

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. 3343

Пухов А.Д.

Преподаватель

Государкин Я.С.

Санкт-Петербург

2024

Цель работы.

Изучение работы с файлами и директориями. Написание программы на языке Си для рекурсивного поиска нужных файлов среди директорий.

Задание.

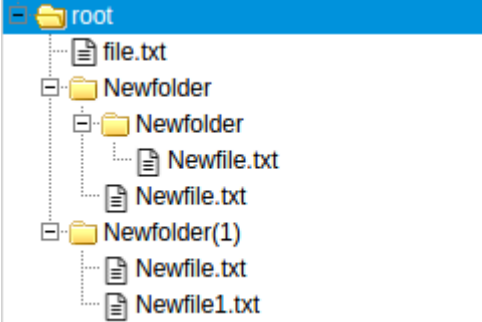
Дана некоторая корневая директория, в которой может находиться некоторое количество папок, в том числе вложенных. В этих папках хранятся некоторые текстовые файлы, имеющие имя вида *<filename>.txt*

В каждом текстовом файле хранится одна строка, начинающаяся с числа вида:

<число><пробел><латинские буквы, цифры, знаки препинания> ("124 string example!")

Требуется написать программу, которая, будучи запущенной в корневой директории, выведет строки из файлов всех поддиректорий в порядке возрастания числа, с которого строки начинаются

Пример

	<pre>root/file.txt: 4 Where am I? root/Newfolder/Newfile.txt: 2 Simple text root/Newfolder/Newfolder/Newfile.txt: 5 So much files! root/Newfolder(1)/Newfile.txt: 3 Wow? Text? root/Newfolder(1)/Newfile1.txt: 1 Small text</pre>
--	---

Решение:

- 1 Small text
- 2 Simple text
- 3 Wow? Text?
- 4 Where am I?
- 5 So much files!

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`.

Выполнение работы.

Структура **Text**:

- **char text[200]** — хранит строку
- **int number** — хранит число с которого начинается строка

Функции:

- **Text *readfile(const char *file_name)** — открывает и считывает содержимое файла.
- **Text **list_dir(const char *dir_name, int *size, Text **strings)** — рекурсивно обходит директории и ищет файлы в нужном формате(txt).
- **char *pathcat(const char *path1, const char *path2)** — склеивает две части пути к директории или к файлу, и возвращает полный путь.
- **write(const char *name_file, int size, Text **strings)** — записывает результат программ в файл.
- **int compar(const void *a, const void *b)** — компаратор для qsort.

Написанный программный код находится в приложении А

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	root/file.txt: 4 Where am I? root/Newfolder/Newfile.txt: 2 Simple text root/Newfolder/Newfolder/Newfile .txt: 5 So much files! root/Newfolder(1)/Newfile.txt: 3 Wow? Text? root/Newfolder(1)/Newfile1.txt: 1 Small text	1 Small text 2 Simple text 3 Wow? Text? 4 Where am I? 5 So much files!	OK

Выводы.

В данной лабораторной работе была изучена работа с файлами и с директориями спомощи библиотеки `dirent.h` в языке Си.

ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb3.c

```
#include <stdio.h>
#include <dirent.h>
#include <stdlib.h>
#include <string.h>

typedef struct Text
{
    char text[200];
    int number;
} Text;

char *pathcat(const char *path1, const char *path2)
{
    int res_path_len = strlen(path1) + strlen(path2) + 2;
    char *res_path = malloc(res_path_len * sizeof(char));
    sprintf(res_path, "%s/%s", path1, path2);
    return res_path;
}

Text *readfile(const char *file_name)
{
    char str[200];
    FILE *new_file = fopen(file_name, "r");
    Text *file = NULL;
    if (new_file)
    {
        file = (Text *)malloc(sizeof(Text));
        fscanf(new_file, "%d", &(file->number));
        fgets(str, 200, new_file);
        strncpy(file->text, str, 200);
        fclose(new_file);
    }
    return file;
}

Text **list_dir(const char *dir_name, int *size, Text **strings)
{
    DIR *dir = opendir(dir_name);
    if (dir)
    {
        struct dirent *de = readdir(dir);
```

```

while (de)
{
    if (de->d_type == DT_REG && strstr(de->d_name, ".txt") != NULL &&
        strstr(de->d_name, "result.txt") == NULL)
    {
        char *file_name = pathcat(dir_name, de->d_name);
        Text *s = readfile(file_name);
        if (s)
        {
            strings[( *size )++] = s;
        }
        free(file_name);
    }
    else if (de->d_type == DT_DIR && strcmp(de->d_name, ".") != 0 &&
        strcmp(de->d_name, "..") != 0)
    {
        char *new_dir = pathcat(dir_name, de->d_name);
        list_dir(new_dir, size, strings);
        free(new_dir);
    }
    de = readdir(dir);
}
closedir(dir);
}
else
{
    printf("error open dir %s\n", dir_name);
}
}

```

```

void write(const char *name_file, int size, Text **strings)
{
    FILE *f = fopen(name_file, "w");
    if (f)
    {
        for (int i = 0; i < size; i++)
        {
            fprintf(f, "%d%s\n", strings[i]->number, strings[i]->text);
            free(strings[i]);
        }
        free(strings);
        fclose(f);
    }
    else
    {

```



```

        printf("error open result.txt");
    }
}

int compar(const void *a, const void *b)
{
    const Text *Text_a = *(Text **)a;
    const Text *Text_b = *(Text **)b;
    if (Text_a->number < Text_b->number)
        return -1;
    if (Text_a->number > Text_b->number)
        return 1;
    return Text_a->number == Text_b->number;
}

int main()
{
    int size;
    size = 0;
    Text **strings = (Text **)malloc(200 * 200 * sizeof(Text));
    list_dir(".", &size, strings);
    qsort(strings, size, sizeof(Text *), compar);
    write("result.txt", size, strings);
    return 0;
}

```