Lab 3 Report

Nicholas Jurden 2415098

1. Briefly describe the design of the program. How many processes and how many pipes are created with your solution? How does data flow from process to process?

    - My program creates 3 pipes between four processes. Data flows from the write end of one pipe to the read end of that same pipe to be used by the second process. At before calling execl, I close all my pipes because they are not being utilized to prevent a system hang. I use dup2 to declare the read end of my pipes before printing commands to the cmdbuf, and the write ends using dup2 after writing to the cmdbuf. Dup2 replaces the file descriptor in this case with STDIN_FILENO or STDOUT_FILENO, depending on which end of the pipe is in question.

2. How did you test and debug your solution?

    - For testing and debugging, I started by printing after each sprintf which process I was in. This was useful in making sure that each process was entered. I consulted `man pipe`, `man fork`, and `man dup2` for further answers and info about what each does and snippets of example usage. Eventually, it came down to moving dup2 declarations and close() calls around until things started to work. At first, my program only output every .c and .h file. At this point I knew at least the first process was running correctly. Further tinkering with the close() calls eventually yielded a working program.

3. When he was head of Bell Labs, Doug McIlroy summarized the “Unix philosophy” as follows:

    > *Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface.*

    How do pipes contribute to the Unix philosophy?

- Pipes contribute to the Unix philosophy in their very nature. They are composed of “read ends” and “write ends”, the core of I/O programming. They allow text streams to be passed between processes which enables more complex computing.