САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Программирование»

Тема: Лабораторная работа № 3: Обход файловой системы

Студентка гр. 3343 _____ Ермолаева В. А.

Преподаватель _____ Государкин Я. С.

Санкт-Петербург

2024

Цель работы

Ознакомиться с понятием рекурсии и освоить написание рекурсивных функций на языке Си, а также изучить работу с файловой системой на языке Си и написать программу для рекурсивного обхода файловой системы.

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.

Выполнение работы

Описание функций:

- int main(): главная функция программы, возвращает 0 при успешном завершении. Вызывает рекурсивную функцию и записывает в файл результат выполнения программы.
- void list_dir(const char* root, Array* arr): рекурсивно обходит файловую систему, сохраняя в массив содержимое всех найденных файлов.
- int cmp_file_info(const void* a, const void* b): сравнивает строки файлов по числу, которое они содержат.
- FileInfo get_file_info(const char* filename, const char* dir_name): составляет путь к файлу и возвращает считанную из него информацию.
- void check_and_resize(Array* arr): перевыделяет память под массив, если его размер превышает имеющийся.
- char* pathcat(const char *path1, const char *path2): возвращает строку, содержащую путь к файлу.

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.	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!	Выходные данные соответствуют ожиданиям.

Выводы

В ходе выполнения лабораторной работы были изучены и применены на практике принципы работы с рекурсией и файловой системой на языке Си. Освоены навыки, необходимые для обхода файловой системы и работы с ее содержимым.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

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

#define STEP 10

typedef struct FileInfo {
    char* text;
    int num;
} FileInfo;

typedef struct Array {
    int cur_index;
    int max_count;
    FileInfo* data;
} Array;

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;
}

void check_and_resize(Array* arr) {
    if(arr->cur_index >= arr->max_count) {
        arr->max_count += STEP;
             FileInfo*  tmp  =  realloc(arr->data,  arr->max_count  *
sizeof(FileInfo));
        arr->data = tmp;
    }
}
```

```
FileInfo get_file_info(const char* filename, const char* dir_name)
{
    FileInfo info;
    char* filepath = pathcat(dir_name, filename);
    info.text = malloc(256);

    FILE* f = fopen(filepath, "r");
    char myString[256];
    fgets(myString, 256, f);

    snprintf(info.text, sizeof(myString), "%s", myString);

    char* pch = strtok (myString," ");
    info.num = atoi(pch);

    fclose(f);
    free(filepath);
    return info;
}

int cmp_file_info(const void* a, const void* b) {
    FileInfo* info_a = (FileInfo *)a;
    FileInfo* info_b = (FileInfo *)b;
    if (info_a->num < info_b->num) return -1;
    if (info_a->num > info_b->num) return 1;
    return info_a->num == info_b->num;
}

void list_dir(const char* root, Array* arr) {
    DIR* root_dir = opendir(root);
    if (root_dir == NULL)
        return;

    struct dirent* dir = readdir(root_dir);
    while (dir) {
        char* new_dir = pathcat(root, dir->d_name);

        if (dir->d_type == DT_REG) {
```

```
                arr->data[arr->cur_index++] = get_file_info(dir->d_name, root);
            check_and_resize(arr);
            free(new_dir);
        }

            else if (dir->d_type == DT_DIR && strcmp(dir->d_name, ".") != 0 && strcmp(dir->d_name, "..") != 0) {
            list_dir(new_dir, arr);
            free(new_dir);
        }

        dir = readdir(root_dir);
    }
    closedir(root_dir);
}

int main() {
    Array arr;
    arr.cur_index = 0;
    arr.max_count = STEP;
    arr.data = malloc(arr.max_count * sizeof(FileInfo));

    list_dir("root", &arr);
              qsort(arr.data,   arr.cur_index,   sizeof(FileInfo), cmp_file_info);

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

    for(int i = 0; i < arr.cur_index; i++) {
        fprintf(f, "%s\n", arr.data[i].text);
        free(arr.data[i].text);
    }

    free(arr.data);
    fclose(f);
    return 0;
}
```