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 _____ Иванов Д. М.

Преподаватель _____ Глазунов С. А.

Санкт-Петербург
2024

Цель работы

Изучить рекурсию, чтение файлов и обход файловой системы на языке Си. Применить эти знания для решения поставленной задачи.

Задание

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 упрощения работы код был разбит на несколько функций, каждая из которых выполняет определенную задачу:

1) long long list_dir(const char *dirPath, long long result) — Получает на вход путь к определенной файловой системе. Она открывается и циклом программа читает каждый файл. Если он оказывается директорией, то запускается рекурсия и предеается в виде аргумента новый путь. Если же это текстовый файл, то происходит чтение данных из него и сохранения суммы или произведения чисел внутри него. То же самое происходит в следующих итерациях с другими файлами.

2) void memory_error() - Вывод ошибки в случае неверного выделения памяти

3) long long sum(char* str) — Вывод значения суммы чисел из текстового файла

4) long long pr(char* str) — Аналогично для произведения

5) int main() - Начальное открытие директории tmp начальный вызов list_dir. Также открытие файла result.txt и запись туда ответа.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарий
1.	tmp: add: file.txt (1) file1.txt (1) mul: file2.txt (2 2) file3.txt (7) add: file4.txt (1 2 3) file5.txt (3 -1)	226	Верный вывод

Выводы

Была разработана программа, читающая все файлы из определенной директории и находящая нужные значения из них. Изучены файловые системы и рекурсия на языке Си.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <dirent.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>

void memory_error(){
    fprintf(stderr, "Error with memory allocation!");
    exit(1);
}

long long sum(char* str){
    long long res = 0;
    char* istr = strtok(str, " ");
    while (istr != NULL)
    {
        res += atoi(istr);
        istr = strtok (NULL, " ");
    }
    return res;
}

long long pr(char* str){
    long long res = 1;
    char* istr = strtok(str, " ");
    while (istr != NULL)
    {
        res *= atoi(istr);
        istr = strtok (NULL, " ");
    }
    return res;
}

long long list_dir(const char *dirPath, long long result)
{
    DIR *dir = opendir(dirPath);
    if(dir) {
        struct  dirent *de = readdir(dir);
        while (de) {
            char* new_dir = calloc(strlen(dirPath) + 4, sizeof(char));
            char* file_dir;
            char* line = calloc(100, sizeof(char));
            if (new_dir == NULL || line == NULL){
                memory_error();
            }
            FILE *file;
            if (de->d_type == 4 && strstr(de->d_name, "add") != NULL){
                sprintf(new_dir, "%s/%s", dirPath, de->d_name);
                if (dirPath[strlen(dirPath) - 1] == 'd'){
                    result += list_dir(new_dir, 0);
                }
                if (dirPath[strlen(dirPath) - 1] == 'l'){
```

```
                    result *= list_dir(new_dir, 0);
                }
            }
            if (de->d_type == 4 && strstr(de->d_name, "mul") != NULL){
                sprintf(new_dir, "%s/%s", dirPath, de->d_name);
                if (dirPath[strlen(dirPath) - 1] == 'd'){
                    result += list_dir(new_dir, 1);
                }
                if (dirPath[strlen(dirPath) - 1] == 'l'){
                    result *= list_dir(new_dir, 1);
                }
            }
            if (de->d_type == 8){
                file_dir = calloc(strlen(dirPath) + strlen(de->d_name)
+ 3, sizeof(char));
                if (file_dir == NULL){
                    memory_error();
                }
                sprintf(file_dir, "%s/%s", dirPath, de->d_name);
                file = fopen(file_dir, "r");
                fgets(line, 100, file);
                if (dirPath[strlen(dirPath) - 1] == 'd'){
                    result += sum(line);
                }
                if (dirPath[strlen(dirPath) - 1] == 'l'){
                    result *= pr(line);
                }
                fclose(file);
                free(file_dir);
            }
            de = readdir(dir);
            free(line);
            free(new_dir);
        }
    }
    closedir(dir);
    return result;
}

int main(){
    DIR *dir = opendir("./tmp");
    struct  dirent *de = readdir(dir);
    long long result = 0;
    while (de){
        if (strstr(de->d_name, "add") != NULL){
            result = list_dir("./tmp/add", 0);
            break;
        }
        if (strstr(de->d_name, "mul") != NULL){
            result = list_dir("./tmp/mul", 1);
            break;
        }
        de = readdir(dir);
    }
    char * filename = "result.txt";
    FILE *fp = fopen(filename, "w");
    if(fp)
    {
```

```
            char result_str[128];
            sprintf(result_str, "%lld", result);
            fputs(result_str, fp);
            fclose(fp);
        }
        closedir(dir);
        return 0;
    }
```