

**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»**

**Лабораторная работа №1.1 «Работа с файлами и каталогами»
Вариант 2(Нечетное)**

Выполнил:
студент гр. N32511
Копылов Н.М.



Проверил:
Горлина А.В

Подпись: _____

Санкт-Петербург
2023 г.

Цель работы: Написать программу, которая будет выполнять рекурсивный поиск в каталогах и находить заданные байты в файлах.

Описание проекта:

Лабораторная работа предполагает написание программы на языке программирования C, которая будет осуществлять рекурсивный поиск в каталогах при помощи функции `nftw()` и находить заданные байты в файлах.

Для выполнения поиска в каталогах будет использоваться функция `nftw()`, которая позволяет обойти все файлы и подкаталоги в заданном каталоге и его подкаталогах. Для каждого найденного файла программа будет выполнять проверку на наличие заданных байт внутри файла.

Программа на вход принимает 3 аргумента `<program_name> <DIR> <Bytes>`, после программа начинает рекурсивно искать заданные байты, начиная с `<DIR>`. Так же в проект добавлены опции:

- `-h(--help)` – выводит справочную информацию о программе
- `-v(--version)` – выводит версию программы.

И добавлена возможность использовать переменную окружения `LAB11DEBUG`. С помощью ее можно вывести отладочную информацию на консоль. А именно вывести точное место, где был найдено совпадение байтов.

Отчет valgrind:

HEAP SUMMARY:

==2876== in use at exit: 0 bytes in 0 blocks

==2876== total heap usage: 17,463 allocs, 17,463 frees, 45,810,123 bytes allocated

==2876==

==2876== All heap blocks were freed -- no leaks are possible

Коды файлов проекта:

Makefile:

1. `CC = gcc`
2. `CFLAGS = -O3 -Wall -Wextra`
3. `OBJS = lab11knmN32511.o`
- 4.
5. `.PHONY: all clean`
- 6.
7. `all: lab11knmN32511`
- 8.

```

9. lab1: $(OBJJS)
10.    $(CC) $(CFLAGS) -o $@ $^
11.
12. lab1.o: lab11dabN32511.c
13.    $(CC) $(CFLAGS) -c $< -o $@
14.
15. clean:
16.    rm -f $(OBJJS) lab11knmN32511

```

Код программы(main.c):

```

1. #define _XOPEN_SOURCE 500    // определяет значение, которое должна использовать
   компиляция
2.
3. #include <stdlib.h>          // стандартная библиотека C для управления памятью, строками
   и прочим
4. #include <stdio.h>           // стандартная библиотека C для ввода/вывода
5. #include <string.h>          // стандартная библиотека C для работы со строками
6. #include <errno.h>           // стандартная библиотека C, которая определяет макросы для
   вывода сообщений об ошибках
7. #include <getopt.h>          // стандартная библиотека C для работы с опциями командной
   строки
8. #include <ftw.h>             // стандартная библиотека C для обхода файловой системы
9. #include <sys/param.h>       // системный заголовочный файл для определения константы
   MIN() и структуры данных struct stat
10.
11. #define MAX_INDENT_LEVEL 128 // макрос, определяющий максимальный уровень вложенности
   директорий в обходе файловой системы
12. //Прототипы функций
13. void searching_func(const char *, const char* );
14. void print_entry(int __attribute__((unused)), int __attribute__((unused)), const char *);
15. int walk_func(const char *,const struct stat *sb __attribute__((unused)), int, struct FTW
   *ftwbuf);
16. void walk_dir(char *);
17. void print_help(const char*);
18. void print_version(const char*);
19. char *hex_to_bytes(const char *hexstr);
20. const char *program_name; //Переменная для имени программы
21. char *bytes; //Переменная для искомых байтов
22.
23. void searching_func(const char *fpath, const char* bytes) { //Функция для поиска байтов в
   файле.
24.     FILE *fp; // Указатель на начало файла.
25.     char buffer[1024]; // Переменная char массива символов для чтения файла
26.     int offset = 0; // Пременная для хранения текущего смещения в файле
27.     int read_len; // Переменная для хранения количества прочитанных байт
28.     int count = 0; // Переменная для хранения количества найденных подстрок
29.     if ((fp = fopen(fpath, "rb")) == NULL) { // Открываем файл
30.         printf("Cannot open file %s\n", fpath); // Ошибка открытия файла
31.         return;
32.     }
33.     char* debug = getenv("LAB11DEBUG"); /// получаем значение переменной окружения
   LAB11DEBUG
34.     while ((read_len = fread(buffer, 1, sizeof(buffer), fp)) > 0) { // Цикл для
   считывания данных из файла
35.         for (int i = 0; i < read_len - 5; i++) { // Цикл для обработки данных из файла
36.             if (memcmp(buffer + i, bytes, 3) == 0) { // Сравнение данных из файла и
   нужной подстроки
37.                 count++; // Счетчик++
38.                 if (debug != NULL){ // Проверка на режим отладки
39.                     printf("Found at offset %d\n", offset + i); // Где найдено совпадение
40.                 }
41.             }
42.         }
43.         offset += read_len; // Прибавляем к смещению длину подстроки.
44.     }
45.     fclose(fp); // Закрытие файла
46.     if (count != 0){

```

```

48.     printf("%s: Found %d times\n", fpath, count); // Вывод сколько раз встретилась
        подстрока.
49.     }
50. }
51.
52. void print_entry(int level __attribute__((unused)), int type __attribute__((unused)),
    const char *path) {
53.     if (!strcmp(path, ".") || !strcmp(path, "..")) // Проверяем, является ли запись
        текущей директорией или родительской директорией, если да, то пропускаем
54.         return;
55.     searching_func(path, bytes); // Вызываем функцию поиска searching_func для поиска
        байтовой последовательности в содержимом файла
56. }
57.
58. int walk_func(const char *fpath, const struct stat *sb __attribute__((unused)), int
    typeflag, struct FTW *ftwbuf) {
59.     print_entry(ftwbuf->level, typeflag, fpath); // Функция для обработки текущей записи
60.     return 0;
61. }
62.
63. void walk_dir(char *dir) {
64.     int res = nftw(dir, walk_func, 10, FTW_PHYS); // Вызов функции nftw() для обхода
        директории
65.     if (res < 0) { // Если директория не была найдена, выводим сообщение об ошибке
66.         fprintf(stderr, "ntfw() failed: %s\n", strerror(errno));
67.     }
68. }
69.
70. int main(int argc, char *argv[]) { // main with argv(directory)
71.
72.
73.     int opt = 0;
74.     program_name = argv[0]; // Получаем имя программы для использования в выводе
        сообщений
75.     static struct option long_options[] = {"help", no_argument, 0, 'h'}, {"version",
        no_argument, 0, 'v'}, {0, 0, 0, 0}}; // Определение длинных версий опций -h -v
76.     while ((opt = getopt_long(argc, argv, "hv", long_options, NULL)) != -1) { //
        Обрабатываем аргументы командной строки
77.         switch (opt) {
78.             case 'h':
79.                 print_help(program_name); // Выводим информацию о помощи
80.                 exit(EXIT_SUCCESS); // Выходим из программы с успешным кодом
81.             case 'v':
82.                 print_version(program_name); // Выводим информацию о версии
83.                 exit(EXIT_SUCCESS); // Выходим из программы с успешным кодом
84.             default:
85.                 printf("Try '%s --help' for more information.\n", program_name); //
        Выводим сообщение о том, что переданы неправильные аргументы
86.                 exit(EXIT_FAILURE); // Выходим из программы с ошибкой
87.         }
88.     }
89.     if (argc != 3) { // Проверяем, что количество переданных аргументов правильное
90.         printf("Usage: %s <dir> <string>\n\n", argv[0]); // Выводим информацию о том, как
        использовать программу
91.         exit(EXIT_FAILURE);
92.     }
93.
94.     bytes = hex_to_bytes(argv[2]); // Получаем байтовую последовательность из переданной
        строки
95.     walk_dir(argv[1]); // Запускаем обход директории
96.     free(bytes); // Освобождаем выделенную память
97.     return EXIT_SUCCESS; // Возвращаем успешный код завершения программы
98. }
99.
100. char *hex_to_bytes(const char *hexstr) {
101.     // Получаем длину строки
102.     size_t len = strlen(hexstr);
103.     // Проверяем строку на валидность (четность длины, префикс "0x", минимальная
        длина)

```

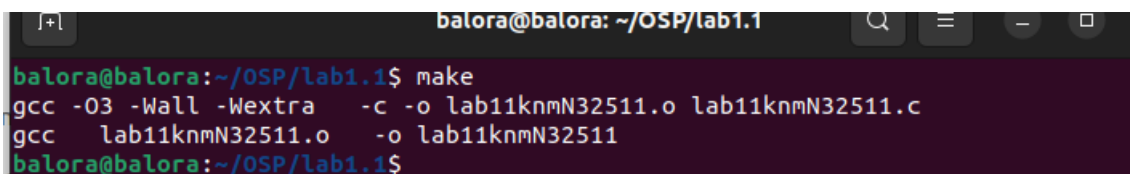
```

104.     if (len % 2 != 0 || len < 2 || hexstr[0] != '0' || hexstr[1] != 'x') {
105.         perror("Invalid hex string");
106.         exit(EXIT_FAILURE);
107.     }
108.     // Вычисляем длину строки без префикса "0x"
109.     size_t hex_len = strlen(hexstr) - 2; // Exclude "0x" prefix
110.     // Выделяем память под буфер, в который будут записаны байты
111.     char *hex_buf = malloc(hex_len / 2);
112.     // Проверяем, удалось ли выделить память
113.     if (hex_buf == NULL) {
114.         perror("Memory allocation error");
115.         exit(EXIT_FAILURE);
116.     }
117.     // Преобразуем каждые два символа в соответствующий байт и записываем его в буфер
118.     for (size_t i = 0; i < hex_len; i += 2) {
119.         sscanf(hexstr + 2 + i, "%2hhx", &hex_buf[i / 2]);
120.     }
121.     // Возвращаем указатель на буфер
122.     return hex_buf;
123. }
124.
125. void print_help(const char* program_name) {
126.     printf("Usage: %s [OPTION]...\n", program_name);
127.     printf("Options:\n");
128.     printf(" -h, --help display this help and exit\n");
129.     printf(" -v, --version output version information and exit\n");
130.     exit(EXIT_SUCCESS);
131. }
132. void print_version(const char* program_name ) {
133.     printf("%s version 1.0\n", program_name);
134.     printf("info: Finds bytes in all files of the directory\n");
135.     printf("Author: Kopylov Nikita\n");
136.     printf("Group: N32511\n");
137.     printf("Lab: 1.01 variant 4\n");
138.     exit(EXIT_SUCCESS);
139. }

```

Скриншоты работы программы:

1. Make all

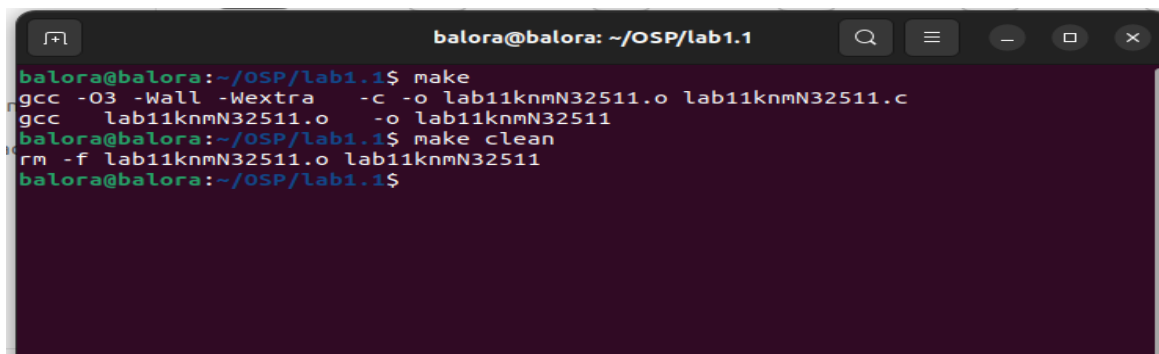


```

balora@balora: ~/OSP/lab1.1
balora@balora:~/OSP/lab1.1$ make
gcc -O3 -Wall -Wextra -c -o lab11knmN32511.o lab11knmN32511.c
gcc lab11knmN32511.o -o lab11knmN32511
balora@balora:~/OSP/lab1.1$

```

2. Make clean



```

balora@balora: ~/OSP/lab1.1
balora@balora:~/OSP/lab1.1$ make
gcc -O3 -Wall -Wextra -c -o lab11knmN32511.o lab11knmN32511.c
gcc lab11knmN32511.o -o lab11knmN32511
balora@balora:~/OSP/lab1.1$ make clean
rm -f lab11knmN32511.o lab11knmN32511
balora@balora:~/OSP/lab1.1$

```

3. Опции

```
balora@balora:~/OSP/lab1.1$ ./lab11knmN32511 -v
./lab11knmN32511 version 1.0
info: Finds bytes in all files of the directory
Author: Kopylov Nikita
Group: N32511
Lab: 1.01 variant 4
balora@balora:~/OSP/lab1.1$ ./lab11knmN32511 --version
./lab11knmN32511 version 1.0
info: Finds bytes in all files of the directory
Author: Kopylov Nikita
Group: N32511
Lab: 1.01 variant 4
balora@balora:~/OSP/lab1.1$ ./lab11knmN32511 -h
Usage: ./lab11knmN32511 [OPTION]...
Options:
  -h, --help display this help and exit
  -v, --version output version information and exit
balora@balora:~/OSP/lab1.1$ ./lab11knmN32511 --help
Usage: ./lab11knmN32511 [OPTION]...
Options:
  -h, --help display this help and exit
  -v, --version output version information and exit
balora@balora:~/OSP/lab1.1$
```

4. Пример работы программы без переменной окружения

```
balora@balora:~/OSP/lab1.1$
balora@balora:~/OSP/lab1.1$
balora@balora:~/OSP/lab1.1$
balora@balora:~/OSP/lab1.1$
balora@balora:~/OSP/lab1.1$
balora@balora:~/OSP/lab1.1$ ./lab11knmN32511 /home "0xff23fc"
/home/balora/Downloads/2728.dmp: Found 6 times
/home/balora/Downloads/BURP/burpsuite_pro_v2023.2.2/burpsuite_pro_v2023.2.2.jar: Found 88 times
/home/balora/Downloads/dump.vmem: Found 116 times
/home/balora/Downloads/memory.zip: Found 30 times
/home/balora/Downloads/dump.vmem.rar: Found 38 times
/home/balora/.local/lib/python3.10/site-packages/unicorn/lib/libunicorn.a: Found 2 times
/home/balora/.local/share/Trash/files/burpsuite_pro_v2023.2.2.zip: Found 34 times
/home/balora/idafree-7.7/til/pc/mssdk_win7.til: Found 1 times
/home/balora/idafree-7.7/til/pc/mssdk.til: Found 1 times
/home/balora/idafree-7.7/til/pc/mssdk64_win7.til: Found 1 times
/home/balora/idafree-7.7/libclpx.so: Found 1 times
/home/balora/volatility/.git/objects/pack/pack-1bd84065b49de7439049272def023a709622dd05.pack: Found 1 times
/home/balora/snap/firefox/common/.mozilla/firefox/6lh4k40d.default/storage/default/https+++web.telegram.org/cache/morgue/198/f1e035590-1729-44c5-9a5e-83b2145eabc6}.final: Found 1 times
```

5. Пример работы программы с переменной окружения

```
Found at offset 13733310
Found at offset 469385946
Found at offset 476781967
Found at offset 487274692
/home/balora/Downloads/dump.vmem.rar: Found 34 times
SFound at offset 6882413
Found at offset 6882701
Found at offset 9679857
Found at offset 9680145
/home/balora/.local/lib/python3.10/site-packages/unicorn/lib/libunicorn.a: Found 4 times
Found at offset 1829
Found at offset 13388329
Found at offset 13388617
Found at offset 14422201
Found at offset 14422489
/home/balora/.local/lib/python3.10/site-packages/unicorn/lib/libunicorn.so.2: Found 5 times
Found at offset 6434001
Found at offset 67619976
Found at offset 78872156
Found at offset 80528752
Found at offset 92849805
Found at offset 100732918
```

Вывод: в ходе лабораторной работы создал программу для рекурсивного обхода директорий при помощи функции `nftw()`. А также познакомился с файловой системой Linux. А также со множеством новых библиотек языка C.