**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. 0382 _____ Куликов М.Д.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2021

**Цель работы.**

Изучение функций для работы с файловой системой

**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 .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

**Основные теоретические положения.**

Для работы с деревом файловой системы используется библиотека dirent.h.

Основные используемые функции:

FILE * fopen( const char * fname, const char * modeopen ); - функция открывает файл fname, при успешном считывании возвращает указатель на него, при неудачном возвращается нулевой указатель.

int fclose( FILE * filestream ); - функция, закрывающая файл.

char * fgets( char * string, int num, FILE * filestream ); - функция, которая позволяет считать строку из файла.

struct dirent *readdir(DIR *dirp); - Функция readdir() возвращает указатель на структуру dirent, представляющую следующую запись каталога в потоке каталога, указанного в dirp. Функция возвращает NULL по достижении последней записи в потоке каталога или если произошла ошибка.

**Выполнение работы.**

В ходе выполнения работы были написаны 3 функции, одной из которых является функция-компаратор для функции qsort.

int cmp(const void *a, const void *b) — функция, сравнивающая 2 числа, стоящие в начале строки файла. Возвращает 1 - если первое число больше второго, -1 - если первое меньше, 0 — если они равны.

char **get_text(const char *path, char **text, int *counter) — функция, считывающая строку из файла и записывающая ее в массив строк text.

char **listDir(const char *path, char **text, int *counter) — функция, работающая с директориями и файлами. Если функция находит файл, то она вызывает для него функцию get_text и продолжает поиск, если функция находит директорию, то она вызывает сама себя для найденной директории.

В функции main вызывается функция qsort с написанным компоратором и сформированным массивом строк, после чего результат записывается в файл result.txt и выделенная память освобождается.

**Тестирование.**

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

| № п/п | Входные данные | Выходные данные | Комментарии |
|---|---|---|---|
| 1. | Test:<br>dir1:<br>dir3:<br>23232.txt<br>kek.txt<br>dir2:<br>23123.txt<br>aboba.txt<br>12321.txt<br>file.txt | -5222 ffff<br><br>2 vtoroy file!<br><br>3 tretiy file!<br><br>52 perviy file<br><br>55 kkkk<br><br>115 fgff | Программа работает корректно |

**Выводы.**

В ходе работы был изучен принцип работы с файловой системой и написана программа, выводящая строки из файлов, находящихся в директориях, в определенном порядке.

## ПРИЛОЖЕНИЕ А
## ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#define _GNU_SOURCE
#include <stdio.h>
#include <dirent.h>
#include <stdlib.h>
#include <sys/types.h>
#include <string.h>


#define BUF 5000


int cmp(const void *a, const void *b) {
    const char *aa = *(char **) a;
    const char *bb = *(char **) b;
    long int a1 = atol(aa);
    long int b1 = atol(bb);
    if (a1 > b1)
        return 1;
    if (a1 < b1)
        return -1;
    return 0;
}

char **get_text(const char *path, char **text, int *counter) {
    char s[100];

    FILE *f = fopen(path, "r");
    if (!f) {
        return text;
    }

    while (fgets(s, 100, f)) {
        strcat(text[*counter], s);
    }

    fclose(f);
    return text;
```

```
}

char **listDir(const char *path, char **text, int *counter) {
    char next[256] = {0};
    strcpy(next, path);
    strcat(next, "/");

    DIR *dir = opendir(path);
    if (!dir) {
        return text;
    }
    struct dirent *de = readdir(dir);

    while (de) {
        if (de->d_type == DT_REG) {
            int len = strlen(next);
            strcat(next, de->d_name);
            get_text(next, text, counter);
            *counter = *counter + 1;
            next[len] = '\0';
        }

          if ((de->d_type == DT_DIR) && strcmp(de->d_name, ".") != 0 &&
strcmp(de->d_name, "..") != 0) {
            int len = strlen(next);
            strcat(next, de->d_name);
            listDir(next, text, counter);
            next[len] = '\0';
        }
        de = readdir(dir);
    }

    closedir(dir);
    return text;
}

int main() {
    int counter = 0;
    char **text = calloc(BUF, sizeof(char *));
    for (int i = 0; i < BUF; i++) {
        text[i] = calloc(BUF, sizeof(char));
    }
```

```
    text = listDir("root", text, &counter);
    qsort(text, counter, sizeof(char *), cmp);

    FILE *f = fopen("result.txt", "w");
    for (int i = 0; i < counter; i++) {
        if (text[i][0] != '\0')
            fprintf(f, "%s\n", text[i]);
    }
    fclose(f);

    for (int i = 0; i < BUF; i++) {
        free(text[i]);
    }
    free(text);
    return 0;
}
```