

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

Студентка гр. 0382

Деткова А.С.

Преподаватель

Берленко Т.А.

Санкт-Петербург

2018

Цель работы.

Изучить способы обхода файловой системы. Научиться работать с данными получаемыми при обходе.

Задание.

Вариант 3

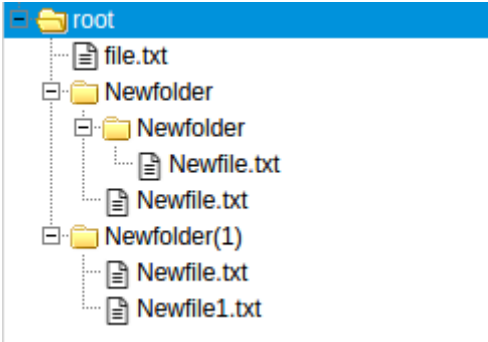
Дана некоторая корневая директория, в которой может находиться некоторое количество папок, в том числе вложенных. В этих папках хранятся некоторые текстовые файлы, имеющие имя вида .txt

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.

Основные теоретические положения.

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

Функция `opendir()` открывает директорию с переданным в функцию именем директорию и возвращает указатель на нее типа `DIR*` (дескриптор директории — структура, хранящая данным о файле (директории)). Функция `readdir()` считывает директорию или файл из текущей директории, на вход получает указатель на дескриптор директории. Возвращает указатель на `struct dirent` (хранит информацию об объекте (имя, тип и т. д.)). `readdir()` вызывается до тех пор пока не будут считаны и обработаны все файлы или директории из рабочей директории, в таком случае вернет `NULL`. `closedir()` закрывает директорию после работы с ней.

Для работы с файлами используется библиотека `stdio.h` (также в ней описаны функции ввода/вывода).

Открывает файл `fopen()` - возвращает указатель на файловый дескриптор `FILE*`. Функция работает в нескольких режимах („r“ - чтение; „w“ — запись и др.). Заккрытие файла — `fclose()`. И другие функции (`fprintf()`, `fscanf()`, `fputs()`, `fgets()` и т. д.).

Выполнение работы.

Для хранения строк из файлов создается двумерный массив `result` размера 5000*1000 (5000 файлов по 1000 символов в них, значения взяты большие, если файлов в директории много). Функция `void scanDir(const char`

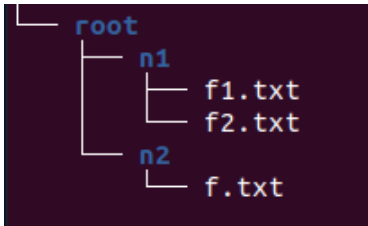
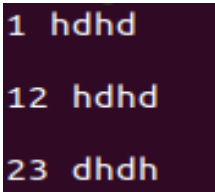
**path, char **res*) — ничего не возвращает, получает имя директории, по дереву которой требуется пройти, и указатель на массив строк. Глобальная переменная *count* считает количество строк, полученных из файлов. Функция *scandir()* открывает текущую директорию и последовательно считывает из нее элементы, если встретится директория, то функция вызывается повторно, генерируя имя для вложенной директории, тем самым углубляясь; если встречается файл, то генерируется путь к нему и вызывается функция *scanFile*. *void scanFile(const char *path, char **txt)* — ничего не возвращает, получает аргументы: путь к файлу и указатель на массив строк. Функция открывает файл по переданному пути к нему и считывает данные в массив строк под номером *count*. В функции *main()* происходит сортировка строк массива строк по принципу величины первого числа в строке. Результат записывается в файл. В конце работы программы закрывается файл, чистится память, выделенная для строк.

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.	 <pre>anna@anna-Aspire-E5-575G:~/dir\$ cat root/n1/f1.txt 12 hdhd anna@anna-Aspire-E5-575G:~/dir\$ cat root/n1/f2.txt 23 dhhd anna@anna-Aspire-E5-575G:~/dir\$ cat root/n2/f.txt 1 hdhd</pre>		Корректная работа программы

Выводы.

Были изучены способы работы с файловой системой.

Была разработана программа, которая осуществляет обход заданной директории с помощью средств, предоставляемых библиотекой `dirent.h` и `stdio.h`, считывает строки из файлов всей файловой системы и сортирует их по заданному параметру, результат выводит в файл.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

#define BUF_SIZE 5000

int count = 0;

int compare(const void *a, const void *b){
    const char *aa = *((const char**)a);
    const char *bb = *((const char**)b);
    long int nmb1 = atol(aa);
    long int nmb2 = atol(bb);

    if(nmb1 > nmb2){
        return 1;
    } else if(nmb1 < nmb2){
        return -1;
    } else {
        return 0;
    }
}

void scanFile(const char *path, char **txt){
    FILE *file = fopen(path, "r");

    if(NULL != fgets(txt[count++], BUF_SIZE/5, file))

        fclose(file);
}

void scanDir(const char *path, char **res){

    char next[BUF_SIZE] = "";
    strcat(next, path);
    strcat(next, "/");

    DIR *dir = opendir(path);
    if(!dir){
        return;
    }

    struct dirent *de = readdir(dir);
    if(dir){
        while(de){
            if(de->d_type == DT_REG){
                char file_path[BUF_SIZE] = "";
                strcat(file_path, next);
                strcat(file_path, de->d_name);
            }
            de = readdir(dir);
        }
    }
}
```

```

        scanFile(file_path,res);
    }

    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);
        scanDir(next,res);
        next[len] = '\0';
    }
    de = readdir(dir);
}
}
closedir(dir);
}

int main() {
    char** result = malloc(sizeof(char *) * BUF_SIZE);
    int i;
    for(i=0; i<BUF_SIZE; i++){
        result[i] = malloc(sizeof(char) * (BUF_SIZE/5));
        result[i][0] = '\0';
    }

    scanDir("root", result);
    qsort(result, count, sizeof(char*), compare);
    FILE *res = fopen("result.txt","w");
    for(i = 0; i<count; i++){
        fprintf(res,"%s\n",result[i]);
    }
    fclose(res);

    for(i=0; i<BUF_SIZE; i++){
        free(result[i]);
    }
    free(result);
    return 0;
}

```