CSCE-110 Fall 2019: Term Project

The Safe Box

Phase 4 - Bonus

The Shell

Assigned: Friday, December 6, 2019

Due Date: Wednesday, December 11, 2019

Overview

In this phase you are required to build a shell for the Safe Box where you can issue commands. You should package everything you developed so far to be invoked as commands from within the target shell. The shell should support the four commands introduced so far in addition to the **exit** and **commit** commands.

Details

In this phase, you will work on the Bonus part. You should implement the Safe Box Shell using a container based approach, like the one we discussed in class. You are required to use the dynamic shared objects (DSO) approach to implement the target commands.

Upon starting the safe_box program, the program will provide a command line shell to the user, where the use can enter different commands, one after the other, to be executed in an interactive mode, and the results will be displayed on the screen, and the prompt will appear again after that allowing the user to enter the next command. In that manner, the shell will establish a session that accepts commands until it receives the **exit** command which will flush the Shelf Index AVL tree on disk and exit the program. The user should provide on the command line of the safe_box shell program two parameters, working directory and key file, which will be used with all commands within the running session; the user does not need to pass those parameters on each command she/he issues on the command line.

The newly added **commit** command is introduced to flush the Shelf Index AVL tree to the shelf.index file on demand at any point in time. It is very important to state that if the shell program is terminated in any way other than executing the exit command, all the modifications done from the last commit or exit are expected to be lost.

Finally, you need to handle all errors and report them to the user. You are required to use exception handling to achieve that and to build an exception handling hierarchy to be able to extend the handling of exceptions and error handling as functionalities are added.

Shell Command Examples

- SB$> list asc
- SB$> list desc
- SB$> import /var/tmp/myfile.data 3 1024 my_encrypted_file.data
- SB$> export my_encrypted_file.data /var/tmp/my_decrypted_file.data
- SB$> delete my_encrypted_file.data
- SB$> commit
- SB$> exit

What to submit

1. All your code.
2. Your code should be split among header files (.h) and source files (.cpp), and all your implementation need to be in the source files. You need to have a very good reason for including implementation in header files, and an explanation of such motives need to be included in your report.
3. Very detailed in-code documentation.
4. A modular Makefile hierarchy.
5. A report describing how to build the software using g++, including all design decisions taken and their alternatives, and explaining anything unusual to the teaching assistant. The report should be in PDF format.
6. Contributions files; each group member will provide a file named after her/his name explaining what she/he did and her/his contribution in this project phase.
7. Do not submit any object, or executable files. Any file uploaded and can be reproduced will result in grade deductions.

How to submit:

First of all, you will need to commit your code to your GIT repository. You will need to upload a text file named phase4_git_commit_id.txt, that includes your GIT commit ID to blackboard. The commit ID submitted by your group on blackboard will be used by the TA to grab your code which will be considered and graded, so make sure to include the correct version commit id. The TA will compare the commit ID date on the GIT repository with the deadline to know if the code was committed on the GIT repository before or after the deadline.

Moreover, you are required also to compress all your work: source code, report, readme file, and any extra information into a zip archive. You should name your archive in the specific format

<Section_ID>_<Team_ID>_Term_Project_F19_Phase4.zip. Finally, upload your archive to blackboard.

If you were able to package your work into a docker environment, which is optional, you can mention your docker repository in your readme file, but it is a must to upload all your source code assignment material to blackboard and commit you code as well to your GIT repository.

This is a group project and you should elect a group captain among your group members, and all the submissions are expected to be performed through her/his blackboard account. IMPORTANT: EVERY GROUP SHOULD SUBMIT THROUGH THE GROUP CAPTAIN ONLY.

IMPORTANT: You will have to present your work and run a demo to the TA and/or the Professor in order to get your grade.

Grading

This bonus is worth 3% of the overall course grade. It will be graded on a 100% grade scale, and then will be scaled down. The grading of this phase will be broken down according to the following criteria:

a.	Safe Box Program		80%
	i.	Functionally running	50%
	ii.	Code Quality	15%
	iii.	In-Line Documentation	15%
b.	Report		20%

The work will be evaluated based on the correct execution of the safe box program based on the functionalities stated in this phase, the code correctness and neatness, design decision aspects, and how much you have utilized the concepts of abstraction, encapsulation, modularity, and hierarchy. It is very important to highlight that although in-code documentation is worth 10%, yet missing or unclear documentation might result in more grade deductions if the grader cannot verify the correctness of your code and logic, and your provided in-line documentation were not sufficient to clarify such matters.