

Hacettepe University
Computer Science
and Engineering Department

Name and Surname : ATAKAN AYYILDIZ

Identity Number : 21526681

Course : BBM203 PROGRAMMING LAB

Subject : Login System with Character Tree

Project : ASSIGNMENT 4

Experiment : Developing a simple search engine.

Data Due : 06.01.2019 (23:55)

Main Program : ass4.c

PROBLEMS:

1.I have to create a Trie that stores a Word, password, boolean control statement and subTrie (that stores other words)

2.I should have written a Create function to make it easy the work.

3.I have to read input File but its length not static.

4.I need a function for passing datas to the trie. But there are several problems (incorrect-not enough...).Also search-query-delete-list.

5.To passing datas to teh tree I need a index number because i create *pointer[26]. So i create a char to index function.

6.After reading file i need to decide which function should that go.

SOLUTIONS:

1.First of all i need to read input file line by line.Therefore i used feof (it was not guaranteed if there would be newline at the end of line i get segmentation fault).After that i read them with fgets i parse them with strtok.

2.After seperation i pass them to the function. Also if key[1]== 'd' i checked if there is no record do not enter the function print no record.

3.I create 5 function for them.

FUNCTIONS:

1.First function:

Add :

It takes 4 parameters. First parameter is our Trie structure. Second parameter is intended name of input. Last parameter is the password of its. Last parameter is output file.

It returns changed Trie.

Add function create a temporary Trie structure (named *temp). Then it goes for loop strlen(name) times.

First it search its index number (ascii number of intended word).

Then it checked is that NULL if it is it calls Create function.

Then Word passes.

It would continue until the end of name.

Then it controls is it reserved.

After that password is passing.

Then function returns.

2.Second Function:

Search:

It takes 3 parameters. First parameter is our trie structure. Second parameter is intended name of input. Last parameter is output file.

It returns void.

First of all we control that first character is on trie if not print no record and return.

Otherwise we control other character of names if next character is not on the trie print incorrect and return.

If the last character has not password print not enough username and return.

If everything is correct and has password print password then return.

3.Third Function:

Query:

It takes 4 parameters. First parameter is our Trie structure. Second parameter is intended name of input. Third parameter is the password of its. Last parameter is output file

It returns void. It works same as second function search.

First of all we control that first character is on trie if not print no record and return.

Otherwise we control other character of names if next character is not on the trie print incorrect and return.

If we are on the last Word and has no password print not enough username.

Every character is on trie but password is not correct print incorrect password.

Everything is correct then print succesful login.

4.Fourth Function:

Delete:

It takes 4 parameters. First parameter is our Trie structure. Second parameter is intended name of input. Third parameter is a counter that which word will be deleted. Last parameter is output file.

It return void. Also this function is recursive

It works same for no record, incorrect username and not enough username.

If we are on the last word and trie node has password delete node . Also we control is that node is leaf or not. If it is leaf node delete it and free. After that we control first word to delete it.

5.Fifth Function:

List:

It takes 5 parameters. First parameter is our Trie structure. Second parameter is temporary array. Third parameter is name counter(it goes until the last character of the name). Fourth parameter is boolean isOnly that control this name has branch on the trie. Last parameter is output file.

It returns void. Also this function is recursive.

Firstly function checked the subtrie number.

If subTrie number more than one and we are not on the root convert isonly to false and print that.

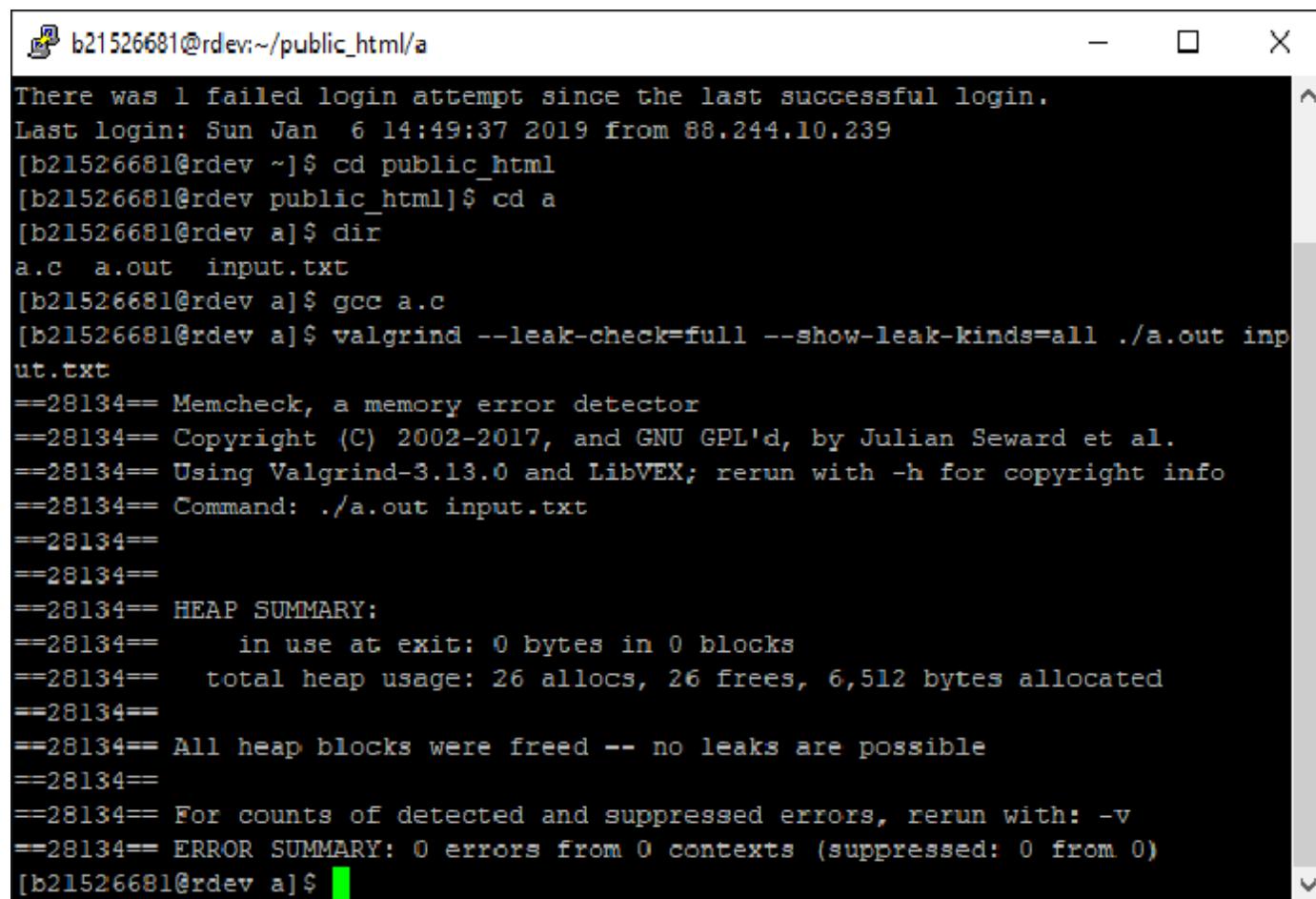
Else is only is true (there is no branch) and we are on the last word print that.

Also print that (with branch) .

Otherwise go recursion.

Then free whole root.

Also leak checked.



```
b21526681@rdev:~/public_html/a
There was 1 failed login attempt since the last successful login.
Last login: Sun Jan  6 14:49:37 2019 from 88.244.10.239
[b21526681@rdev ~]$ cd public_html
[b21526681@rdev public_html]$ cd a
[b21526681@rdev a]$ dir
a.c  a.out  input.txt
[b21526681@rdev a]$ gcc a.c
[b21526681@rdev a]$ valgrind --leak-check=full --show-leak-kinds=all ./a.out inp
ut.txt
==28134== Memcheck, a memory error detector
==28134== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==28134== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==28134== Command: ./a.out input.txt
==28134==
==28134==
==28134== HEAP SUMMARY:
==28134==    in use at exit: 0 bytes in 0 blocks
==28134==   total heap usage: 26 allocs, 26 frees, 6,512 bytes allocated
==28134==
==28134== All heap blocks were freed -- no leaks are possible
==28134==
==28134== For counts of detected and suppressed errors, rerun with: -v
==28134== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[b21526681@rdev a]$
```