CSE 231: Operating System 04/08/2023 **Assignment-2 (Section-A)**

SimpleShell: A Unix Shell in C from Scratch

Due by 11:59pm on 10th September 2023
(Total **8%** weightage)
Instructor: Vivek Kumar

No extensions will be provided. Any submission after the deadline will not be evaluated. If you see an ambiguity or inconsistency in a question, please seek clarification from the teaching staff.

Plagiarism: **This is a pair programming-based assignment that you must do with the group member that you have already chosen. No change to the group is allowed. You are not allowed to discuss the approach/solution outside your group.** You should never misrepresent some other group's work as your own. In case any plagiarism case is detected, it will be dealt as per the new plagiarism policy of IIITD **and will be applied to each member in the group.**

Open-sourcing of this assignment solution is not allowed, even after the course gets over.

Even if you are a single member group, there will not be any relaxations in marking scheme/deadlines, and the same rubric will be followed for each group.

General Instructions

a) Hardware requirements:
   a) You will require a machine having any Operating System that supports Unix APIs. You should not use MacOS for OS assignments. You can either have a dual boot system having any Linux OS (e.g., Ubuntu), or install a WSL.

b) Software requirements are:
   a) C compiler and GNU make.
   b) You **must** do version controlling of all your code using github. You should only use a **PRIVATE** repository. If you are found to be using a **PUBLIC** access repository, then it will be considered plagiarism. NOTE that TAs will check your github repository during the demo.

Assignment Details

1. Summary

You have to implement a **SimpleShell** that waits for user input, executes commands provided in the user input, and then repeats until terminated using ctrl-c. The pseudocode for the Shell was discussed in the Lecture 06 and Lecture 07 slides. This assignment will teach you how to use different system calls. You don't have to implement the individual Unix commands that will execute on your SimpleShell.

2. SimpleShell Implementation ("simple-shell.c")

We are not providing any starter code for this assignment, but a detailed description is provided herewith for your implementation:

a. The main job of any shell (including SimpleShell) is to read the user command from the standard input, parse the command string (command and its arguments if any), and execute the command along with the command line arguments provided by the user. All these three steps should be carried out in an infinite do-while loop as shown in the Lecture 06 slides.

b. The shell must display a command prompt (of your choice) where user can provide the command as

mentioned above.

c. The user command has certain restrictions to simplify your implementation. The user command is not supposed to include backslash or quotes. The command and its argument will simply be separated by a whitespace as shown here:

echo you should be aware of the plagiarism policy

The above command should simply be printed by SimpleShell as:

you should be aware of the plagiarism policy

d. As shown in Lecture 06 slides, the command provided by the user will be executed by calling the **launch** method that would create a child process to execute the command provided by the user. Feel free to use any of the seven exec functions for executing the user command.

e. The commands (and the style) that should be supported by the SimpleShell as follows. As you don't have to actually implement the command (e.g., you don't have to write the implementation of "ls" in C), your SimpleShell should be able to execute more commands than the ones listed below (of course not all the Unix commands). You should list **some of the** commands that will not be supported in your design document along with a convincing reason behind it (e.g., the reason should not be stated as some bug in your code).

ls
ls /home
echo you should be aware of the plagiarism policy
wc -l fib.c
wc -c fib.c
grep printf helloworld.c
ls -R
ls -l
./fib 40
./helloworld
sort fib.c
uniq file.txt
cat fib.c | wc -l
cat helloworld.c | grep print | wc -l

The "fib" and "helloworld" are the executables of a Fibonacci number calculator and hello world programs respectively (c-code) that would be made available in the directory where your simple-shell.c will reside. The file "file.txt" is some file that you can create with repetitive lines to test "uniq" command. Note that you might have to know the location of the ELF file for the Unix commands. The "which" command is helpful that would show you that the commands are stored inside the directory **/usr/bin**

f. The concepts and system calls discussed in Lecture 06 and 07 is required for the implementation of your SimpleShell.

g. SimpleShell should also support **history** command that should **only** show the commands entered on the SimpleShell command prompt (along with their command line arguments).

h. Terminating the SimpleShell should display additional details on the execution of each command, e.g., process pid, time at which the command was executed, total duration the command took for execution, etc. You don't need to display details of the commands executed in the past invocations of the SimpleShell. Basically, display all the mentioned details only for the entries in the history.

3. Bonus

a. Support "&" for background processes in your SimpleShell [+1 marks]
b. Your SimpleShell can execute the commands from inside a Shell Script (by reading that file) [+1 marks]

4. Requirements

a. You should strictly follow the instructions provided above.
b. Proper error checking must be done at all places. Its up to you to decide what are those necessary checks.
c. Proper documentation should be done in your coding.
d. Your assignment submission should consist of two parts:
    a. A zip file containing your source files as mentioned above. Name the zip file as “group-ID.zip”, where “ID” is your group ID specified in the spreadsheet shared by the TF.
    b. A design document **inside the above “zip”** file detailing the contribution of each member in the group, detailing your SimpleShell implementation, and the link to your **private** github repository where your assignment is saved.
        i. Your design document should also list the limitations of the SimpleShell, e.g., the user commands that it cannot support along with the reason for that.
e. There should be **ONLY ONE** submission per group.
f. In case your group member is not responding to your messages or is not contributing to the assignment then please get in touch with the teaching staff immediately.

5. <u>Reading / Reference Materials</u>

a. Lecture 06 and Lecture 07 slides
b. man pages of the system call mentioned in the lecture slides.