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 _____ Львов А.В.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Ознакомление с рекурсией, её применение для обхода файловой системы с помощью языка 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

В начале работы программа вызывает функцию listdir (void listdir(char * name, long long int * result, int * call)), которая принимает на вход path – путь к директории, result – переменная, в которую необходимо сохранить результат и call – аргумент, отвечающий за то, дошла ли функция до папки, в которой нет подпапок. Функция рекурсивно обходит файловое дерево, пока не встретит описанную выше папку. Затем переменной присваивается начальное значение в зависимости от названия папки, в которой функция находится в данный момент. После этого, функция перебирает все файлы и выполняет с ними действия, требуемые в задании.

Функция getoperation (int getoperation(char * path)) получает на вход путь к директории и с помощью strtok находит последнюю подстроку, полученную делением исходной строки символом «/» и возвращает соответствующий результат – 0, если операция “add”, 1, если операция “mul” и 2 в ином случае. Для удобства было создано перечисление operation.

Фукнция getresult (int getresult(array * nums, int operation)) принимает на вход переменную типа array * (array имеет поля arr – целочисленный массив и size – его размер) и операцию, которую требуется произвести расчеты. Функция возвращает результат применения операции ко всем числам nums.

Функция getnumsfile (array * getnumsfile(char * pathToFile)) получает на вход путь к файлу и с помощью fscanf считывает числа из файла в структуру array, указатель на которую и возвращает.

Разработанный программный код см. в приложении А.

Выводы

Было проведено ознакомление с рекурсией. Разработана программа на языке C с использованием библиотеки dirent.h для реализации обхода файловой системы.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

#define MAX_SIZE 1024

enum operation {
    ADD,
    MUL,
    UNDEFINED
};

typedef struct array {
    int * arr;
    int size;
} array;

void listdir(char * name, long long int * result, int * call);
array * getnumsfile(char * pathtofile);
int getoperation(char * path);
int getresult(array * nums, int operation);

int main() {
    long long int res;
    int call = 0;

    listdir("./tmp", &res, &call);
    FILE * file = fopen("./result.txt", "w");
    if (file == NULL) {
        printf("Cannot open the file!");
        exit(1);
    }
    fprintf(file, "%lld", res);
    fclose(file);
    return 0;
}

void listdir(char *name, long long int * result, int * call) {
    DIR *dir;
    struct dirent *entry;
    char path[MAX_SIZE];

    if (!(dir = opendir(name)))
        return;

    while ((entry = readdir(dir)) != NULL) {
        if (entry->d_type == DT_DIR) {
                    if   (strcmp(entry->d_name,   ".")   ==   0   ||
strcmp(entry->d_name, "..") == 0)
```

```
                    continue;
                snprintf(path, sizeof(path), "%s/%s", name, entry->d_name);
                listdir(path, result, call);
            }
        }

        char strforcopy[sizeof(name)]; // strtok изменяет исходную строку
        strcpy(strforcopy, name);
        enum operation op = getoperation(strforcopy);

        if (*call == 0) { // если мы дошли до папки, в которой нет
подпапок
            if (op == ADD) {
                *result = 0;
            } else if (op == MUL){
                *result = 1;
            } else {
                return;
            }
            *call = 1;
        }

        if (op == UNDEFINED){
            return;
        }

        rewinddir(dir);

        while ((entry = readdir(dir)) != NULL) {
            if (entry->d_type == DT_REG) {
                char pathtofile[MAX_SIZE];
                    snprintf(pathtofile, sizeof(path), "%s/%s", name,
entry->d_name);
                array * nums = getnumsfile(pathtofile);
                int tmpres = getresult(nums, op);
                if (op == ADD) {
                    *result += tmpres;
                } else {
                    *result *= tmpres;
                }
            }
        }
        closedir(dir);
    }

    int getoperation(char * path) {
        enum operation op = UNDEFINED;
        char * tmp = strtok(path, "/");
        while (tmp != NULL) {
            if (strcmp(tmp, "add") == 0) {
                op = ADD;
            } else if (strcmp(tmp, "mul") == 0){
                op = MUL;
            }
            tmp = strtok(NULL, "/");
        }
        return op;
    }
```

```
int getresult(array * nums, int operation) {
    int tmp;
    if (operation == ADD) {
        tmp = 0;
        for (int i = 0; i < nums->size; i++) {
            tmp += nums->arr[i];
        }
    } else {
        tmp = 1;
        for (int i = 0; i < nums->size; i++) {
            tmp *= nums->arr[i];
        }
    }
    return tmp;
}

array * getnumsfile(char * pathtofile) {
    array * nums = (array *)calloc(1, sizeof(array));
    if (nums == NULL) {
        printf("Cannot allocate memory!");
        exit(1);
    }

    int arr[MAX_SIZE];
    int size = 0;
    FILE * file = fopen(pathtofile, "r");
    if (file == NULL) {
        printf("Cannot open file!");
        return NULL;
    }

    while (fscanf(file, "%d", &arr[size++]) == 1) {}

    nums -> arr = arr;
    nums -> size = size - 1;
    fclose(file);
    return nums;
}
```