Assignment 1: Building a Basic Shell

Professor: Ashutosh Rai
TA: Yash Shirke, Bhavik Sankhla, Aakrity Pandey
Deadline: 11th August 11:59 p.m

Introduction

In this assignment, you will create a basic shell similar to the bash shell in Linux. Your shell will be responsible for executing user commands, handling pipes between commands, and implementing some built-in functionalities. Follow the steps below to complete the assignment:

Objectives

1. **Basic Shell Implementation:**

   - Implement a shell that reads user input and executes commands using the `exec` system call.
   - The shell should display the prompt `"MTL458 >"` and continuously process commands until interrupted by the user.
   - Handle errors during command execution by displaying appropriate error messages.

2. **Command Execution:**

   - Your shell must support executing standard Linux commands such as `ls`, `cat`, `echo`, and `sleep`.

3. **Pipes:**

   - Implement support for piping between commands. For example, your shell should correctly handle a command like `grep -o foo file | wc -l`, where the output of one command is used as input to the next. Assume single pipes only and not multiple pipes.

4. **Built-in Commands:**

- In addition to executing external commands, your shell should support some built-in commands. Implement the following built-in functionalities:

  - `cd directoryname`: Change the current working directory of the shell. You can assume the folder names do not contain spaces in them.
  - `history`: Display the history of commands entered by the user.

5. **User Interrupt:**

- Ensure that the shell program terminates after receiving the command `exit`.

6. **Error Handling:**

- Ensure that your shell handles incorrect arguments or command formats gracefully. Display error messages without crashing the shell and continue to prompt for the next command.

Submission

- Submit your source code file in a zipped folder named as your entry number followed by _ and `assignment1`, example: `2020MT60867_assignment1`.

- The submission should be in C, and the file should be named `entrynumber_shell.c`.

- Ensure your code is well-commented and follows best practices for readability and maintainability.

Test Cases

Assume the following folder structure in your home directory:

```
/home/user/
  file1.txt
  file2.txt
  directory1/
      file3.txt
  directory2/
    file4.txt
    subdirectory1/
        file5.txt
  script.sh
```

Here's the content of each file for reference:

- `file1.txt`: Contains the text `hello world`

- `file2.txt`: Contains the text `foo bar`

- `file3.txt`: Contains the text `foo foo foo`

- `file4.txt`: Contains the text `bar bar bar`

- `file5.txt`: Contains the text `foo bar foo`

- `script.sh`: Contains the text `echo "This is a script"`

1 Test Cases and Expected Outputs

1.1 Basic Command Execution

Test Case 1.1:
Execute a simple command `ls`

```
Input: ls
Expected Output:
  file1.txt
  file2.txt
  directory1
  directory2
  script.sh
```

Test Case 1.2:
Execute a command with arguments `cat file1.txt`

```
Input: cat file1.txt
Expected Output:
  hello world
```

Test Case 1.3:
Execute a command with no arguments `echo "Hello"`

```
Input: echo "Hello"
Expected Output:
  Hello
```

1.2 Pipes Between Commands

Test Case 2.1:
Pipe output of `ls` to `grep`

```
Input: ls | grep file
Expected Output:
  file1.txt
  file2.txt
  script.sh
```

Test Case 2.2:
Pipe output of `cat file2.txt` to `grep foo`

```
Input: cat file2.txt | grep foo
Expected Output:
  foo bar
```

1.3   Built-in Commands

Test Case 3.1:
Check current directory after `cd`

```
Input: pwd
Expected Output:
  /home/user/directory2
```

Test Case 3.2:
Show command history with `history`

```
Input: history
Expected Output: A list of recent commands.
```

Test Case 3.3:
Use `cd` with invalid directory

```
Input: cd non_existent_directory
Expected Output:
  The File Path Does not exist
```

Good luck, and happy coding!