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. 1304 _____ Ефремов А.А.

Преподаватель _____ Чайка К.В.

Санкт-Петербург
2022

Цель работы.

Научиться работать с директориями и файлами при помощи языка Си. Изучить рекурсивный метод обхода дерева в глубину.

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 .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

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.

Использованные функции:

1. int main() — функция, которая инициализирует динамический массив arr_of_strs, в котором в последствии будут храниться строки из текстовых файлов, далее вызывает функции list_dir, qsort, writeintotxt и в конце затирает выделенную динамически память. Возвращает 0.
2. Void list_dir(const char *dirPath, char **arr_of_strs, long long int *len_size) — функция, совершающая обход по данной директории и всем поддиректориям. Принимает путь к данной директории, а также

указатель на массив arr_of_strs и его размер len_size. Сначала функция открывает данную директорию при помощи opendir(), затем начинает считывать файлы оттуда при помощи readdir(). Обходя файлы в директории, совершает проверку, если файл оказался директорией, то происходит рекурсивный вызов list_dir, если же файл текстовый, то происходит вызов функций add_to_arr. Если считываемых файлов не осталось, закрывает директорию при помощи closedir().

3. char *read_file(char *filename) — функция, принимающая название текстового файла, открывающая его с помощью fopen(), считывающая строку, которая записана в данном текстовом файле, далее закрывающая файл с помощью fclose() и возвращающая считанную строку.
4. void add_to_arr(char *string, char **arr, long long int *len) — функция, добавляющая принимаемую строку string в массив arr, при этом увеличивает значение длины массива len на единицу.
5. int cmp(const void *a, const void *b) — функция, которую принимает функция сортировки строк в массиве qsort(). Принимает два аргумента и приводит их к типу char**, а далее к строке. С помощью atoi() получает два числовых значения, каждое из которых было записано в своей строке. Сравнивая числовые значения, возвращает 1, 0 или -1, если первое число больше другого, если они равны и если второе больше первого соответственно.
6. void writeintotxt(char **arr, long long int *len) — функция, открывающая файл result.txt и записывающая в него принимаемый массив строк при помощи функции fprintf().

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!	Успешно

Выводы.

Были изучены принцип работы с директориями и файлами на языке Си и рекурсивный метод обхода дерева в глубину.

Итогом применения полученных знаний на практике стала написанная программа, которая перебирает все файлы текущей директории и во всех поддиректориях, считывает информацию из файлов, обрабатывает и записывает результат в отдельный текстовый файл.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: solution.c

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

int cmp(const void *a, const void *b)
{
  char *fstr = *(char **)a;
  char *sstr = *(char **)b;
  long long int f = atoi(fstr);
  long long int s = atoi(sstr);
  if (f == s)
    return 0;
  if (f > s)
    return 1;
  if (f < s)
    return -1;
}

void add_to_arr(char *string, char **arr, long long int *len)
{
  arr[(*len)++] = string;
}

void writeintotxt(char **arr, long long int *len)
{
  FILE *f = fopen("./result.txt", "w");
  if (f == NULL)
  {
    puts("Error, can't open txt file to write");
    return;
  }
  int i;
  for (i = 0; i < *len; i++)
    fprintf(f, "%s\n", arr[i]);
  fclose(f);
}

char *read_file(char *filename)
{
  FILE *f = fopen(filename, "r");
  if (f == NULL)
  {
    puts("Error, can't open txt file to read");
    exit(1);
  }
  char *s = malloc(100 * sizeof(char));
  fgets(s, 100, f);
```

```
  fclose(f);
  return s;
}

void list_dir(const char *dirPath, char **arr_of_strs, long long int
*len_size)
{
  DIR *dir = opendir(dirPath);
  if (dir == NULL)
  {
    puts("Error, can't open dir");
    return;
  }
  struct dirent *de = readdir(dir);
  while (de)
  {
    if (de->d_type == DT_DIR && strcmp(de->d_name, ".") != 0 &&
strcmp(de->d_name, "..") != 0)
    {
      char path[100] = {0};
      strcat(path, dirPath);
      strcat(path, "/");
      strcat(path, de->d_name);
      list_dir(path, arr_of_strs, len_size);
    }
    if (strcmp(de->d_name, "result.txt") != 0 && strstr(de->d_name,
".txt") != NULL)
    {
      char path[100] = {0};
      strcat(path, dirPath);
      strcat(path, "/");
      strcat(path, de->d_name);
      add_to_arr(read_file(path), arr_of_strs, len_size);
    }
    de = readdir(dir);
  }
  closedir(dir);
}

int main()
{
  long long int len_size = 0;
  long long int i;
  char **arr_of_strs = malloc(5000 * sizeof(char *));
  if (arr_of_strs != NULL)
    list_dir(".", arr_of_strs, &len_size);
  qsort(arr_of_strs, len_size, sizeof(char *), cmp);
  writeintotxt(arr_of_strs, &len_size);
  for (i = 0; i < len_size; i++)
    free(arr_of_strs[i]);
  free(arr_of_strs);
  return 0;
}
```