**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 |  | Иванов П.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 <stdio.h>

#include <dirent.h>

#include <string.h>

#include <stdlib.h>

#include <sys/types.h>

#define MAX\_LINE\_LENGTH 250

#define DEFAULT\_DIRECTORY "labyrinth"

#define FINAL\_TARGET "Minotaur"

#define DEADLOCK\_TARGET "Deadlock"

char \*combinePaths(const char \*path1, const char \*path2)

{

int resultLength = strlen(path1) + strlen(path2) + 2;

char \*resultPath = malloc(resultLength \* sizeof(char));

sprintf(resultPath, "%s/%s", path1, path2);

return resultPath;

}

char \*findTargetFile(const char \*directoryName, const char \*targetFileName)

{

char \*foundFilePath = NULL;

DIR \*directory = opendir(directoryName);

if (directory)

{

struct dirent \*dirEntry = readdir(directory);

while (dirEntry)

{

if (dirEntry->d\_type == DT\_REG && !strcmp(dirEntry->d\_name, targetFileName))

{

foundFilePath = combinePaths(directoryName, targetFileName);

break;

}

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

{

char \*newDirectory = combinePaths(directoryName, dirEntry->d\_name);

foundFilePath = findTargetFile(newDirectory, targetFileName);

free(newDirectory);

}

if (foundFilePath)

break;

dirEntry = readdir(directory);

}

closedir(directory);

}

else

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

return foundFilePath;

}

int checkTarget(const char \*filePath, char \*\*result)

{

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

if (!file)

{

printf("Failed to open %s file\n", filePath);

exit(0);

}

char line[MAX\_LINE\_LENGTH];

char \*readResult;

while ((readResult = fgets(line, MAX\_LINE\_LENGTH, file)) != NULL)

{

if (strcmp(line, FINAL\_TARGET) == 0)

{

fclose(file);

return 1;

}

else if (strcmp(line, DEADLOCK\_TARGET) == 0)

{

break;

}

else

{

sscanf(line, "@include %s", readResult);

char \*fileName = findTargetFile(DEFAULT\_DIRECTORY, line);

if (checkTarget(fileName, result))

{

strcat(\*result, "./");

strcat(\*result, fileName);

strcat(\*result, "\n");

free(fileName);

fclose(file);

return 1;

}

free(fileName);

}

}

fclose(file);

return 0;

}

int main()

{

char \*resultString = malloc(sizeof(char) \* 1000);

char \*targetFile = findTargetFile(DEFAULT\_DIRECTORY, "file.txt");

checkTarget(targetFile, &resultString);

strcat(resultString, "./");

strcat(resultString, targetFile);

strcat(resultString, "\n");

int count = 0;

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

char \*token = strtok(resultString, "\n");

while (token != NULL)

{

resultTable[count++] = token;

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

token = strtok(NULL, "\n");

}

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

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

{

if (i == 0)

{

fprintf(resultFile, "%s", resultTable[i]);

}

else

{

fprintf(resultFile, "%s\n", resultTable[i]);

}

}

fclose(resultFile);

free(resultString);

free(resultTable);

free(targetFile);

return 0;

}