**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 |  | Лучкин М.А. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Целью данной лабораторной работы является создание программы на языке C для поиска файла-минотавра в структуре файловой системы. Программа должна рекурсивно обходить директории, начиная с корневой директории "labyrinth", и находить файл с именем "file.txt". Затем она должна анализировать содержимое этого файла и, если обнаруживает строку "Minotaur", записывать путь к этому файлу вместе с цепочкой всех файлов, которые привели к обнаружению файла-минотавра.

## 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. Определяется структура ContainerString для хранения массива строк.
2. Реализуется функция pushBack для добавления строки в массив.
3. Создается функция getDataFromFile для считывания данных из файла в структуру ContainerString.
4. Функция output используется для записи данных в файл.
5. Реализуется функция joinStrings для объединения строк с разделительным символом.
6. Функция process выполняет рекурсивный обход файловой системы по указанному пути.
7. При обнаружении файла с именем "file.txt", происходит обработка его содержимого согласно условиям в коде.
8. Основная функция main запускает процесс обработки файлов, начиная с указанного пути "./labyrinth" и целевым именем файла "file.txt".
9. 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

Была разработана программа на языке C, которая обходит структуру файловой системы, начиная с заданной корневой директории.

# Приложение А Исходный код программы

Название файла: main.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <dirent.h>

#define START\_PATH "./labyrinth"

typedef struct ContainerString

{

int size;

int capacity;

char\*\* array;

} ContainerString;

void pushBack(ContainerString\* v, char\* el)

{

if (v->size >= v->capacity) {

v->capacity = (v->size == 0) ? 2 : v->capacity\*v->capacity;

v->array = realloc(v->array, v->capacity\*sizeof(char\*));

if (v->array == NULL){

exit(0);

}

}

v->array[v->size++] = el;

}

ContainerString\* getDataFromFile(char\* file\_path)

{

ContainerString\* content = (ContainerString\*) calloc(1, sizeof(ContainerString));

if (content == NULL){

exit(0);

}

FILE\* fin = fopen(file\_path, "r");

char\* string = (char\*) malloc(128\*sizeof(char));

if (string == NULL){

exit(0);

}

while (fgets(string, 128, fin)) {

if (strchr(string, '\n')) {

\*(strchr(string, '\n')) = '\0';

}

pushBack(content, string);

string = (char\*) malloc(128\*sizeof(char));

if (string == NULL){

exit(0);

}

}

fclose(fin);

free(string);

free(content);

return content;

}

void output(char\* file\_path, char\* labyrinth\_path)

{

FILE\* fout = fopen(file\_path, "w");

if (fout == NULL){

exit(0);

}

fprintf(fout, labyrinth\_path, "%s\n");

fclose(fout);

}

char\* joinStrings(char\* str1, char\* str2, char symbol)

{

char\* subPath = (char\*) calloc((strlen(str1) + strlen(str2) + 2), sizeof(char));

if (subPath == NULL){

exit(0);

}

strcpy(subPath, str1);

if (strcmp(str1, "") != 0) {

subPath[strlen(str1)] = symbol;

}

strcat(subPath, str2);

return subPath;

}

void process(char\* dir\_name, char\* target\_file\_name, char\* labyrinth\_path)

{

DIR\* dir = opendir(dir\_name);

if (dir == NULL){

exit(0);

}

struct dirent\* de;

while ( de = readdir(dir) ) {

if (de->d\_type == DT\_DIR) {

if ((strcmp(de->d\_name, ".") != 0 && strcmp(de->d\_name, "..") != 0)) {

process(joinStrings(dir\_name, de->d\_name, '/'), target\_file\_name, labyrinth\_path);

}

}

if (de->d\_type == DT\_REG && (strcmp(de->d\_name, target\_file\_name) == 0)) {

char\* subPath = joinStrings(dir\_name, target\_file\_name, '/');

ContainerString\* data = getDataFromFile(subPath );

if (strcmp(data->array[0], "Deadlock") == 0) {

closedir(dir);

for (int i = 0; i < data->size; i++){

free(data->array[i]);

}

free(data->array);

free(data);

return;

}

if (strcmp(data->array[0], "Minotaur") == 0) {

output("result.txt", joinStrings(labyrinth\_path, subPath, '\n'));

closedir(dir);

for (int i = 0; i < data->size; i++){

free(data->array[i]);

}

free(data->array);

free(data);

} else {

for (int i=0; i<data->size; i++) {

process(START\_PATH, strchr(data->array[i], ' ') + 1,

joinStrings(labyrinth\_path, subPath, '\n'));

}

}

closedir(dir);

for (int i = 0; i < data->size; i++){

free(data->array[i]);

}

free(data->array);

free(data);

}

}

}

int main()

{

process(START\_PATH, "file.txt", "");

}