PA2: Implementing a Linux Shell

Maria Dmitrievskaia

UIN: 927009911

CSCE 313 – 512

Due Date: September 20, 2020

Design:

The written code implements a Linux shell. The shell is able to handle pipes, input output redirection, and background processes. In order for the implementation to work correctly, a trim and a split function had to be written to parse through the user input correctly.

The trim function gets rid of all leading and trailing spaces between words, while the separate function then puts the input into a vector based on either spaces or the piping symbol ("|"). In order to trim the input, I go through the input string, and if the character is not a space, I push the character input a "trimmed input" string. If the "i + 1"th character is a space, I also push it back into the "trimmed input" string, thus ensuring that the trimmed string has only 1 space in between each word with no leading and training spaces at the beginning or end of the input.

The split function was written to handle separation of the input string variables by spaces and by quotes. The quote implementation does not completely work, however, the logic is the following. I set a boolean value "inside quotes" to false, which is set to true once a quotation character is seen in the FOR loop of going through each character of the input string. If a quotation chacter is seen, then I increment the count for the number of quotes. I keep track of the number of even/odd quotes, so that if I have an odd number of quotes, then the "inside quotes" booleen is set to true and I get a start value of where the quote starts in the input string. I then continue until I hit the end of the quote, at which point I run from my "start" value till the end of the quote and push that into my separated input vector as one argument. Similarly, if there are no quotes, I go through each character of the input string, and, once there is a space, I go through "word" and push it inpto my separated input vector.

In order to print out the current date and time, created a gettime() function that is called every time the user is prompted for an input.

If there is a background process, indicated by the '&' symbol, then I detect the background process and trim the input string to omit the symbol and continue working with the input string from there. If there is a background process, I push the pid of that process into a vector, and as soon as the process ends, I omit it from the vector.

The change directory is handled by keeping track of the current and previous directories. If a "-" is detected, then I change the directory to the previous one, and update the new previous directory with the old current directory.

The input output redirection is handled in the child process of the shell. If a redirection symbol is found, then I create a string through trimming called "command" in which I place the command, and a string called "filename" in which I place the filename. I then set the input string equal to my command string. I then open a file and either read to or from the file, depending on the redirection symbol.

YouTube Video Link: https://youtu.be/cS2OVYwN28M

Resources:

The following resource was used for my gettime() function.

//http://www.cplusplus.com/reference/ctime/ctime/