Gülşen Sabak / 2020400072 /13.12.2023

C Shells by the C Shore

Introduction-My algorithm steps-Program execution:

In C shells by the C Shore project, I implemented a shell called “myshell” in fork-exec manner via using the C programming language. It accepts commands which contain at most 1024 characters, and it acts as a normal shell. Myshell controls some steps to determine whether to execute that command or not.

Control criteria:
1) Does that command in the file that I keep aliases? If it is there, then find and execute the left part of alias since I kept aliases inside my file in that form: myls = ls
2) Does that command start with exit? If it is then successfully exit from the shell and print “Shell terminated”.
3) Does that command start with the bello? If it is, then send command to the execute_command function. Inside execute_command send command to bello function. I did it like that since I want to control background processes for bello. (like bello &)

4) Does that command start with an alias? If it is, then send command to the handle_alias function.

If the command does not fit one of these criteria, then I split the PATH and search the command inside these PATH parts. Actually, I search the path just for the first element of command. The flags, bello and redirections are handled inside execute_command function. If the command exists in PATH, then it executes this command via looking some criteria:
1) Does that command contains redirection? If it is, handle it.
2) Does that command bello? If it is, send it to bello command.
3) Else: execute that command and its flags
If it does not exist in PATH, then it prints “Command not found”.

Also, when execution of one command is done, I print the exit status of command. If it is successfully executed, I print to the terminal “Child process exited with status 0”. If it is not, I print “Child process exited with status 1”. (just example)

Redirection: Also, myshell handles redirections. > and >> achieve same task as in bash. I also implement >>>. When the input command contains this, I handled it inside execute_command function. I created another function called extract_left_part so that I can extract the left part of that >>> redirection. I executed the left part, and I reversed the output of that left part. At the end, I add it to the output file.

My algorithm steps: Firstly, I write a code for printing the prompt that you want in description. Then, I create a while loop so that user can enter command until an exception comes. I removed trailing newline character and whitespaces. Then I call process_user_input function it determines which function to execute based on user input, including checking aliases and executing commands. After that, depending on the type of the input I call functions that I have created. Also, I create a function called execute_command that executes commands including background execution and various types of redirections. My fork-exec manner is working inside this function. In addition, to manually control whether the command exists in path or not I create is_command_available function. I also have a function called handle_alias which processes alias commands, updating or creating aliases in the aliases.txt file. (this file is created automatically when I run myshell)

The difficulties: Actually, I have faced difficulties, since it is not an easy project. The main difficulty that I have faced is manual iteration of PATH. Creating a function for that is not that hard but calling it in correct place inside my process_user_input is a little bit hard for me. To handle it, I changed the order of checking. At the beginning I checked firstly whether the command is bello, exit or alias. Then I checked that the command is inside the aliases.txt file. This was okay, but I didn’t decide where to put the error message. So, I changed the algorithm. In a new algorithm, I checked firstly whether that command is in aliases.txt (I put a flag for that). If it is inside aliases.txt I didn’t look at the other criteria. If it is not, I checked other criteria.

Also, I get segmentation fault, then I realized that I forget to free the memory for strdup via trying to implement >>>. I free it and it is solved.

Execution: go to the directory that includes myshell. Run make command, after that run ./myshell command. Myshell will be opened. You can try commands right now.

Some important notes for alias and bello:
In bello, I return the last successfully executed command. However, when I call a command that makes background processing and call bello right after that, it does not return the command that makes background processing, it returns the previous one before background processing command. After waiting so that background process is done, it gives last executed command as that background process.

I adjust that alias only cares the first and last quotation. For example, when I enter that command: alias gulsen = “echo “hi”” It puts that command to the file in that form: gulsen = echo “hi”
Also, when user wants to change the corresponding command for an alias, he/she can change it. I adjust my code to update aliases when user wants.
Also, in myshell the aliases have priority over commands. For example, I wrote alias pwd = ls When I call pwd as command, myshell return the output of ls, not return the output of pwd.