

**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

Корниенко А.Е.

Преподаватель

Глазунов С.А.

Санкт-Петербург

2024

## **Цель работы**

Ознакомление с рекурсией, которая используется в нашей работе для обхода файловой системы с помощью C.

## Задание

### Вариант 2.

Задана иерархия папок и файлов по следующим правилам:

- название папок может быть только "add" или "mul"
- В папках могут находиться другие вложенные папки и/или текстовые файлы
- Текстовые файлы имеют произвольное имя с расширением .txt
- Содержимое текстовых файлов представляет собой строку, в которой через пробел записано некоторое количество целых чисел

Требуется написать программу, которая, запускается в корневой директории, содержащей одну папку с именем "add" или "mul" и вычисляет и выводит на экран результат выражения состоящего из чисел в поддиректориях по следующим правилам:

- Если в папке находится один или несколько текстовых файлов, то математическая операция определяемая названием папки (add = сложение, mul = умножение) применяется ко всем числам всех файлов в этой папке
- Если в папке находится еще одна или несколько папок, то сначала вычисляются значения выражений, определяемые ими, а после используются уже эти значения

## **Выполнение работы**

Для получения результата из файла используется функция `readFile(char* filename, char* name_command)`, которая принимает два аргумента: имя файла, команду, которую надо выполнить для всех чисел( сложить или умножить ).

Для обхода файловой системы используется функция `listDir(char* Dir, char* name_command)`, которая принимает два аргумента: имя директории, команду, которую нужно выполнить для всех поддиректорий. Далее при помощи цикла и рекурсии, мы проходим по файловой системе, если тип директории – файл, то используем функция `readFile`, иначе идём дальше вглубь рекурсии.

В конце записываем, полученный результат в файл `result.txt`.

Для открытия и закрытия файлов и директорий используем функции: `fopen`, `fclose`, `opendir`, `closedir`, и для чтения директории: `readdir`.

Разработанный программный код см. в приложении А.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
1.	<div><div><div>- root</div><div><div>add</div><div><div>add</div><div><div>file.txt</div><div>file1.txt</div></div><div><div>mul</div><div><div>file2.txt</div><div>file3.txt</div></div><div><div>add</div><div><div>file4.txt</div><div>file5.txt</div></div></div></div></div></div><div>file.txt: 1 file1.txt: 1 file2.txt: 2 2 file3.txt: 7 file4.txt: 1 2 3 file5.txt: 3 -1</div></div></div>	226

## **Выводы**

Разработана программа на языке программирования С с использованием библиотеки `dirent.h` для реализации обхода файловой системы при помощи рекурсии

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

#define IF_dir de->d_type == DT_DIR && strcmp(de->d_name, ".") &&
strcmp(de->d_name, "..")

int readFile(char* filename, char* name_command){
    int res = 0;

    if(strcmp(name_command, "mul") == 0)
        res = 1;

    FILE *f = fopen(filename, "r");

    if(!f)
        return 0;

    char s[100];
    while(fgets(s, 100, f)){

        char num[10];
        int size = 0;
        for(int i = 0; i < strlen(s); i++){
            if(isdigit(s[i]) || s[i] == '-'){
                num[size] = s[i];
                size++;
            }
            if(s[i] == ' ' || i == strlen(s) - 1){
                num[size] = '\0';
                if(strcmp(name_command, "add") == 0)
                    res += atoi(num);
                else if(strcmp(name_command, "mul") == 0)
                    res *= atoi(num);

                size = 0;
            }
        }
        fclose(f);

        return res;
    }
}

int listDir(char* Dir, char* name_command){
    int result = 0;
    if(name_command != NULL && strcmp(name_command, "mul") == 0)
```

```

        result = 1;
        char next[200] = {0};
        strcpy(next, Dir);
        DIR *dir = opendir(Dir);
        if(!dir)
            return 0;

        struct dirent *de = readdir(dir);

        while(de){

            if(IF_dir)
            {
                int len = strlen(next);
                strcat(next, "/");
                strcat(next, de->d_name);
                if(name_command != NULL && strcmp(name_command, "add")
== 0)

                    result += listDir(next, de->d_name);
                else if(name_command != NULL && strcmp(name_command,
"mul") == 0)

                    result *= listDir(next, de->d_name);
                else
                    result = listDir(next, de->d_name);
                next[len] = '\0';
            }
            if(de->d_type == DT_REG){
                int len = strlen(next);
                strcat(next, "/");
                strcat(next, de->d_name);
                if(name_command != NULL && strcmp(name_command, "add")
== 0)

                    result += readFile(next, name_command);
                else if(name_command != NULL && strcmp(name_command,
"mul") == 0)

                    result *= readFile(next, name_command);
                next[len] = '\0';
            }
            de = readdir(dir);
        }
        closedir(dir);
        return result;
    }

    int main(){

        int answer = listDir("tmp", NULL);

        FILE *f = fopen("result.txt", "w");
        fprintf(f, "%d", answer);
        fclose(f);

        return 0;
    }

```