HACETTEPE UNIVERSITY COMPUTER ENGINEERING

BBM203 Programming Lab.

ASSIGNMENT 4

MELTEM TOKGÖZ

21527381

b21527381@cs.hacettepe.edu.tr

Subject : Login System with Character Tree

TAs: R.A.Pelin Canbay

Programming Lanuage: C

Due Date: 06.01.2019

1. Software Usage

Software input is input.txt. Input files reads such that an argument.
Software output is write output.txt. I wrote with CodeBlocks in C.

2. Software Design Note

In this assignment, I designed a Login System with Character Tree. I practiced on the tree data structure.

2.1 What is the problem?

In this assignment, I am expected to write an application that constructs a Login System with Character Tree. The application will take the input.txt file in the current directory and read its contents. In order to create the Character tree, there are some commands for adding, deleting and refreshing.

Structure of Commands:

-a username password #add username to the tree with the given password

-s username #search with the given username and return the password if it is

-q username password #send query for the password with the given username

-d username #delete username and its password

-l #list the tree

2.2 Solution

I used the trie structure in my homework and did all of the functions by navigating over those nodes in trie.

I first read the input.txt file in my homework. Then I created structure to create a trie. This trie structure has the following properties.

//Trie structure:

struct trie *next[R]; //this next node in trie

bool End; // The last node?

char *password; //This is password for user

bool Leaf; //The leaf node?

I created a new trie from the trie structure I established.(root node). I've separated each line by spaces and I've sent root node and appropriate argument to the functions I've written according to the appropriate command names. Now let me talk about these functions and their implementations.

3.Functions

1. **int main(int argc, char *argv[])**

   This function is the main function in which the program is started. It takes one input file name as an arguments and one output file.(output.txt) In this function, the trie is created and the createNode function is called. This function, input files are read in line by line and the required functions are called.

2. **struct trie *createNode(void)**

   This function was done to create a new node.It creates a new node with dynamic memory allocation and then returns that node as a result.

3. **void addUser(struct trie *root,char *user, char *password,FILE* f)**

   This function takes the output file as a parameter. Used to add a new user to the tree.When doing so, it uses the createNode function.Uses the searching function to check whether the desired name has already been added.

4. **int searching(struct trie *root,char *user)**

   This function check username.If username exist in trie, returned 3.
   In doing so, it compares the letters in the username with the nodes of the tree starting from the root.

5. **bool searchUser(struct trie *root,char *user,FILE* f)**

   This function is made to find out if the user is in the tree.
   If a node that starts with a letter in the user name does not return from the root, there is no user and printing "no record".
   if the root node is the required node, but it doesn't match the given username, I'm printing an "incorrect username". I do this by drawing attention to the level of the node.

If the username exists in the tree but does not have a password, this is called "not enough username". If the user name and password are present then the user is in the tree, and returns true.

6. **void loginUser(struct trie *root,char* username, char* password,FILE* f)**

   If the process you want is in the tree and the password matches the given password, the user will login with this function. I check the status of the user and password with the check function and according to him the are given to login.

7. **int check(struct trie *root,char *user,char *password)**

   The operation logic of the check function is the same as search.
   It only checks the accuracy of the password as extra.

8. **void deleteUser(struct trie* root, char* user, int depth,FILE* f)**

   With the searching( ) function, the user checks whether it is suitable for deletion from the tree.If the user to delete the user to deleteThisUser( ) function sends it.

9. **bool isEmptyTrie(struct trie* root)**

   This function checks whether the tree is empty.

10. **struct trie* deleteThisUser(struct trie* root, char* user, int depth)**

    This function removes the user nodes from the tree with the free function according to the situation.

11. **void listUser(struct trie* root, char str[], int level,FILE* f)**

    This function lists all the users in the tree. In doing so I used the level of the tree and did it as recursive.

12. **bool isLeafNode(struct trie* root)**

    Controls whether the given node is a leaf node