Introduction

This program creates 10 unique child processes in which the user is notified when finished. When a child process is successfully started, the ID will be printed and commands will be executed. If there are errors, they will be caught accordingly. Because of the header files, this program is OS specific and made for Linux.

Implementation

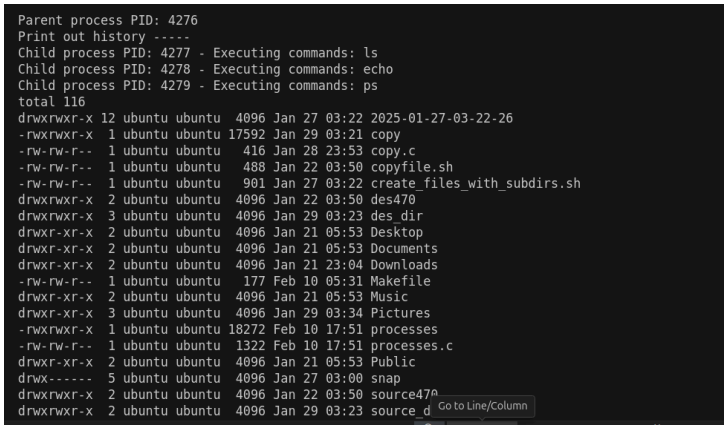
This program consists of two files; one is a C file that implements bash functionality and another is a basic Makefile.

The C program creates a 2D char array of 10 child processes, each a unique bash command. The ID of the parent process is printed and then we move into a for loop that goes through each child process. The ID is tied to the fork() function in C which will basically relay the status of the process. If the number is below 0, the user is notified of the failure and the program exits. If the ID is 0, the ID of the child process will be printed and the command will be executed. Within this if statement, is handling for an error if the execution is unsuccessful. Outside of the loop, once processes are finished, the user will be notified per child process.

The Makefile is a simple file that executes the gcc command renaming the file to a processes.exe file, which can be executed.

Results and Observations

The file executed quite easily. The only thing is that the commands were executed in the same terminal so the user will get a bit of clutter, especially in my example where the ls -l command was taking up a lot of space between the other echoes:



Here is the first picture of the execution where it's cut off since the ls command is so large.

Conclusion

From this lab, I have a much better understanding of process creation and how parent and child processes interact with each other.