CSCE-313

PA 2

Aakash Tyagi

Priyanshu Barnwal

10/8/2021

INTRODUCTION

In this PA, we were tasked with creating a Linux Shell. This should carry out many of the functions of an actual Linux shell. Piping and other concepts from PA 1 were used in this PA as well.

THE CODE

To do so, I started with creating a trim function, a vec_to_char_array function, and finally a split function. The trim function simply trims the whitespace from the beginning and end of the inputted string. The vec_to_char_array does exactly what it sounds like and converts a vector to a character array of pointers. Finally, the split function cuts up the input into words and deals with any quotes found inside the input.

Moving on to the main function, we start off by creating some variables that we will use later on. Then, I defined the directory paths with a temporary previous directory variable to use for the "cd .." command. Afterwards, I created a while loop to permanently loop and take in inputs. I then made a for loop to detect any processes. Then, I used the chrono library to get the current times and used flags to format the date and time as shown in the instructions. After outputting the prompt, I took in the line and used the trim function to trim all whitespace from the input. I also check if the input was "exit" and if so, I broke the loop and ended. I then checked if the line ended with an ampersand. If so, I knew that there was supposed to be a child process running and set my process variable to true. I then take off the ampersand. I then create some more variables to use later and check if it's a child process. If so, I execute the next set of code, which is the meat. It executes the directory navigation, where it converts current and previous directory using the path of the current directory. This implements cd, ls, and all related operations. Afterwards, I redirected I/O to a file to implement reading and writing to a file. Using the specifiers of '<' and '>' to check whether it is an input or an output, I set the appropriate number in dup2. Using the flags from the PowerPoint, the code then is able to write to a file, read from a file, and copy from file to file. This is all in a while loop so that the code can execute for any number of arguments. At the end, an execvp function is used to actually write to the file. Finally, I implemented piping in an else statement for the parent only. This requires the parent to wait for any child process using the waitpid function. This also implements the sleep functionality. Finally, we close fd.

THE VIDEO:

Video: https://youtu.be/9NELb8XR7Js

Git Link: https://github.com/CSCE-313-Tyagi-Fall-2021/pa2-implementing-a-linux-shell-Preebie.git