

**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: Работа**  
**с файловой системой.**

Студент гр. 0382

Тюленев. Т.В.

Преподаватель

Берленко Т.А.

Санкт-Петербург

2021

### **Цель работы.**

Изучение функций для работы с файловой системой.

### **Задание.**

#### **Вариант 3**

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**

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

### **Решение:**

1 Small text

2 Simple text

3 Wow? Text?

4 Where am I?

5 So much files!

## **Основные теоретические положения.**

Для работы с файловой системой используется библиотека `dirent.h`.

Ниже представлены ее функции:

```
DIR *opendir(char *dirname)
struct dirent *readdir(DIR *ptr)
void rewinddir(DIR *ptr)
int closedir(DIR *ptr)
```

Функция `opendir()` открывает поток каталога и возвращает указатель на структуру типа `DIR`, которая содержит информацию о каталоге. Не следует модифицировать содержимое данной структуры.

Функция `closedir()` закрывает поток каталога, на который указывает `ptr`.

Функция `readdir()` возвращает название следующего файла в каталоге. Иными словами, функция `readdir()` читает оглавление каталога по одному файлу за раз. Параметр `ptr` должен указывать на поток каталога, открытый с помощью `opendir()`. Структура `dirent` определена для DOS следующим образом:

```
struct dirent{
    char d_name[13];
};
```

Таким образом, после вызова функции `readdir()` параметр `d_name` содержит имя следующего файла в каталоге. Для Windows длина `d_name` равна 260 байтам. Для OS/2 длина равна 256 байтам.

Функция `rewinddir()` вызывает возвращение в начало каталога, на который указывает `ptr` и который был предварительно получен с помощью `opendir()`. Это означает возвращение к первой позиции в каталоге. Благодаря этому каталог может быть прочитан снова.

Функция `closedir()` в случае успеха возвращает 0 и —1 в противном случае. При неудаче переменная `errno` устанавливается равной `EBADF` (недействительный каталог). Функция `opendir()` возвращает `NULL` в том

случае, если каталог не может быть открыт. При этом переменная `errno` устанавливается равной либо `ENOENT` (каталог не найден), либо `ENOMEM` (недостаточно памяти).

Функция `readdir()` возвращает `NULL`, когда достигается конец каталога.

### **Ход работы:**

В начале программы выделяем память под двумерный массив `answer` типа `char`, он нам понадобится, чтобы запомнить и отсортировать содержимое файлов.

Далее вызываем функцию `printDir` для определенной корневой директории, в данной функции будут рекурсивно перебираться файлы и папки, а также генерироваться пути для соответствующих папок и файлов с помощью функции `strcpy` и `strcat`. В процессе данной рекурсии, если программа находит папку, то генерируется путь к данной директории и вызов функции `printDir` с новым путем (рекурсия), если же программа нашла файл, то происходит также генерация пути, но еще в конце мы с помощью функции `check_file` записываем содержимое файла в массив `answer`.

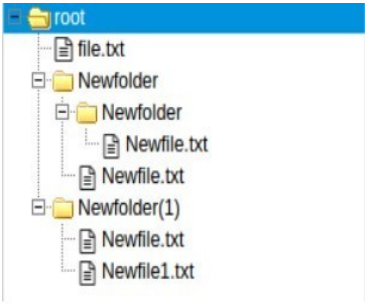
После завершения работы `printDir`, сортируем массив `answer` относительно первых чисел в каждом элементе, для этого вызываем функцию быстрой сортировки, а в функции сравнения будем сравнивать строки относительно начальных чисел (для этого используется функция `atoi`).

Далее запишем отсортированный массив `answer` в файл `result.txt` каждый элемент с новой строки, очистим память.

Исходный код см. в приложении А.

## Тестирование.

Таблица 1 – Результаты тестирования.

№ п/п	Входные данные	Выходные данные	Комментарии
1.		1 Small text 2 Simple text 3 Wow? Text? 4 Where am I? 5 So much files!	Программа нашла файлы и отсортировала их содержимое в порядке возрастания первых цифр в них.

## **Вывод**

Были изучены функции для работы с файловой системой. Разработана программа, которая пробегает по каталогу файлов с помощью функций библиотеки `dirent.h`, находит файлы и сортирует их содержимое по определенным условиям.

# ПРИЛОЖЕНИЕ А

## ИСХОДНЫЙ КОД

### ПРОГРАММЫ

**Название файла: main.c**

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

#define LEN 5000

int count = 0;

int cmp(const void* a, const void* b){
    const char* aa = *((const char**)a);
    const char* bb = *((const char**)b);
    long int num1 = atol(aa);
    long int num2 = atol(bb);
    if(num1 > num2)
        return 1;
    if(num1 < num2)
        return -1;
    return 0;
}

void check_file(const char *filePath, char** answer){
    FILE *fp = fopen(filePath, "r");

    if(NULL != fgets(answer[count++], LEN, fp))

        fclose(fp);
}

void printDir(const char *dirPath, char** answer){
    char next[LEN] = "";
    strcpy(next, dirPath);
    strcat(next, "/");

    DIR *dir = opendir(dirPath);
    struct dirent* de = readdir(dir);

    if(dir){
        while(de){
            if (de->d_type == DT_REG){
                char file_path[LEN] = "";
                strcat(file_path, next);
                strcat(file_path, de->d_name);
                check_file(file_path, answer);
            }

            if (de->d_type == DT_DIR && strcmp(de->d_name, ".") != 0 && strcmp(de->d_name, "..") != 0){
                int len = (int)strlen(next);
```



```

        strcat(next, de->d_name);
        printDir(next, answer);
        next[len] = '\0';
    }
    de = readdir(dir);
}
closedir(dir);
}

int main(){
    char** answer = malloc(sizeof(char*) * LEN);

    for (int i = 0; i < LEN; i++){
        answer[i] = malloc(sizeof(char) * LEN);
        answer[i][0] = '\0';
    }

    printDir("root", answer);

    qsort(answer, count, sizeof(char*), cmp);

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

    for (int i = 0; i < count; i++)
        fprintf(result, "%s\n", answer[i]);

    fclose(result);

    for (int i = 0; i < LEN; i++){
        free(answer[i]);
    }
    free(answer);

    return 0;
}

```