**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: Рекурсия, циклы, рекурсивный обход файловой системы

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3341 |  | Яковлева А.А. |
| 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 4.

Дана некоторая корневая директория, в которой может находиться некоторое количество папок, в том числе вложенных. В этих папках хранятся некоторые текстовые файлы, имеющие имя вида *<filename>.txt*. В качестве имени файла используется символ латинского алфавита.

На вход программе подается строка. Требуется найти и вывести последовательность полных путей файлов, имена которых образуют эту строку.

! Регистрозависимость

! Могут встречаться файлы, в имени которых есть несколько букв и эти файлы использовать нельзя.

! Одна буква может встречаться один раз.

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

Макросы:

* *START\_DIR -* название директории, которую обрабатывает программа
* *NAME\_RESULT\_FILE -* имя файла, в который будет записан результат работы программы
* *FILE\_NAME\_TEMPLATE -* шаблон имени файла " .txt", т.к. требуется найти файлы данного вида
* *END\_LINE -* символ переноса строки
* *MAX\_STR\_LENGTH -* максимальная длина считанной строки
* *MAX\_FILE\_NAME\_LENGTH -* максимальная длина имени файла (длина всегда равна 6, т.к. ищем файлы вида " .txt")
* *CURRENT\_DIR -* ссылка на текущую директорию "."
* *PARENT\_DIR -* ссылка на родительскую директорию ".."

Функции:

* *char \*path\_cat(const char \*path1, const char \*path2)* принимает два указателя на две части пути, выделяет память размера длин двух строк + 2(для символов '/' и '\0'), функцией *sprintf* создаёт строку, состоящую из первого пути, символа '/' и второго пути, возвращает указатель на полученную строку.
* *int is\_not\_curr\_or\_parent\_dir(const char \*dir\_name)* принимает название директории. Если это не ссылка на текущую или родительскую директорию возвращает 1, иначе возвращает 0.
* *void search\_full\_path\_file(const char \*dir\_name, char\* name\_required\_file, FILE \*result\_file)* принимает указатель на название директории, название искомого файла и на файл с результатом работы программы. Сначала с помощью функции *opendir* открывает директорию *dir\_name,* циклом *while* проходит по всем элементам из потока директории. Если тип файла данного элемента *DT\_REG,* т.е. это обычный файл, и имя файла совпадает с искомым, то то в *result\_file* функцией *fputs* записывается полный путь к файлу, полученный функцией *path\_cat,* и *END\_LINE.* Иначе если тип файла данного элемента *DT\_DIR,* т.е. это директория, и это не ссылка на текущую или родительскую директорию, то функцей *path\_cat* конкатенируются имена родительской и вложенной директории, далее рекурсивного вызывается функция для обработки вложенной директории. После прохода по всем элементам директории её поток закрывается функцией *closedir.*
* *main.* Сначала создаётся строка *name\_required\_file,* в которую функцией *strcpy* копируется шаблон искомого файла *FILE\_NAME\_TEMPLATE.* Функцией *fopen* открывается файл *NAME\_RESULT\_FILE* на запись. Далее создаётся и считывается строка *str\_containing\_names\_of\_files.* Затем циклом *for* проходим по всем символом считанной строки, для получения имени искомого файла первый символ *name\_required\_file* заменяем текущим символом и вызываем функцию *search\_full\_path\_file.* После цикла закрываем файловый поток функцией *fclose.*

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 |
|  | HeLlO | tmp/asdfgh/mkoipu/H.txt tmp/qwerty/e.txt tmp/qwerty/qwert/L.txt tmp/asdfgh/l.txt tmp/asdfgh/O.txt | Выводятся полные пути файлов H.txt, e.txt, L.txt, l.txt, O.txt. |
|  | 123 | tmp/1.txt  tmp/third/fourth/2.txt  tmp/first/sec/3.txt | Выводятся полные пути файлов 1.txt, 2.txt, 3.txt. |

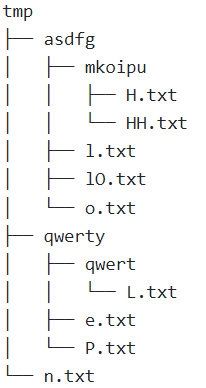


Рисунок 1 – расположение файлов в 1 тесте

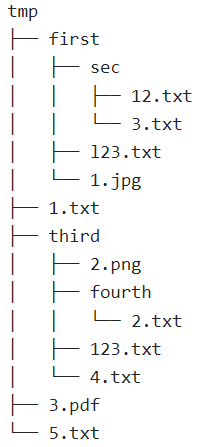


Рисунок 2 – расположение файлов во 2 тесте

## Выводы

В ходе выполнения работы были решены следующие задачи:

* изучены рекурсивные функции и их написание в языке Си;
* изучена работа с файловой системой в языке Си;
* написана программа для рекурсивного обхода всех файлов в папке, в том числе во вложенных папках, и поиска нужного файла.

# Приложение А Исходный код программы

Название файла: solution.c

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <dirent.h>

#define START\_DIR "./tmp"

#define NAME\_RESULT\_FILE "result.txt"

#define FILE\_NAME\_TEMPLATE " .txt"

#define END\_LINE "\n"

#define MAX\_STR\_LENGTH 100

#define MAX\_FILE\_NAME\_LENGTH 6

#define CURRENT\_DIR "."

#define PARENT\_DIR ".."

char \*path\_cat(const char \*path1, const char \*path2)

{

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;

}

int is\_not\_curr\_or\_parent\_dir(const char \*dir\_name)

{

if(strcmp(dir\_name, CURRENT\_DIR) && strcmp(dir\_name, PARENT\_DIR)) return 1;

return 0;

}

void search\_full\_path\_file(const char \*dir\_name, char\* name\_required\_file, FILE \*result\_file)

{

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

struct dirent \*de = readdir(dir);

while (de)

{

if(de->d\_type == DT\_REG && !strcmp(de->d\_name, name\_required\_file))

{

fputs(path\_cat(dir\_name, de->d\_name), result\_file);

fputs(END\_LINE, result\_file);

}

else if (de->d\_type == DT\_DIR && is\_not\_curr\_or\_parent\_dir(de->d\_name))

{

char \*new\_dir = path\_cat(dir\_name, de->d\_name);

search\_full\_path\_file(new\_dir, name\_required\_file, result\_file);

free(new\_dir);

}

de = readdir(dir);

}

closedir(dir);

}

int main()

{

char\* name\_required\_file = strcpy(name\_required\_file, FILE\_NAME\_TEMPLATE);

FILE \*result\_file = fopen(NAME\_RESULT\_FILE, "w");

char str\_containing\_names\_of\_files[MAX\_STR\_LENGTH];

fgets(str\_containing\_names\_of\_files, MAX\_STR\_LENGTH, stdin);

for(int i = 0; i < strlen(str\_containing\_names\_of\_files); i++)

{

name\_required\_file[0] = str\_containing\_names\_of\_files[i];

search\_full\_path\_file(START\_DIR, name\_required\_file, result\_file);

}

free(name\_required\_file);

fclose(result\_file);

return 0;

}