**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Программирование»**

Тема: «Обход файловой системы»

|  |  |  |
| --- | --- | --- |
| Студент гр. 3343 |  | Гребнев Е.Д. |
| Преподаватель |  | Государкин Я.С. |

Санкт-Петербург

2024

# Цель работы

Создать программу на языке C, которая рекурсивно ищет определенные файлы в директориях, используя знания о работе с файлами и директориями.

# Задание

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

Реализованный код на языке C выполняет поиск файла file.txt в директории "labyrinth" и ее поддиректориях. Если файл найден, программа открывает его и ищет строку, соответствующую FINAL\_TARGET ("Minotaur"). Если такая строка найдена, программа заканчивает выполнение. Если в файле встречается строка, соответствующая DEADLOCK\_TARGET ("Deadlock"), выполнение также завершается. В противном случае, программа ищет строки вида @include <file\_name>, где <file\_name> - это имя файла, и рекурсивно вызывает саму себя для поиска файла <file\_name>. Поиск осуществляется в директории "labyrinth" и ее поддиректориях.

Когда все цели найдены, программа записывает пути к найденным файлам в обратном порядке в файл "result.txt".

# Выводы

В процессе выполнения лабораторной работы был изучен синтаксис языка C для работы с файлами и директориями, а также была реализована программа, которая рекурсивно обходит файловую систему для поиска конкретных файлов или строк в ней.

ПРИЛОЖЕНИЕ А  
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

#include <dirent.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_INCLUDES 20000

#define MAX\_PATH\_LEN 1024

#define INITIAL\_PATH "./labyrinth"

#define INITIAL\_FILE "file.txt"

#define OUTPUT\_FILE "result.txt"

char \*deadlocks[MAX\_INCLUDES];

int deadlock\_count = 0;

char trace[MAX\_INCLUDES][MAX\_PATH\_LEN];

int trace\_count = 0;

void add\_deadlock(const char \*deadlock) {

for (int i = 0; i < deadlock\_count; ++i) {

if (strcmp(deadlocks[i], deadlock) == 0) {

return;

}

}

deadlocks[deadlock\_count] = (char \*)malloc((strlen(deadlock) + 1) \* sizeof(char));

strcpy(deadlocks[deadlock\_count], deadlock);

deadlock\_count++;

}

void get\_deadlocks(const char \*path) {

DIR \*dir = opendir(path);

if (!dir) return;

struct dirent \*entry;

while ((entry = readdir(dir))) {

if (strcmp(entry->d\_name, ".") == 0 || strcmp(entry->d\_name, "..") == 0)

continue;

char fullpath[MAX\_PATH\_LEN];

snprintf(fullpath, sizeof(fullpath), "%s/%s", path, entry->d\_name);

if (entry->d\_type == 4) {

get\_deadlocks(fullpath);

get\_deadlocks(fullpath);

} else {

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

if (!file) continue;

char line[MAX\_PATH\_LEN], file\_include[MAX\_PATH\_LEN];

int include\_count = 0;

while (fgets(line, sizeof(line), file)) {

if (strcmp(line, "Deadlock") == 0) {

add\_deadlock(entry->d\_name);

break;

}

if (strcmp(line, "Minotaur") == 0) {

include\_count++;

break;

}

if (sscanf(line, "@include %s", file\_include) == 1) {

int isDeadlock = 0;

for (size\_t i = 0; i < deadlock\_count; i++) {

if (strcmp(file\_include, deadlocks[i]) == 0) {

isDeadlock = 1;

break;

}

}

if (!isDeadlock) include\_count++;

}

}

if (include\_count == 0) {

add\_deadlock(entry->d\_name);

}

fclose(file);

}

}

closedir(dir);

}

void process\_directory(const char \*path, const char \*filename) {

DIR \*dir = opendir(path);

if (!dir) return;

struct dirent \*entry;

while ((entry = readdir(dir))) {

if (strcmp(entry->d\_name, ".") == 0 || strcmp(entry->d\_name, "..") == 0)

continue;

char fullpath[MAX\_PATH\_LEN];

snprintf(fullpath, sizeof(fullpath), "%s/%s", path, entry->d\_name);

if (entry->d\_type == 4)

process\_directory(fullpath, filename);

else if (strcmp(entry->d\_name, filename) == 0) {

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

if (!file) continue;

char line[MAX\_PATH\_LEN], file\_include[MAX\_PATH\_LEN];

while (fgets(line, sizeof(line), file)) {

if (strcmp(line, "Minotaur") == 0) {

FILE \*output\_file = fopen(OUTPUT\_FILE, "w");

for (int i = 0; i < trace\_count; i++) {

fprintf(output\_file, "%s\n", trace[i]);

}

fprintf(output\_file, "%s", fullpath);

fclose(output\_file);

return;

}

if (sscanf(line, "@include %s", file\_include) == 1) {

int isDeadlock = 0;

for (size\_t i = 0; i < deadlock\_count; i++) {

if (strcmp(file\_include, deadlocks[i]) == 0) {

isDeadlock = 1;

break;

}

}

if (!isDeadlock) {

strcpy(trace[trace\_count++], fullpath);

process\_directory(INITIAL\_PATH, file\_include);

}

}

}

fclose(file);

}

}

closedir(dir);

}

int main() {

get\_deadlocks(INITIAL\_PATH);

get\_deadlocks(INITIAL\_PATH);

process\_directory(INITIAL\_PATH, INITIAL\_FILE);

return 0;

}