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 _____ Галеев А.Д.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Исследование и разработка методов обхода файловой системы с целью выполнения определенных задач. Задачи могут включать в себя изучение структуры файловой системы, определение методов доступа к файлам и директориям, а также разработку алгоритмов и программного обеспечения для эффективного обхода файлов и директорий. Целью является создание программы, способной автоматизировать обход 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.

Основные теоретические положения

Для решения задач в программе использовались функции стандартной библиотеки языка C, а так же библиотека dirent.h предназначенная для работы с директориями и файлами. Она предоставляет функции для работы с файловой структурой директорий, позволяя осуществлять чтение содержимого директорий, а также получать информацию о файлах в них.

Выполнение работы

Функция main:

В функции main() сначала вызывается dir_lookup() для обхода директории и чтения файлов. Затем вызывается sortSentences() для сортировки прочитанных строк. Открывается файл "result.txt", и отсортированные строки записываются в этот файл с помощью fputs(). После записи строк в файл освобождается выделенная память для массива строк array и его элементов.

Функция обхода содержимого директории:

dir_lookup рекурсивно обходит содержимое директории root. При обнаружении файла с расширением ".txt", вызывает read_file() для чтения содержимого файла. При обнаружении поддиректории, вызывает саму себя рекурсивно для обхода этой поддиректории.

Функция чтения текстовых файлов:

read_file открывает файл, читает первую строку из него, выделяет память для хранения этой строки в массиве array, копирует строку в массив array, затем закрывает файл.

Функции сортировки:

Compare сравнивает две строки как числа, извлекая числа из начала каждой строки. Эта функция используется в qsort для сортировки массива строк.
SortSentences сортирует массив строк sentences с помощью функции compare.

Функция проверки корректности файлов:

file_validator проверяет, что файл имеет расширение .txt. Возвращает 1, если это так, и 0 в противном случае.
dir_validator проверяет, что имя директории не равно "." или "..". Возвращает 1, если это так, и 0 в противном случае.

Тестирование

Результаты тестирования представлены в табл. 1

Таблица 1 – Результаты тестирования

№ проверки	Входные данные	Выходные данные
1.	9 skfjhskdf -3453 dfgkjhd 100 dkfjgdk -3 slkdjfhk 5 kdfjg	-3453 dfgkjhd -3 slkdjfhk 5 kdfjg 9 skfjhskdf 100 dkfjgdk

Выводы

В ходе выполнения лабораторной работы были исследованы различные методы доступа к файлам и директориям, а также разработана программа для их эффективного обхода.

Была изучена структура файловой системы и рассмотрены основные принципы её организации. Затем были разработаны алгоритмы обхода директорий с использованием стандартной библиотеки dirent.h, что позволило получать информацию о содержимом файловой системы и выполнять необходимые операции с файлами и директориями.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main_lb3

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>

#define MAX_FILENAME 128
#define MAX_PATH 256

char **array = NULL;
int size = 0;

int file_validator(char *filename) {
    int res = 1;
    if (strstr(filename, ".txt") == NULL) {
        res = 0;
    }
    return res;
}

int dir_validator(char *dirname) {
    int res = 0;
    if(strcmp(dirname, ".") && strcmp(dirname, "..")) {
        res = 1;
    }
    return res;
}

void read_file(char *filepath) {
    size++;
    FILE *f = fopen(filepath, "r");
    if (f == NULL) {
        printf("Не удалось открыть файл %s\n", filepath);
        return;
    }
    char sentence[MAX_PATH];
    fgets(sentence, sizeof(sentence), f);

    char **temp_array = (char **)realloc(array, size * sizeof(char *));
    if (temp_array == NULL) {
        printf("Не удалось выделить память\n");
        fclose(f);
        return;
    }
    array = temp_array;
```

```
    array[size - 1] = (char *)malloc((strlen(sentence) + 1) *
sizeof(char));
    if (array[size - 1] == NULL) {
        printf("Не удалось выделить память\n");
        fclose(f);
        return;
    }
    strcpy(array[size - 1], sentence);
    fclose(f);
}

void dir_lookup(char *root) {
    char tmp_dir[MAX_PATH];
    strncpy(tmp_dir, root, MAX_FILENAME);

    DIR *root_dir = opendir(tmp_dir);
    if (root_dir == NULL) {
        return;
    }
    struct dirent *dir = readdir(root_dir);

    while (dir) {
        if (dir->d_type == DT_REG && file_validator(dir->d_name)) {
            if (strlen(dir->d_name) > MAX_FILENAME) {
                perror("Слишком большая длина имени");
                exit(0);
            }
            strncat(tmp_dir, "/", strlen(tmp_dir) + 1);
            strncat(tmp_dir, dir->d_name, strlen(dir->d_name));

            read_file(tmp_dir);
            tmp_dir[strlen(tmp_dir) - 1 - strlen(dir->d_name)] = '\0';

        } else if (dir->d_type == DT_DIR && dir_validator(dir->d_name)) {
            if (strlen(dir->d_name) > MAX_FILENAME) {
                perror("Слишком большая длина имени");
                exit(0);
            }
            strncat(tmp_dir, "/", strlen(tmp_dir) + 1);
            strncat(tmp_dir, dir->d_name, strlen(dir->d_name));
            printf("DIR: [%s]\n", tmp_dir);

            dir_lookup(tmp_dir);
            tmp_dir[strlen(tmp_dir) - 1 - strlen(dir->d_name)] = '\0';
        }
        dir = readdir(root_dir);
    }
    closedir(root_dir);
}
```

```
int compare(const void *a, const void *b) {
    const char *sentenceA = *(const char **)a;
    const char *sentenceB = *(const char **)b;
    int numberA, numberB;
    sscanf(sentenceA, "%d", &numberA);
    sscanf(sentenceB, "%d", &numberB);
    if (numberA < numberB) {
        return -1;
    } else if (numberA > numberB) {
        return 1;
    } else {
    return 0;
    }
}

void sortSentences(char **sentences, int numSentences) {
    qsort(sentences, numSentences, sizeof(char *), compare);
}

int main() {
    dir lookup("./root");
    sortSentences(array, size);

    FILE *f = fopen("result.txt", "w");
    for (int i = 0; i < size; i++) {
        if (f == NULL) {
            printf("Не удалось открыть файл");
            return 0;
        }
        fputs(array[i], f);
        fputs("\n", f);
        free(array[i]);
    }

    fclose(f);
    free(array);
    return 0;
}
```