**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. 3342 |  | Лапшов К.Н. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Получить навыки написания программ для работы с файловой системой. Изучить рекурсивные функции.

## 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!

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.

## Выполнение работы

Была создана структура для хранения сообщений из текстовых файлов. FIleItem имеет поля number – число, которое стоит в начале предложения, message – само сообщение. Функция getDataFromFIle принимает на вход путь до файла, который надо обработать. Выполняется считывание файла, создается указатель на тип FileItem, далее файл обрабатывается, и необходимые данные записываются в структуру, после чего функцию возвращает указатель на структуру.

Функция comparator осуществляет сравнение структур по полю number. Является необходимой для работы функции qsort.

Функция dirLookup принимает на вход строку – путь до директории относительно той директории, в которой была запущена программа, и указатель на указатель массив структур, в который будут сохранены сообщения, а так же размер самого массива. Функция с помощью библиотечной функции readdir проходит по элементам директории и, если он является текстовым файлом, с помощью функции getDataFromFIle добавляет сообщение в массив. Если элемент является поддиректорией, вызывает саму себя от пути данной поддиректории и того же указателя массив. Т.е. функция рекурсивно проходит по всем поддиректориям.

Функция main осуществляет вызов функции dirLookup от текущей директории (строки «./») и от проинициализированого массива структур, далее, с помощью цикла for содержимое списка записывается в файл result.txt, после чего память, занимаемая списком освобождается.

Разработанный программный код см. в приложении А.

## Выводы

Были получены навыки работы с рекурсивными функциями и файловой системой с помощью библиотеки языка Си dirent.h. Была написана программа, рекурсивно обходящая директории и записывающая строки из текстовых файлов, имеющих вид <filename>.txt, всех поддиректорий в порядке возрастания числа, с которого строки начинаются.

**Тестирование**

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| № п/п | Входные данные | Выходные данные |
|  | 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! |

# Приложение А Исходный код программы

Название файла: main.c

#include <stdio.h>

#include <string.h>

#include <dirent.h>

#include <stdlib.h>

typedef struct fileItem{

long number;

char\* massage;

}fileItem;

void memoryError(){

printf("Memory allocation error!");

exit(0);

}

void fileError(){

printf("File open error!");

exit(0);

}

int

comparator(const void \*a, const void \*b) {

const fileItem \*item\_a = a;

const fileItem \*item\_b = b;

if (item\_a->number < item\_b->number)

return -1;

if (item\_a->number > item\_b->number)

return 1;

return 0;

}

fileItem\*

getDataFromFile(char \*filepath){

fileItem\* newFileItem = malloc(sizeof(fileItem));

if(newFileItem == NULL){

memoryError();

}

newFileItem->number = 0;

newFileItem->massage = NULL;

FILE\* currentFile = fopen(filepath, "r");

if(currentFile == NULL){

fclose(currentFile);

fileError();

}

//Считываем цифру

char stringNumber[64];

char \*endPtr;

if(fscanf(currentFile, "%s", stringNumber) != -1){

newFileItem->number = strtol(stringNumber, &endPtr, 10);

};

//Пропускаем символ

char c = fgetc(currentFile);

//Считываем текст

size\_t n = 0;

size\_t capacity = 16;

char \*text = malloc(capacity);

if(text == NULL){

memoryError();

}

while ((c = fgetc(currentFile)) != EOF && c != '\n') {

if (n + 1 >= capacity) {

capacity \*= 2;

text = realloc(text, capacity);

}

text[n++] = c;

}

text[n] = '\0';

newFileItem->massage = text;

fclose(currentFile);

return newFileItem;

}

void dirLookup(const char\* dir\_name, fileItem\*\* currentArr, size\_t \* sizeOfArr){

DIR\* dir = opendir(dir\_name);

struct dirent\* de;

while(de = readdir(dir)){

if(de->d\_type == 8 && strstr(de->d\_name, ".txt")){

char str[120];

strcpy(str, dir\_name);

strcat(str, "/");

strcat(str, de->d\_name);

fileItem\* newFileItem = getDataFromFile(str);

if(newFileItem->number != 0){

\*currentArr = realloc(\*currentArr, sizeof(fileItem)\* ((\*sizeOfArr)+1));

if(currentArr == NULL){

memoryError();

}

(\*currentArr)[(\*sizeOfArr)] = \*newFileItem;

(\*sizeOfArr)++;

}

}

if(de->d\_type == 4 && strcmp(de->d\_name, ".") && strcmp(de->d\_name, "..")){

char\* str = calloc(120, sizeof(char));

if(str == NULL){

memoryError();

}

strcpy(str, dir\_name);

strcat(str, "/");

strcat(str, de->d\_name);

dirLookup(str, currentArr, sizeOfArr);

free(str);

}

}

closedir(dir);

}

int

main(){

size\_t sizeOfArr = 0;

fileItem \* fileItemArr = malloc(sizeof(fileItem));

if(fileItemArr == NULL){

memoryError();

}

dirLookup("./", &fileItemArr, &sizeOfArr);

qsort(fileItemArr, sizeOfArr, sizeof(fileItem), comparator);

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

if(file == NULL){

fclose(file);

fileError();

}

for (int i = 0; i < sizeOfArr; i++) {

fprintf(file,"%ld %s\n", fileItemArr[i].number, fileItemArr[i].massage);

}

fclose(file);

for (int i = 0; i < sizeOfArr; i++) {

free(fileItemArr[i].massage);

}

free(fileItemArr);

return 0;

}