Ryan Russell

Dr. Ahmed

CSCE 313.503

21 September 2020

Programming Assignment Two Report

When it came to implement how the shell handled echo commands, I decided to test to see if the input string contained the word echo in it. If that was the case, then I set a boolean variable called result equal to true to show that the string was an echo command. Then, I found the position of the first quotation mark and the next quotation mark. Then I tested to see if the pipe character was found in the string and if it was between where the quotation marks where in the string. If it was in between these quotation marks, then I created a backup string which would equal the input string and then changed the input string so that the pipe character was no longer part of the string. This process was done again except with single quotation marks instead of double quotation marks. After both processes were complete, I erased all of the quotation marks (double and single) in the input string with the erase method. Now the command is ready to handle the input string if the string contained an echo command once the command gets to the execvp method.

After implementing the echo command handling in the shell, I tested to see if an & character was in the command (for background process handling). If the position of this character was not equal to negative one (meaning it was in the string) and the boolean variable called result from the previous paragraph was equal to false; then I created a substring which eliminated the & character and then set a boolean variable called bg equal to true which meant that the command dealt with background processes. The next thing that was checked was if the string was going to be a change directory command which I implemented by doing the same thing as checking for the & character (i.e. checking to see if the string included cd and if the echo result equaled false). If both these conditions were met then I got the directory name from the string and used the chdir method to execute the command. This process was done for both the pwd and sleep command with the only difference being the change of content in the if loop.

The next things that I checked for after creating a child process using the fork method was that if the process was a child; then the shell program would test for I/O redirecting. This was done by checking if the echo result was equal to false and if the input (>) character was in the string. If that was the case, then the command and filename portions of the string were split up and the file was either created and written in or opened by the open command. A boolean variable named inresult would also be set equal to true to indicate that the string contained an input redirection command. This process was done also for the output (<) character except the open command could only read the file this time and a boolean variable named outresult would be set equal to true to indicate that the string contained an output redirection command. After both characters were tested to see if they were in the input string, then I tested to see boolean variable result equaled true. If it did, then the backup string (created in the echo command handling portion of the program) would erase any double or single quotation marks with the erase function and then create a vector by splitting the string on a space. This vector would then be converted into a char array and then executed with the execvp function. If either the inresult or outresult variables equaled true, then the same process would done except with the updated input string variable. If none of the boolean variables equaled true, then the same thing would be done as it was handled with inresult and outresult if loop except that the input string would be trimmed of extra whitespaces.

Demo Video Link:

https://drive.google.com/file/d/176VXIzJMzKqdT05fAZ-f1Bkemi9DooVm/view?usp=sharing