**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. 3344** |  | **Сьомак Д.А.** |
| **Преподаватель** |  | **Глазунов** **С.А.** |

**Санкт-Петербург**

**2024**

**Цель работы**

Получение практических навыков работы с файловой системой на языке Си. Освоение рекурсивного метода обхода файлового дерева путём написания программы.

**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.

**Выполнение работы**

Были подключены стандартные библиотеки stdio.h, string.h, stdlib.h и dirent.h для работы с файловыми директориями. Также были созданы глобальные переменны для хранения количества путей и состояния флага. Были реализованы:

CatPaths - вспомогательная функция конкатенации строк. На вход получает две строки, в виде путей к файлам или составляющих этих путей и возвращает путь, образованный этими строками.

findFileByName - функция поиска пути к файлу по его названию. Функция получает на вход путь к директории и название файла, который необходимо в ней найти. Если функция не находит файла, но находит директорию, то начинается рекурсия, при этом функция получает на вход путь к другой директории (полученный благодаря первой функции). Как только функция находит файл, рекурсия заканчивается, директория закрывается, и функция возвращает путь к этому файлу.

findPath - функция поиска всех путей к файлам, содержащим минотавра. Функция получает на вход название файла и массив строк, который будет содержать пути к файлам с минотавром. Сначала вызывается вторая функция, в которую передаётся начальная директория и входное название файла. По найденному пути открывается файл, из него построчно считывается содержимое. Если строка это: Deadlock(функция ничего не возвращает), Minotaur (поднимается флаг и в массив записывается путь к этому файлу), include... (название файла, которое идёт после include, используется для рекурсивного вызова этой функции. Если по итогу рекурсии минотавр был найден, он записывается в массив). Когда функция проверит содержимое всех файлов и запишет все пути, файловый поток будет закрыт, и функция завершит работу.

main - основная функция, внутри которой выделяется память для массива с нужными путями и происходит вызов третьей функции, с названием файла, с которого по заданию стоит начинать поиск минотавров. Далее открывается файл для записи результатов и туда переносятся пути, которые были найдены в результате работы третьей функции. В конце файловый поток закрывается, и программа завершает свою работу.

Исходный код см. в приложении А.

**Тестирование**

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
| 1. | 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 | корректно |

## **Выводы**

Был получен практический опыт работы с файловой системой. Была написана программа, внутри которой был реализован рекурсивный обход файлового дерева для нахождения пути к обозначенным файлам.

# **ПРИЛОЖЕНИЕ А** **ИСХОДНЫЙ КОД ПРОГРАММЫ**

Название файла: Somak\_Demid\_lb3.c

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <dirent.h>

int minotaur\_flag = 0;

int count = 0;

char \*catPaths(const char \*first\_path,const char \*second\_path){

char\* res\_path = malloc(sizeof(char) \* (strlen(first\_path)+ strlen(second\_path)+2));

sprintf(res\_path, "%s/%s", first\_path, second\_path);

return res\_path;

}

char \*findFileByName(const char \*root, const char \*fileName)

{

DIR \*root\_dir = opendir(root);

if (root\_dir == NULL)

printf("Failed to open directory: %s \n", root);

struct dirent\* dir = readdir(root\_dir);

char \*path = NULL;

while (dir)

{

if (dir->d\_type == DT\_REG && strcmp(dir->d\_name, fileName) == 0)

{

path = catPaths(root,fileName);

}

else if (strcmp(dir->d\_name, ".") != 0 && strcmp(dir->d\_name, "..") != 0 && dir->d\_type == DT\_DIR)

{

char \*new\_root = catPaths(root,dir->d\_name);

path = findFileByName(new\_root, fileName);

free(new\_root);

}

if (path)

break;

dir = readdir(root\_dir);

}

closedir(root\_dir);

return path;

}

void findPath(char \*fileName, char massOfPaths)

{

char\* file\_Path = findFileByName(".",fileName);

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

if (file == NULL)

return;

char file\_string[256];

while (fgets(file\_string, 256, file) != NULL && minotaur\_flag != 1)

{

if (strstr(file\_string, "Deadlock") )

{

return;

}

else if (strstr(file\_string, "Minotaur"))

{

minotaur\_flag = 1;

massOfPaths[count] = malloc((strlen(file\_Path)+1) \* sizeof(char));

strcpy(massOfPaths[count++], file\_Path);

massOfPaths = realloc(massOfPaths, (count+1) \* sizeof(char\*));

}

else if (strncmp(file\_string, "@include", 8) == 0)

{

if(file\_string[strlen(file\_string) - 1] == '\n')

file\_string[strlen(file\_string)-1] = '\0';

memmove(&file\_string[0], &file\_string[9], sizeof(char) \* (strlen(file\_Path)+1));

findPath(file\_string, massOfPaths);

if (minotaur\_flag)

{

massOfPaths[count] = malloc((strlen(file\_Path)+1) \* sizeof(char));

strcpy(massOfPaths[count++], file\_Path);

massOfPaths = realloc(massOfPaths, (count+1) \* sizeof(char\*));

}

}

}

fclose(file);

return;

}

int main(void)

{

char paths = (char\*\*)malloc(sizeof(char\*));

findPath("file.txt", paths);

FILE \*file\_result = fopen("result.txt", "w");

if (file\_result == NULL)

{

printf("Failed to open file.txt");

return 0;

}

for (int i = count-1; i >= 0; i--)

{

fprintf(file\_result, "%s\n", paths[i]);

}

fclose(file\_result);

return 0;

}