Operating Systems (CS:3620)

Assignment 6 (Implementing A Password Manager) Total points: 100

Due by 11:59 pm of December 9 (Friday), 2016

This assignment will contribute 10% to your final grades.

You can participate in a group of maximum two members.

You can maintain the same group from the previous assignment or form a new group.

Objective: The objective of this assignment is to implement a password manager and familiarize yourself with the OpenSSL cryptography library. Your implementation should take a password file as a command line argument and modify the file according to the **operations** the user asks you to perform. Examples of such user operations include *registering a user account with password*, *deleting a user account*, *checking whether a user account exists*, and *changing a user's account password*. **Your goal is to write the code that implements these operations**.

You will be given a skelton implementation and you have to fill out the code for 4 functions. The function bodies contain all the instructions on how to implement the functionality.

Input—Password File: The password file has the following format. Each line of the password file is of the form: userName : salt : hashedPasswordWithSalt\$n$. The character ":" is the field separator. There will be no space or tab character around the field separator ":". "\$n$" is used to represent a newline character here for clarity and it will be represented as a newline character in the file. The field userName is the username is ASCII representation. The field salt is the 64-byte hexadecimal representation in ASCII of the randomly generated 32-byte binary value. You will be given a utility function which will enable you to generate random bytes. The field hashedPasswordWithSalt is the 128-byte hexadecimal representation in ASCII of the `SHA512(user_password • salt)` where • means the concatenation operator. If we have two strings "`dsdskjjfdkg`" and "`123456`", then `dsdskjjfdkg•123456 = dsdskjjfdkg123456`. Note that, **the input password file can be empty**.

Constraints:

1. The userName field should have a length greater than 5 but less than 32 characters.
2. The allowed characters in the userName field is "a-z", "A-Z", and "0-9".
3. The password generated by the user should have a length greater than 8 but less than 32 characters.
4. The allowed characters in the password are: "a-z", "A-Z", "0-9", and "@#$%&*()+=".
5. For each user the salt will be randomly generated unique value. It will be of size 32 bytes.

Executing Your Program: If the executable file generated after compiling your source code is called `PassManager` then we will invoke it as ./`PassManager PasswordFile`
Advice—Please try to understand the code first and then implement the required functions.

Skeleton: The skeleton provided contains the following files: `Makefile`, `PassManager.c` (the main source where you have to fill out the 4 functions), `util.h` (contains all the utility functions used by PassManager.c and you will need to implement the 4 functions), `sha512usage.c` (shows how to use sha512 hashing function and how to generate random bytes), `sample_password_file` (a sample password file), and a file containing the password for all user accounts in the sample password file.

To compile the skeleton file type `make` in the command line inside the directory containing the skeleton files. This will generate an executable named `pmanager`. To execute type `./pmanager sample_password_file`.

For compiling the `sha512usage.c` file type `gcc sha512usage.c -o password -lcrypto` in the command line. This will generate an executable `password`. When you execute `password`, it will wait for you to type in a password and it will then randomly generate a salt, and then output the hashed password in the terminal. It will do this in a loop until you hit `Ctrl+D`.

The functions to implement: You will implement the following four functions. Please consult the `PassManager.c` for further instructions.

int register_user(unsigned char *`user`, unsigned char *`password`) This function will return 1 on success and -1 on failure. It will create a password file entry into the internal data structure (linked list) for the user specified with `user` and `password`.

int delete_user(unsigned char *`user`, unsigned char * `password`) This function will return 1 on success and -1 on failure. It will delete the password file entry from the internal data structure (linked list) for the user specified with `user` and `password`.

int match_user(unsigned char *`user`, unsigned char * `passwd`) This function will return 1 on success and -1 on failure. It will consult the internal data structure (linked list) storing the password file entry to check whether the user specified with `user` has the password specified by `passwd`.

int change_user_password(unsigned char *`user`, unsigned char * `passwd_current`, unsigned char * `passwd_new`) This function will return 1 on success and -1 on failure. This function will change the user password from `passwd_current` to `passwd_new` in the internal data structure. Note that, you will generate a new salt while storing the new password.

Submission: Please submit your source code through ICON. Your submissions should be compliant with the following directory structure. If you are doing this assignment individually, please follow the individual submission guideline. In case, you are doing this assignment as a group, please follow the group submission guideline.

Individual Submission: The top level directory of your submission should be named ⟨last-name⟩-⟨first-name⟩-Assignment-6. If a student's name is Bob Marley and he were to submit the assignment, his top level folder name will have the name `Marley-Bob-Assignment-6`. All your source code will be inside of this top level directory. You will then zip your top-level directory (e.g., `Marley-Bob-Assignment-6.zip`) and submit them.

Submission for groups of 2 students: The top level directory of your submission should be named ⟨last-name$_1$⟩-⟨last-name$_2$⟩-Assignment-6. If a group have two students, namely, `Bob Marley` and `Omar Chowdhury`, their top level folder name will have the name `Marley-Chowdhury-Assignment-6` or `Chowdhury-Marley-Assignment-6`. All your source code will be inside of this top level directory. You will then zip your top-level directory (e.g., `Marley-Chowdhury-Assignment-6.zip`) and submit them.

Programming Language: You are required to use either C or C++.

Cheating and Collaboration: *This is a group exercise, you can discuss with other groups but cannot copy their source code. Please do not copy source code from Internet.* The scheduling algorithm implementations that can be found in the Internet have a lot of subtle bugs which can be used to fingerprint those implementations.

Grading Rubric: Each of your implementations will be checked with 50 test cases. You will get points for a particular test case if your program behaves according to the specification.