Eshwar Muneshar

CS 340 Project 1

For project 1, the assignment given was to write a C or a C++ program that would read from one txt file and print to another txt file that would utilize both a parent process and a child process. The parent process would read from the input file and the child process would write to both the output file and then console. To accomplish this, I took inspiration from the cpfile.c that was demonstrated in class and the class textbook.

The first step to accomplishing this task was to declare the variables. Among them we have f1 and f2, and buf which is an array that stores chars. Next up we perform a series of checks. These checks are to make sure that we have enough command line arguments to successfully run the program. The next check is to make sure the output file is capable of being made. From there we create the pipe by declaring the ends of the pipe as an array. This array will be utilized later in several crucial processes. Next we fork the program, this is where we create a child process from the parent process. When you perform a fork function you create a child process who gets an exact copy of all the variables and data available to the original parent process. With that in mind this also means that the child process will get a copy of the pipe variables.

After that we need to isolate the child and parent processes from each other. To do this we need to utilize the fact that the child process is known to have a pid of 0 and the parent process is not. I used an if statement to isolate the parent process. Here in the parent process we close the read end of the pipe because the parent process will not need to read anything from the child processing. Here we use a while loop to continuously read from the input file. Simultaneously we use a write statement to write to the pipe. We read from the file in 128bit chunks because the pipe needs to send data the child in 128bit chunks. Once all the input is read from the file, the child process comes in and closes the write end of the pipe as it will not be used by the child. The child process uses another for loop to write the contents supplied from the pipe to the screen and the output file which was previously created. Once the child process is done it closes all files and the read end of the pipe. The program then prints that the copy operation was successful and exits, outputting the copy of the input file in the process.