CS-333: OPERATING SYSTEMS

Shell and System Calls Project

Name: Hisham Osama Abd Al-salam Ghazy
ID: 81

Introduction

In this project we are implementing a Unix shell which is a command-line interpreter that provides a traditional user interface for the Unix operating system and for Unix-like systems, the implemented shell can run in two modes interactive mode and batch mode.

Code Organization

The code is divided into 6 main folders:

1. **Parser**

   This folder is considered the **first step** in the project flow where it is responsible for **parsing** the user command step by step and **grabbing** the needed information from this command to be used later in execution process

   It consists of 2 files:

   a. Command Parser performing the actual parsing and returning the grabbed information from the command

   b. Utils containing helper methods in the parsing process

2. **Executor**

   This folder is considered the second step in the project flow where it is responsible for **handling** and **executing** of the user command and **displaying** the desired result for the user

   It consists of 2 files:

   a. Command Executer can be considered as a factory where it produces the way of executing the user command

   b. Utils containing helper methods in the execution process where some command are implemented from scratch and the other are used through system calls to the linux system

3. **Variables**

   This folder is responsible for maintaining the environment variables and the session variables where the user commands can request **creating**, **accessing** or **modifying** this data

4. **Files**

   This folder is responsible for the file managing through the project where it is possible to read the command from file and we provide the user with two files (history file and log file)
   It consists of 2 files:

   a. File Processing responsible for creating, opening, reading, appending data to any file

   b. File Setup responsible for creating the files on the project initialization to be available for the user

5. **Utils**

   This folder contains a file carrying the constants used through all the project

6. **Data**

   This folder is a carrier for the history file and the log file presented by the project for the user to manage and trace his work

Code Main Functions

For the main functions found in our project:

1. **Parser**
    - int **isValidLength**(char* command)
      This function is responsible for checking if the length of the command is within the specific allowed range or not
    - void **splitCommand**(char* command, char* delimiter, char* arguments[])
      This function is responsible for splitting the command into array containing the arguments for this command to be user later in the execution process
    - These function are used to return all the data grabbed after parsing the user command step by step
        i. char* **getCommand**()
        ii. void **getCommandArguments**(char* arguments[])
        iii. int **getGroundState**()

2. **Executor**

    - void **execute**(char* command, char* arguments[])
      This function is working as a factory for executing the user command where the function logic changes according to the command
      Some of the commands are implemented from scratch while the other are executed through system calls to the linux system where a child process is created and an attempt to execute the user command is done in this state

    - These function are implemented to support its execution for the user

        i. void **cd**(char* path)

        ii. void **echo**(char* command)

3. **Variables**

    - int **setVariable**(char* equality)
      This function is responsible for testing the equality introduced by the user and if it is a valid equality, it will be added to our session variable in order to be used later in other commands

    - char* **lookupVariable**(char* key)
      This function is used to lookup the value of the environment variables in addition to the session variables created by the user through the program

    - char* **substituteVariables**(char* command)
      This is an important function that part of it was in the parsing process where the user command is substituted with the values of the variables and any special characters to ensure proper execution for the command

4. **Files**

   Functions handling the file management are found here as

    - void **createFile**(char* path)
    - FILE* **openFile**(char* path)
    - void **appendFile**(char* path, char* message)
    - void **printFile**(char* path)

User Guide

To compile and run the code, follow the following steps :

1. Download the project to your machine
2. Open the terminal in the working directory of the project
3. Type “make”
    - This step will open the makefile supported with the project
    - The already found makefile will compile the .c files and link them together
    - It will and produce an executable file to run the program from, this file will be named “shell”
4. Type “./shell” to open the program in the interactive mode
5. Start typing your command and enjoy the results
6. Type “exit” or “CTRL-D” at any time to end the program
7. You can type “./shell PATH” where PATH is considered as the path of the batch file to execute the command form in the batch mode
8. Type “exit” in the batch file or wait for EOF to end the program
9. Check the output folder in the working directory of the project where you can find 2 files supported by the project, one for the history of your commands and the other is for logging the behaviour of the child process through the running session