Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет безопасности информационных технологий

Дисциплина:

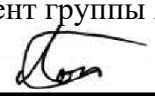
«Основы системного программирования»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1.1

«Работа с файлами и каталогами»

Выполнили:

Ложкин Владислав Сергеевич, студент группы N3245



(подпись)

Проверил:

Грозов Владимир Андреевич,

(отметка о выполнении)

(подпись)

Санкт-Петербург

2024 г.

СОДЕРЖАНИЕ

ЗАДАНИЕ

Цель работы – разработать на языке C для ОС Linux программу, позволяющую выполнять рекурсивный поиск файлов, начиная с указанного каталога, в соответствии с условием и вариантом.

Формат аргумента цель_поиска:

Количество букв в **фамилии** студента, выполняющего работу, **четное**.	Выполняется поиск заданной последовательности байтов. Аргумент `цель_поиска` имеет формат строки в кодировке UTF-8. *Пример запуска программы*: `./lab11psiN32451 /home "лавандовый раф"` (Начиная с каталога `/home`, рекурсивно выполняется поиск файлов, содержащих последовательность байтов 0xd0 0xbb ('л'), 0xd0 0xb0 ('а'), ..., 0xd1 0x84('ф').)

Вариант 3 – ftw()

Программа должна представлять собой консольную утилиту, настройка работы которой осуществляется путем передачи аргументов в строке запуска и/или с помощью переменных окружения (опции необязательны, аргументы *каталог* и *цель_поиска* — обязательны):

lab11abcNXXXXX [опции] каталог цель_поиска

Программа должна выполнять рекурсивный поиск файлов, отвечающих критерию, который задается аргументом цель_поиска в соответствии с условием. При обнаружении файла, отвечающего заданным критериям поиска, программа должна вывести в стандартный поток вывода полный путь к этому файлу. При указании опций -h или -v (или их "длинных" аналогов --help или --version) выполняется вывод информации, заданной опцией, и работа программы завершается.

При определении переменной окружения LAB11DEBUG в стандартный поток ошибок должна выводиться информация о том, что и в каком месте файла нашлось (чтобы было легче понять, почему файл отвечает критериям поиска), а также может выводиться любая дополнительная отладочная информация.

Проект (исходные коды, заголовочные файлы, Makefile и прочие файлы, необходимые для сборки) должен содержаться в отдельном каталоге с именем, совпадающим с названием программы (lab11abcNXXXXX) и собираться с помощью стандартной утилиты make. Исходные файлы программы на языке C должны

компилироваться с помощью gcc. Makefile должен поддерживать как минимум цели all и clean.

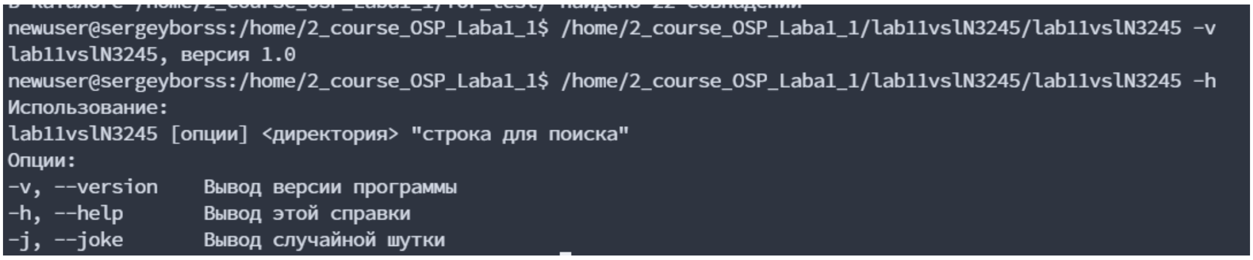
МАКЕ-ФАЙЛ

```
# Определяем компилятор и флаги
CC=gcc
CFLAGS=-Wall -Wextra -Werror -O3

# Название файла
TARGET=lab11vslN3245

# Цель
all: clean $(TARGET)

$(TARGET): lab11vslN3245.c
    $(CC) $(CFLAGS) lab11vslN3245.c -o $(TARGET)

# Цель для очистки проекта
clean:
    rm -f $(TARGET)
```

ОТЧЕТ VALGRIND

```
==669611== HEAP SUMMARY:
==669611==     in use at exit: 0 bytes in 0 blocks
==669611==   total heap usage: 32 allocs, 32 frees, 121,120 bytes allocated
==669611==
==669611== All heap blocks were freed -- no leaks are possible
==669611==
==669611== For lists of detected and suppressed errors, rerun with: -s
==669611== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

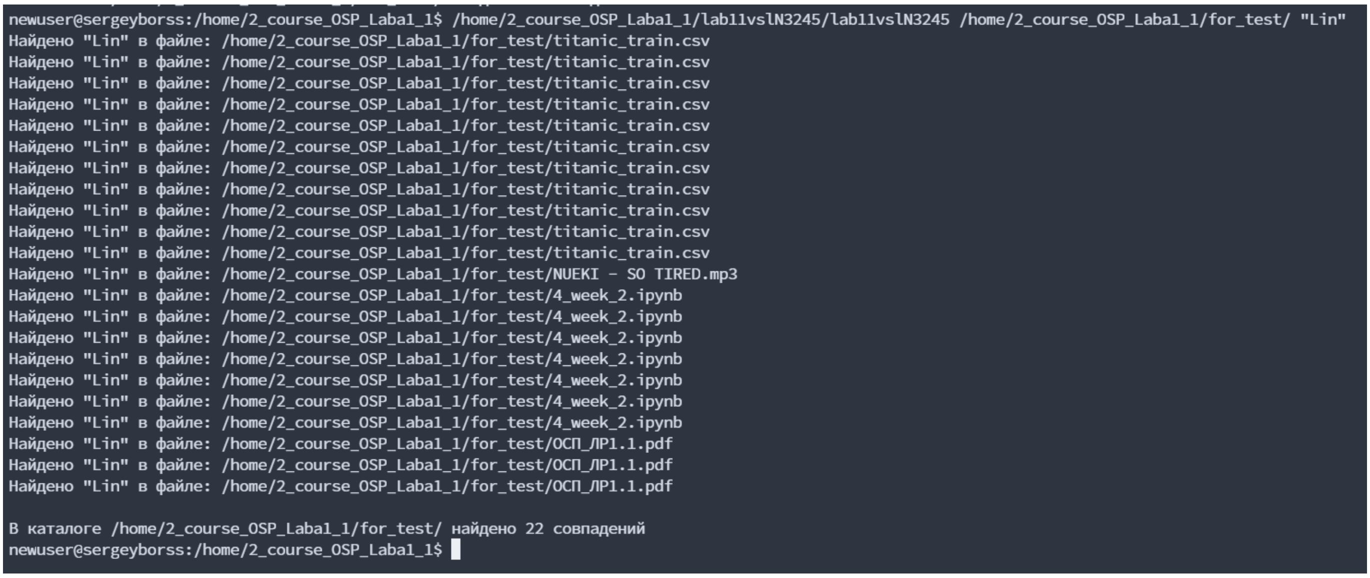


```
newuser@sergeyborss:/home/2_course_OSP_Laba1_1$ /home/2_course_OSP_Laba1_1/lab11vslN3245/lab11vslN3245 -v
lab11vslN3245, версия 1.0
newuser@sergeyborss:/home/2_course_OSP_Laba1_1$ /home/2_course_OSP_Laba1_1/lab11vslN3245/lab11vslN3245 -h
Использование:
lab11vslN3245 [опции] <директория> "строка для поиска"
Опции:
-v, --version    Вывод версии программы
-h, --help       Вывод этой справки
-j, --joke       Вывод случайной шутки
```

Рис. 1 – меню опций

```
newuser@sergeyborss:/home/2_course_OSP_Laba1_1$ /home/2_course_OSP_Laba1_1/lab11vslN3245/lab11vslN3245 /home/2_course_OSP_Laba1_1/for_test/ "Юрий"
Найдено "Юрий" в файле: /home/2_course_OSP_Laba1_1/for_test/Гагарин.txt
Найдено "Юрий" в файле: /home/2_course_OSP_Laba1_1/for_test/Гагарин.txt
Найдено "Юрий" в файле: /home/2_course_OSP_Laba1_1/for_test/Гагарин.txt
Найдено "Юрий" в файле: /home/2_course_OSP_Laba1_1/for_test/Гагарин.txt
Найдено "Юрий" в файле: /home/2_course_OSP_Laba1_1/for_test/Гагарин.txt
Найдено "Юрий" в файле: /home/2_course_OSP_Laba1_1/for_test/link_to_project/valgrind.txt
Найдено "Юрий" в файле: /home/2_course_OSP_Laba1_1/for_test/link_to_project/valgrind.txt
Найдено "Юрий" в файле: /home/2_course_OSP_Laba1_1/for_test/link_to_project/valgrind.txt
Найдено "Юрий" в файле: /home/2_course_OSP_Laba1_1/for_test/link_to_project/valgrind.txt
Найдено "Юрий" в файле: /home/2_course_OSP_Laba1_1/for_test/link_to_project/valgrind.txt
Найдено "Юрий" в файле: /home/2_course_OSP_Laba1_1/for_test/link_to_project/valgrind.txt
Найдено "Юрий" в файле: /home/2_course_OSP_Laba1_1/for_test/link_to_project/valgrind.txt
Найдено "Юрий" в файле: /home/2_course_OSP_Laba1_1/for_test/link_to_project/valgrind.txt
Найдено "Юрий" в файле: /home/2_course_OSP_Laba1_1/for_test/link_to_project/valgrind.txt
Найдено "Юрий" в файле: /home/2_course_OSP_Laba1_1/for_test/link_to_project/valgrind.txt
Найдено "Юрий" в файле: /home/2_course_OSP_Laba1_1/for_test/link_to_project/valgrind.txt
Найдено "Юрий" в файле: /home/2_course_OSP_Laba1_1/for_test/link_to_project/valgrind.txt

В каталоге /home/2_course_OSP_Laba1_1/for_test/ найдено 17 совпадений
newuser@sergeyborss:/home/2_course_OSP_Laba1_1$
```

Рис. 2 – проверка поиска



```
newuser@sergeyborss:/home/2_course_OSP_Laba1_1$ /home/2_course_OSP_Laba1_1/lab11vslN3245/lab11vslN3245 /home/2_course_OSP_Laba1_1/for_test/ "Lin"
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/titanic_train.csv
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/titanic_train.csv
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/titanic_train.csv
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/titanic_train.csv
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/titanic_train.csv
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/titanic_train.csv
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/titanic_train.csv
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/titanic_train.csv
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/titanic_train.csv
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/titanic_train.csv
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/titanic_train.csv
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/NUEKI - SO TIRED.mp3
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/4_week_2.ipynb
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/4_week_2.ipynb
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/4_week_2.ipynb
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/4_week_2.ipynb
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/4_week_2.ipynb
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/4_week_2.ipynb
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/4_week_2.ipynb
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/ОСП_ЛР1.1.pdf
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/ОСП_ЛР1.1.pdf
Найдено "Lin" в файле: /home/2_course_OSP_Laba1_1/for_test/ОСП_ЛР1.1.pdf

В каталоге /home/2_course_OSP_Laba1_1/for_test/ найдено 22 совпадений
newuser@sergeyborss:/home/2_course_OSP_Laba1_1$
```

Рис. 3 – поиск с другим аргументом

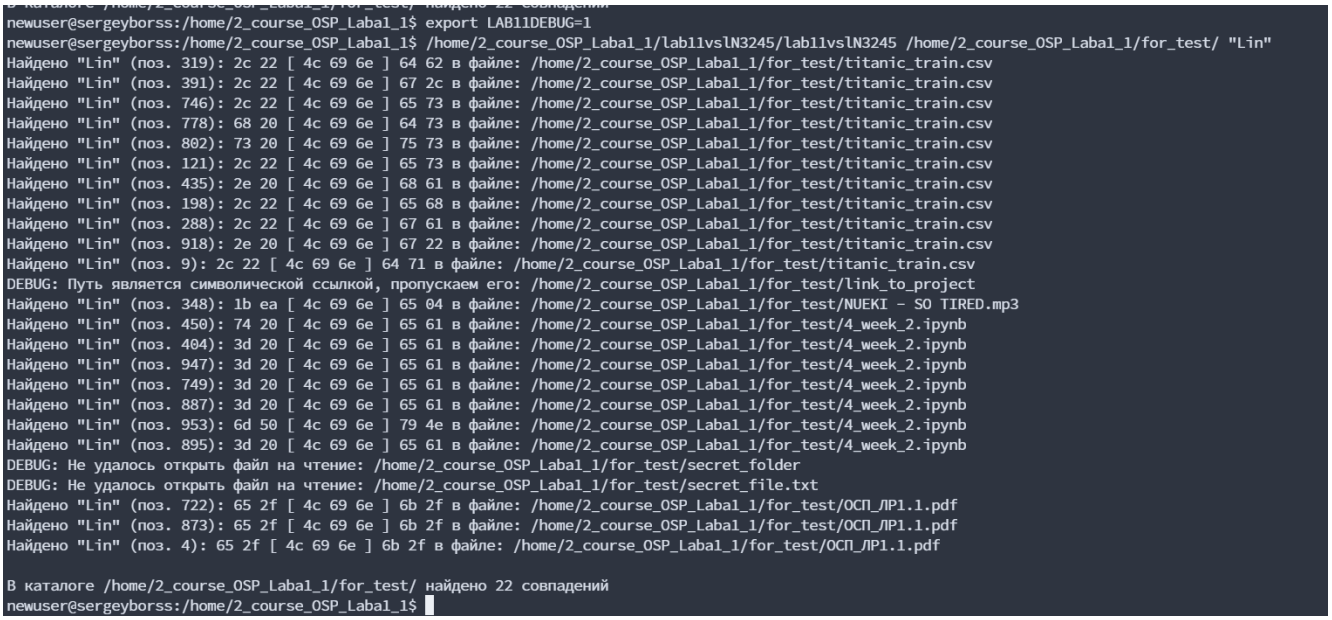


Рис. 4 – с переменными среды

ИСХОДНЫЙ КОД

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ftw.h>
#include <getopt.h>
#include <time.h>
#include <sys/stat.h>
#include <unistd.h>
#include <stdarg.h>
#include <errno.h>


const char *searchString; // Глобальная переменная для хранения строки для поиска
int found = 0;

// Список шуток
const char *jokes[] = {
    "Штирлиц на даче открыл дверь. Включился свет. Штирлиц закрыл дверь. Свет выключился. Холодильник, догадался он.",
    "Из окна дуло. Штирлиц подошел к окну. Дуло исчезло.",
    "Штирлиц топил печку. Через час печка утонула.",
    "Штирлиц мог спать сутками. Но утки с ним спать не хотели.",
    "Встретив гестаповцев, Штирлиц выхватил шашку и закричал: “Порублю!”. Гестаповцы скинулись по рублю и убежали.",
    "Штирлиц вошёл в комнату, из окна дуло. Штирлиц закрыл окно, дуло исчезло.",
    "Штирлиц попал в глубокую яму и чудом из нее вылез. «Чудес не бывает», — подумал Штирлиц и на всякий случай залез обратно.",
    "Штирлицу за шиворот упала гусеница. «Неподалеку взорвался танк», — догадался Штирлиц."};
const int jokesCount = sizeof(jokes) / sizeof(jokes[0]);

// Вывод рандоиной шутки (самое главное в этой программе)
void printJoke()
{
    srand(time(NULL));
    int jokeIndex = rand() % jokesCount;
    printf("%s\n", jokes[jokeIndex]);
}

// Работа с переменными среды
void debugPrint(const char *format, ...) {
    if (getenv("LAB11DEBUG") != NULL) {
        fflush(stdout);  // Проверка на то, что stdout очищен перед записью в stderr
        va_list args;
        va_start(args, format);
        vfprintf(stderr, format, args); // Вывод сообщения в поток ошибок
```

```
        va_end(args);
    }
}

// Функция для поиска строки в файле
int searchStringInFile(const char *filepath, const char *searchStr)
{
    FILE *file = fopen(filepath, "rb"); // Открываем на чтение
    if (!file)
    {
        debugPrint("DEBUG: Ошибка при открытии файла: %s\n", filepath);
        return -1;
    }

    const size_t searchLength = strlen(searchStr); // Вычисляем длину строки для
поиска
    if (searchLength == 0)
    {
        fclose(file); // Если строка поиска пуста, закрываем файл и выходим
        return 0;
    }

    unsigned char buffer[1024 + searchLength - 1]; // Увеличенный буфер (для
"скользящего окна")
    size_t prevPortion = 0;                        // Размер сохраненных данных
от предыдущего чтения

    while (1)
    {
        size_t bytesRead = fread(buffer + prevPortion, 1, sizeof(buffer) -
prevPortion, file); // читаем часть из файла
        size_t totalBytes = bytesRead + prevPortion; // Общее количество байт в
буфере

        if (totalBytes == 0)
            break; // Если ничего не прочитано и ничего не сохранено, выходим

        for (size_t i = 0; i < totalBytes; ++i)
        {
            if (i + searchLength <= totalBytes && memcmp(&buffer[i], searchStr,
searchLength) == 0) // Проверяем, что не вышли за границу и что нашли совпадение
            {
                printf("Найдено \"%s\" ", searchStr); // Печатаем это
                debugPrint("(поз. %zu): ", i);
                for (int j = -2; j < (int)searchLength + 2; ++j) // Делаем
красивый вывод байтов
                {
                    if ((j >= 0 || (size_t)(-j) <= i) && (i + j < totalBytes))
                    {
```

```
                        if (j == 0)
                        {
                            debugPrint("[ "); // Начало искомой
последовательности
                        }
                        debugPrint("%02x ", buffer[i + j]);
                        if (j == (int)searchLength - 1)
                        {
                            debugPrint("] "); // Конец искомой последовательности
                        }
                    }
                }
                printf("в файле: %s\n", filepath);
                found++;
                i += searchLength - 1; // Перемещаем индекс за конец найденной
последовательности
            }
        }

        if (bytesRead == 0)
            break; // Если ничего не прочитано, выходим из цикла

        if (bytesRead < sizeof(buffer) - prevPortion)
        {
            // Если это последнее чтение и оно не заполнило весь буфер, завершаем
            break;
        }

        prevPortion = searchLength - 1; // Сохраняем последние байты для
следующего чтения
        memmove(buffer, buffer + sizeof(buffer) - prevPortion, prevPortion);
    }

    fclose(file);
    return 0;
}

// Функция, которая вызывается ftw()
int processEntry(const char *fpath, const struct stat *sb, int typeflag) {
    (void)sb;

    // Проверка на ошибку получения данных о файле
    struct stat path_stat;
    if (lstat(fpath, &path_stat) < 0) {
        debugPrint("DEBUG: Не удалось получить информацию о файле: %s\n", fpath);
        return 0;
    }

    // Пропускаем символические ссылки
```

```
    if (S_ISLNK(path_stat.st_mode)) {
        debugPrint("DEBUG: Путь является символической ссылкой, пропускаем его:
%s\n", fpath);
        return 0;
    }

    // Пропускаем, если нет доступа на чтение
    if (access(fpath, R_OK) < 0) {
        debugPrint("DEBUG: Не удалось открыть файл на чтение: %s\n", fpath);
        return 0;
    }

    // Если все ок, запускаем поиск
    if (typeflag == FTW_F) {
        searchStringInFile(fpath, searchString);
    }

    return 0;
}

// Опция -h
void printHelp()
{
    printf("Использование:\n");
    printf("lab11vslN3245 [опции] <директория> \"строка для поиска\"\n");
    printf("Опции:\n");
    printf("-v, --version    Вывод версии программы\n");
    printf("-h, --help       Вывод этой справки\n");
    printf("-j, --joke       Вывод случайной шутки\n");
}

// Опция -v
void printVersion()
{
    printf("lab11vslN3245, версия 1.0\n");
}

int main(int argc, char *argv[])
{
    int opt;
    static struct option long_options[] = { // Определение основных команд
        {"help", no_argument, 0, 'h'},
        {"version", no_argument, 0, 'v'},
        {"joke", no_argument, 0, 'j'},
        {0, 0, 0, 0}};

    while ((opt = getopt_long(argc, argv, "hvj", long_options, NULL)) != -1) //
Получаем аргументы программы
    {
```

```
        switch (opt)
        {
        case 'h':
            printHelp();
            return EXIT_SUCCESS;
        case 'v':
            printVersion();
            return EXIT_SUCCESS;
        case 'j':
            printJoke();
            return EXIT_SUCCESS;
        default:
            printHelp();
            return EXIT_FAILURE;
        }
    }

    if (argc - optind < 2)
    {
        fprintf(stderr, "Недостаточно аргументов\n\n");
        printHelp();
        return EXIT_FAILURE;
    }

    searchString = argv[optind + 1];

    // Используем lstat для проверки на символические ссылки и наличие файла
    struct stat path_stat;
    if (lstat(argv[optind], &path_stat) < 0)
    {
        fprintf(stderr, "Ошибка при получении информации об указанной
директории\n");
        return EXIT_FAILURE;
    }

    // Проверяем, является ли путь директорией
    if (!S_ISDIR(path_stat.st_mode))
    {
        fprintf(stderr, "Указанный аргумент не является директорией\n");
        return EXIT_FAILURE;
    }

    // Проверка на доступность директории для чтения
    if (access(argv[optind], R_OK) < 0)
    {
        fprintf(stderr, "Указанная директория не доступна для чтения\n");
        return EXIT_FAILURE;
    }
```

```
    // Проверка на другие ошибки, связанные с ftw
    if (ftw(argv[optind], processEntry, 30) == -1)
    {
        fprintf(stderr, "ftw ошибка: %s\n", strerror(errno));
        return EXIT_FAILURE;
    }

    printf("\nВ каталоге %s найдено %d совпадений\n", argv[optind], found);

    return EXIT_SUCCESS;
}
```

ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы была реализована программа для поиска введенной строки в указанной директории. Для компиляции программы был создан Makefile, утечек памяти выявлено не было, код работает с переменными среды. Программа проходит все придуманные тесты.