**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Программирование»**

**Тема: Обход файловой системы**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3342 |  | Роднов И.С. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Целью работы является реализование программы на языке C для поиска файла-минотавра в файловой системе, при помощи ее обхода с использованием рекурсивных алгоритмов, начиная с корневой директории "labyrinth”.

## Задание

Вариант 1.

Дана некоторая корневая директория, в которой может находиться некоторое количество папок, в том числе вложенных. В этих папках хранятся некоторые текстовые файлы, имеющие имя вида ​​.txt.  
Требуется найти файл, который содержит строку "Minotaur" (файл-минотавр).  
Файл, с которого следует начинать поиск, всегда называется file.txt (но полный путь к нему неизвестен). Каждый текстовый файл, кроме искомого, может содержать в себе ссылку на название другого файла (эта ссылка не содержит пути к файлу). Таких ссылок может быть несколько.

Программа должна вывести правильную цепочку файлов (с путями), которая привела к поимке файла-минотавра.

Ваше решение должно находиться в директории **/home/box**, файл с решением должен называться **solution.c**. Результат работы программы должен быть записан в файл **result.txt**. Ваша программа должна обрабатывать директорию, которая называется **labyrinth**.

## Выполнение работы

1. Программа начинает свою работу с поиска файла "file.txt" в корневой директории "labyrinth".
2. После нахождения файла "file.txt" программа сохраняет путь до него.
3. Затем программа, начиная с этого файла, изучается содержимое каждого файла и рекурсивно проходит по файловой системе.
4. Программа продолжает поиск до тех пор, пока не будет обнаружен файл-минотавр или не будет достигнут конец структуры файловой системы.
5. Результат работы программы записывается в файл "result.txt", который находится в директории /home/box, как требуется по условию задачи.

Разработанный программный код см. в приложении А.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| № п/п | Входные данные | Выходные данные |
|  | file.txt:  @include file1.txt  @include file4.txt  @include file5.txt  file1.txt:  Deadlock  file2.txt:  @include file3.txt  file3.txt:  Minotaur  file4.txt:  @include file2.txt  @include file1.txt  file5.txt:  Deadlock | ./root/add/add/file.txt  ./root/add/mul/add/file4.txt  ./root/add/mul/file2.txt  ./root/add/mul/file3.txt |

## Выводы

Реализованна программа на языке программирования C, которая обходит файловую систему, начиная с заданной корневой директории "labyrinth" и ищет файл минотавр.

# Приложение А Исходный код программы

Название файла: main.c

#include <stdio.h>

#include <stdlib.h>

#include <dirent.h>

#include <string.h>

#include <regex.h>

char\* find\_path(const char \* dirPath, const char \* cmpPath){

if(strstr(dirPath, cmpPath) != NULL){

return strdup(dirPath);

}

DIR \*dir = opendir(dirPath);

if(dir){

struct dirent \*de = readdir(dir);

while(de){

if(strcmp(".", de->d\_name) && strcmp("..", de->d\_name)){

char\* new\_dirPath = malloc(strlen(dirPath) + strlen(de->d\_name) + 2); // +2 for '/' and null terminator

if(new\_dirPath == NULL){

perror("Memory allocation failed");

exit(EXIT\_FAILURE);

}

sprintf(new\_dirPath, "%s/%s", dirPath, de->d\_name);

char\* result = find\_path(new\_dirPath, cmpPath);

free(new\_dirPath);

if(result != NULL){

closedir(dir);

return result;

}

}

de = readdir(dir);

}

closedir(dir);

}

return NULL;

}

void print\_res(char\*\* files, int k){

FILE \*fw = fopen("result.txt", "at");

if(fw == NULL){

perror("Failed to open result.txt for writing");

exit(EXIT\_FAILURE);

}

int i;

for(i = 0; i < k; i++){

fputs(files[i], fw);

}

fclose(fw);

}

char\*\* find\_file(const char \* path, const char\* filename, char\*\* path\_to\_files, int\* k){

char\* path\_to\_first = find\_path(path, filename);

if(path\_to\_first == NULL){

fprintf(stderr, "File %s not found in directory %s\n", filename, path);

return NULL;

}

FILE\* fr = fopen(path\_to\_first, "r");

if(fr == NULL){

perror("Failed to open file for reading");

free(path\_to\_first);

return NULL;

}

strcat(path\_to\_first, "\n");

char result[100];

char\* pattern = "([!-~]+\\.txt)";

regex\_t re;

if(regcomp(&re, pattern, REG\_EXTENDED)){

perror("Failed to compile regular expression");

exit(EXIT\_FAILURE);

}

int f = 0;

while(fgets(result, 100, fr)){

regmatch\_t pmatch[1];

if(regexec(&re, result, 1, pmatch, 0)==0){

if(f == 0){

free(path\_to\_files);

path\_to\_files[\*k] = path\_to\_first;

(\*k)++;

}

f++;

char new\_file[10];

int j = 0;

for(int i = pmatch[0].rm\_so; i!= pmatch[0].rm\_eo; i++){

new\_file[j++] = result[i];

}

new\_file[j] = '\0';

char\*\* new\_path\_to\_files = find\_file(path, new\_file, path\_to\_files, k);

if(new\_path\_to\_files != NULL){

regfree(&re);

return new\_path\_to\_files;

}

}

}

if(!(strcmp(result, "Minotaur"))){

path\_to\_files[\*k] = path\_to\_first;

(\*k)++;

}

fclose(fr);

regfree(&re);

return path\_to\_files;

}

void free\_res(char\*\* path\_to files, int k){

for (int i = 0; i < k; ++i) {

free(path\_to\_files[i]);

}

free(path\_to\_files);

}

int main(){

const char path[100] = "./labyrinth";

char \*\* path\_to\_files = malloc(sizeof(char\*) \* 100);

if(path\_to\_files == NULL){

perror("Memory allocation failed");

return EXIT\_FAILURE;

}

int k = 0;

path\_to\_files = find\_file(path, "file.txt", path\_to\_files, &k);

if(path\_to\_files == NULL){

fprintf(stderr, "No files found\n");

return EXIT\_FAILURE;

}

print\_res(path\_to\_files, k);

free\_res(path\_to\_files, k);

return 0;

}