Individual Project Report

The Individual Project is a Python project designed to do the three things requested of the project reports, thus coming with three modes.

A screenshot of a computer

Description automatically generated

# Task Manager Mode

The first mode is the Task Manager. It has five functions of its own, which are:

1. Searching processes by name

A screen shot of a computer

Description automatically generated

1. Getting more information out of a process by searching via ID

A computer screen shot of a program code

Description automatically generated

1. Creating a new process

A screen shot of a computer program

Description automatically generated

1. Suspending or resuming a process

A screen shot of a computer

Description automatically generated

1. Termination of a process

A screen shot of a computer

Description automatically generated

## Issues

I was not able to perform all the tasks required, which included the creation of threads at user-level. This is due to me believing that I was unable to create threads for processes, as the threading module in Python led me to believe that all threads required a function to exist, and that letting users create a function would be a security violation at best, and out of the scope at worst. So, it was never implemented. Additionally, the program does not like peering at specific processes, namely the one belonging to the system itself. This is a limitation as well, especially as it induces a crash in the program The detailed process information could also have additional information, as the psutil library allows for much more than viewing only parents, children, and threads.

# Inter-process Communication Mode

The second mode was not completed and was meant to be the comparison of four types of inter-process communication: Shared memory and message passing of both threads and processes. There is still rudimentary code in for a single message between processes using message passing and a full-blown set of small messages being sent between two threads, however there is no comparison material.

# Text Processing Mode

The last mode is the text processing mode, which has its own litany of issues. However, it is functional and uses threads. It takes in a complete text file directory and counts the number of times each character is used in that file. Lowercase characters are turned uppercase before they are counted, but does not affect the original file.

A computer screen shot of a program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A computer screen shot of a program code

Description automatically generated

A screen shot of a computer code

Description automatically generated

A screen shot of a computer

Description automatically generated

## Issues

No testing was done to determine the scalability or memory usage of the process. Only one file was used, which was 151 characters long. There were no statistics for that process either, although the module for determining such statistics was imported. It went unused due to time constraints caused entirely by myself starting very, very late. Additionally, the threads are not very well used. It waits for the user to input their file name. Due to an issue with putting an inputted argument into a thread, the thread itself starts with the prompt, effectively turning it into a glorified function and not even close to the thread it wants to be.

# Overall Issues

Python is not the best way to do a project such as this. One of the bigger issues is how threads in Python work. Python has a Global Interpreter Lock, which prevents multiple threads from running simultaneously. This only applies to threads, and processes using the multiprocessing library would be unaffected. If I had perhaps given myself a little more time, I may have attempted to use Processes for the text interpreter to avoid this lock.

So, why Python? I did it in Python after finding out that threading features do exist within Python, as rudimentary as they are and then later found out about the multiprocessing library. Additionally, as much as I would like to attempt to use C, I simply never built a C compiler, which would cause a potential complication. C++ was also on the mind a bit but was also scrapped because I didn’t believe it could handle the situation.