

**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 №1.2

«Использование динамических библиотек»

Вариант 24

Выполнил:

Суханкулиев Мухаммет,
студент группы N3246



(подпись)

Проверил:

Грозов В. А.,
преподаватель практики

(отметка о выполнении)

(подпись)

Санкт-Петербург

2025 г.

СОДЕРЖАНИЕ

Введение	3
1 Описание реализации	4
1.1 Структура проекта	4
1.2 Обработка опций и загрузки плагинов	4
1.3 Реализация плагина	5
2 Сборка	6
2.1 Порядок сборки проекта	6
2.2 Makefile	6
3 Тестирование	7
3.1 Скриншоты тестирования	7
3.1.1 Тестирование библиотеки-плагина	7
3.1.2 Тестирование основной программы	8
3.2 Тестирование утечек памяти (valgrind)	12
Заключение	13
Список использованных источников	14

ВВЕДЕНИЕ

Цель работы – разработать на языке C для ОС Linux:

- программу, позволяющую выполнять рекурсивный поиск файлов, начиная с указанного каталога, с помощью динамических (разделяемых) библиотек-плагинов; Программа должна использовать `nftw()` для обхода файловой системы.
- динамическую библиотеку, реализующую критерий поиска: поиск файлов, содержащих корректные номера кредитных карт в бинарной (little-endian или big-endian) форме (8 байтов, по 4 бита на цифру). Корректность номера проверяется алгоритмом Луна.

Для достижения поставленной цели необходимо решить следующие задачи:

- разработать и отладить плагин, реализующий требуемый критерий поиска, с возможностью самостоятельного тестирования при помощи утилиты `lab1test` и `lab1call`;
- адаптировать и значительно доработать программу обхода файловой системы на базе решения, выполненного в лабораторной работе 1.1;
- обеспечить корректную интеграцию между основной программой и подключаемыми модулями, соблюдая требования, описанные в `plugin_api.h`.

1 ОПИСАНИЕ РЕАЛИЗАЦИИ

1.1 Структура проекта

Проект состоит из нескольких исходных файлов:

- **main.c** — основной файл программы, содержащий логику работы с командной строкой, загрузку плагинов, обработку опций и запуск поиска.
- **utils.c** — вспомогательные функции для работы с плагинами, обработки опций, вывода справки и версии программы.
- **utils.h** — заголовочный файл с объявлениями внешних переменных, функций и структур данных.
- **plugin_api.h** — интерфейс для взаимодействия с плагинами (структура плагинов и функции API, предоставлено преподавателем).
- **libmsN3246.c** — плагин для поиска номеров кредитных карт в бинарной форме. Он использует API плагинов для взаимодействия с основной программой.

1.2 Обработка опций и загрузки плагинов

Основная логика обработки командной строки состоит в парсинге переданных параметров с использованием библиотеки `getopt`. Функция `parse_command_line()` обрабатывает базовые опции, такие как:

- **-P dir:** Указывает каталог для плагинов.
- **-A:** Использует логическое "И" для объединения условий плагинов.
- **-O:** Использует логическое "ИЛИ" для объединения условий плагинов.
- **-N:** Инвертирует итоговое условие.
- **-v:** Показывает информацию о студенте.
- **-h:** Показывает справку по использованию.

После первичной обработки командной строки и установления базовых опций, программа переходит к загрузке плагинов из указанного каталога. Плагины загружаются с помощью функции `load_plugins()`, которая использует `dlopen()` для динамической загрузки `.so` файлов. Затем для каждого подключённого плагина вызывается функция `plugin_get_info()`, возвращающая структуру с описанием и поддерживаемыми опциями плагина.

После этого:

- Выполняется проверка на конфликтующие опции между плагинами с помощью `check_option_conflicts()`.

- Если конфликтов нет, все опции плагинов собираются в единый список.
- Проводится вторичный парсинг командной строки, на этот раз включая опции, добавленные плагинами, с установкой их значений.

В результате утилита поддерживает гибкую архитектуру, позволяя одновременно использовать несколько плагинов, каждый из которых определяет собственные условия фильтрации.

Примечание: Опции плагинов должны указываться со знаком " = ".

1.3 Реализация плагина

Плагин `libmsN3246.so` реализует функциональность поиска корректных номеров кредитных карт в бинарной форме. Ключевые особенности реализации:

- Определена одна опция `--ccards-bin`, активирующая плагин (в основной программе опция не обязательна для указания в командной строке: плагин активируется автоматически при наличии в каталоге плагинов, т. к. его значение передается как `null`).
- Каждый номер представляет собой 8 байт, где каждая четверть байта (4 бита) содержит одну BCD-цифру.
- Проверка корректности реализована по следующим критериям:
 - o Все цифры должны быть BCD (0–9).
 - o Недопустимо избыточное количество нулей (≥ 10).
 - o Номер не должен начинаться с нуля.
 - o Проходит проверку по алгоритму Луна.

Файл читается побайтово, с шагом в 1 байт, что обеспечивает нахождение всех потенциальных номеров. Параллельно проверяется как прямое, так и обратное представление каждого блока (little-endian – big-endian).

При наличии переменной окружения `LAB1DEBUG` плагин выводит в `stderr` отладочную информацию о процессе обработки.

2 СБОРКА

2.1 Порядок сборки проекта

Для компиляции используется следующая команда:

```
make all
```

Компилирует основную lab12msN3246 и libmsN3246.so

2.2 Makefile

```
CC = gcc
CFLAGS = -Wall -Wextra -Werror -O3 -D_XOPEN_SOURCE=700
LDFLAGS = -ldl
SRC = main.c utils.c
OBJ = $(SRC:.c=.o)
EXEC = lab12msN3246
PLUGIN_SRC = libmsN3246.c
PLUGIN_SO = libmsN3246.so

all: $(EXEC) $(PLUGIN_SO)

$(EXEC): $(OBJ)
    $(CC) $(OBJ) -o $(EXEC) $(LDFLAGS)

$(PLUGIN_SO): $(PLUGIN_SRC)
    $(CC) -shared -fPIC $(CFLAGS) $< -o $@

%.o: %.c
    $(CC) $(CFLAGS) -c $< -o $@

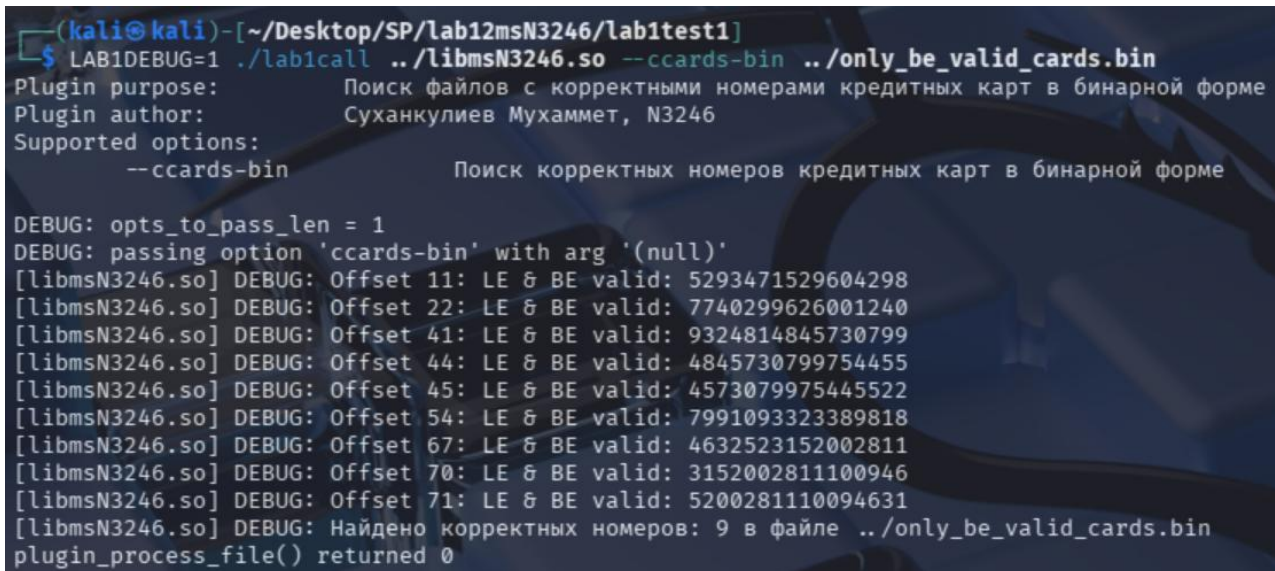
clean:
    rm -f $(OBJ) $(EXEC) $(PLUGIN_SO)

.PHONY: all clean
```

3 ТЕСТИРОВАНИЕ

3.1 Скриншоты тестирования

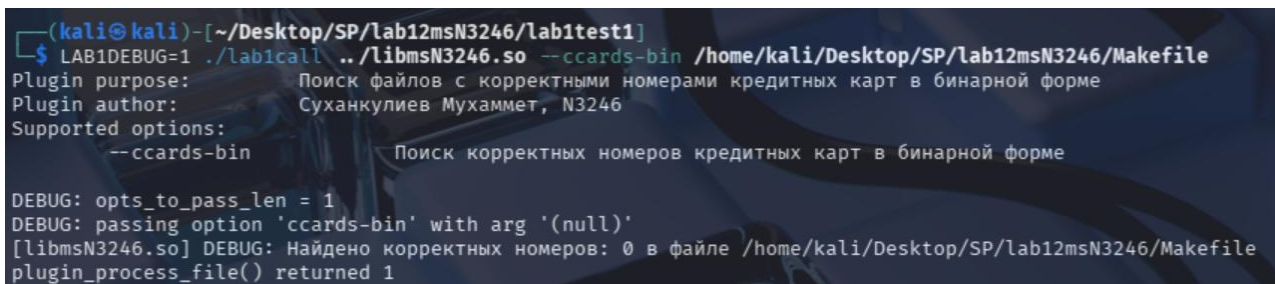
3.1.1 Тестирование библиотеки-плагина



```
(kali㉿kali)-[~/Desktop/SP/lab12msN3246/lab1test1]
$ LAB1DEBUG=1 ./lab1call ../libmsN3246.so --ccards-bin ../only_be_valid_cards.bin
Plugin purpose:      Поиск файлов с корректными номерами кредитных карт в бинарной форме
Plugin author:       Суханкулиев Мухаммет, N3246
Supported options:
    --ccards-bin      Поиск корректных номеров кредитных карт в бинарной форме

DEBUG: opts_to_pass_len = 1
DEBUG: passing option 'ccards-bin' with arg '(null)'
[libmsN3246.so] DEBUG: Offset 11: LE & BE valid: 5293471529604298
[libmsN3246.so] DEBUG: Offset 22: LE & BE valid: 7740299626001240
[libmsN3246.so] DEBUG: Offset 41: LE & BE valid: 9324814845730799
[libmsN3246.so] DEBUG: Offset 44: LE & BE valid: 4845730799754455
[libmsN3246.so] DEBUG: Offset 45: LE & BE valid: 4573079975445522
[libmsN3246.so] DEBUG: Offset 54: LE & BE valid: 7991093323389818
[libmsN3246.so] DEBUG: Offset 67: LE & BE valid: 4632523152002811
[libmsN3246.so] DEBUG: Offset 70: LE & BE valid: 3152002811100946
[libmsN3246.so] DEBUG: Offset 71: LE & BE valid: 5200281110094631
[libmsN3246.so] DEBUG: Найдено корректных номеров: 9 в файле ../only_be_valid_cards.bin
plugin_process_file() returned 0
```

Рисунок 1 – Файл, соответствующий критерию



```
(kali㉿kali)-[~/Desktop/SP/lab12msN3246/lab1test1]
$ LAB1DEBUG=1 ./lab1call ../libmsN3246.so --ccards-bin /home/kali/Desktop/SP/lab12msN3246/Makefile
Plugin purpose:      Поиск файлов с корректными номерами кредитных карт в бинарной форме
Plugin author:       Суханкулиев Мухаммет, N3246
Supported options:
    --ccards-bin      Поиск корректных номеров кредитных карт в бинарной форме

DEBUG: opts_to_pass_len = 1
DEBUG: passing option 'ccards-bin' with arg '(null)'
[libmsN3246.so] DEBUG: Найдено корректных номеров: 0 в файле /home/kali/Desktop/SP/lab12msN3246/Makefile
plugin_process_file() returned 1
```

Рисунок 2 – Файл не соответствует критерию

3.1.2 Тестирование основной программы

```
(kali@kali)-[~/Desktop/SP/lab12msN3246/main]
$ ./lab12msN3246
Usage: lab12msN3246 [options] [directory]
Options:
  -P dir    Указать каталог с плагинами
  -A        Объединение условий по AND (по умолчанию)
  -O        Объединение условий по OR
  -N        Инвертировать итоговое условие
  -v        Show author information
  -h        Show this help message

(kali@kali)-[~/Desktop/SP/lab12msN3246/main]
$ ./lab12msN3246 -v
lab12msN3246 version 1.0
Author: Суханкулиев Мухаммет
Group: N3246
Поток: ОСП N23 1.2
Variant: 24
```

Рисунок 3 – Проверка print_help и print_version

```
(kali@kali)-[~/Desktop/SP/lab12msN3246/main]
$ ./lab12msN3246 ../lab1test1 --ccards-bin
Loaded 1 plugin(s):
  Plugin [/home/kali/Desktop/SP/lab12msN3246/main/libmsN3246.so]
    Purpose: Поиск файлов с корректными номерами кредитных карт в бинарной форме
    Author : Суханкулиев Мухаммет, N3246
    --ccards-bin : Поиск корректных номеров кредитных карт в бинарной форме
Parsed options:
  -P plugin_dir = /home/kali/Desktop/SP/lab12msN3246/main
  -A = true
  -O = false
  -N = false
  -v = false
  -h = false
  --ccards-bin = 1

/home/kali/Desktop/SP/lab12msN3246/lab1test1/lab1call.c
/home/kali/Desktop/SP/lab12msN3246/lab1test1/Makefile
/home/kali/Desktop/SP/lab12msN3246/lab1test1/libavg.so
/home/kali/Desktop/SP/lab12msN3246/lab1test1/lab1call
/home/kali/Desktop/SP/lab12msN3246/lab1test1/libavg.c
/home/kali/Desktop/SP/lab12msN3246/lab1test1/lab1test.c
/home/kali/Desktop/SP/lab12msN3246/lab1test1/plugin_api.h
/home/kali/Desktop/SP/lab12msN3246/lab1test1/lab1test
```

Рисунок 4 – Работа с плагином


```

(kali@kali)-[~/Desktop/SP/lab12msN3246/main]
$ ./lab12msN3246 -O -A -P . . --ccards-bin
Error: options -A and -O cannot be used together

(kali@kali)-[~/Desktop/SP/lab12msN3246/main]
$ ./lab12msN3246 -N -P . . --ccards-bin
Loaded 1 plugin(s):
  Plugin [./libmsN3246.so]
    Purpose: Поиск файлов с корректными номерами кредитных карт в бинарной форме
    Author : Суханкулиев Мухаммет, N3246
    --ccards-bin : Поиск корректных номеров кредитных карт в бинарной форме
Parsed options:
  -P plugin_dir = .
  -A = true
  -O = false
  -N = true
  -v = false
  -h = false
  --ccards-bin = 1

/home/kali/Desktop/SP/lab12msN3246/main/Makefile

```

Рисунок 5 – Парсинг опций

```

(kali@kali)-[~/Desktop/SP/lab12msN3246/main]
$ ./lab12msN3246 -O -P . ./lab1test1 --ccards-bin
Loaded 1 plugin(s):
  Plugin [./libmsN3246.so]
    Purpose: Поиск файлов с корректными номерами кредитных карт в бинарной форме
    Author : Суханкулиев Мухаммет, N3246
    --ccards-bin : Поиск корректных номеров кредитных карт в бинарной форме
Parsed options:
  -P plugin_dir = .
  -A = false
  -O = true
  -N = false
  -v = false
  -h = false
  --ccards-bin = 1

/home/kali/Desktop/SP/lab12msN3246/lab1test1/lab1call.c
/home/kali/Desktop/SP/lab12msN3246/lab1test1/Makefile
/home/kali/Desktop/SP/lab12msN3246/lab1test1/libavg.so
/home/kali/Desktop/SP/lab12msN3246/lab1test1/lab1call
/home/kali/Desktop/SP/lab12msN3246/lab1test1/libavg.c
/home/kali/Desktop/SP/lab12msN3246/lab1test1/lab1test.c
/home/kali/Desktop/SP/lab12msN3246/lab1test1/README.txt
/home/kali/Desktop/SP/lab12msN3246/lab1test1/plugin_api.h
/home/kali/Desktop/SP/lab12msN3246/lab1test1/lab1test

```

Рисунок 6 – Парсинг опций 2

```

(kali@kali)-[~/Desktop/SP/lab12msN3246/main]
$ LAB1DEBUG=1 ./lab12msN3246 -c . --ccards-bin
Недопустимая опция: -c
Usage: lab12abcNXXXXX [options] [directory]
Options:
  -P dir    Указать каталог с плагинами
  -A        Объединение условий по AND (по умолчанию)
  -O        Объединение условий по OR
  -N        Инвертировать итоговое условие
  -v        Show author information
  -h        Show this help message

(kali@kali)-[~/Desktop/SP/lab12msN3246/main]
$ LAB1DEBUG=1 ./lab12msN3246 . --ccards-bin --asdasd=kek
Loaded 1 plugin(s):
  Plugin [/home/kali/Desktop/SP/lab12msN3246/main/libmsN3246.so]
    Purpose: Поиск файлов с корректными номерами кредитных карт в бинарной форме
    Author : Суханкулиев Мухаммет, N3246
      --ccards-bin : Поиск корректных номеров кредитных карт в бинарной форме
Error: unsupported option(s): passed 2 plugin options, but only 1 supported

```

Рисунок 7 – Обработка некорректных опций

```

(kali@kali)-[~/Desktop/SP/lab12msN3246/main]
$ ./lab12msN3246 -P ../pluginsasd .
Cannot open plugin dir '../pluginsasd': No such file or directory

(kali@kali)-[~/Desktop/SP/lab12msN3246/main]
$ ./lab12msN3246 -P . ./kek
Loaded 1 plugin(s):
  Plugin [./libmsN3246.so]
    Purpose: Поиск файлов с корректными номерами кредитных карт в бинарной форме
    Author : Суханкулиев Мухаммет, N3246
      --ccards-bin : Поиск корректных номеров кредитных карт в бинарной форме
Parsed options:
  -P plugin_dir = .
  -A = true
  -O = false
  -N = false
  -v = false
  -h = false
  --ccards-bin = null
Error: cannot resolve './kek': No such file or directory

```

Рисунок 8 – Обработка аргументов опций

```

(kali@kali)-[~/Desktop/SP/lab12msN3246/main]
$ LAB1DEBUG=1 ./lab12msN3246 -P ../lab1test1 ../lab1test1 --entropy=1 --offset-from=0 --offset-to=100
Loaded 1 plugin(s):
  Plugin [../lab1test1/libavg.so]
    Purpose: Check if entropy of a file or its part is less than the given value
    Author : Alexei Guirik
      --entropy : Target value of entropy
      --offset-from : Start offset
      --offset-to : End offset
Parsed options:
-P plugin_dir = ../lab1test1
-A = true
-O = false
-N = false
-v = false
-h = false
--entropy = 1
--offset-from = 0
--offset-to = 100

DEBUG: libavg.so: Got option 'entropy' with arg '1'
DEBUG: libavg.so: Got option 'offset-from' with arg '0'
DEBUG: libavg.so: Got option 'offset-to' with arg '100'
DEBUG: libavg.so: Inputs: entropy = 1.000000, offset_from = 0, offset_to = 100
DEBUG: libavg.so: Calculated entropy = 0.559680
/home/kali/Desktop/SP/lab12msN3246/lab1test1/lab1call.c
DEBUG: libavg.so: Got option 'entropy' with arg '1'
DEBUG: libavg.so: Got option 'offset-from' with arg '0'
DEBUG: libavg.so: Got option 'offset-to' with arg '100'
DEBUG: libavg.so: Inputs: entropy = 1.000000, offset_from = 0, offset_to = 100
DEBUG: libavg.so: Calculated entropy = 0.607415
/home/kali/Desktop/SP/lab12msN3246/lab1test1/Makefile
DEBUG: libavg.so: Got option 'entropy' with arg '1'
DEBUG: libavg.so: Got option 'offset-from' with arg '0'
DEBUG: libavg.so: Got option 'offset-to' with arg '100'
DEBUG: libavg.so: Inputs: entropy = 1.000000, offset_from = 0, offset_to = 100
DEBUG: libavg.so: Calculated entropy = 0.209732
/home/kali/Desktop/SP/lab12msN3246/lab1test1/libavg.so
DEBUG: libavg.so: Got option 'entropy' with arg '1'

```

Рисунок 9 – Работа с другим плагином и отладка

```

(kali@kali)-[~/Desktop/SP/lab12msN3246/main]
$ ./lab12msN3246 -N -P ../lab1test1 . --entropy=1 --offset-from=0 --offset-to=100 --ccards-bin
Loaded 2 plugin(s):
  Plugin [../lab1test1/libavg.so]
    Purpose: Check if entropy of a file or its part is less than the given value
    Author : Alexei Guirik
      --entropy : Target value of entropy
      --offset-from : Start offset
      --offset-to : End offset
  Plugin [../lab1test1/libmsN3246.so]
    Purpose: Поиск файлов с корректными номерами кредитных карт в бинарной форме
    Author : Суханкулиев Мухаммет, N3246
      --ccards-bin : Поиск корректных номеров кредитных карт в бинарной форме
Parsed options:
-P plugin_dir = ../lab1test1
-A = true
-O = false
-N = true
-v = false
-h = false
--entropy = 1
--offset-from = 0
--offset-to = 100
--ccards-bin = 1

/home/kali/Desktop/SP/lab12msN3246/main/Makefile

```

Рисунок 10 – Два плагина


```

(kali@kali)=[~/Desktop/SP/lab12msN3246/main]
$ LAB1DEBUG=1 ./lab12msN3246 -N -P .
Loaded 1 plugin(s):
  Plugin [./libmsN3246.so]
    Purpose: Поиск файлов с корректными номерами кредитных карт в бинарной форме
    Author : Суханкулиев Мухаммет, N3246
    --ccards-bin : Поиск корректных номеров кредитных карт в бинарной форме
Parsed options:
  -P plugin_dir = .
  -A = true
  -O = false
  -N = true
  -v = false
  -h = false
  --ccards-bin = null

[libmsN3246.so] DEBUG: Найдено корректных номеров: 0 в файле /home/kali/Desktop/SP/lab12msN3246/main/Makefile
[libmsN3246.so] DEBUG: Offset 1902: LE valid: 7466007075747300
[libmsN3246.so] DEBUG: Offset 1903: LE & BE valid: 6600707574730066
[libmsN3246.so] DEBUG: Offset 2057: LE & BE valid: 6464697200676574
[libmsN3246.so] DEBUG: Offset 2075: BE valid: 7600637475706600
[libmsN3246.so] DEBUG: Offset 2108: LE & BE valid: 7269746500737472
[libmsN3246.so] DEBUG: Offset 2218: LE & BE valid: 6572656769737465
[libmsN3246.so] DEBUG: Offset 6402: LE & BE valid: 1502170000740748
[libmsN3246.so] DEBUG: Offset 7112: LE & BE valid: 8974242848895424
[libmsN3246.so] DEBUG: Offset 7113: LE & BE valid: 7424284889542430
[libmsN3246.so] DEBUG: Offset 7281: LE & BE valid: 5741564155415449
[libmsN3246.so] DEBUG: Offset 7347: LE & BE valid: 4048894424284489
[libmsN3246.so] DEBUG: Offset 7449: LE & BE valid: 2335000048894424
[libmsN3246.so] DEBUG: Offset 7451: BE valid: 4810244489480000
[libmsN3246.so] DEBUG: Offset 7636: LE & BE valid: 1147104839742410
[libmsN3246.so] DEBUG: Offset 8561: LE & BE valid: 5741564155415449
[libmsN3246.so] DEBUG: Offset 8634: BE valid: 8948906600000000
[libmsN3246.so] DEBUG: Offset 10096: LE & BE valid: 4889051137000031
[libmsN3246.so] DEBUG: Offset 12398: LE & BE valid: 7870656374656420
[libmsN3246.so] DEBUG: Offset 19164: BE valid: 7369676572656400
[libmsN3246.so] DEBUG: Offset 19166: LE & BE valid: 6572656769737465
[libmsN3246.so] DEBUG: Offset 19572: LE & BE valid: 6572656769737465
[libmsN3246.so] DEBUG: Offset 19809: LE & BE valid: 7472726368724047
[libmsN3246.so] DEBUG: Offset 20158: LE & BE valid: 7265616464697240
[libmsN3246.so] DEBUG: Offset 20253: LE & BE valid: 6563757273697665
[libmsN3246.so] DEBUG: Offset 20346: LE & BE valid: 6677726974654047
[libmsN3246.so] DEBUG: Найдено корректных номеров: 25 в файле /home/kali/Desktop/SP/lab12msN3246/main/lab12msN3246
[libmsN3246.so] DEBUG: Offset 72: LE & BE valid: 8974242848895424
[libmsN3246.so] DEBUG: Offset 73: LE & BE valid: 7424284889542430

```

Рисунок 11 – Отладочный режим

3.2 Тестирование утечек памяти (valgrind)

```
valgrind --leak-check=full ... ./lab12msN3246 ... ~/Desktop ...
```

```

==284865== HEAP SUMMARY:
==284865==      in use at exit: 0 bytes in 0 blocks
==284865==    total heap usage: 461 allocs, 461 frees, 136,673,224 bytes
allocated
==284865==
==284865== All heap blocks were freed -- no leaks are possible
==284865==
==284865== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

```

ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы была разработана модульная утилита на языке C для Linux, реализующая рекурсивный поиск файлов с использованием динамически подключаемых плагинов и обходом файловой системы с помощью `nftw()`.

Основная программа обеспечивает корректную загрузку плагинов, парсинг базовых и пользовательских опций.

Разработанный плагин реализует поиск номеров кредитных карт в бинарной форме с проверкой по алгоритму Луна и успешно интегрирован в систему.

Проведённое тестирование, включая анализ утечек памяти с использованием `valgrind`, подтвердило корректность и стабильность реализации.

Это позволило освоить принципы работы с динамически загружаемыми библиотеками, организовать безопасную обработку пользовательских опций и расширяемую архитектуру программ.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Керниган, Брайан У., Ритчи, Деннис М.: Язык программирования С, 2-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс» – 2009. – 304 с.: ил. – Парал. тит. англ. – URL: [Керниган, Ритчи. Язык программирования С.pdf - Google Диск](#)
2. Гирик А. В.: Основы системного программирования. Файлы и каталоги. Управление памятью – Университет ИТМО – 2025. – URL: [02. ОСП. Файлы и каталоги. Управление памятью.pdf - Google Диск](#)
3. Гирик А. В.: Основы системного программирования. Исполняемые файлы и разделяемые библиотеки – Университет ИТМО – 2025. – URL: [03. ОСП. Исполняемые файлы и разделяемые библиотеки.pdf - Google Диск](#)
4. Файлы для тестирования плагинов-библиотек – 2021. – URL: [lab1test – Google Диск](#)

ПРИЛОЖЕНИЕ А

(обязательное)

Исходные коды с комментариями

Листинг А.1 – libmsN3246.c

```
//
// gcc -shared -fPIC -Wall -Wextra -Werror -O3 libmsN3246.c -o libmsN3246.so
//

#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include "plugin_api.h"

#define CARD_BYTES 8 // 8 байт = 16 BCD-цифр (по 4 бита на цифру)

static char *g_lib_name = "libmsN3246.so";
static char *g_plugin_purpose = "Поиск файлов с корректными номерами кредитных карт в бинарной форме";
static char *g_plugin_author = "Суханкулиев Мухаммет, N3246";

static struct plugin_option g_po_arr[] = {
    {
        {
            "ccards-bin", no_argument, 0, 0
        },
        "Поиск корректных номеров кредитных карт в бинарной форме"
    }
};

static int g_po_arr_len = sizeof(g_po_arr) / sizeof(g_po_arr[0]);

//
// Приватные функции
//
static inline int is_debug_mode(void) {
    return getenv("LAB1DEBUG") != NULL;
}

static int is_valid_bcd(const unsigned char *buf, size_t len) {
    for (size_t i = 0; i < len; ++i) {
        unsigned char hi = (buf[i] & 0xF0) >> 4;
        unsigned char lo = buf[i] & 0x0F;
        if (hi > 9 || lo > 9)
            return 0;
    }
    return 1;
}

static int is_valid_luhn(const unsigned char *buf) {
    unsigned char digits[16];
    for (size_t i = 0; i < CARD_BYTES; ++i) {
        digits[2 * i] = (buf[i] & 0xF0) >> 4;
        digits[2 * i + 1] = buf[i] & 0x0F;
    }

    int sum = 0;
    for (int i = 15; i >= 0; --i) {
        int digit = digits[i];
        if ((15 - i) % 2 == 1) {
            digit *= 2;
            if (digit > 9) digit -= 9;
        }
    }
}
```

```

        sum += digit;
    }
    return (sum % 10 == 0);
}

static int has_too_many_zeros(const unsigned char *buf) {
    int zero_count = 0;
    for (size_t i = 0; i < CARD_BYTES; ++i) {
        if ((buf[i] & 0xF0) >> 4) == 0) ++zero_count;
        if ((buf[i] & 0x0F) == 0) ++zero_count;
    }
    return zero_count >= 10;
}

static int starts_with_zero(const unsigned char *buf) {
    return ((buf[0] & 0xF0) >> 4) == 0;
}

static void print_card_number(FILE *stream, const unsigned char *buf) {
    for (int i = 0; i < CARD_BYTES; ++i) {
        fprintf(stream, "%X%X", (buf[i] & 0xF0) >> 4, buf[i] & 0x0F);
    }
    fputc('\n', stream);
}

static int is_valid_card(const unsigned char *buf) {
    return is_valid_bcd(buf, CARD_BYTES) &&
        !has_too_many_zeros(buf) &&
        !starts_with_zero(buf) &&
        is_valid_luhn(buf);
}

//
// API функции
//
int plugin_get_info(struct plugin_info* ppi) {
    if (!ppi) {
        fprintf(stderr, "[%s] ERROR: Неверный аргумент\n", g_lib_name);
        return -1;
    }

    ppi->plugin_purpose = g_plugin_purpose;
    ppi->plugin_author = g_plugin_author;
    ppi->sup_opts_len = g_po_arr_len;
    ppi->sup_opts = g_po_arr;

    return 0;
}

int plugin_process_file(const char *fname, struct option in_opts[], size_t
in_opts_len) {
    // Проверка, включён ли нужный флаг
    int enabled = 0;
    for (size_t i = 0; i < in_opts_len; ++i) {
        if (strcmp(in_opts[i].name, "ccards-bin") == 0) {
            enabled = 1;
            break;
        }
    }
    if (!enabled) return 0;

    // Открытие файла
    FILE *file = fopen(fname, "rb");
    if (!file) {
        fprintf(stderr, "[%s] ERROR: Не удалось открыть файл '%s': %s\n", g_lib_name,
fname, strerror(errno));
        return -1;
    }
}

```



```

unsigned char buffer[CARD_BYTES];
unsigned char reversed[CARD_BYTES];
size_t offset = 0;
int found = 0;
int debug = is_debug_mode();

// Основная обработка файла
while (fread(buffer, 1, CARD_BYTES, file) == CARD_BYTES) {
    // Пропуск полностью нулевых блоков
    if (memcmp(buffer, "\0\0\0\0\0\0\0\0", CARD_BYTES) == 0) {
        ++offset;
        fseek(file, offset, SEEK_SET);
        continue;
    }

    // Подготовка реверсированного буфера
    for (int i = 0; i < CARD_BYTES; ++i)
        reversed[i] = buffer[CARD_BYTES - 1 - i];

    int valid_le = is_valid_card(buffer);
    int valid_be = is_valid_card(reversed);

    if (debug && (valid_le || valid_be)) {
        fprintf(stderr, "[%s] DEBUG: Offset %zu: ", g_lib_name, offset);
        if (valid_le && valid_be) {
            fprintf(stderr, "LE & BE valid: ");
            print_card_number(stderr, buffer);
        } else if (valid_le) {
            fprintf(stderr, "LE valid: ");
            print_card_number(stderr, buffer);
        } else {
            fprintf(stderr, "BE valid: ");
            print_card_number(stderr, reversed);
        }
    }

    if (valid_le || valid_be)
        ++found;

    ++offset;
    fseek(file, offset, SEEK_SET); // сдвиг на один байт
}

if (ferror(file)) {
    fprintf(stderr, "[%s] ERROR: Ошибка чтения файла '%s': %s\n", g_lib_name,
fname, strerror(errno));
    fclose(file);
    return -1;
}

fclose(file);

if (debug){
    printf("[%s] DEBUG: Найдено корректных номеров: %d в файле %s\n", g_lib_name,
found, fname);
}
return (found > 0) ? 0 : 1;
}

```

Листинг А.2 – main.c

```

#include "utils.h"

int main(int argc, char *argv[]) {
    if (getenv("LAB1DEBUG")) debug_enabled = 1;
    int ret = EXIT_SUCCESS;
    // Каталог плагинов: по умолчанию каталог исполняемого файла

```

```

char exe_path[PATH_MAX];
ssize_t len = readlink("/proc/self/exe", exe_path, sizeof(exe_path) - 1);
if (len != -1) {
    exe_path[len] = '\0';
    char *last_slash = strrchr(exe_path, '/');
    if (last_slash) *last_slash = '\0';
    else strcpy(exe_path, ".");
    plugin_dir = strdup(exe_path);
} else { plugin_dir = strdup("."); }

// Первичный парсинг базовых опций
if (parse_command_line(argc, argv) != 0) { free(plugin_dir); return EXIT_FAILURE;
}

if (show_version) { print_version(); free(plugin_dir); return EXIT_SUCCESS; }
if (show_help || !start_dir) { print_help(); free(plugin_dir); return
EXIT_SUCCESS; }

// Собираем все --длинные опции из argv[]
for (int i = 1; i < argc; i++) {
    if (strncmp(argv[i], "--", 2) == 0) {
        char *eq = strchr(argv[i] + 2, '=');
        size_t len = eq ? (size_t)(eq - (argv[i] + 2)) : strlen(argv[i] + 2);
        if (passed_long_opts_count < MAX_PASSED_OPTS) {
            passed_long_opts[passed_long_opts_count++] = strndup(argv[i] + 2,
len);
        }
    }
}

// Загрузка плагинов
if (load_plugins(plugin_dir) != 0) { free(plugin_dir); return EXIT_FAILURE; }
// Сразу показываем, какие плагины загрузились
print_loaded_plugins();
// Сбор всех опций из плагинов
if (collect_all_plugin_options() != 0) { unload_plugins(); free(plugin_dir);
return EXIT_FAILURE; }
// Проверка на конфликты между опциями
if (check_option_conflicts() != 0) { unload_plugins(); free(all_plugin_options);
free(plugin_dir); return EXIT_FAILURE; }
// Массив указателей на значения опций плагинов
char **plugin_opt_args = calloc(all_plugin_options_count, sizeof(char *));
if (!plugin_opt_args) {
    fprintf(stderr, "Memory allocation failed\n");
    unload_plugins();
    free(all_plugin_options);
    free(plugin_dir);
    return EXIT_FAILURE;
}
// Подготовка объединённого списка опций: базовые + плагиновые
size_t base_opts_len = 6;
struct option *combined_options = calloc(base_opts_len + all_plugin_options_count
+ 1, sizeof(struct option));
if (!combined_options) {
    fprintf(stderr, "Memory allocation failed\n");
    free(plugin_opt_args);
    unload_plugins();
    free(all_plugin_options);
    free(plugin_dir);
    return EXIT_FAILURE;
}
struct option base_options[] = {
    {"P",    required_argument, NULL, 'P'},
    {"A",    no_argument,      NULL, 'A'},
    {"O",    no_argument,      NULL, 'O'},
    {"N",    no_argument,      NULL, 'N'},
    {"v",    no_argument,      NULL, 'v'},
    {"h",    no_argument,      NULL, 'h'}
};

```

```

memcpy(combined_options, base_options, sizeof(base_options));
for (size_t i = 0; i < all_plugin_options_count; i++) {
    combined_options[base_opts_len + i] = all_plugin_options[i];
    combined_options[base_opts_len + i].val = 256 + (int)i;
}

// Повторный парсинг опций с учётом опций плагинов
optind = 1;
int opt, option_index;
int count_A = 0, count_O = 0;
while ((opt = getopt_long(argc, argv, "P:AONvh", combined_options, &option_index))
!= -1) {
    size_t plugin_idx = opt - 256;
    if (plugin_idx < all_plugin_options_count) {
        if (combined_options[base_opts_len + plugin_idx].has_arg ==
required_argument) {
            if (!optarg) {
                fprintf(stderr, "Missing argument for --%s\n",
combined_options[base_opts_len + plugin_idx].name);
                goto fail;
            }
            plugin_opt_args[plugin_idx] = strdup(optarg);
        } else { plugin_opt_args[plugin_idx] = strdup("1"); }
    }
}

if (count_A > 0 && count_O > 0) {
    fprintf(stderr, "Error: -A and -O cannot be used together\n");
    goto fail;
}
if (optind < argc) start_dir = argv[optind];
int passed_long = 0;
for (int i = 1; i < optind; i++) {
    if (strncmp(argv[i], "--", 2) == 0) {
        passed_long++;
    }
}
// подсчитать, сколько из них (опций) распознано
int recognized = 0;
for (size_t i = 0; i < all_plugin_options_count; i++) {
    if (plugin_opt_args[i] != NULL) {
        recognized++;
    }
}
if (passed_long > recognized) {
    fprintf(stderr,
        "Error: unsupported option(s): passed %d plugin options, but only %d
supported\n",
        passed_long, recognized);
    goto fail;
}
for (size_t i = 0; i < all_plugin_options_count; i++) {
    if (all_plugin_options[i].has_arg == required_argument && (!plugin_opt_args[i]
|| plugin_opt_args[i][0] == '\0')) {
        fprintf(stderr, "Missing argument for plugin(s)");
        goto fail;
    }
    all_plugin_options[i].flag = (int *)plugin_opt_args[i];
}

fprintf(stderr, "Parsed options:\n");
if (plugin_dir)    fprintf(stderr, "  -P plugin_dir = %s\n", plugin_dir);
fprintf(stderr, "  -A = %s\n", combine_and ? "true" : "false");
fprintf(stderr, "  -O = %s\n", combine_or ? "true" : "false");
fprintf(stderr, "  -N = %s\n", invert_result ? "true" : "false");
fprintf(stderr, "  -v = %s\n", show_version ? "true" : "false");
fprintf(stderr, "  -h = %s\n", show_help ? "true" : "false");

```

```

    for (size_t i = 0; i < all_plugin_options_count; ++i) {
        const char *name = all_plugin_options[i].name;
        const char *val = plugin_opt_args[i];
        if (val) {
            fprintf(stderr, " --%s = %s\n", name, val);
        } else { // no_argument
            // опция-флаг: покажем явно, даже если val == NULL
            fprintf(stderr, " --%s = %s\n", name, "null");
        }
    }
    fprintf(stderr, "\n");

    if (optind < argc) start_dir = argv[optind]; // развернуть start_dir в абсолютный
путь
    {
        char buf[PATH_MAX];
        if (!realpath(start_dir, buf)) {
            fprintf(stderr, "Error: cannot resolve '%s': %s\n",
                start_dir, strerror(errno));
            goto fail;
        }
        start_dir = strdup(buf);
    }
    // Запуск поиска
    int result = recursive_search(start_dir, all_plugin_options,
all_plugin_options_count);
    if (result != 0)
        ret = EXIT_FAILURE;

cleanup:
    for (size_t i = 0; i < passed_long_opts_count; ++i) {
        free(passed_long_opts[i]);
    }
    unload_plugins();
    if (plugin_opt_args) {
        for (size_t i = 0; i < all_plugin_options_count; i++)
            free(plugin_opt_args[i]);
        free(plugin_opt_args);
    }
    free(combined_options);
    free(all_plugin_options);
    free(plugin_dir);
    if (start_dir && start_dir != argv[optind]) free(start_dir);
    return ret;

fail:
    ret = EXIT_FAILURE;
    goto cleanup;
}

```

Листинг A.3 – utils.c

```

#include "utils.h"
#include <ftw.h>
#include <dirent.h>
#include <dlfcn.h>
#include <stdarg.h>

// Определения extern-переменных
char *plugin_dir = NULL;
int combine_and = 1, combine_or = 0, invert_result = 0;
int show_version = 0, show_help = 0;
char *start_dir = NULL;
int debug_enabled = 0;

plugin_t plugins[MAX_PLUGINS];
size_t plugins_count = 0;

```

```

struct option *all_plugin_options = NULL;
size_t all_plugin_options_count = 0;

struct option *global_opts = NULL;
size_t global_opts_len = 0;

char *passed_long_opts[MAX_PASSED_OPTS];
size_t passed_long_opts_count = 0;

// Проверяет, есть ли у плагина хотя бы одна опция из passed_long_opts[]
static int plugin_matches_passed_opts(const struct plugin_info *pi) {
    for (size_t i = 0; i < pi->sup_opts_len; i++) {
        const char *name = pi->sup_opts[i].opt.name;
        for (size_t j = 0; j < passed_long_opts_count; j++) {
            if (strcmp(name, passed_long_opts[j]) == 0)
                return 1;
        }
    }
    return 0;
}

// Вывод отладочных сообщений, если задана LAB1DEBUG
void debug_printf(const char *fmt, ...) {
    if (!debug_enabled) return;
    va_list ap;
    va_start(ap, fmt);
    vfprintf(stderr, fmt, ap);
    va_end(ap);
}

void print_version(void) {
    printf("lab12msN3246 version 1.1\n");
    printf("Author: Суханкулиев Мухаммет\nGroup: N3246\nПоток: ОСП N23 1.2\nVariant: 24\n");
}

// Вывод справки по опциям и плагинам
void print_help(void) {
    printf("Usage: lab12msN3246 [options] [directory]\n");
    printf("Options:\n");
    printf("  -P dir    Указать каталог с плагинами\n");
    printf("  -A        Объединение условий по AND (по умолчанию)\n");
    printf("  -O        Объединение условий по OR\n");
    printf("  -N        Инвертировать итоговое условие\n");
    printf("  -v        Show author information\n");
    printf("  -h        Show this help message\n\n");
}

// Вывод загруженных плагинов и их описаний
void print_loaded_plugins(void) {
    fprintf(stderr, "Loaded %zu plugin(s):\n", plugins_count);
    for (size_t i = 0; i < plugins_count; i++) {
        struct plugin_info *pi = &plugins[i].info;
        fprintf(stderr, "  Plugin [%s]\n", plugins[i].path);
        fprintf(stderr, "    Purpose: %s\n", pi->plugin_purpose);
        fprintf(stderr, "    Author : %s\n", pi->plugin_author);
        for (size_t j = 0; j < pi->sup_opts_len; j++) {
            fprintf(stderr, "      --%s : %s\n", pi->sup_opts[j].opt.name, pi->sup_opts[j].opt_descr);
        }
    }
}

// Загрузка плагинов из указанного каталога
int load_plugins(const char *dir) {
    DIR *dp = opendir(dir);

```

```

if (!dp) {
    fprintf(stderr, "Cannot open plugin dir '%s': %s\n", dir, strerror(errno));
    return -1;
}

struct dirent *entry;
plugins_count = 0;
while ((entry = readdir(dp)) != NULL) {
    if (plugins_count >= MAX_PLUGINS) break;
    size_t len = strlen(entry->d_name);
    if (len <= 3 || strcmp(entry->d_name + len - 3, ".so") != 0) continue;

    char fullpath[MAX_PATH_LEN];
    snprintf(fullpath, sizeof(fullpath), "%s/%s", dir, entry->d_name);

    void *handle = dlopen(fullpath, RTLD_LAZY);
    if (!handle) {
        fprintf(stderr, "dlopen failed '%s': %s\n", fullpath, dlerror());
        continue;
    }

    int (*get_info)(struct plugin_info*) = dlsym(handle, "plugin_get_info");
    if (!get_info) { dlclose(handle); continue; }

    struct plugin_info pi;
    if (get_info(&pi) < 0) { dlclose(handle); continue; }

    // **Фильтрация по passed_long_opts**
    if (passed_long_opts_count > 0 && !plugin_matches_passed_opts(&pi)) {
        debug_printf("Skipping plugin %s: no matching --opts\n", fullpath);
        dlclose(handle);
        continue;
    }

    // Только после проверки сохраняем плагин
    plugins[plugins_count].handle = handle;
    plugins[plugins_count].info = pi;
    plugins[plugins_count].path = strdup(fullpath);
    plugins_count++;
}
closedir(dp);
if (plugins_count == 0) {
    fprintf(stderr, "No valid plugins loaded from '%s'\n", dir);
    return -1;
}
return 0;
}

// Выгрузка плагинов
void unload_plugins(void) {
    for (size_t i = 0; i < plugins_count; i++) {
        if (plugins[i].handle) dlclose(plugins[i].handle);
        free(plugins[i].path);
    }
    plugins_count = 0;
}

// Сбор всех опций от плагинов
int collect_all_plugin_options(void) {
    size_t total_opts = 0;
    for (size_t i = 0; i < plugins_count; i++)
        total_opts += plugins[i].info.sup_opts_len;
    if (total_opts == 0) {
        fprintf(stderr, "No plugin options found\n");
        return -1;
    }
    all_plugin_options = calloc(total_opts + 1, sizeof(struct option)); // +1 для
завершающего {0}
    if (!all_plugin_options) {

```

```

        fprintf(stderr, "Memory allocation failed\n");
        return -1;
    }
    size_t idx = 0;
    for (size_t i = 0; i < plugins_count; i++)
        for (size_t j = 0; j < plugins[i].info.sup_opts_len; j++)
            all_plugin_options[idx++] = plugins[i].info.sup_opts[j].opt;
    all_plugin_options[total_opts] = (struct option){0}; // Завершающий нулевой
элемент
    all_plugin_options_count = total_opts;
    return 0;
}

// Проверка на конфликты опций
int check_option_conflicts(void) {
    for (size_t i = 0; i < all_plugin_options_count; i++)
        for (size_t j = i + 1; j < all_plugin_options_count; j++)
            if (strcmp(all_plugin_options[i].name, all_plugin_options[j].name) == 0) {
                fprintf(stderr, "Option name conflict: --%s\n",
all_plugin_options[i].name);
                return -1;
            }
    return 0;
}

int parse_command_line(int argc, char *argv[]) {
    static struct option base_options[] = {
        {"P", required_argument, NULL, 'P'},
        {"A", no_argument, NULL, 'A'},
        {"O", no_argument, NULL, 'O'},
        {"N", no_argument, NULL, 'N'},
        {"v", no_argument, NULL, 'v'},
        {"h", no_argument, NULL, 'h'},
        {0, 0, 0, 0}
    };
    int opt, option_index;
    opterr = 0;
    int count_A = 0, count_O = 0;
    // Разбираем только базовые опции, неизвестные игнорируем
    while ((opt = getopt_long(argc, argv, "P:AONvh", base_options, &option_index)) !=
-1) {
        switch (opt) { case 'P': free(plugin_dir); plugin_dir = strdup(optarg); break;
            case 'A': combine_and = 1; combine_or = 0; count_A++; break;
            case 'O': combine_or = 1; combine_and = 0; count_O++; break;
            case 'N': invert_result = 1; break;
            case 'v': show_version = 1; break;
            case 'h': show_help = 1; break;
            case '?': if (optopt != 0) { fprintf(stderr, "Unknown option: -%c\n",
optopt); print_help(); return -1;
                } else { break; }
            default: fprintf(stderr, "Unknown option\n"); return -1;
        }
    }
    if (count_A > 0 && count_O > 0) {
        fprintf(stderr, "Error: options -A and -O cannot be used together\n");
        return -1;
    }
    if (optind < argc) start_dir = argv[optind];
    return 0;
}

// Обход файловой системы: вызывается для каждого файла/каталога
int nftw_callback(const char *fpath, const struct stat *sb, int typeflag, struct FTW
*ftwbuf) {
    (void) sb; (void) ftwbuf;
    if (typeflag == FTW_SL) {
        debug_printf("Ignoring symlink: %s\n", fpath);
        return 0;
    }

```

```

    }
    if (typeflag == FTW_F) {
        int match = 1;
        // Обработка файла каждым подключённым плагином
        for (size_t i = 0; i < plugins_count; i++) {
            int (*plugin_process_file)(const char *, struct option *, size_t) =
                dlsym(plugins[i].handle, "plugin_process_file");
            if (!plugin_process_file) {
                fprintf(stderr, "plugin_process_file not found in plugin '%s'\n",
plugins[i].path);
                return -1;
            }
            // Выбор только тех опций, которые поддерживает данный плагин
            size_t plugin_opts_len = plugins[i].info.sup_opts_len;
            struct option *plugin_opts = calloc(plugin_opts_len, sizeof(struct
option));
            if (!plugin_opts) {
                fprintf(stderr, "Memory allocation failed\n");
                return -1;
            }
            size_t count = 0;
            for (size_t j = 0; j < global_opts_len; j++) {
                for (size_t k = 0; k < plugin_opts_len; k++) {
                    if (strcmp(global_opts[j].name,
plugins[i].info.sup_opts[k].opt.name) == 0) {
                        plugin_opts[count++] = global_opts[j];
                        break;
                    }
                }
            }
            // Вызов функции плагина
            int pres = plugin_process_file(fpath, plugin_opts, count);
            free(plugin_opts);

            if (pres < 0) {
                if (debug_enabled)
                    fprintf(stderr, "Plugin '%s' error on '%s': %s\n",
plugins[i].path, fpath, strerror(errno));
                return -1;
            }
            if (combine_and)
                match = match && (pres == 0);
            else if (combine_or)
                match = match || (pres == 0);
        }
        if (invert_result)
            match = !match;
        if (match)
            printf("%s\n", fpath);
    }
    return 0;
}

// Запуск обхода с использованием nftw()
int recursive_search(const char *dir_path, struct option *opts, size_t opts_len) {
    global_opts = opts;
    global_opts_len = opts_len;
    int flags = FTW_PHYS; // Не следовать по символическим ссылкам
    int max_fd = 20; // Максимум одновременно открытых файловых дескрипторов
    return nftw(dir_path, nftw_callback, max_fd, flags);
}

```

Листинг А.4 – utils.h

```

#ifndef UTILS_H
#define UTILS_H

#include <stdio.h>

```



```

#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <limits.h>
#include <getopt.h>
#include <sys/types.h>
#include "plugin_api.h"

#define MAX_PLUGINS    64
#define MAX_PATH_LEN  PATH_MAX

typedef struct {
    void          *handle; // дескриптор из dlopen()
    struct plugin_info info; // данные, полученные plugin_get_info()
    char          *path;    // полный путь до .so-файла
} plugin_t;

// Массив имён всех --длинных опций, переданных пользователем
#define MAX_PASSED_OPTS 128
extern char *passed_long_opts[MAX_PASSED_OPTS];
extern size_t passed_long_opts_count;

//
// Глобальные параметры утилиты (extern-переменные)
//
extern char *plugin_dir; // каталог для поиска плагинов
extern int  combine_and; // -A (AND) режим объединения плагиновых условий
extern int  combine_or;  // -O (OR) режим
extern int  invert_result; // -N инверсия итогового результата
extern int  show_version; // -v вывод версии и автора
extern int  show_help;    // -h вывод справки
extern char *start_dir;   // начальный каталог для поиска
extern int  debug_enabled; // LAB1DEBUG вкл/выкл отладку

//
// Массив загруженных плагинов
//
extern plugin_t  plugins[MAX_PLUGINS];
extern size_t    plugins_count;

//
// Собранные опции всех плагинов (после collect_all_plugin_options)
//
extern struct option *all_plugin_options;
extern size_t        all_plugin_options_count;

// Отладочный вывод (использует debug_enabled)
void debug_printf(const char *fmt, ...);

// Информационные сообщения утилиты
void print_version(void);
void print_help(void);
void print_loaded_plugins(void);

// Парсинг только коротких базовых опций
int parse_command_line(int argc, char *argv[]);

// Загрузка/выгрузка плагинов из plugin_dir
int load_plugins(const char *dir);
void unload_plugins(void);

// Сбор опций из каждого плагина в единый массив
int collect_all_plugin_options(void);

// Проверка дублирования имён опций разных плагинов
int check_option_conflicts(void);

```

```
// Запуск рекурсивного поиска с переданными опциями
int recursive_search(const char *dir_path,
                    struct option *opts,
                    size_t      opts_len);

#endif // UTILS_H
```