**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 |
|  | 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;

}