Jingning Zhang

EC 440

U64723279

HW3 – Build A Bash report

# Overview

In this homework we are trying to make our own version of the bash. By doing this homework, desiging, coding and debugging. I am getting a sense of what is actually happening behind the bash shell. Many system call function are used in this homework part (b), and many system provided executable files(under /bin) are being used.

# Outline

In my code I first have all the needed libraries included.

I declared several global variables to be used in the [^C] handling.

Then I have several functions each does specific jobs.

I have one function for redirecting type 1, one function that generates a pipe and then execute the “ls” command, 2 void functions that have the ctrl+c signal handled for 3 times.

I also have a main() function that prompts the user for inputs, spilt the inputs into string array and call on the rest of the functions to actually do a bash’s job.

# Procedure

This project takes several weeks to be finished. I did not understand what we are going to do at first. So the first week, I just did the parser that takes the user input.

I successfully separate the tokens but I have no way to put them into one string array. I have only one string memory location to put the tokens, and it is always being overwritten by the next loop.

I saved the parser like that because I think when we actually do the command part, the token is to be used one by one so I gave up forcing the tokens going into array slots.

The next week when I was trying to do some system calls, I find out the inconvenience of having only one token available at a time. I changed my parser’s algorithm and this time I use getchar() instead of scanf() and putting the tokens into the 2 dimensional char matrix.

After debugging the parser, I applied the handler for the “[^c]” signal. I did this first because there is an example online, I read the example carefully and understand how does this work. I write the handler in reference of the example. I set the maximum handling time to be 3.

I call the void function in the main function before implementing the parser so that my bash will not quit when I give the ctrl+c signal before I type any command. I saved the code and tested it. The handler works well. Then I am going on to implement the pipe.

The pipe and exec part confuse me a lot, so it take a long time for me to do. I read the examples in the textbook and then have a sense of how the pipe is generated and deleted. However I did not know how it is connected to the execution. I then tried the pipe, fork() and execlp() inside a same function. When I test the code I find that the fork() worked and exec worked and the waitpid() at the end of the child process worked. I could do the fork an child process and have the parent wait for it to end then continue executing.

I want to test the pipe so I added some logic that if pipe were failed it would print out an error message.

I did not get the error message, which means my pipe is also working. Then I made the first type of command, which is going to create a file if the user prompts to do so. I finished the function, but I am not being able to link the function with the taken inputs, which were stored in the char matrix. I can print the tokens out to the console one by one using a loop, but when I want to compare the special sign tokens with the specific character or using strncmp() to compare the special sign token. It always return zero. It is probably because the 2d matrix position is a pointer which has the different address with what is being compared.

# Success and Failures

For this project, I made the parser, pipe, fork, exec, wait all working properly.

The redirection is not working because my code is not analyzing the tokens in the right way.

I set the exec inputs fixed because I am not being able to control which process to run using the special characters in the token array matrix.

# Added Feature and testing

I added the ls command which is directly being called in the main function.

I was testing the code by compiling and run (print out that is being processed). After I run the bash, I use the ls command to see if the newly generated file is actually here.