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
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!	Тест с e.moevm.info, Содержимое выходного файла
2.	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;
    }
```