CMPUT 379 Assignment 1 Project Report

Name: Tymoore Jamal

Student ID: 1452978

LEC A1

Objectives:

This project had 3 main objectives. To familiarize us with forking to create child processes, to learn how to execute a different process inside of a program through execlp, and how to send signals to processes in order to suspend, resume, or terminate them.

This provides us with meaningful experience interacting with the GNU operating system through making system calls to the Unix shell. This allows us to have significantly more freedom when programming in the Unix environment, as system calls are quite powerful and allow us to achieve results we would not otherwise be able to do (e.g. creating a process inside of another process).

Through this new knowledge, we can now create processes easily, whether they are the same process as the parent process, or an entirely new process, as well as we now have the ability to send signals to processes to change their state.

Design Overview:

The important features of each program are outlined below:

a1jobs:

- Setting the CPU clock limit to be 10 minutes as a preventative measure, to ensure no process executes indefinitely.
- Getting the initial CPU times to monitor them.
- Providing the user the ability to run a separate process.
- Providing the user the ability to view all processes created by the user.
- Providing the user the ability to suspend, resume, or terminate a process they created.
- Proving the user the ability to exit, either terminating all processes, or to keep them running.
- Display the elapsed CPU time, of the process and its children.

a1mon:

- Setting the CPU clock limit to be 10 minutes as a preventative measure, to ensure no process executes indefinitely.
- Allowing the user to specify a process to monitor, and the interval to monitor it.
- Displaying to the user all executing processes.
- Upon detection of the target process terminating, a clean up occurs, where all processes rooted at the target process are terminated.

Project Status:

a1jobs:

Some difficulties included in this program include:

- Parsing the input to store it as a vector of strings and determine the number of arguments.
- Ensuring that all invalid inputs are detected and flagged as invalid.
- Properly tracking all processes.

However, the main functionality of this program (creating processes, and sending signals to them) did not pose much issues to implement.

a1mon:

Some difficulties included in this program include:

- Determining if a process has terminated.
- Maintaining a data structure that contains the command issued and parent's pid for each pid running.
- Ensuring all processes rooted at the target are terminated.

Testing and Results:

a1jobs:

I tested a1jobs first as it could easily be tested independently without a1mon completed. To test it I used xeyes. I used xeyes for multiple reasons, it does not produce any output to stdout, it is obvious if the process is terminated or suspended, and it is customizable by color to identify separate processes. To test a1jobs, I created multiple instances of xeyes, and tested suspending, resuming, and terminating them. Also testing if they remained after using quit vs exit.

a1mon:

To test a1mon, I set the target pid to be the pid of a1jobs, I then created multiple child processes in a1jobs. I also created grandchildren of a1jobs by having a1jobs run itself and then have its child create a child process. Finally, I terminated a1jobs through a 3rd terminal using kill -SIGKILL. From there I noticed that a1mon had terminated all the child and grandchildren processes.

Acknowledgements:

I did this assignment individually and did not collaborate with anyone.

Assumptions Made:

a1jobs:

- A run process will be added to the list regardless of if the execlp command was executed properly.
    - I made this assumption due to a conversation I had with Dr. Elmallah, where we concluded that this would not be easy to implement and would not be marked.

a1mon:

- If any changes are made within an interval and then the target process is terminated within that same interval, those changes will not be noticed by a1mon. Therefore, if a child is created and the parent (target) is terminated within the same interval the child will not be terminated by a1mon.
    - I made this assumption due to a discussion I created on the discussion forum which got replies from Tobias Renwick, and Dr. Elmallah.