

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 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 функцией *path_cat*

конкатенируются имена родительской и вложенной директории, далее рекурсивно 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 символам 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
1.	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.
2.	123	tmp/1.txt tmp/third/fourth/2.txt tmp/first/sec/3.txt	Выводятся полные пути файлов 1.txt, 2.txt, 3.txt.

```
tmp
├── asdfg
│   ├── mkoipu
│   │   ├── H.txt
│   │   └── HH.txt
│   ├── 1.txt
│   ├── 10.txt
│   └── o.txt
├── qwerty
│   ├── qwert
│   │   └── L.txt
│   ├── e.txt
│   └── P.txt
└── n.txt
```

Рисунок 1 – расположение файлов в 1 тесте

```
tmp
├─ first
│   └─ sec
│       ├── 12.txt
│       └─ 3.txt
│   ├── 123.txt
│   └─ 1.jpg
├─ 1.txt
├─ third
│   ├── 2.png
│   ├── fourth
│   │   └─ 2.txt
│   ├── 123.txt
│   └─ 4.txt
├─ 3.pdf
└─ 5.txt
```

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;
}

```