**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: Строки. Рекурсия, циклы, обход дерева

|  |  |  |
| --- | --- | --- |
| Студент гр. 3341 |  | Самокрутов А.Р. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Целью работы является освоение работы с рекурсивными функциями и

файловой системой, а также ее рекурсивным обходом.

Для достижения поставленной цели требуется решить следующие задачи:

1. Ознакомиться с понятием рекурсии;
2. Освоить написание рекурсивных функций в языке Си;
3. Изучить работу с файловой системой в языке Си;
4. Написать программу для рекурсивного обхода всех файлов в папке в том числе во вложенных папках.

## 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)

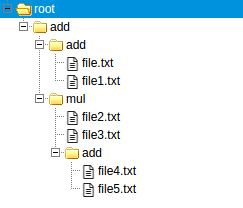


Рисунок 1 – Пример

file.txt: 1

file1.txt: 1

file2.txt: 2 2

file3.txt: 7

file4.txt: 1 2 3

file5.txt: 3 -1

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

Сначала с помощью директивы include подключаются библиотеки *sys/types.h*, *dirent.h*, *stdio.h*, *stdlib.h*, *string.h*.

Далее описаны несколько функций для работы с файлами.

Функция *char \*pathcat(const char \*path1, const char \*path2)* принимает на вход две строки — путь к директории и имя файла в этой директории и возвращает динамически выделенную строку, в которой хранится адрес файла, образованное по принципу *<путь к директории>/<имя файла>*.

Функция *long long int add(const char \*dir\_name)* принимает имя директории и возвращает сумму всех элементов в ней по правилу, описанному в задании. Объявляется целое число *sum* — сумма, инициализированная нулём. Далее с помощью функции *opendir()* по переданному имени открывается директория, после чего с помощью цикла *while()* читается её содержимое. С помощью функций *is\_child\_dir()* и *is\_txt\_file()* определяется, является ли элемент в директории дочерней директорией или текстовым файлом, после чего вызывается соответственно функция *add\_dir()* или *add\_file()*, которые считают сумму своего содержимого и записывают её в переменную *sum*. Функция *long long int mul(const char \*dir\_name)* аналогична и отличается лишь тем, что считает не сумму, а произведение с помощью функции *mul\_dir()* или *mul\_file()*.

Функция *int is\_child\_dir(const struct dirent \*de)* проверяет, что элемент *de* является директорией (*de->d\_type == DT\_DIR*) и при этом не является родительской или текущей директорией. Возвращает *0* в случае невыполнения условий, не *0* иначе. Функция *int is\_txt\_file(const struct dirent \*de)* проверяет, что элемент является файлом (*de->d\_type == DT\_REG*), оканчивающимся на *“.txt”* (вообще просто содержащим это сочетание символов в названии, но иметь его не в конце файл не может). Аналогично возвращает *0*, если условия соблюдены, и не *0* в любом другом случае.

Функция *void assign\_dir(const char \*path, const struct dirent \*de, long long int \*res)* в зависимости от названия директории вызывает функцию *add()* или *mul()* и присваивает её значение содержимому указателя res.

Функция *void add\_file(const char \*dir\_name, const struct dirent \*de, long long int \*sum)* открывает файл и с помощью функции *fscanf()* считывает числа в нём, пока они не кончатся. Эти числа суммируются к содержимому указателя *sum*. Функция *void mul\_file(const char \*dir\_name, const struct dirent \*de, long long int \*prod)* аналогичным образом находит произведение всех чисел, записанных в файле.

Функция *void add\_dir(const char \*path, const struct dirent \*de, long long int \*sum)* вычисляет сумму или произведение содержимого директории с именем *dir\_name* с помощью функций *add()* и *mul()* и добавляет это в содержимому *sum*. Аналогично произведение всех элементов считает функция *mul\_dir()*.

Функция *void process\_dir(const char \*dir\_name, long long int \*res)* открывает директорию, проверяет, что она является дочерней, и вызывает функцию *assign\_dir()*.

Функции *void dir\_error\_check(DIR \*dir) и void file\_error\_check(FILE \*file)* служат для проверки успешного открытия соответственно директории и файла.

Функция *void write\_ans(const long long int res)* записывает число *res* в файл *result.txt*.

В функции *main()* инициализируется 64-битное целое число *res* со значением *0*. Вызываются функции *process\_dir()* и *write\_ans()*.

Разработанный программный код см. в **приложении А**.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | add  file.txt – 1  file1.txt – 2 | result.txt – 3 | Проверка программы на работоспособность. |
|  | add  file.txt –  file1.txt – 1 | result.txt – 1 | Проверка корректной работы программы, если какие-либо файлы пустые. |
|  | add  mul  file.txt – 3 2  add | result.txt – 6 | Проверка корректной работы программы, если какая-либо директория пуста. |
|  | mul  mul  mul  mul  file.txt – 1 2 3 4 | result.txt – 24 | Проверка корректной работы программы с вложенными директориями. |

## Выводы

В ходе выполнения были изучены принципы рекурсии и рекурсивных функций, обхода файловой системы типа POSIX на языке программирования Си. Были приобретены навыки использования рекурсивных методов для обхода файловых систем, анализа содержимого текстовых файлов и директорий.

Была разработана программа, обрабатывающая рекурсивными методами информацию из различных файлов и директорий.

# Приложение А Исходный код программы

Название файла: main.c

#include <sys/types.h>

#include <dirent.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

char \*pathcat(const char \*path1, const char \*path2);

long long int add(const char \*);

long long int mul(const char \*);

int is\_child\_dir(const struct dirent \*);

int is\_txt\_file(const struct dirent \*);

void assign\_dir(const char \*, const struct dirent \*, long long int \*);

void add\_file(const char \*, const struct dirent \*, long long int \*);

void mul\_file(const char \*, const struct dirent \*, long long int \*);

void add\_dir(const char \*, const struct dirent \*, long long int \*);

void mul\_dir(const char \*, const struct dirent \*, long long int \*);

void process\_dir(const char \*, long long int \*);

void dir\_error\_check(DIR \*);

void file\_error\_check(FILE \*);

void write\_ans(const long long int);

long long int main() {

long long int res = 0;

process\_dir("./tmp", &res);

write\_ans(res);

return 0;

}

char \*pathcat(const char \*path1, const char \*path2) {

long long int res\_path\_len = strlen(path1) + strlen(path2) + 2;

char \*res\_path = (char \*)malloc(res\_path\_len \* sizeof(char));

sprintf(res\_path, "%s/%s", path1, path2);

return res\_path;

}

long long int add(const char \*dir\_name) {

long long int sum = 0;

DIR \*dir = opendir(dir\_name);

dir\_error\_check(dir);

struct dirent \*de = readdir(dir);

while ((de) != NULL) {

if (is\_child\_dir(de)) {

add\_dir(dir\_name, de, &sum);

} else if (is\_txt\_file(de)) {

add\_file(dir\_name, de, &sum);

}

de = readdir(dir);

}

closedir(dir);

return sum;

}

long long int mul(const char \*dir\_name) {

long long int prod = 1;

DIR \*dir = opendir(dir\_name);

dir\_error\_check(dir);

struct dirent \*de = readdir(dir);

while ((de) != NULL) {

if (is\_child\_dir(de)) {

mul\_dir(dir\_name, de, &prod);

} else if (is\_txt\_file(de)) {

mul\_file(dir\_name, de, &prod);

}

de = readdir(dir);

}

closedir(dir);

return prod;

}

int is\_child\_dir(const struct dirent \*de) {

if (de->d\_type == DT\_DIR)

return (strcmp(de->d\_name, ".") != 0 && strcmp(de->d\_name, ".."));

else

return 0;

}

int is\_txt\_file(const struct dirent \*de) {

if (de->d\_type == DT\_REG)

return (strstr(de->d\_name, ".txt") != NULL);

else

return 0;

}

void assign\_dir(const char \*path, const struct dirent \*de, long long int \*res) {

char \*dir\_name = pathcat(path, de->d\_name);

if (strcmp(de->d\_name, "add") == 0) {

\*res = add(dir\_name);

} else if (strcmp(de->d\_name, "mul") == 0) {

\*res = mul(dir\_name);

}

free(dir\_name);

}

void add\_file(const char \*dir\_name, const struct dirent \*de, long long int \*sum) {

long long int contents;

char \*file\_name = pathcat(dir\_name, de->d\_name);

FILE \*file = fopen(file\_name, "r");

file\_error\_check(file);

while (fscanf(file, "%lli", &contents) == 1) {

\*sum += contents;

}

fclose(file);

}

void mul\_file(const char \*dir\_name, const struct dirent \*de, long long int \*prod) {

long long int contents;

char \*file\_name = pathcat(dir\_name, de->d\_name);

FILE \*file = fopen(file\_name, "r");

file\_error\_check(file);

while (fscanf(file, "%lli", &contents) == 1) {

\*prod \*= contents;

}

fclose(file);

}

void add\_dir(const char \*path, const struct dirent \*de, long long int \*sum) {

char \*dir\_name = pathcat(path, de->d\_name);

if (strcmp(de->d\_name, "add") == 0) {

\*sum += add(dir\_name);

} else if (strcmp(de->d\_name, "mul") == 0) {

\*sum += mul(dir\_name);

}

free(dir\_name);

}

void mul\_dir(const char \*path, const struct dirent \*de, long long int \*prod) {

char \*dir\_name = pathcat(path, de->d\_name);

if (strcmp(de->d\_name, "add") == 0) {

\*prod \*= add(dir\_name);

} else if (strcmp(de->d\_name, "mul") == 0) {

\*prod \*= mul(dir\_name);

}

free(dir\_name);

}

void process\_dir(const char \*dir\_name, long long int \*res) {

DIR \*dir = opendir(dir\_name);

dir\_error\_check(dir);

struct dirent \*de = readdir(dir);

while ((de) != NULL) {

if (is\_child\_dir(de)) {

assign\_dir(dir\_name, de, res);

}

de = readdir(dir);

}

closedir(dir);

}

void dir\_error\_check(DIR \*dir) {

if (dir == NULL) {

puts("Error accessing a directory...\n");

exit(1);

}

}

void file\_error\_check(FILE \*file) {

if (file == NULL) {

puts("Error opening a file...\n");

exit(1);

}

}

void write\_ans(const long long int res) {

FILE \*file = fopen("result.txt", "w");

file\_error\_check(file);

fprintf(file, "%lli", res);

fclose(file);

}