lab-2.md

Lab 2

These are the instructions for Lab 2 of SOFE3950/CSCI3020 "Operating Systems".

Overview

Lab 2 will be graded in two portions:

- Work shown to a TA during the lab period (the "In-Class Activity")
- Work submitted online after the lab (the "Lab Project")

The Lab Project is a continuation of the In-Class Activity, both of which are requirements for the MyShell project described in the provided PDF.

NB: The MyShell Project does not contain the final lab instructions. Refer carefully to **this document** for clarifications and modifications.

Objectives

- Learn to work in groups to develop software using Git
- Gain experience developing multi-source file projects in C
- Experience using Makefiles and other software build tools

Important Notes

- Work in your designated lab group

Submission Guidelines

1. Add your files to a public or private Git repository.
2. If you choose to use a private repository, please add all specified TA usernames as collaborators. These usernames will be provided in the submission box text on Blackboard.
3. If there are other labs or assignments in your repository, make sure it is obvious which folder contains your lab2 files. You may be docked marks for ambiguous naming in your file stucture.
4. Make sure you also submit your source files as an archive on Blackboard. Acceptable formats are `.zip`, `.tar`, and `.tar.gz` exclusively.

Helpful Resources

It is recommended for this lab activity and others that you save/bookmark the following resources as they are very useful for C programming.

- http://en.cppreference.com/w/c
- http://www.cplusplus.com/reference/clibrary/
- http://users.ece.utexas.edu/~adnan/c-refcard.pdf
- http://gribblelab.org/CBootcamp

The following resources are helpful as you will need to perform string tokenization and use POSIX functions.

- http://www.tutorialspoint.com/c_standard_library/c_function_strtok.htm
- http://www.thegeekstuff.com/2012/06/c-directory/

What is a String Tokenizer?

A string tokenizer is a program or function that splits text into smaller parts called "tokens". For the purposes of this lab, tokens are always separated by whitespace.

For example, if you enter "ls -la /home" in your shell, the text will be converted into an array of strings: "ls", "-la", "/home".

In-Class Activity

1. Either create a new Git repository for this lab, or create a folder in your existing GitHub repository that you created in the previous lab.
2. Download the example source code (provided on Blackboard) and Makefile and use git to add the contents and push it to your Git repository. (Note: it is not necessary to use the template if you do not want to)
3. Ensure that you are able to use the Makefile and C source code examples to build the code using make. If you are having issues you may need to modify the Makefile and set the `CC` variable to gcc instead of clang. (or install `clang`)
4. Before writing any of the source code for your project, review the entire shell project description and ensure that you understand what is required.
5. Next, work in groups to write the documentation for the README file for the shell project, which will be displayed when the help command is used.
6. Finally, using the resources listed above, work on implementing the string tokenization for the shell to process each of the commands.

In-Class Deliverables

Please complete the following deliverables during the lab session and demonstrate your work to the TA before the end of the lab.

1. Demonstrate that you have pushed code to either the Git repository you made in Lab 1 or a new Git repository.
2. Demonstrate that you are able to compile your code.
3. Demonstrate your current README that you have written for the shell project.
4. Demonstrate that you have a working string tokenizer that can be used to tokenize the shell commands. It is imperative that you have a basic string tokenizer working by the end of the lab as it is one of the most challenging aspects of the shell project. The TA is present in the lab to assist you in completing the string tokenizer.

Lab Project

The lab project consists of completing the entire MyShell project with some clarifications and items removed. you will be required to submit your completed MyShell project several weeks later following the lab session, it is not due **during the lab session**.

Please take the time to read these project clarifications during the lab period. Asking questions during the lab period is best, as the entire class can be informed about important distinctions.

Project Clarifications

1. The lab project consists of completing and submitting the entire MyShell project, and is an extension of the lab 2 in-class session where you were given guidance on the project and only required to submit a subset of the entire project.

2. For the purpose of the lab project, make sure to use the existing Git repository created during the lab 2 in-class activity. You will be working in the same groups for the project as in the labs, as this is considered a project for your lab groups.
3. For the project you must ensure that all of the shell features listed below are implemented as specified. Most of these features are merely commands that your shell will provide to the user.
4. Shell command clarifications:
    - each time the directory is changed you must set the character array PWD to the new current directory (this is a shell environment). You must use POSIX functions to change directories on the local filesystem.
    - for dir you must use POSIX functions, see the resources above.
    - for environ list all of your environment strings and their values in the terminal (e.g. PWD, SHELL, etc.)
    - you must use a POSIX or other similar method in C to determine the directory the shell was executed from, the character array SHELL must then be set to this path.
5. item 2, item 4, and item 5 of the MyShell description are NOT REQUIRED for the submission, as they require the use of forking, which has not been covered yet.
6. You should be able to use I/O redirection in Bash to pipe a file containing commands (that your shell supports) to your shell. This should be possible without any modification to your program, but be aware that this may be done during grading.

Lab Project Deliverables

Please complete the deliverables as specified and include your README document, all source files, and the Makefile. You should **submit both** an archive (zip, gzip, or tar) and Git URL on Blackboard.

1. All sources files, all of the Project Requirements described in the shell project document must be met, however recall that items 2 and 5, which require forking are NOT REQUIRED. The primary evaluation will be that all of the shell features are implemented as specified.
2. A Makefile is included so that the source code can be compiled, if your Makefile does not work then marks will be deducted, make sure that your code can be compiled using a Makefile.
3. A detailed README document as outlined in the shell project document, which explains all of the functionality and usage of your shell.