CS3401 : Operating Systems

Project 1:

Linux Terminal Shell

Mohamed Eliskandrani

900140998

**Introduction:**

The aim of this project was to create a shell for the Linux Terminal with an added history feature. The history feature shows the user the last 10 commands inputted and allows them to reinput one of these said commands. The program should also support concurrency of processes and allow the user to choose whether to run this program concurrently or have it wait for the child process to terminate.

**Code:**

The code is comprised of 4 functions including the *main* function. The *main* function takes the user input and calls the function *tokenize*. After the input has been fully tokenized it forks to create a new process and invokes the *execute* function and checks if the user has inputted the command “exit”. The program then loops back to the start of the *main* function if the user has not entered the exit command.

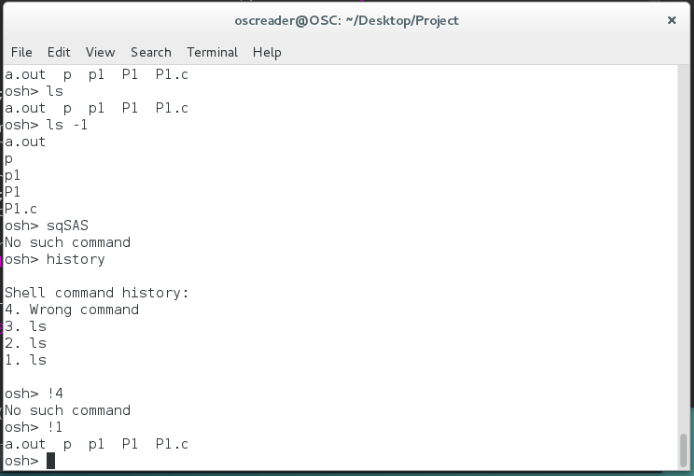
The tokenize function parses the string and reads the arguments separated by spaces and terminated by ‘\0’. After this it checks if the user has invoked a command using the & operator indicating that the command should be run concurrently. It then checks for the ! operator indicating that the terminal should run a previously entered command. The function also checks for the history command and calls the function *displayHistory* which simply displays the last 10 commands inputted by the user. After this the function updates the history table and returns to the *main* function for execution.

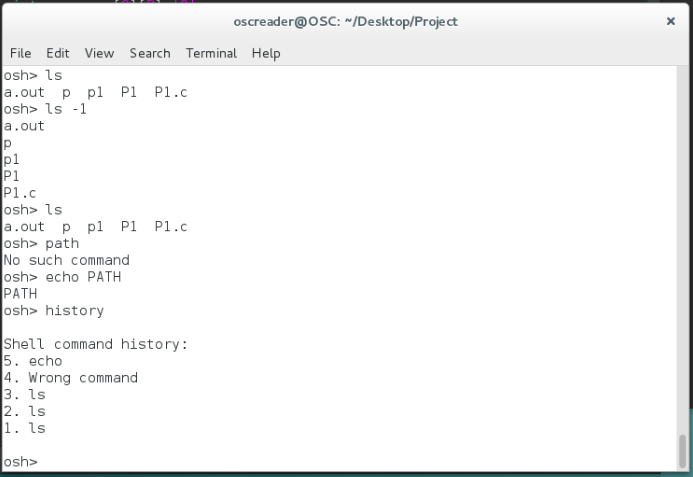
In the *execute* function the arguments are placed in the execvp function that runs the linux terminal commands. The function checks whether or not the user has asked for this to be run concurrently or not and invokes waitpid as necessary.

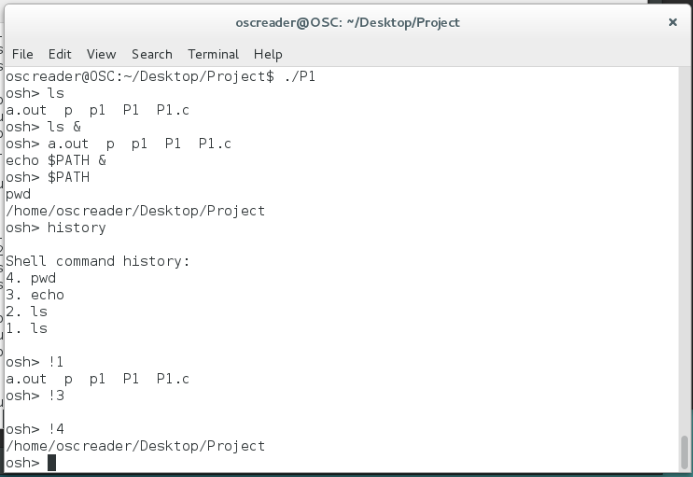
**Difficulty and Complications:**

During the creation of this project, I found great difficulty in creating the tokenize function to match all of the requirements needed by the other function calls. Moreover, the program cannot create a history record for commands having more than one argument and as such most of the testing was done using ls, ls -1 and env. This is due to the fact that if the line inputted by the user was saved in the history as is, the arguments would no longer be valid and as such a buffer for the arguments needs to be created and synchronized with the history functionality. Alternatively, another tokenizing function could be made to tokenize the history but that would increase the processing needed when we could alternatively rely on memory. Getting the program to save Wrong Command into the history of the terminal rather than the wrong command itself proved to be extremely challenging as well. If I had time to recreate this project I would make the program more modular in order to avoid some of the problems that I faced especially in the tokenization of the input. There is also an additional IO bug when using commands concurrently in which osc> and the previous command’s reply appear on the same line.

**Test Cases:**







**Directory:**

The compressed file contains the source code of the project (P1.c) the Linux executable (P1) and this report.