Assignment 1: MyShell

The purpose of this assignment is to use the fork, wait, and execvp to write a simple Linux shell. This shell is called "MyShell", and the goal of MyShell is to continually prompt the user to supply a command, execute that command, and show the next prompt, once the command is executed.

The file MyShell.c is already given to you. The file contains the functions shown in the below table. You should NOT change the method signatures. You should implement the missing portions of the program as per the guidelines given below. You may add additional functions if necessary.

Functions	**Parameters**

`int main(int argc, char** argv)`	`Parameters: int argc, char **argv`
	`Return: EXIT_SUCCESS`

`char** parse(void)`	`Parameters: none`
	`Return: char** pointer`

`int execute(char** args)`	`Parameters: char** args`
	`Return: 0/1`

int main(int argc, char** argv)

- The main function should display the *MyShell>* prompt. You should not change the name of the prompt.
- The main function should indefinitely run until terminated by CTRL+C or when the user types the *exit* command in the *MyShell>* prompt.
- It should call the parse() function in order to accept and process the command the user enters
- It should call the execute() function to actually execute the command

char** parse(void)

- Gets the input typed by the user in the *MyShell>* prompt in the form of a line
- Splits the line into tokens
- The tokens are used to create a char** pointer which will serve as an argument for execvp

int execute(char** args)

- char** args is obtained from the parse function
- You should fork a child and execute the command typed in by the user using EXECVP. You should only use execvp and not other versions.
- When fork fails, an error message should be printed. However, the next *MyShell>* prompt should be shown.

- When execvp is not successful, an error message should be printed, and the child should terminate. However, the next *MyShell*> prompt should be shown.
- When an empty command is entered, the next *MyShell*> prompt should be shown.
- When the command *exit* is entered by the user, the main program must exit [You may or may not fork in order to implement the exit command].
- The parent should wait for the child to terminate before it accepts another command. You can use wait or waitpid for this.

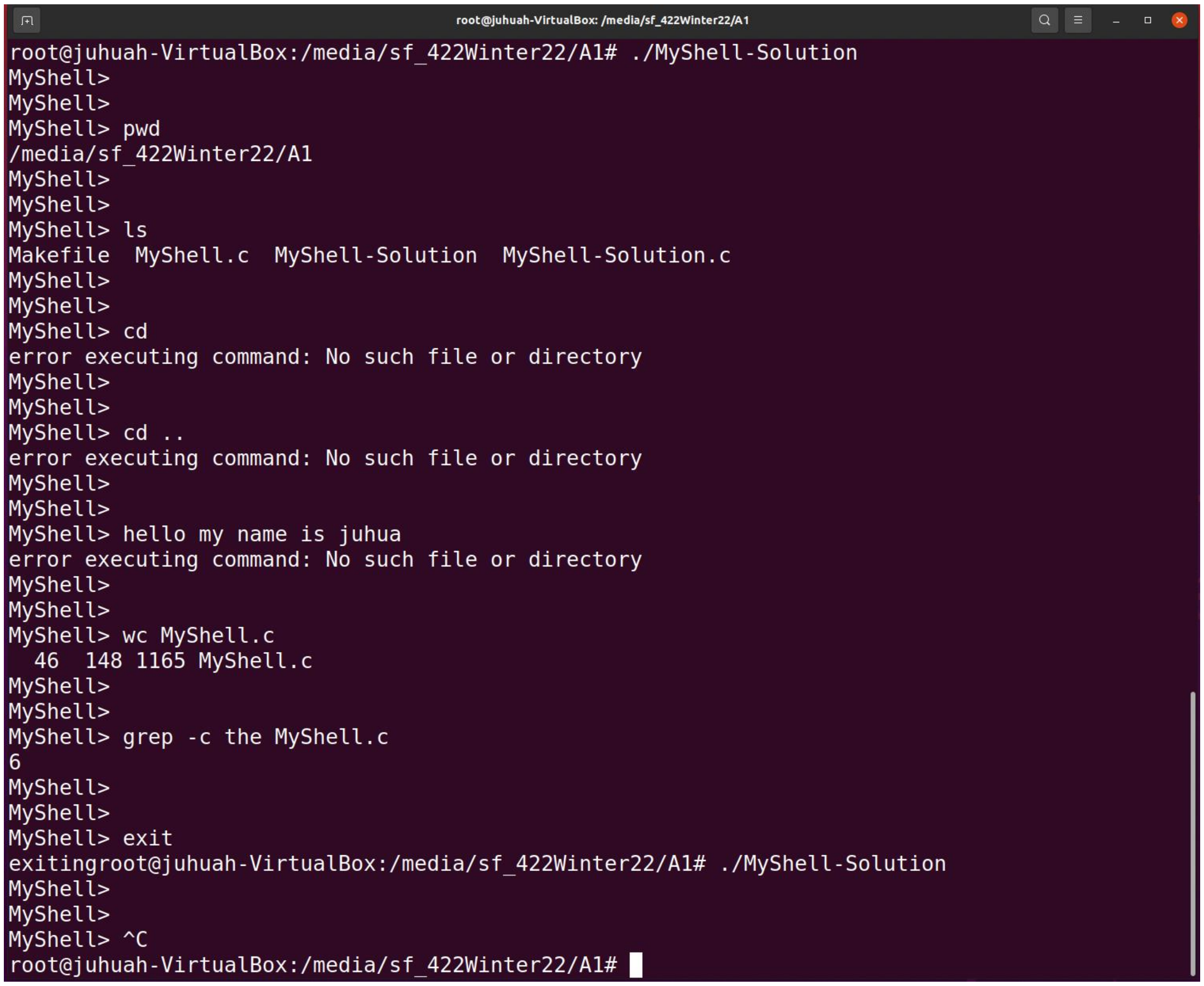
Other guidelines or useful hints:

- User command plus arguments will not exceed 255 characters
- Each command may have different number of arguments
- Variables stored in Stack cannot be accessed outside of a function/procedure, while data dynamically allocated in Heap can be accessed by any function within one program.
- Only one command should be typed at the *MyShell*> prompt, no pipelining of commands
- Commands run in *MyShell*> will assume the user's original path: Type "echo $PATH" to see the current path variable setting. Myshell is not responsible for finding the path for the commands or files.
- If the user makes a mistake typing a command and/or its arguments, execvp should simply fail to run the command. A simple error should be shown, but next MyShell> prompt should be displayed.
- Please note that complicated commands like "cd" may not work as shown in the sample output.
- If execvp was not successful (e.g., a wrong command), remember to 'exit(0)' the child process.

What you should submit:

A single, completed file MyShell.c to canvas. Note: make sure it is compliable for Juhua using gcc 9.3.0.

Sample Output:


root@juhuah-VirtualBox: /media/sf_422Winter22/A1

```
root@juhuah-VirtualBox:/media/sf_422Winter22/A1# ./MyShell-Solution
MyShell>
MyShell>
MyShell> pwd
/media/sf_422Winter22/A1
MyShell>
MyShell>
MyShell> ls
Makefile  MyShell.c  MyShell-Solution  MyShell-Solution.c
MyShell>
MyShell>
MyShell> cd
error executing command: No such file or directory
MyShell>
MyShell>
MyShell> cd ..
error executing command: No such file or directory
MyShell>
MyShell>
MyShell> hello my name is juhua
error executing command: No such file or directory
MyShell>
MyShell>
MyShell> wc MyShell.c
  46  148 1165 MyShell.c
MyShell>
MyShell>
MyShell> grep -c the MyShell.c
6
MyShell>
MyShell>
MyShell> exit
exitingroot@juhuah-VirtualBox:/media/sf_422Winter22/A1# ./MyShell-Solution
MyShell>
MyShell>
MyShell> ^C
root@juhuah-VirtualBox:/media/sf_422Winter22/A1# 
```

Grading Rubric

This assignment will be scored out of 100* points. (100/100) =100%
110 points are possible.

External Correctness:

- MyShell> is displayed correctly 5
- MyShell> accepts user input 5
- MyShell> is displayed after executing a command 10
- MyShell> is displayed after execvp fails 5
- MyShell> is not displayed after the exit command 5
- The command user types are executed 10
- The command user types are executed, and the results are displayed 10
- When the user types an empty command, no error message is printed and the next MyShell> prompt is displayed 10
- When the user types *exit* the program terminates 10
- When the user presses CTRL+C, the program terminates 10

Internal Correctness:

- The execvp command is appropriately implemented with the correct arguments 5
- After fork, the parent waits for the child to terminate 5
- The user input is parsed and tokenized to suit the execvp arguments 5
- There is code to display MyShell> prompt infinitely, except when exit command or CTRL+C is given 5

Extra Credit:

- If your program accepts user input (with unlimited length including unlimited number of arguments) you will receive additional 10 points
- Note: make sure your basic version works before you work on this extra credit one.