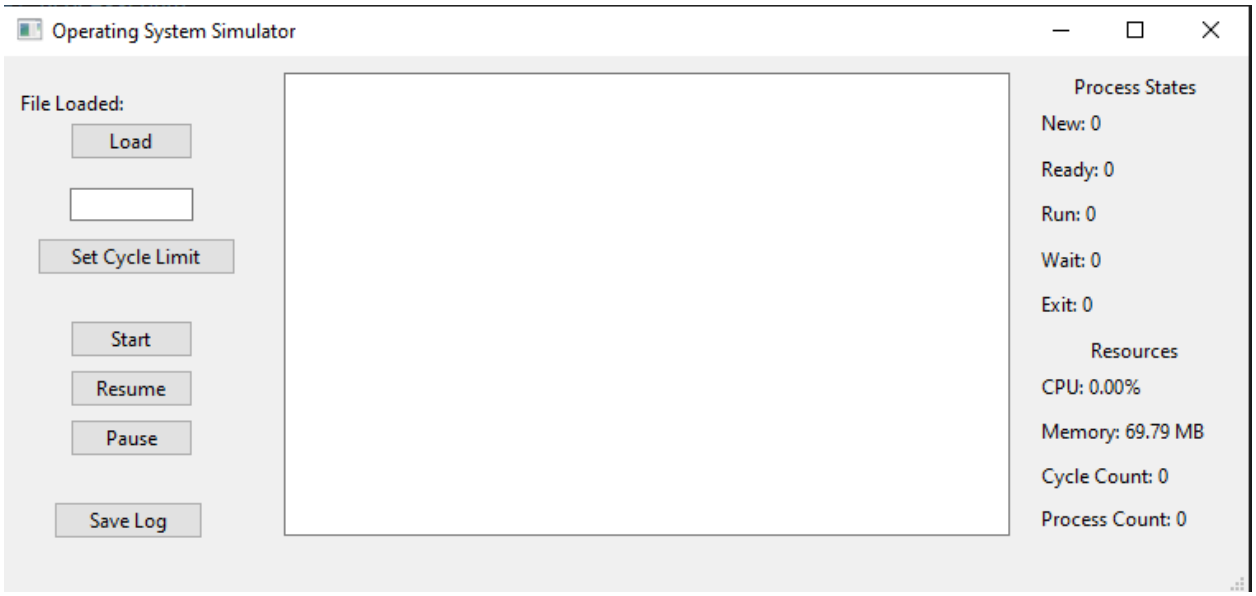
For this project, I will be using Python as the main scripting language, using the PyQT6 library to help simulate the cycles of the environment and the GUI. Unfortunately I found out too late that QThreads do not handle forks well, so they are not implemented for my own sanity. Users can load a text file in the required format in a txt file. The CPU percent is listed, though due to the nature of python, it is hardly used and will likely stay at 0% for the entire process running time.

There are 3 files that are needed to run: main.py, which is the majority of the functionality for the simulator. ui_cmsc312.py, which is the ui builder. And any named text file to be used as a template. Everything else in the repository is either an output file, or folders to launch the compiled EXE file.

The program itself uses QThreads, which are the same as regular threads but allow the gui to be updated during its running cycle. When run, the main window outputs the current life cycles for

each process being run, and outputs each life cycle runtime. Beside the window, are the values listed for each state and how many processes are currently in each one. The New, Ready, and Exit lifecycles are set to a random value between 5 and 15, while the run and wait are set to a random int between the min and max values given for the command.

User interface example:



Template format:
Operation List,Min Cycles,Max Cycles
Calculate,5,100
Calculate,25,50
I/O,10,20
Calculate,5,20
I/O,15,25

Instructions

1. Run **python3 -m venv /path/to/environment/** to create the virtual environment for python
2. Run **pip install -r /path/to/requirements.txt** to install required libraries
3. From there the main.py file can be run independently with python3 main.py (make sure virtual env is running first)
    a. ALTERNATIVE: Run the App Launcher.exe file
4. Click load on the ui and input the path to the template text file, hit ok
5. Enter number of desired processes to run
6. Set cycle limit to pause threads if desired, default 0 with no pausing until finished
7. Click start
    a. Click pause to pause cycles
    b. Click resume to continue
    c. Program runs until all processes exit
8. Click save log to save output into txt file located in the same directory as main.py