**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 |  | Стрижков И.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 FileArray {

int cur\_index;

int max\_count;

FileInfo\* data;

} FileArray;

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(FileArray\* 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, FileArray\* 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() {

FileArray 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;

}