CIS 3207 – Lab 2

**OBJECTIVE**

Design a C shell that takes command input from user input or a batch file. The shell will incorporate 9 built in commands { "exit", "clear", "help", "dir", "pause", "shellPath", "environ", "echo", "cd"} via function calls. The user can also pass commands found in the "/bin/" directory. The shell should also be able to perform input and output redirection with files, and commands using the symbols: "<",">",">>". Moreover, the shell should be able to perform piping functions using the symbol "|". Lastly, if the user wants to run a command in the background they must enter an ampersand ("&") at the end of their input.

**LEARNING OUTCOMES**

I learned how to use the dup2 functions, how piping works, and how redirection works.

**LIBRARIES USED**

<stdio.h>

<string.h>

<unistd.h>

<stdlib.h>

<dirent.h>

<sys/stat.h>

<fcntl.h>

**PROJECT DESIGN**

The shell will take inputs from either a batch file or user file. Once I receive the input, I must parse the input. Next, the shell checks if the parsed user input contains piping symbols, or I/O redirection symbols. If the parsed input does contain either symbols. The parsed input is split into subarray of strings.

For example: COMMAND1 ARG1 | COMMAND2 ARG2

Sub Array 1: COMMAND ARG1

Sub Array 2: COMMAND ARG2

If I/O values are found then the standard output/input is changed to a file, and the commands are invoked with the new standard input/output.

If the piping symbol is found then two child processes are created. One child process will change its standard output to the other child process, and the other child process will change its standard input to the first child process.

The way commands are invoked in the shell is by either executing built in functions, or using execvp() to run external commands found in the bin folder.

Also note that if an ampersand is found in the end of the command input, then the command is run in the background

**TESTING**

To test my shell, I used various printf statements to check if I my program was outputting the correct information. To test if a command runs in the background in the presence of an ampersand I made a program with an infinite loop, and checked to see the difference between running the infinite loop in the background versus the foreground.

**PROBLEMS I ENCOUNTERED**

Problems I encountered: I had trouble splitting the parsed input, and storing the split parsed input in an array. I realized that cause of the problem was because I wasn’t allocating memory to each index of the subarray.

I had trouble with changing the STDIN and STDOUT when working with I/O redirection. I realized that I was using dup2 in the parent process, which changed the STDIN and STDOUT of the shell. I fixed this problem by using dup2 in a child process by forking.