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. 3344 _____ Волков А.А.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Освоить работу с файловой системой при помощи языка Си. Изучить рекурсивный подход в алгоритмах. Обучиться обходить дерево файловой системы. Реализовать алгоритм сортировки информации в заданных файлах и записать полученный результат в результирующий файл.

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

Подключаются требуемые заголовочные файлы <stdio.h>, <stdlib.h> (для работы с динамической памятью и функциями открытия, закрытия файлов), <string.h> (для работы со строками), <dirent.h> (для работы с директориями и их содержимым). Вводится макроопределение BUF, которое определяет размер считываемых символов за один раз из файла.

В функции main() задается указатель str_list, в который нужно записать массив строк, а также целочисленная переменная str_count, хранящая текущее количество элементов в этом массиве. Вызывается рекурсивная функция check_dir(), в которую передается путь на директорию, с которой нужно начать обход вглубь файловой системы, указатель на список строк и указатель на str_count.

Внутри функции check_dir() с помощью функций из <dirent.h> открывается директория, к которой ведет текущий путь и начинается считывание содержимого этой директории. Если же обнаружена еще одна директория, то check_dir() вызывается и для нее с обновленным путем. В случае обнаружения файла подходящего формата, расширяется массив строк и вызывается функция check_file(), в которую передается путь файлу, список строк и указатель на str_count. Полностью пройденная директория закрывается в конце обхода.

Внутри функции check_file() с помощью функций из <stdlib.h> открывается файл и при помощи временной строки и функции fgets() из <string.h> считывается записанная в нем строка, после чего она помещается в массив str_list.

В конце функции main() полученный str_list сортируется при помощи qsort() и записывается в файл result.txt при помощи fprintf().

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	tmp1.txt: -543 first tmp2.txt: 1500 thi!rd tmp3.txt: 754 second	-543 first 754 second: 1500 thi!rd	Файлы расположены внутри корневой директории, внутри которой есть другие директории. Поиск файлов и сортировка происходят успешно.
2.	tmp1.txt: -543 first tmp2.txt: 1500 thi!rd tmp3.txt: 754 second tmp4.txt: 1786 qweruiqweyfkjbasdbfhsdggfsdsdfuiogudosifugoisdufgoisdufgpoisdufpgoiusdfpoigudsfpoigudpsoifgu	-543 first 754 second: 1500 thi!rd 1786 qweruiqweyfkjbasdbfhsdggfsdsdfuiogudosifugoisdufgoisdufgpoisdufpgoiusdfpoigudsfpoigudpsoifgu	Длинные строки корректно обрабатываются.

Выводы

Был изучен рекурсивный подход при написании программы, освоена работа с файловой системой при помощи языка Си.

Была разработана программа, которая обходит корневую директории и выполняет поставленную задачу, результат работы программы записывается в файл в нужном формате.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb_3.c

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

#define BUF 64

int cmp_by_number(const void *a, const void *b)
{
    const char *f_line = *(const char **)a;
 const char *s_line = *(const char **)b;
    long f_num = atol(f_line);
 long s_num = atol(s_line);
 if (f_num - s_num > 0)
  return 1;
 if (f_num - s_num < 0)
  return -1;
 return 0;
}

void check_file(char *filename, char ***str_list, int *str_count)
{
    FILE *file = fopen(filename, "r");
    char *line = (char *)malloc(BUF);
 char *tmp_line = (char *)malloc(BUF);
    fgets(line, BUF, file);
    while (fgets(tmp_line, BUF, file)) {
  line = (char *)realloc(line, strlen(line) + strlen(tmp_line) + 1);
  strcat(line, tmp_line);
 }
    (*str_list)[*str_count-1] = line;
 free(tmp_line);
 free(filename);
 fclose(file);
}

void check_dir(char *path, char ***str_list, int *str_count)
{
 DIR *dir = opendir(path);
 if (dir) {
  struct dirent *obj;
  while (obj = readdir(dir)) {
   char *obj_name = obj->d_name;
      if  (!strcmp(obj_name,  "..")  ||  !strcmp(obj_name,  ".")  ||  !
strcmp(obj_name, "result.txt")) {
    continue;
   }
   if (obj->d_type == DT_DIR) {
    char *new_path = (char *)malloc(strlen(path) + strlen(obj->d_name) +
2);
```

```
    sprintf(new_path, "%s/%s", path, obj->d_name);
    check_dir(new_path, str_list, str_count);
    free(new_path);
   }
   if (obj->d_type == DT_REG && strstr(obj_name, ".txt")) {
    if (*str_count == 0) {
     *str_list = (char **)malloc((++(*str_count)) * sizeof(char *));
    } else {
          *str_list  =  (char  **)realloc(*str_list,  (++(*str_count))  *
sizeof(char *));
    }
    char *filename = (char *)malloc(strlen(path) + strlen(obj_name) + 2);
    sprintf(filename, "%s/%s", path, obj_name);
    check_file(filename, str_list, str_count);
   }
  }
 }
 closedir(dir);
}

int main()
{
 char **str_list = NULL;
 int str_count = 0;
 check_dir(".", &str_list, &str_count);
 qsort(str_list, str_count, sizeof(char*), cmp_by_number);
 FILE *res = fopen("result.txt", "w");
 for (int i = 0; i < str_count; i++) {
  fprintf(res, "%s\n", str_list[i]);
  free(str_list[i]);
 }
 free(str_list);
 fclose(res);
 return 0;
}
```