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

Студентка гр. 3342 _____ Епонишникова А.И

Преподаватель _____ Глазунов С.А

Санкт-Петербург
2024

Цель работы

Целью работы является на практике изучить работу с рекурсивными функциями, файловой системой и ее обходом.

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).

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

file_path(char *dir_name, char *filename, char *path) – в качестве аргументов подается название директории, файла и путь к нему. Сначала открываем директорию dir_name, далее считываем элементы из потока директории. Проверяем файл это или директория. Если директория, то теперь проверка на то что это ссылка на текущую директорию или ссылка на родительскую директорию, если они – continue. Создается next_dir, куда записывается путь до этой директории. Далее рекурсивно вызывается file_path. Если файл, то проверяем тот ли этой файл, который требуется. Создается curr_path, куда записывается путь до файла. Копируем в path curr_path и возвращается 1.

find_minotaur(char *dir_name, char *filename, char** answer, int ind) - в качестве аргументов название директории, файла, массив answer(куда записывается ответ) и индекс(в массиве). Создается path, str. Если функция file_path равна 0, возвращаем 0. Далее проверка на то, чтобы в answer были записаны пути до файлов. Открываем файл, с помощью fgets в str. Если в файле “Deadlock”, то в answer записывается NULL, файл закрывается и возвращается 0, если Minotaur, то закрывается файл и возвращается 1. Если в файле есть @include и заканчивается на \n, то нужно получить название файла, здесь вызываем функцию find_minotaur, для проверки данного файла.

В функции main() осуществляется вызов функции find_minotaur() и выводится ответ.

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

На практике научились работать с рекурсивными функциями, файловой системой и ее обходом. Была реализована программа, которая искала необходимую строчку среди файлов и записывала пути до них.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3.c

```
#include <stdio.h>
#include <string.h>
#include <dirent.h>

#define MAX_LEN 512

int file_path(char *dir_name, char *filename, char *path) {
    DIR *dir;
    if ((dir = opendir(dir_name)) == NULL)
        return 0;
    struct dirent *de;
    while ((de = readdir(dir)) != NULL) {
        if (de->d_type == DT_DIR) {
             if (strcmp(de->d_name, ".") == 0 || strcmp(de->d_name,
"..") == 0)
                continue;
            char next_dir[MAX_LEN];
                snprintf(next_dir, MAX_LEN, "%s/%s", dir_name, de-
>d_name);
            if (file_path(next_dir, filename, path)) {
                closedir(dir);
                return 1;
            }
        } else {
            if (strcmp(de->d_name, filename) == 0) {
                char curr_path[MAX_LEN];
                 snprintf(curr_path, MAX_LEN, "%s/%s", dir_name, de-
>d_name);
                strcpy(path, curr_path);
                return 1;
            }
        }
    }
    closedir(dir);
    return 0;
}

int find_minotaur(char *dir_name, char *filename, char** answer,
int ind) {
    FILE* file;
    char path[MAX_LEN], str[MAX_LEN];
    if (file_path(dir_name, filename, path) == 0)
        return 0;
    int flag = 1;
    for (int i = 0; i < ind; i++) {
        if (strcmp(answer[i], path) == 0) {
            flag = 0;
            break;
        }
    }
    if (flag)
        answer[ind++] = path;
```

```
        file = fopen(path, "r");
        if (file == NULL)
            return 0;
        while (fgets(str, sizeof(str), file)) {
            if (strcmp(str, "Deadlock") == 0) {
                answer[ind] = NULL;
                fclose(file);
                return 0;
            } else if (strcmp(str, "Minotaur") == 0) {
                fclose(file);
                return 1;
                  } else if (strncmp(str, "@include ", 9) == 0 &&
str[strlen(str) - 1] == '\n') {
                int i = 9;
                while (str[i] != '\n')
                    i++;
                str[i] = '\0';
                 if (find_minotaur(dir_name, &str[9], answer, ind) == 1)
{
                    fclose(file);
                    return 1;
                }
            }
        }
        fclose(file);
        return 0;
    }

    int main() {
        char* answer[MAX_LEN];
        int ind = 0;
        FILE *file;
        file = fopen("./result.txt", "w");
        if (find_minotaur("./labyrinth", "file.txt", answer, ind)) {
            for (int i = 0; answer[i]; i++) {
                fputs(answer[i], file);
                fputs("\n", file);
            }
        }
        fclose(file);
        return 0;
    }
```