**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. 3341 |  | Шаповаленко Е.В |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Цель данной работы заключается в изучении принципов обхода файловой системы с использованием языка программирования Си. Основные задачи включают в себя разработку программы, способной обходить файловую систему, работа со структурами каталогов, чтение и обработку файлов.

## 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

Подключаются библиотеки *string.h*, *stdio.h*, *stdlib.h*, *dirent.h* и *regex.h*.

Объявляются константы для имен директорий, размеров буфферов для выделения памяти, имя выходного файла и шаблон для проверки расширения файлов в корневой директории.

Описывается структура *Sentence* с полями *num* и *text*, хранящими число и текст соответственно.

В функции *main* создается массив *result* для хранения содержимого файлов. Инициализируется количество предложений и начальная директория. Компилируется регулярное выражение для проверки расширения файлов *regex\_compiled*. Запускается рекурсивная функция *dir\_processing* для обхода файловой системы. По ее завершению массив *result* сортируется с помощью функции *qsort* и выводится в файл *result.txt* функцией *print\_to\_file*.

Функция *dir\_processing* принимает на вход текущий путь, указатель на массив, в который необходимо сохранять содержимое файлов, указатель на переменную, хранящую количество предложений, и регулярное выражение для проверки расширения файлов.

Открывается директория по текущему пути, обрабатывается ее содержимое:

* Если это директория (кроме *".."* и *"."*)— вызывается функция *dir\_processing* с путем до этой директории;
* Если это *\*.txt* файл, что проверяется функцией *check\_file\_extension* — его содержимое записывается в массив *result*.

После обработки директория закрывается.

Функция *qsort* из стандартной библиотеки *stdlib.h* сортирует массив при помощи функции *compare*, которая сравнивает два элемента по полю *num*.

Функция *print\_to\_file* записывает содержимое полученного массива в файл *result.txt*.

Функция *check\_file\_extension* получает на вход имя файла и регулярное выражение, при помощи которого проверяется расширение файла. Функция возвращает 1 если расширение файла *".txt"* и 0 в противном случае.

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 |
|  | 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! | Тест с e.moevm.info, Содержимое выходного файла |
|  | root/ | - | Если корневая директория пуста, то выходной файл будет пуст |

## Выводы

В ходе выполнения работы были изучены принципы обхода файловой системы на языке программирования Си. На практике были применены знания о структуре файловой системы, освоены основные принципы работы с каталогами и файлами.

# Приложение А Исходный код программы

Название файла: solution.c

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#include <dirent.h>

#include <regex.h>

#define PRESENT\_DIR "."

#define PARENT\_DIR ".."

#define START\_DIR "./root"

#define ENDL "\n"

#define START\_DIR\_BUFFER\_SIZE 7

#define NULL\_AND\_SLASH\_CH\_BUFFER\_SIZE 2

#define START\_SENTENCE\_ARRAY\_SIZE 1

#define NEW\_SENTENCE\_BUFFER\_SIZE 1

const char \*output\_file = "./result.txt";

const char \*file\_extension\_pattern = "^.+\\.txt$";

typedef struct Sentence{

long long num;

char \*text;

} Sentence;

int check\_file\_extension(char \*filename, regex\_t regex\_compiled){

return regexec(&regex\_compiled, filename, 0, NULL, 0) == 0;

}

void dir\_processing(char \*current\_path, Sentence \*\*result, long long \*sentences\_num, regex\_t regex\_compiled){

DIR \*current\_dir = opendir(current\_path);

if(current\_dir){

struct dirent \*sub\_dir = readdir(current\_dir);

while(sub\_dir){

long long path\_len = strlen(current\_path) + NULL\_AND\_SLASH\_CH\_BUFFER\_SIZE;

char \*new\_path = (char \*)calloc(path\_len, sizeof(char));

strcpy(new\_path, current\_path);

strcat(new\_path, "/");

if(strcmp(sub\_dir->d\_name, PRESENT\_DIR) == 0 || strcmp(sub\_dir->d\_name, PARENT\_DIR) == 0){

sub\_dir = readdir(current\_dir);

continue;

}

new\_path = (char \*)realloc(new\_path, (path\_len + strlen(sub\_dir->d\_name)) \* sizeof(char));

strcat(new\_path, sub\_dir->d\_name);

if(sub\_dir->d\_type == DT\_DIR)

dir\_processing(new\_path, result, sentences\_num, regex\_compiled);

else if(sub\_dir->d\_type == DT\_REG && check\_file\_extension(sub\_dir->d\_name, regex\_compiled)){

FILE \*file = fopen(new\_path, "r");

if(file){

fscanf(file, "%lld ", &((\*result)[\*sentences\_num].num));

int text\_len = 0, memory = 1;

char \*buffer = (char \*)calloc(1, sizeof(char));

char ch;

while((ch = fgetc(file)) != EOF){

buffer[text\_len] = ch;

text\_len++;

if(text\_len >= memory){

memory \*= 2;

buffer = (char \*)realloc(buffer, memory \* sizeof(char));

}

}

(\*result)[\*sentences\_num].text = buffer;

(\*sentences\_num)++;

\*result = (Sentence \*)realloc(\*result, (\*sentences\_num + NEW\_SENTENCE\_BUFFER\_SIZE) \* sizeof(Sentence));

fclose(file);

}

}

sub\_dir = readdir(current\_dir);

}

closedir(current\_dir);

}

}

int compare(const void \*first, const void \*second){

if(((Sentence \*)first)->num > ((Sentence \*)second)->num)

return 1;

else if(((Sentence \*)first)->num == ((Sentence \*)second)->num)

return 0;

else

return -1;

}

void print\_to\_file(Sentence \*result, long long sentences\_num){

FILE \*file = fopen(output\_file, "w");

if(file){

for(long long i = 0; i < sentences\_num; i++){

fprintf(file, "%lld %s", result[i].num, result[i].text);

if(i < sentences\_num - 1)

fprintf(file, ENDL);

}

fclose(file);

}

}

int main(){

long long sentences\_num = 0;

Sentence \*result = (Sentence \*)calloc(START\_SENTENCE\_ARRAY\_SIZE, sizeof(Sentence));

char \*current\_path = (char \*)calloc(START\_DIR\_BUFFER\_SIZE, sizeof(char));

strcat(current\_path, START\_DIR);

regex\_t regex\_compiled;

regcomp(&regex\_compiled, file\_extension\_pattern, REG\_EXTENDED);

dir\_processing(current\_path, &result, &sentences\_num, regex\_compiled);

qsort(result, sentences\_num, sizeof(Sentence), compare);

print\_to\_file(result, sentences\_num);

return 0;

}