

**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. 1304

Чернякова В.А.

Преподаватель

Чайка К.В.

Санкт-Петербург

2022

## **Цель работы.**

Освоение работы с основными функциями для работы с деревом файловой системы, объявления которых находятся в заголовочном файле *dirent.h*.

## **Задание.**

### **Вариант 2**

Задана иерархия папок и файлов по следующим правилам:

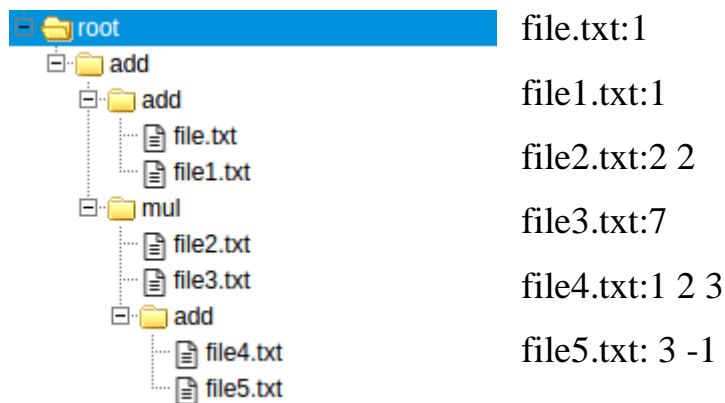
- название папок может быть только "add" или "mul"
- В папках могут находиться другие вложенные папки и/или текстовые файлы
- Текстовые файлы имеют произвольное имя с расширением .txt
- Содержимое текстовых файлов представляет собой строку, в которой через пробел записано некоторое количество целых чисел

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

## **Пример**

(Программа в момент запуска находится в директории root)



### Решение:

226

Выражение в данном случае имеет вид:  $((1+1)) + ((1+2+3+3+1)*7*2*2))$

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 tmp.

### Выполнение работы.

#### Главная функция `int main()`.

В основной функции программы открывается поток записи с помощью функции стандартной библиотеки языка программирования C `fprintf()` в файл `result.txt`. В данный текстовый файл будет записан результат работы функции `add()` над папкой `tmp`.

### Функции.

#### Функция `Long long add(char* curPath)`.

Данная рекурсивная функция, проходится по всем файлам и директориям текущей папки. Дескриптор `readdir()` выполняет роль стека. Если в текущей папке найдены дополнительные папки, то в зависимости от названия `add` или `mul` вызывается функция `add()` или `mul()` соответственно. Если найден файл, то с помощью функции `fscanf()`, числа этого файла добавляются в переменную `res` операцией сложения.

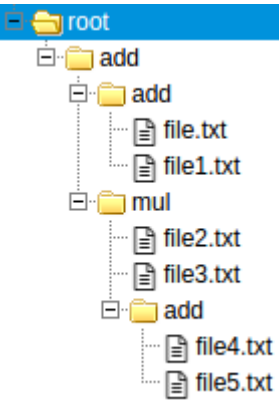
#### Функция `Long long mul(char* curPath)`.

Данная рекурсивная функция работает аналогично функции *add()*, описанной выше, только действия сложения для возвращаемой переменной *res*, заменены на умножение.

### Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.		226	Программа работает корректно.

### Выводы.

В ходе лабораторной работы была освоена работа с функциями, позволяющими совершать обход деревьев файловой системы. Улучшены навыки работы с рекурсией. Написана программа, которая осуществляет работу с файлами и директориями. Реализованы математические операции с данными, содержащимися в текстовых файлах, в зависимости от названия папки.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Chernyakova\_Valeria\_lb3/main.c

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

long long mul(char *actualPath);

long long add(char *actualPath){
    FILE *f;
    DIR *d = opendir(actualPath);
    long long result = 0, n;
    char c;
    if (d) {
        struct dirent *der = readdir(d);
        while (der) {
            if (der->d_name[0] == '.') {
                der = readdir(d);
                continue;
            }
            if (der->d_type == DT_DIR && strcmp(der->d_name,
"add") == 0) {
                strcat(actualPath, "/add");
                result += add(actualPath);
                actualPath[strlen(actualPath) - 4] = '\\0';
            }
            else if (der->d_type == DT_DIR && strcmp(der->d_name,
"mul") == 0) {
                strcat(actualPath, "/mul");
                result += mul(actualPath);
                actualPath[strlen(actualPath) - 4] = '\\0';
            }
            else {
                strcat(actualPath, "/");
                strcat(actualPath, der->d_name);
                f = fopen(actualPath, "r");
                fscanf(f, "%lld", &n);
                c = fgetc(f);
                result += n;
                while (c != EOF && c != '\\n' && fscanf(f, "%lld",
&n) != EOF) {
                    c = fgetc(f);
                    result += n;
                }
                fclose(f);
                actualPath[strlen(actualPath) - strlen(der-
>d_name) - 1] = '\\0';
            }
        }
    }
}
```

```

        }
        der = readdir(d);
    }
}
closedir(d);
return result;
}

long long mul(char *actualPath) {
    FILE *f;
    DIR *d = opendir(actualPath);
    long long result = 0, n, flag = 0;
    char c, cl;
    if (d) {
        struct dirent *der = readdir(d);
        while (der) {
            if (der->d_name[0] == '.') {
                der = readdir(d);
                continue;
            }
            if (der->d_type == DT_DIR && strcmp(der->d_name,
"add") == 0) {
                strcat(actualPath, "/add");
                result *= add(actualPath);
                actualPath[strlen(actualPath) - 4] = '\\0';
            }
            else if (der->d_type == DT_DIR && strcmp(der->d_name,
"mul") == 0) {
                strcat(actualPath, "/mul");
                result *= mul(actualPath);
                actualPath[strlen(actualPath) - 4] = '\\0';
            }
            else {
                if (flag == 0) {
                    flag = 1;
                    result = 1;
                }
                strcat(actualPath, "/");
                strcat(actualPath, der->d_name);
                f = fopen(actualPath, "r");
                fscanf(f, "%lld", &n);
                c = fgetc(f);
                result *= n;
                while (c != EOF && c != '\\n' && fscanf(f, "%lld",
&n) != EOF) {
                    c = fgetc(f);
                    result *= n;
                }
                fclose(f);
            }
        }
    }
}

```

```

        actualPath[strlen(actualPath) - strlen(der-
>d_name) - 1] = '\\0';
    }
    der = readdir(d);
}
}
closedir(d);
return result;
}

int main() {
    char str[10000];
    strcpy(str, ".");
    strcat(str, "/result.txt");
    FILE *fresult = fopen(str, "w");
    str[strlen(str) - 11] = '\\0';
    strcat(str, "/tmp");
    fprintf(fresult, "%lld", add(str));
    return 0;
}

```