

Министерство образования и науки Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение высшего  
профессионального образования  
«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И  
РАДИОЭЛЕКТРОНИКИ» (ТУСУР)  
Кафедра комплексной информационной безопасности электронно-вычислительных систем  
(КИБЭВС)

УТВЕРЖДАЮ

заведующий каф. КИБЭВС

\_\_\_\_\_ А.А. Шелупанов

«\_\_\_\_\_» \_\_\_\_\_ 2016г.

КОМПЬЮТЕРНАЯ ЭКСПЕРТИЗА

Отчет по групповому проектному обучению

Группа КИБЭВС-1401

Ответственный исполнитель

студент гр. 722

\_\_\_\_\_ О.В. Лобанов

«\_\_\_\_\_» \_\_\_\_\_ 2016г.

Научный руководитель

аспирант каф. КИБЭВС

\_\_\_\_\_ А.И. Гуляев

«\_\_\_\_\_» \_\_\_\_\_ 2016г.

## РЕФЕРАТ

Курсовая работа содержит 73 страниц, 61 рисунок, 0 таблицы, 12 источников, 4 приложение.

КОМПЬЮТЕРНАЯ ЭКСПЕРТИЗА, ФОРЕНЗИКА, ЛОГИ, QT, XML, GIT, GITLAB, LATEX, MOZILLA THUNDERBIRD, MOZILLA FIREFOX, MS OUTLOOK, WINDOWS, HTML5, CSS3, БИБЛИОТЕКИ, РЕПОЗИТОРИЙ, ПОЧТОВЫЙ КЛИЕНТ, МЕТА-ДАННЫЕ, ID3, JFIF, RIFF, C++, ISSUE, NGINX, GUI, BASH, APACHE, UNIT-ТЕСТИРОВАНИЕ.

Цель работы — создание программного комплекса, предназначенного для проведения компьютерной экспертизы.

Среди задач, поставленных на данный семестр, было:

- определение индивидуальных задач для каждого участника проектной группы;
- исследование предметных областей в рамках индивидуальных задач;
- создание репозитория проекта;
- дизайн, верстка и развертывание сайта проекта;
- сборка программного пакета проекта;
- доработка программных модулей.

Результаты работы в данном семестре:

- доработан программный модуль, определяющий ОС;
- разработан графический интерфейс пользователя системы;
- доработан программный модуль, осуществляющий нахождение медиа-файлов;
- доработан программный модуль для сбора истории посещений браузера Mozilla Firefox;
- собран установочный .deb-пакет системы компьютерной экспертизы;
- доработан программный модуль для сбора информации из почтового клиента MS Outlook;
- созданы удаленный репозиторий и сайт проекта;
- проведено Unit-тестирование в инструментарии Qt на примере модуля, сканирующего медиа-файлы;
- внесены поправки, изменения и доработки в исходный код проекта.

Пояснительная записка выполнена при помощи системы компьютерной вёрстки L<sup>A</sup>T<sub>E</sub>X.

## Список исполнителей

### **ПРАВИТЬ!!!**

Кучер М.В. – программист, ответственный за разработку программного модуля для извлечения информации о пользовательских аккаунтах в мессенджере Viber.

Мейта М.В. – программист, документатор, ответственный за доработку архитектуры проекта, а также верстку необходимой документации в системе L<sup>A</sup>T<sub>E</sub>X.

Терещенко Ю.А. – программист, ответственный за написание программного модуля для сбора сохраненных логинов и паролей браузера Mozilla Firefox.

Шиповской В.В. – программист, ответственный за доработку графического интерфейса пользователя системы «СОЕХ» и анализ данных, хранимых в NoSql БД Apache Solr.

Боков И.М. – программист, ответственный за написание части системы для для нахождения медиа-файлов (аудио, видео, изображение) и извлечения мета-данных из них.

Лобанов О.В. – программист, ответственный за разработку сайта проекта.

Серяков А.В. – программист, ответственный за написание части системы для сбора информации из почтового клиента MS Outlook и создание программного пакета с исходными файлами системы «СОЕХ».

## Содержание

Введение . . . . .	9
1 Назначение и область применения . . . . .	9
2 Постановка задачи . . . . .	9
3 Инструменты . . . . .	10
3.1 Система контроля версий Git . . . . .	10
3.2 Система компьютерной вёрстки TeX . . . . .	10
3.3 Язык разметки Markdown . . . . .	11
3.4 Qt - кроссплатформенный инструментарий разработки ПО . . . . .	11
3.4.1 Автоматизация поиска журнальных файлов . . . . .	13
3.4.2 Реализация сохранения результатов работы программного комплекса в XML . . . . .	13
3.5 GitHub . . . . .	13
4 Технические характеристики . . . . .	14
4.1 Требования к аппаратному обеспечению . . . . .	14
4.2 Требования к программному обеспечению . . . . .	14
4.3 Выбор единого формата выходных файлов . . . . .	14
5 Разработка программного обеспечения . . . . .	16
5.1 Архитектура . . . . .	16
5.1.1 Основной алгоритм . . . . .	16
5.1.2 Описание основных функций модуля системы . . . . .	18
5.2 Плагин TaskViber . . . . .	19
5.2.1 Расположения файлов мессенджера Viber . . . . .	19
5.2.2 Описание содержимого папки «ViberPC» . . . . .	19
5.2.3 SQL-запросы для получения информации . . . . .	19
5.2.4 Описание плагина . . . . .	21
5.3 Сбор сохраненных логинов и паролей браузера Mozilla Firefox . . . . .	26
5.3.1 Алгоритм работы программного модуля . . . . .	27
5.3.2 Тестирование . . . . .	27
5.4 Графический интерфейс пользователя системы «COEX» . . . . .	30
5.5 Анализ данных с помощью Apache Solr . . . . .	38
5.5.1 Общая информация об Apache Lucene, Solr . . . . .	38
5.5.2 Подготовка окружения и установка Apache Lucene . . . . .	38
5.5.3 Добавление документов в поисковый индекс . . . . .	39
5.5.4 Формирование запросов . . . . .	40
5.6 Создание «бинарного» пакета DEB из исходных файлов программного комплекса «COEX» . . . . .	49
5.7 Документирование плагинов . . . . .	52
5.8 Разработка и внедрение копии жесткого диска . . . . .	54
5.9 Многопоточное программирование . . . . .	61
5.9.1 Сигналы и слоты . . . . .	61
5.9.2 Потoki QThreads . . . . .	62

5.9.3 Итоги работы за семестр . . . . .	66
Заключение . . . . .	67
Приложение А Компакт-диск . . . . .	69
Приложение Б Md to html script . . . . .	70
Приложение В Hdd class . . . . .	71
Приложение Г Disk usage logging script . . . . .	73

## Введение

Компьютерно-техническая экспертиза – это самостоятельный род судебных экспертиз, относящийся к классу инженерно-технических экспертиз, проводимых в следующих целях: определения статуса объекта как компьютерного средства, выявление и изучение его роли в рассматриваемом деле, а так же получения доступа к информации на электронных носителях с последующим всесторонним её исследованием. [1]

Компьютерная экспертиза помогает получить доказательственную информацию и установить факты, имеющие значение для уголовных, гражданских и административных дел, сопряжённых с использованием компьютерных технологий. Для проведения компьютерных экспертиз необходима высокая квалификация экспертов, так как при изучении представленных носителей информации, попытке к ним доступа и сбора информации возможно внесение в информационную среду изменений или полная утрата важных данных.

Компьютерная экспертиза, в отличие от компьютерно-технической экспертизы, затрагивает только информационную составляющую, в то время как аппаратная часть и её связь с программной средой не рассматривается.

На протяжении предыдущих семестров разработчиками данного проекта были рассмотрены такие направления компьютерной экспертизы, как исследование файловых систем, сетевых протоколов, организация работы серверных систем, механизм журналирования событий. Также были изучены основные задачи, которые ставятся перед сотрудниками правоохранительных органов, проводящими компьютерную экспертизу, и набор существующих утилит, способных помочь эксперту в проведении компьютерной экспертизы. Было выявлено, что существует множество разрозненных программ, предназначенных для просмотра лог-файлов системы и таких приложений, как мессенджеры и браузеры, но для каждого вида лог-файлов необходимо искать отдельную программу. Так как ни одна из них не позволяет эксперту собрать воедино и просмотреть все логи системы, браузеров и мессенджеров, было решено создать для этой цели собственный автоматизированный комплекс, не имеющий на данный момент аналогов в РФ.

### 1 Назначение и область применения

Разрабатываемый комплекс предназначен для автоматизации процесса сбора информации с исследуемого образа жёсткого диска.

### 2 Постановка задачи

На данный семестр были поставлены следующие задачи:

Цель: получить информацию из мессенджера Viber.

Целью работы в текущем семестре стала переработка графического интерфейса и добавление функционала для удобства его эксплуатации.

Задача семестра — анализ данных, хранимых в NoSql БД Apache Solr.

- определение индивидуальных задач для каждого участника проектной группы;
- исследование предметных областей в рамках индивидуальных задач;
- создание репозитория проекта;

- дизайн, верстка и развертывание сайта проекта;
- сборка программного пакета проекта;
- доработка программных модулей.

### 3 Инструменты

#### 3.1 Система контроля версий Git

Для разработки программного комплекса для проведения компьютерной экспертизы было решено использовать Git.

Git — распределённая система управления версиями файлов. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux как противоположность системе управления версиями Subversion (также известная как «SVN»). [2]

При работе над одним проектом команде разработчиков необходим инструмент для совместного написания, бэкапирования и тестирования программного обеспечения. Используя Git, мы имеем:

- возможность удаленной работы с исходными кодами;
- возможность создавать свои ветки, не мешая при этом другим разработчикам;
- доступ к последним изменениям в коде, т.к. все исходники хранятся на сервере git.keva.su;
- исходные коды защищены, доступ к ним можно получить лишь имея RSA-ключ;
- возможность откатиться к любой стабильной стадии проекта.

Основные постулаты работы с кодом в системе Git:

- каждая задача решается в своей ветке;
- необходимо делать коммит как только был получен осмысленный результат;
- ветка master мерджится не разработчиком, а вторым человеком, который производит вычитку и тестирование изменения;
- все коммиты должны быть осмысленно подписаны/прокомментированы.

#### 3.2 Система компьютерной вёрстки T<sub>E</sub>X

T<sub>E</sub>X — это созданная американским математиком и программистом Дональдом Кнутот система для вёрстки текстов. Сам по себе T<sub>E</sub>X представляет собой специализированный язык программирования. Каждая издательская система представляет собой пакет макроопределений этого языка.

L<sup>A</sup>T<sub>E</sub>X — это созданная Лэсли Лэмпортом издательская система на базе T<sub>E</sub>X'a [3] L<sup>A</sup>T<sub>E</sub>X позволяет пользователю сконцентрировать свои усилия на содержании и структуре текста, не заботясь о деталях его оформления.

Для подготовки отчётной и иной документации нами был выбран L<sup>A</sup>T<sub>E</sub>X так как совместно с системой контроля версий Git он предоставляет возможность совместного создания и редактирования документов. Огромным достоинством системы L<sup>A</sup>T<sub>E</sub>X то, что создаваемые с её помощью файлы обладают высокой степенью переносимости. [4]

Совместно с L<sup>A</sup>T<sub>E</sub>X часто используется BibT<sub>E</sub>X — программное обеспечение для создания форматированных списков библиографии. Оно входит в состав дистрибутива L<sup>A</sup>T<sub>E</sub>X и позволяет создавать удобную, универсальную и долговечную библиографию. BibT<sub>E</sub>X стал одной из причин, по которой нами был выбран L<sup>A</sup>T<sub>E</sub>X для создания документации.

### 3.3 Язык разметки Markdown

Markdown — это язык форматирования (или язык описания форматирования), по принципу работы похожий на HTML, который используется для определения финального вида текста. Работает это следующим образом — в текстовое поле вводится текст, форматируя его специальными кодами (тегами) Markdown, которые при сохранении формы конвертируются в абсолютно валидный HTML. Преимущества использования Markdown:

- 1) Основан исключительно на текстовом вводе. Нет необходимости использовать сторонние редакторы и инструменты — только обычное текстовое поле. Таким образом можно быть уверенным, что текст отобразится корректно;
- 2) Минимальное количество кода. Всё, что нужно запомнить пользователю — это пара очевидных тегов и несколько правил;
- 3) Исходный код максимально читабелен. Редактируя страницу, можно наблюдать код Markdown и сразу понять, где находится заголовок, жирный текст или таблица. Всё очень наглядно;
- 4) Является open source языком, использующим BSD лицензию.

Недостатки Markdown:

- 1) Строгие конвенции, которые надо соблюдать;
- 2) Определенные символы нужно эскапировать. При необходимости ввода одного из технических символов, используемых в Markdown для разметки, перед ним надо ставить обратный слэш ("\").

### 3.4 Qt - кроссплатформенный инструментарий разработки ПО

Qt — это кроссплатформенная библиотека C++ классов для создания графических пользовательских интерфейсов (GUI) от фирмы Digia. Эта библиотека полностью объектно-ориентированная, что обеспечивает легкое расширение возможностей и создание новых компонентов. Ко всему прочему, она поддерживает огромное количество платформ.

Qt позволяет запускать написанное с его помощью ПО в большинстве современных операционных систем путём простой компиляции программы для каждой ОС без изменения исходного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML. Qt является полностью объектно-ориентированным, легко расширяемым и поддерживающим технику компонентного программирования.

Список использованных классов фреймворка QT

- iostream
- QChar
- QCryptographicHash
- QDateTime
- QDir
- QDirIterator
- QFile
- QFileInfo



- QIODevice
- QList
- QRegExp
- QString
- QTextStream
- QSql/QSqlDatabase
- QVector
- QMap
- QXmlStreamReader
- QXmlStreamWriter
- Conversations

Класс QXmlStreamWriter представляет собой XML писателя с простым потоковым.

Класс QXmlStreamReader представляет собой быстрый синтаксически корректный XML анализатор с простым потоковым API.

QVector представляет собой класс для создания динамических массивов.

Модуль QSql/QSqlDatabase помогает обеспечить однородную интеграцию БД в ваши Qt приложения.

Класс QTextStream предоставляет удобный интерфейс для чтения и записи текста.

QTextStream может взаимодействовать с QIODevice, QByteArray или QString. Используя потоковые операторы QTextStream, вы можете легко читать и записывать слова, строки и числа. При формировании текста QTextStream поддерживает параметры форматирования для заполнения и выравнивания полей и форматирования чисел. [5]

Класс QString предоставляет строку символов Unicode.

Класс QMap — контейнерный класс для хранения элементов различных типов данных.

Класс QDateTime используется для работы с форматом даты, в который записывается информация о файле.

QString хранит строку 16-битных QChar, где каждому QChar соответствует один символ Unicode 4.0. (Символы Unicode со значениями кодов больше 65535 хранятся с использованием суррогатных пар, т.е. двух последовательных QChar.)

Unicode - это международный стандарт, который поддерживает большинство используемых сегодня систем письменности. Это расширение US-ASCII (ANSI X3.4-1986) и Latin-1 (ISO 8859-1), где все символы US-ASCII/Latin-1 доступны на позициях с тем же кодом.

Внутри QString использует неявное совместное использование данных (копирование-приписи), чтобы уменьшить использование памяти и избежать ненужного копирования данных. Это также позволяет снизить накладные расходы, свойственные хранению 16-битных символов вместо 8-битных.

В дополнение к QString Qt также предоставляет класс QByteArray для хранения сырых байт и традиционных нультерминальных строк. В большинстве случаев QString - необходимый для использования класс. Он используется во всем API Qt, а поддержка Unicode гарантирует, что ваши приложения можно будет легко перевести на другой язык, если в какой-то момент вы захотите увеличить их рынок распространения. Два основных случая, когда уместно использование QByteArray: когда вам необходимо хранить сырые двоичные данные и когда критично использование памяти (на-

пример, в Qt для встраиваемых Linux-систем). [6]

Класс `QRegExp` предоставляет сопоставление с образцом при помощи регулярных выражений.

Регулярное выражение, или "regex", представляет собой образец для поиска соответствующей подстроки в тексте. Это полезно во многих ситуациях, например:

Проверка правильности – регулярное выражение может проверить, соответствует ли подстрока каким-либо критериям, например, целое ли она число или не содержит ли пробелов. Поиск – регулярное выражение предоставляет более мощные шаблоны, чем простое соответствие строки, например, соответствие одному из слов `mail`, `letter` или `correspondence`, но не словам `email`, `mailman`, `mailer`, `letterbox` и т.д. Поиск и замена – регулярное выражение может заменить все вхождения подстроки другой подстрокой, например, заменить все вхождения `&` на `&amp;`, исключая случаи, когда за `&` уже следует `amp;`. Разделение строки – регулярное выражение может быть использовано для определения того, где строка должна быть разделена на части, например, разделяя строку по символам табуляции.

`QFileInfo` - Во время поиска возвращает полную информацию о файле.

Класс `QDir` обеспечивает доступ к структуре каталогов и их содержимого.

`QIODevice` представляет собой базовый класс всех устройств ввода/вывода в Qt.

Класс `QCryptographicHash` предоставляет способ генерации криптографических хэшей. `QCryptographicHash` могут быть использованы для генерации криптографических хэшей двоичных или текстовых данных. В настоящее время MD4, MD5, и SHA-1 поддерживаются. [6]

`QChar` обеспечивает поддержку 16-битных символов Unicode.

### 3.4.1 Автоматизация поиска журнальных файлов

Для сканирования образа на наличие интересующих лог файлов использовался класс `QDirIterator`. После вызова происходит поочередный обход по каждому файлу в директории и поддиректории. Проверка полученного полного пути к файлу осуществляется регулярным выражением, если условие выполняется, происходит добавление в список обрабатываемых файлов.

### 3.4.2 Реализация сохранения результатов работы программного комплекса в XML

Сохранение полученных данных происходит в ранее выбранный формат XML (Extensible Markup Language). Для этого используется класс `QXmlStreamReader` и `QXmlStreamWriter`. Класс `QXmlStreamWriter` представляет XML писателя с простым потоковым API.

`QXmlStreamWriter` работает в связке с `QXmlStreamReader` для записи XML. Как и связанный класс, он работает с `QIODevice`, определённым с помощью `setDevice()`.

Сохранение данных реализовано в классе `WriteMessage`. В методе `WriteMessages`, структура которого представлена на UML диаграмме в разделе Архитектура.

## 3.5 GitHub

GitHub — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. Основан на системе контроля версий Git и разработан на Ruby on Rails и Erlang компанией GitHub, Inc (ранее Logical Awesome).

Сервис абсолютно бесплатен для проектов с открытым исходным кодом и предоставляет им все возможности (включая SSL), а для частных проектов предлагаются различные платные тарифные планы.[7]

Создатели сайта называют GitHub «социальной сетью для разработчиков». Кроме размещения кода, участники могут общаться, комментировать правки друг друга, а также следить за новостями знакомых. С помощью широких возможностей Git программисты могут объединять свои репозитории — GitHub предлагает удобный интерфейс для этого и может отображать вклад каждого участника в виде дерева.

Для проектов есть личные страницы, небольшие Вики и система отслеживания ошибок. Прямо на сайте можно просмотреть файлы проектов с подсветкой синтаксиса для большинства языков программирования. На платных тарифных планах можно создавать приватные репозитории, доступные ограниченному кругу пользователей.

Начиная с 5 декабря 2012 года, на сервисе добавлена возможность прямого добавления новых файлов в свой репозиторий через веб-интерфейс сервиса.

Код проектов можно не только скопировать через Git, но и скачать в виде обычных архивов с сайта. (Для этого достаточно добавить /zipball/master/ в конец адресной строки.)

Кроме Git, сервис поддерживает получение и редактирование кода через SVN и Mercurial.

Для работы над проектом «СОЕХ» проектной группой был поднят собственный репозиторий на сервере github.com.

Исходные файлы проекта и файлы для тестирования можно найти здесь:

<https://github.com/tusur-coex>.

## 4 Технические характеристики

### 4.1 Требования к аппаратному обеспечению

Минимальные системные требования:

- процессор 1ГГц Pentium 4;
- оперативная память 512 Мб;
- место на жёстком диске – 9 Гб.

### 4.2 Требования к программному обеспечению

Для корректной работы разрабатываемого программного комплекса на компьютере должна быть установлена операционная система Debian Squeeze или выше, данная система должна иметь набор библиотек QT.

### 4.3 Выбор единого формата выходных файлов

Для вывода результата был выбран формат XML-документов, так как с данным форматом легко работать при помощи программ, а результат работы данного комплекса в дальнейшем планируется обрабатывать при помощи программ.

XML - eXtensible Markup Language или расширяемый язык разметки. Язык XML представляет собой простой и гибкий текстовый формат, подходящий в качестве основы для создания новых

языков разметки, которые могут использоваться в публикации документов и обмене данными. [8] Задумка языка в том, что он позволяет дополнять данные метаданными, которые разделяют документ на объекты с атрибутами. Это позволяет упростить программную обработку документов, так как структурирует информацию.

Простейший XML-документ может выглядеть так:

```
<?xml version="1.0"?>
<list_of_items>
<item id="1"\><first/>Первый</item\>
<item id="2"\>Второй <subsub_item\>подпункт 1</subsub_item\></item\>
<item id="3"\>Третий</item\>
<item id="4"\><last/\>Последний</item\>
</list_of_items>
```

Первая строка - это объявление начала XML-документа, дальше идут элементы документа `<list_of_items>` - тег описывающий начало элемента `list_of_items`, `</list_of_items>` - тег конца элемента. Между этими тегами заключается описание элемента, которое может содержать текстовую информацию или другие элементы (как в нашем примере). Внутри тега начала элемента так же могут указывать атрибуты элемента, как например атрибут `id` элемента `item`, атрибуту должно быть присвоено определенное значение.

## 5 Разработка программного обеспечения

### 5.1 Архитектура

#### 5.1.1 Основной алгоритм

В ходе разработки был применен видоизменённый шаблон проектирования Factory method.

Данный шаблон относится к классу порождающих шаблонов. Шаблоны данного класса - это шаблоны проектирования, которые абстрагируют процесс инстанцирования (создания экземпляра класса). Они позволяют сделать систему независимой от способа создания, композиции и представления объектов. Шаблон, порождающий классы, использует наследование, чтобы изменять инстанцируемый класс, а шаблон, порождающий объекты, делегирует инстанцирование другому объекту. Пример организации проекта при использовании шаблона проектирования Factory method представлен на рисунке 5.1.

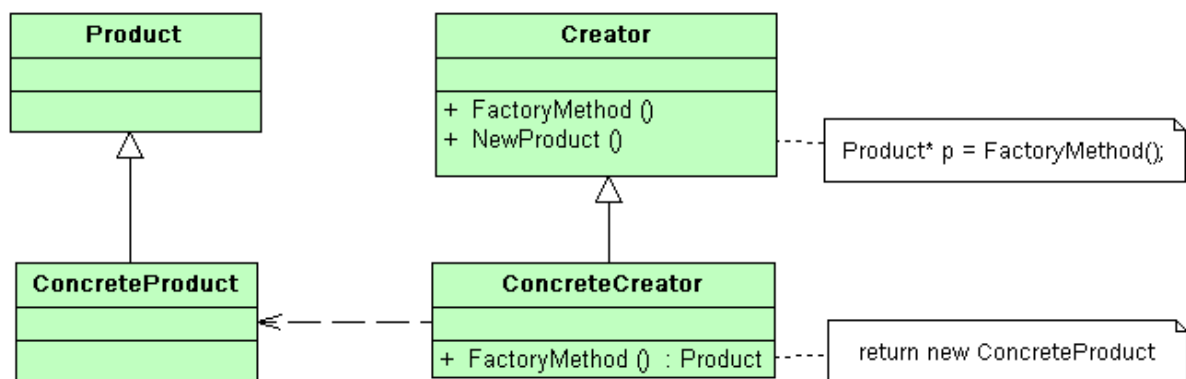


Рисунок 5.1 – Пример организации проекта при использовании шаблона проектирования Factory method

Использование данного шаблона позволило разбить проект на независимые модули, что весьма упростило задачу разработки, так как написание алгоритма для конкретного taska не влияло на остальную часть проекта. При разработке был реализован базовый класс для работы с образом диска. Данный клас предназначался для формирования списка настроек, определения операционной системы на смонтированном образе и инстанционировании и накапливание всех необходимых классов-taskов в очереди taskов. После чего каждый task из очереди отправлялся на выполнение. Блоксхема работы алгоритма представлена на рисунке 5.2.

Каждый класс-task порождался путем наследования от базового абстрактного класса который имеет 8 методов и 3 атрибута:

- 1) QString manual() - возвращает справку о входных параметрах данного taska;
- 2) void setOption(QStringList list) - установка флагов для поданных на вход параметров;
- 3) QString command() - возвращает команду для инициализации taska вручную;
- 4) bool supportOS(const coex::typeOS &os) - возвращает флаг, указывающий на возможность использования данного taska для конкретной операционной системы;
- 5) QString name() - возвращает имя данного taska;

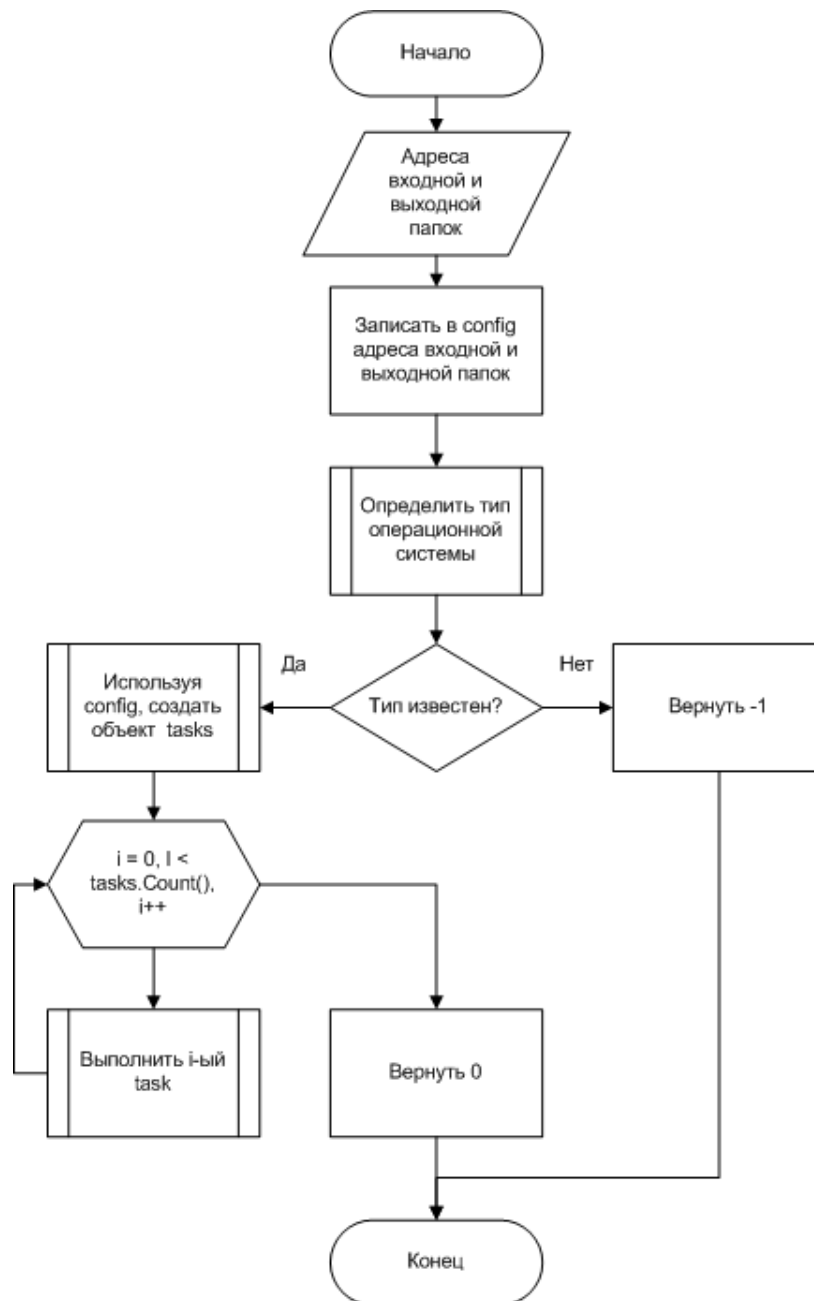


Рисунок 5.2 – Алгоритм работы с образом диска

- 6) QString description() - возвращает краткое описание таска;
- 7) bool test() - предназначена для теста на доступность таска;
- 8) bool execute(const coex::config &config) - запуск таска на выполнение;
- 9) QString m\_strName - хранит имя таска;
- 10) QString m\_strDescription - хранит описание таска;
- 11) bool m\_bDebug - флаг для параметра -debug;

На данный момент в проекте используется восемь классов. UML-диаграмма классов представлена на рисунке 5.3.

Классы taskSearchSyslogsWin, taskSearchPidginWin и taskSearchSkypeWin - наследники от класса task являются тасками. Класс winEventLog и \_EVENTLOGRECORD предназначены для конвертации журнальных файлов операционной системы Windows XP, а класс writerMessages для преобразования истории переписки.

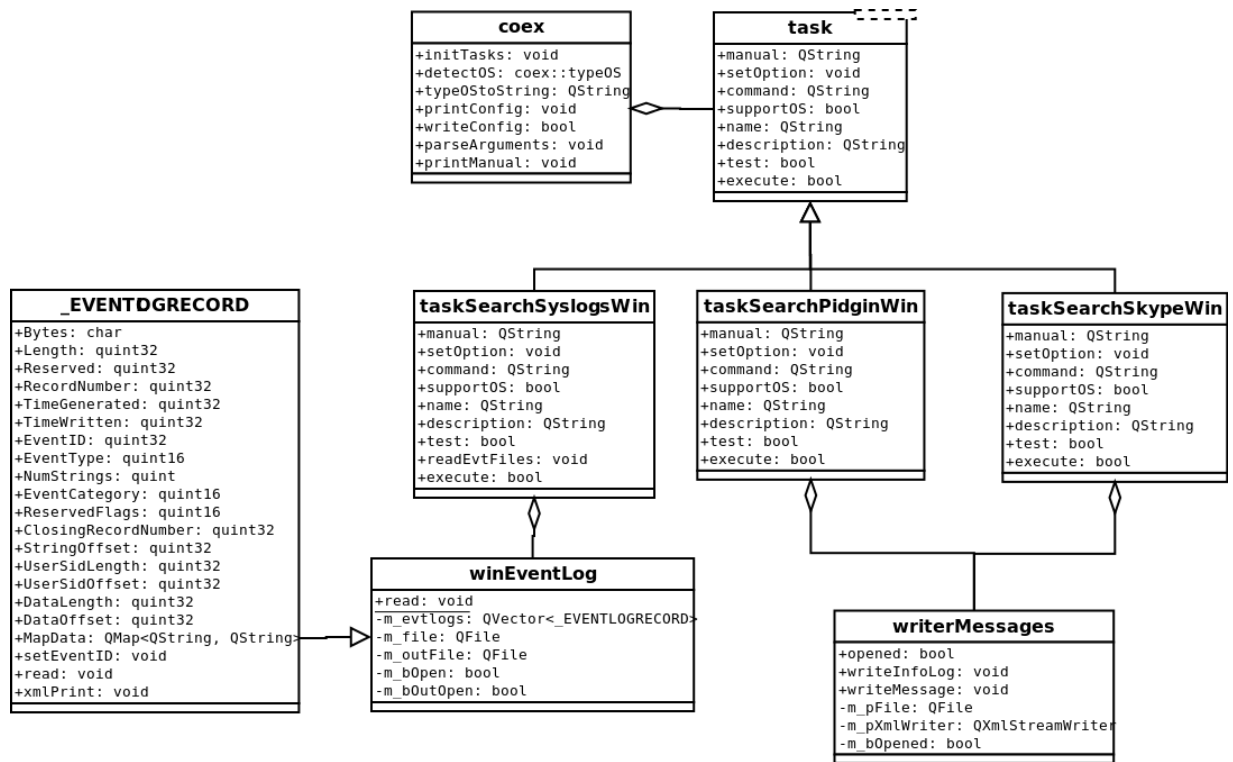


Рисунок 5.3 – UML-диаграмма классов

### 5.1.2 Описание основных функций модуля системы

Любой модуль системы является классом-наследником от некоторого абстрактного класса используемого как основу для всех модулей программы (шаблон проектирования Factory method). Модуль содержит в себе 8 методов и 3 атрибута:

QString manual() - возвращает справку о входных параметрах данного taska

void setOption(QStringList list) - установка флагов для поданных на вход параметров

QString command() - возвращает команду для инициализации taska вручную

bool supportOS(const coex::typeOS &os) - возвращает флаг указывающий на возможность использования данного taska для конкретной операционной системы

QString name() - возвращает имя данного taska

QString description() - возвращает краткое описание taska

bool test() - предназначена для проверки работоспособности taska

bool execute(const coex::config &config) - запуск taska на выполнение

QString m\_strName - хранит имя taska

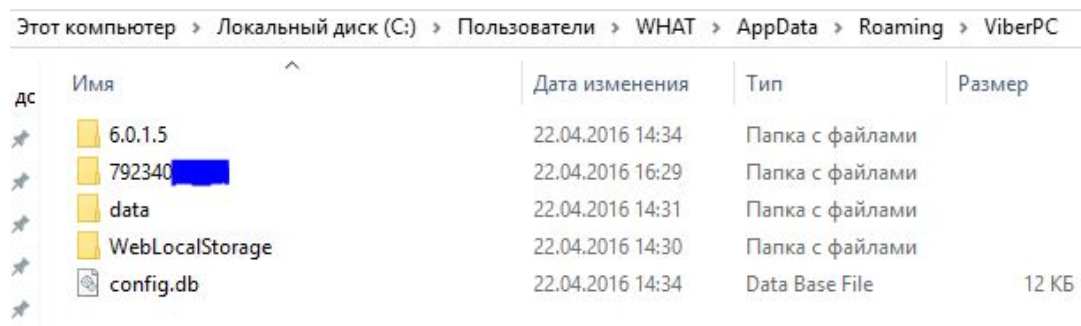
QString m\_strDescription - хранит описание taska

bool m\_bDebug - флаг для параметра -debug

## 5.2 Плагин TaskViber

### 5.2.1 Расположения файлов мессенджера Viber

В зависимости от операционной системы (в дальнейшем ОС) у Viber разные пути установки. Для Windows XP: C:\Documents and Settings\%Username%\Application Data\ViberPC. Для Windows 7, 8, 8.1, 10: C:\Users\%Username%\AppData\Roaming\ViberPC (рис. 5.4).



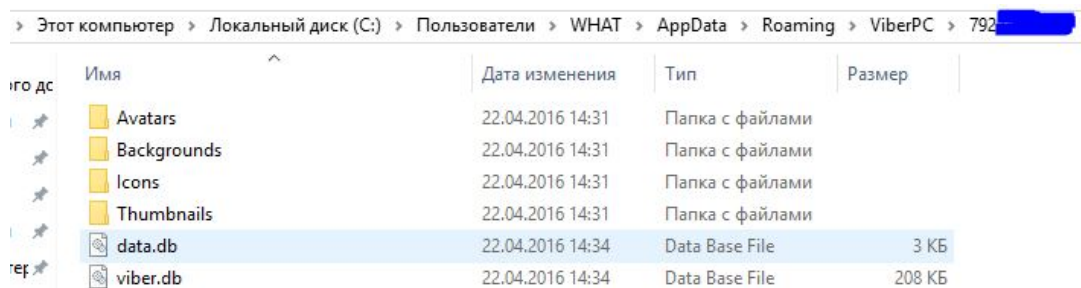
Этот компьютер > Локальный диск (C:) > Пользователи > WHAT > AppData > Roaming > ViberPC

Имя	Дата изменения	Тип	Размер
6.0.1.5	22.04.2016 14:34	Папка с файлами	
792340 [redacted]	22.04.2016 16:29	Папка с файлами	
data	22.04.2016 14:31	Папка с файлами	
WebLocalStorage	22.04.2016 14:30	Папка с файлами	
config.db	22.04.2016 14:34	Data Base File	12 КБ

Рисунок 5.4 – Путь к файлам Viber

### 5.2.2 Описание содержимого папки «ViberPC»

При изучении содержимого папки «ViberPC» было обнаружено, что интересующая информация содержится в папках, название которых – номер телефона (рис. 5.5).



Этот компьютер > Локальный диск (C:) > Пользователи > WHAT > AppData > Roaming > ViberPC > 792340 [redacted]

Имя	Дата изменения	Тип	Размер
Avatars	22.04.2016 14:31	Папка с файлами	
Backgrounds	22.04.2016 14:31	Папка с файлами	
Icons	22.04.2016 14:31	Папка с файлами	
Thumbnails	22.04.2016 14:31	Папка с файлами	
data.db	22.04.2016 14:34	Data Base File	3 КБ
viber.db	22.04.2016 14:34	Data Base File	208 КБ

Рисунок 5.5 – Содержимое папки с номером телефона

- Папка «Avatars» – содержит изображения пользователей;
- Папка «Thumbnails» – содержит все изображения, которые были отправлены и получены в ходе переписки; «viber.db» – база данных (далее БД), в которой хранится информация о контактах, переписках, звонках. БД «viber.db» – имеет формат SQLite format 3 (рис. 5.6).

### 5.2.3 SQL-запросы для получения информации

Чтобы получить все контакты и их имена был написан следующий SQL-запрос: *Select ContactRelation.Number, Contact.FirstName from ContactRelation, Contact where Contact.ContactID = ContactRelation.ContactID.*

Чтобы получить все контакты и имена, на которые можно позвонить в Viber, нужен следующий SQL-запрос: *Select Contact.FirstName, ContactRelation.Number from contact,*



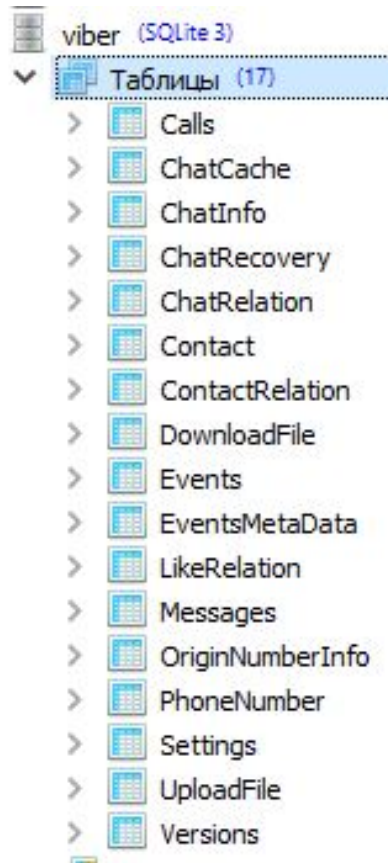


Рисунок 5.6 – Содержимое БД «viber.db»

*PhoneNumber, ContactRelation where PhoneNumber.IsViberNumber = 1 and PhoneNumber.Number = ContactRelation.Number and ContactRelation.ContactID = Contact.ContactID.*

Чтобы связать изображение пользователя с номером телефона и именем, нужен следующий SQL-запрос: *Select Contact.FirstName, ContactRelation.Number, OriginNumberInfo.AvatarPath From OriginNumberInfo, ContactRelation, Contact Where OriginNumberInfo.Number = ContactRelation.Number and ContactRelation.ContactID = Contact.ContactID.*

Для получения информации о звонках которые осуществлялись через Viber, нужен следующий запрос: *select Contact.FirstName, Events.Direction, datetime(Events.TimeStamp, 'unixepoch') from Contact, Events, ContactRelation where Events.EventID = (select Calls.EventID from Calls) AND Events.Number = ContactRelation.Number and ContactRelation.ContactID = Contact.ContactID.*

Для получения текста переписки с конкретным пользователем нужно знать его номер чата. Для получения всех номеров чата нужно воспользоваться следующим запросом: *Select ChatInfo.ChatID, Contact.FirstName, ChatInfo.TokenFrom ChatInfo, ContactRelation, Contact where ChatInfo.Token = ContactRelation.Number and ContactRelation.ContactID = Contact.ContactID.*

Зная номер чата, можно получить текст переписки: *select Messages.Body, Contact.FirstName, Events.Direction, Messages.ThumbnailPath, datetime(Events.TimeStamp, 'unixepoch') from messages, Events, Contact, ContactRelation where Messages.EventID = Events.EventID and Events.Number = ContactRelation.Number and ContactRelation.ContactID = Contact.ContactID and Events.ChatID = @nomer\_chata.*

#### 5.2.4 Описание плагина

Плагин «TaskViber» получает точку монтирования жесткого диска, с которого, в отличие от ОС, проверяет папку «ViberPC» у всех пользователей в ОС. Если папка «ViberPC» существует, то плагин извлекает информацию из аккаунтов, под которыми авторизовались с данного компьютера. Всю найденную информацию плагин сохраняет по указанному пути программного обеспечения «COEX». В папку «Avatars» (рис. 5.12) копируются все найденные изображения пользователей. В папку «Thumbnails» (рис. 5.13) копируются все изображения, которые были отправлены и получены в ходе переписки. В файле «Avatar Path.txt» находятся связи между изображениями пользователей, именами и номерами телефонов. В файле «Calls.txt» находятся описание звонков, которые осуществлялись через «Viber» (с кем был звонок, во сколько и кто кому звонил). В файле «Phone book.txt» находятся все номера телефонов и имена с мобильного телефона, на котором был зарегистрирован аккаунт в «Viber». В файле «Viber book.txt» находятся все номера телефонов и имена, которым можно позвонить через «Viber». В файле «Имя/номер messages.txt» содержится переписка с пользователем «Имя/номер» (с кем велась переписка, кто кому писал, что писал и во сколько писал).

Блок-схема алгоритма работы плагина «TaskViber» представлена на рисунке 5.7, блок-схема функции WinXP — на рисунке 5.8. Ниже также представлены блок-схемы Win\_7\_8\_10 (рис. 5.9) и Viber\_XP\_7\_8\_10 (рис. 5.10).

Результат работы плагина представлен на рисунке 5.11.

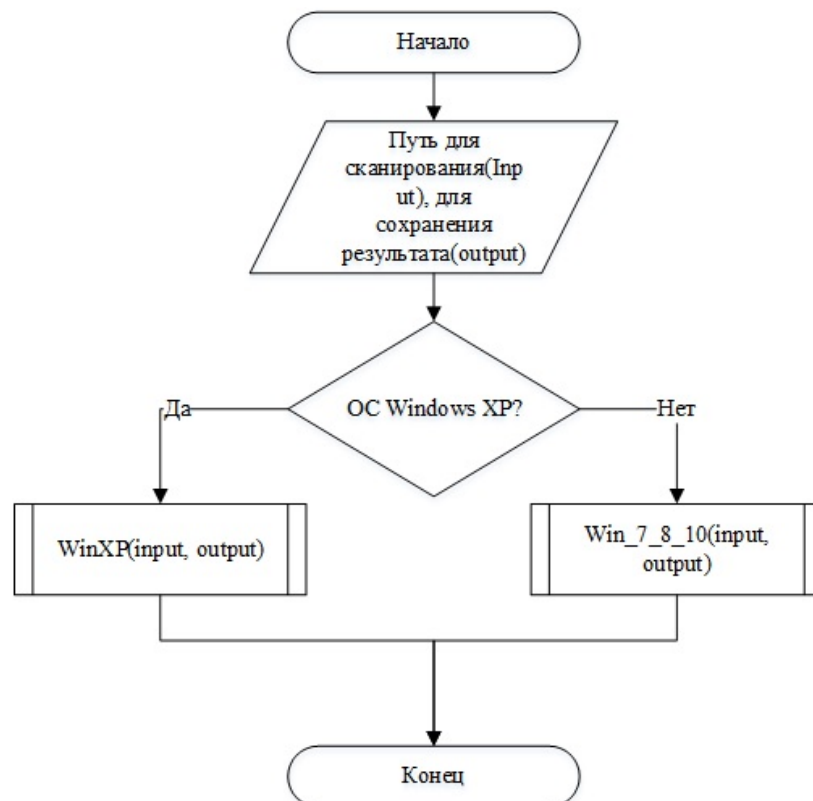


Рисунок 5.7 – Блок-схема алгоритма работы плагина

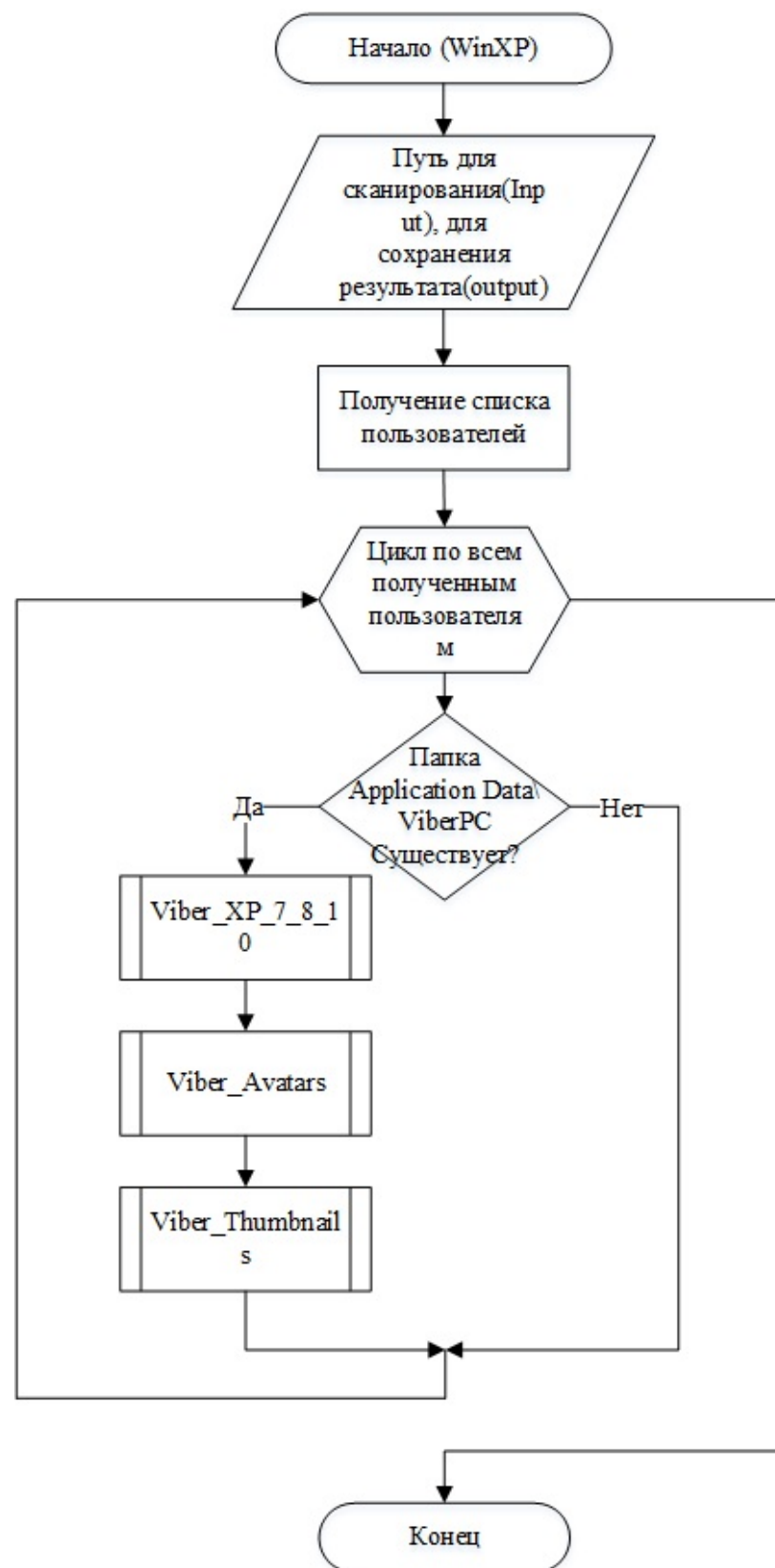


Рисунок 5.8 – Блок-схема функции WinXP

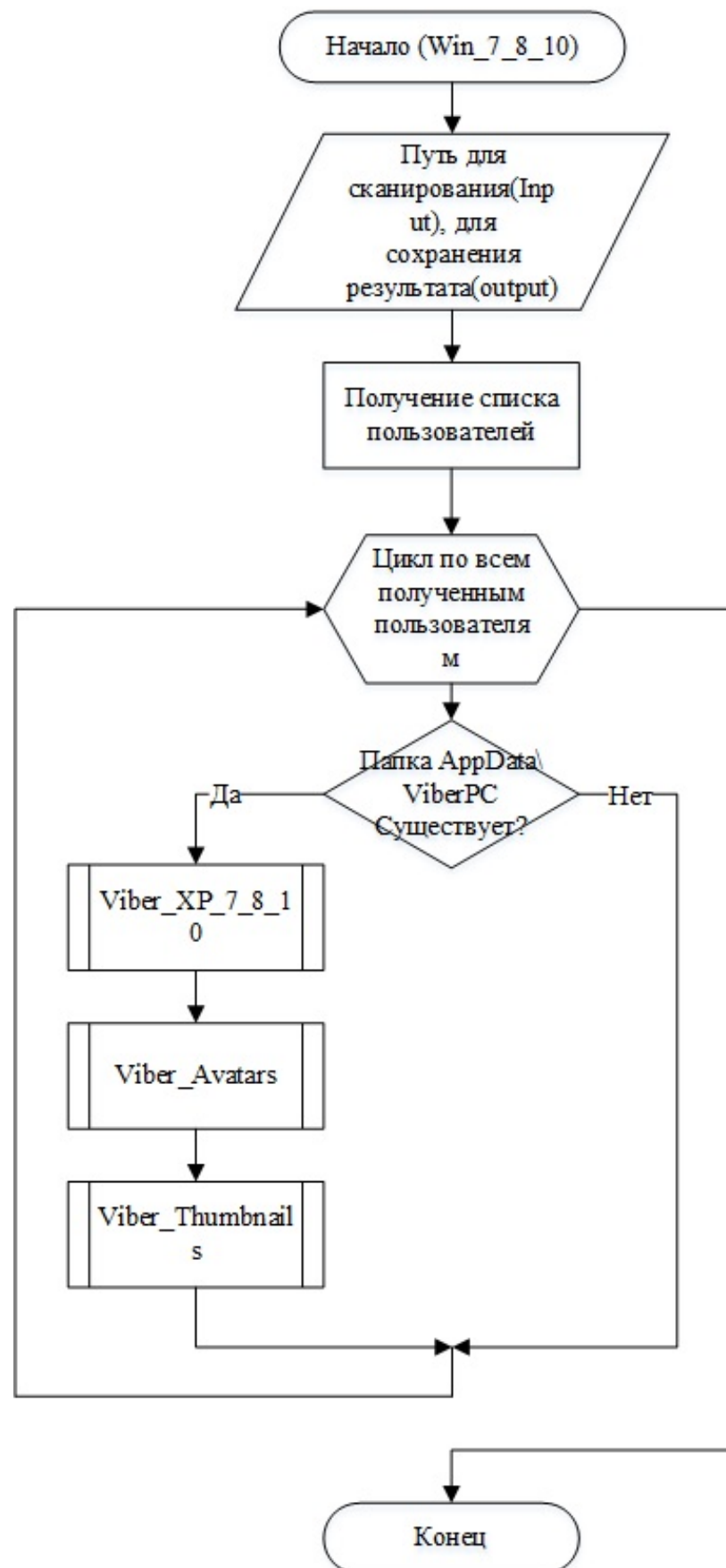


Рисунок 5.9 – Блок-схема Win\_7\_8\_10

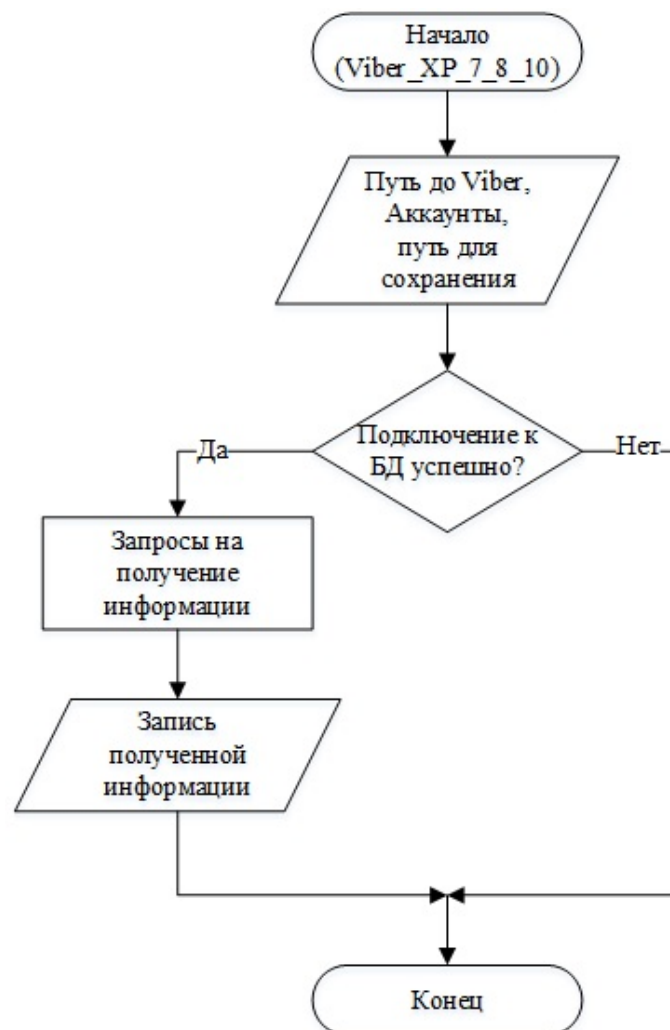


Рисунок 5.10 – Блок-схема Viber\_XP\_7\_8\_10

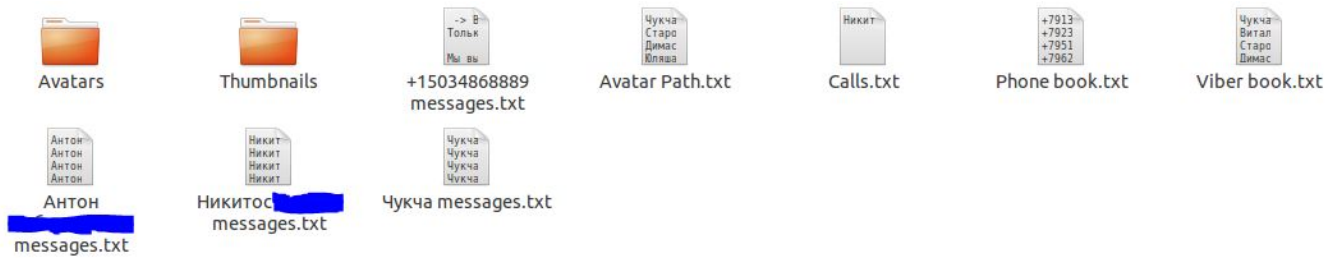


Рисунок 5.11 – Результат работы плагина

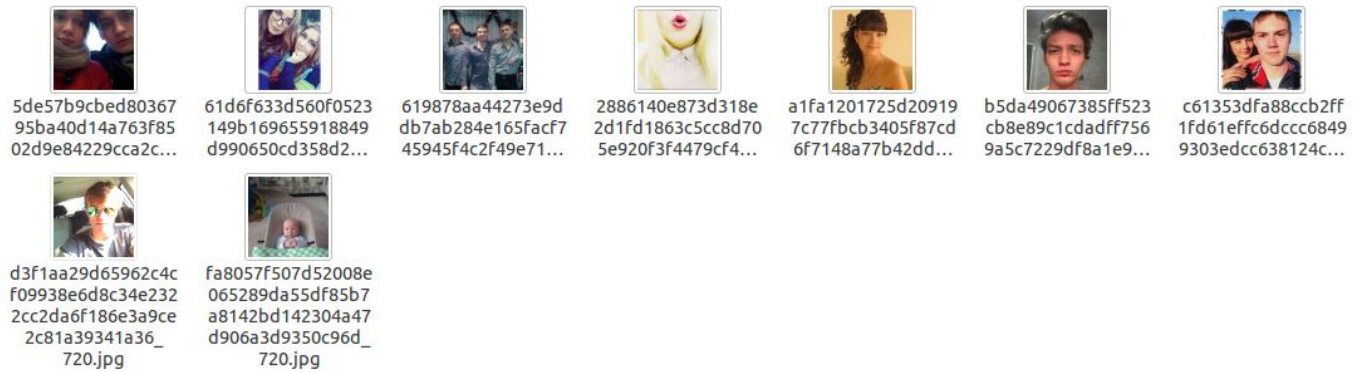


Рисунок 5.12 – Содержимое папки «Avatars»

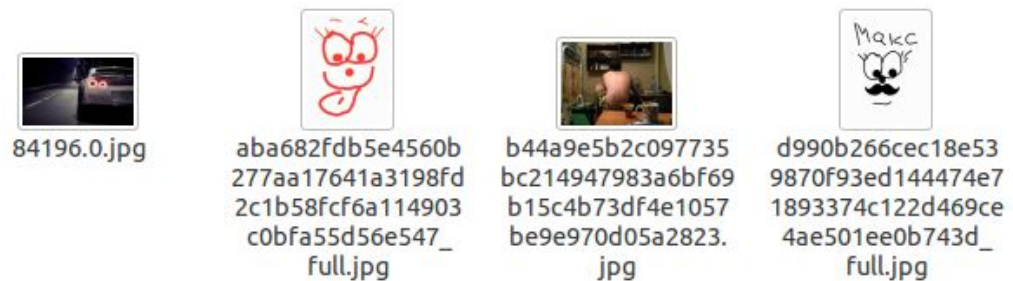


Рисунок 5.13 – Содержимое папки «Thumbnails»

### 5.3 Сбор сохраненных логинов и паролей браузера Mozilla Firefox

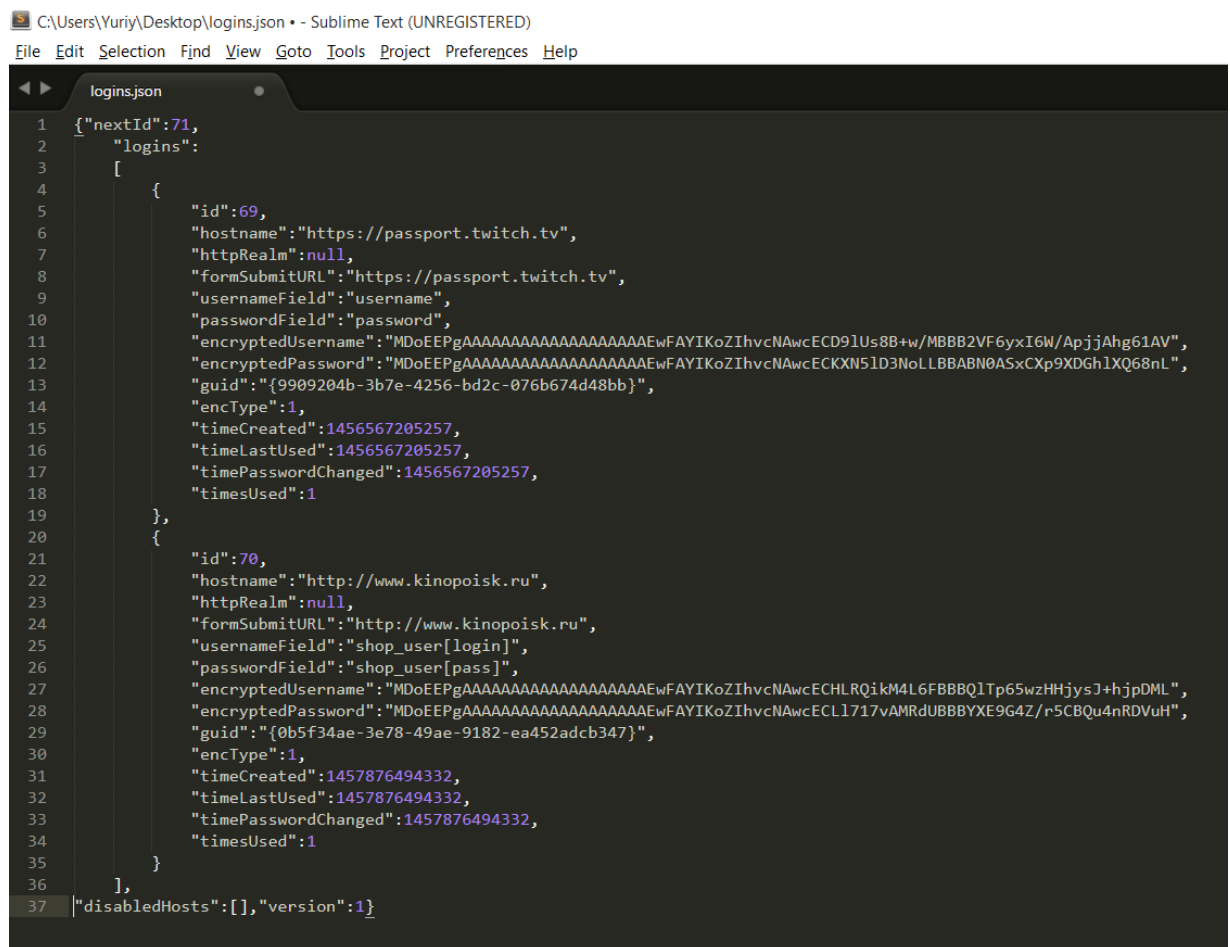
Главной задачей в данном стало написание программного модуля для сбора сохраненных логинов и паролей браузера Mozilla Firefox.

Сохраненные логины и пароли хранятся в папке с профилем Mozilla Firefox. Путь к папке: C:\Users\User\AppData\Roaming\Mozilla\Firefox\Profiles\prof\_n. Где prof\_n генерируется самим браузером.

Для браузеров, версия которых 31 и меньше, логины и пароли хранятся в файле базы данных signons.sqlite. База данных имеет формат sqlite3. В базе данных есть таблица moz\_logins. В ней нас интересуют столбцы: encryptedUsername, encryptedPassword, formSubmitURL, которые содержат зашифрованный логин, зашифрованный пароль и URL сайта соответственно.

Для остальных, логины и пароли хранятся в файле logins.json.

Json – текстовый формат файла, в котором информация представлена в виде структур. На рисунке 5.14 показано, как выглядит файл logins.json.



```

1  {"nextId":71,
2    "logins":
3    [
4      {
5        "id":69,
6        "hostname":"https://passport.twitch.tv",
7        "httpRealm":null,
8        "formSubmitURL":"https://passport.twitch.tv",
9        "usernameField":"username",
10       "passwordField":"password",
11       "encryptedUsername":"MDoEEPgAAAAAAAAAAAAAAAAAAEwFAYIKoZIhvcNAwECDD9lUs8B+w/MBBB2VF6yxI6W/ApjJAhg61AV",
12       "encryptedPassword":"MDoEEPgAAAAAAAAAAAAAAAAAAEwFAYIKoZIhvcNAwECCKXN5lD3NoLLBBABN0ASxCXp9XDGHlXQ68nL",
13       "guid":"{9909204b-3b7e-4256-bd2c-076b674d48bb}",
14       "encType":1,
15       "timeCreated":1456567205257,
16       "timeLastUsed":1456567205257,
17       "timePasswordChanged":1456567205257,
18       "timesUsed":1
19     },
20     {
21       "id":70,
22       "hostname":"http://www.kinopoisk.ru",
23       "httpRealm":null,
24       "formSubmitURL":"http://www.kinopoisk.ru",
25       "usernameField":"shop_user[login]",
26       "passwordField":"shop_user[pass]",
27       "encryptedUsername":"MDoEEPgAAAAAAAAAAAAAAAAAAEwFAYIKoZIhvcNAwECCHLRQikM4L6FBBBQ1Tp65wzHHjysJ+hjpDML",
28       "encryptedPassword":"MDoEEPgAAAAAAAAAAAAAAAAAAEwFAYIKoZIhvcNAwECCL1717vAMRdUBBBYXE9G4Z/r5CBQu4nRnVuH",
29       "guid":"{0b5f34ae-3e78-49ae-9182-ea452adcb347}",
30       "encType":1,
31       "timeCreated":1457876494332,
32       "timeLastUsed":1457876494332,
33       "timePasswordChanged":1457876494332,
34       "timesUsed":1
35     }
36   ],
37   "disabledHosts":[], "version":1}

```

Рисунок 5.14 – Файл logins.json

Здесь нас интересуют те же самые поля, как и в случае с базой данных.

Как видно из рисунка 5.14, логины и пароли зашифрованы. Для их расшифровки нам также необходимы файлы cert8.db, key3.db и secmod.db. В них содержатся ключи для расшифровки значений. Эти файлы также находятся в папке с профилем.

Так же для работы модуля необходима библиотека nss3. Именно с помощью нее браузер зашифровывает логины и пароли.

Nss3 – open source библиотека, разработанная для создания кроссплатформенных защищенных клиентских и серверных приложений.

### 5.3.1 Алгоритм работы программного модуля

Сначала модуль подключает стороннюю библиотеку nss3. Затем выполняется поиск файла compatibility.ini (также находится в папке профиля), содержащего версию браузера.

Для браузеров, версия которых 31 и меньше, ищется файл базы данных signons.sqlite. Модуль подключается к нему и выполняет sql-запрос: «SELECT encryptedUsername, encryptedPassword, formSubmitURL FROM moz\_logins».

Для других версий, ищется файл logins.json, вытаскиваются нужные нам значения.

Затем данные расшифровываются с помощью сторонней библиотеки nss3 и сохраняются в xml-файл.

Блок-схема алгоритма работы программы представлена на рисунке 5.16.

### 5.3.2 Тестирование

Как видно из рисунков 5.15 и 5.17 плагин выполняет поставленные перед ним задачи.

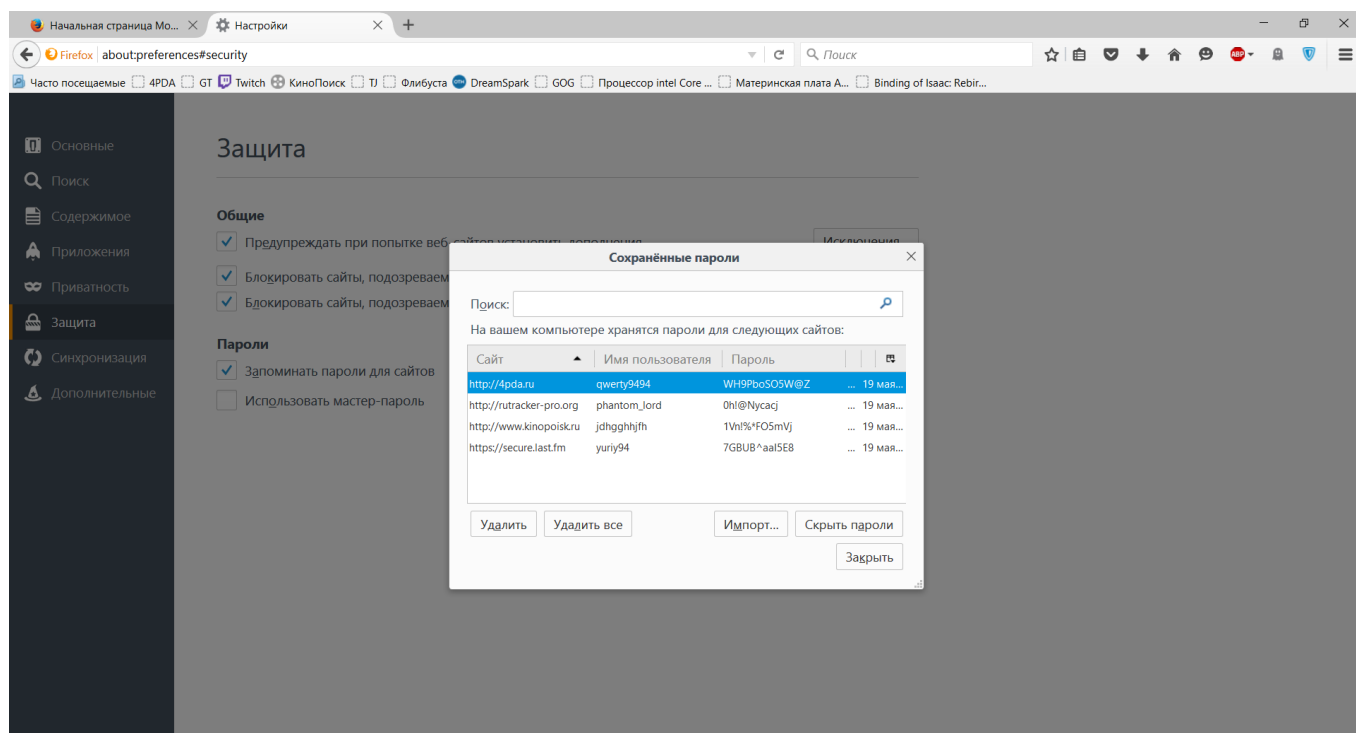


Рисунок 5.15 – Тестовые данные



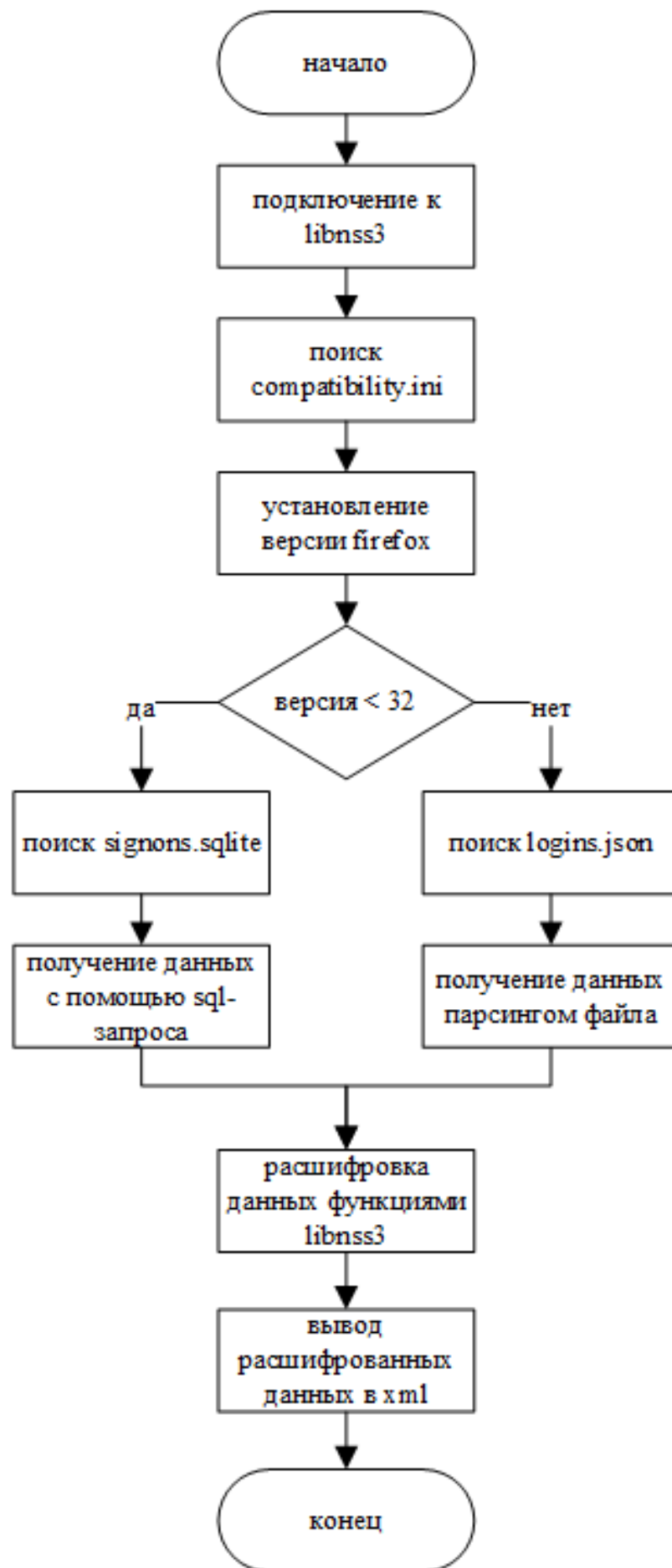


Рисунок 5.16 – Алгоритм работы программного модуля

---

```
-<add>
-<doc>
  <field name="url">http://www.kinopoisk.ru</field>
  <field name="login">jdhgghhj</field>
  <field name="password">1Vn!%*FO5mVj</field>
</doc>
-<doc>
  <field name="url">http://rutracker-pro.org</field>
  <field name="login">phantom_lord</field>
  <field name="password">0h!@Nycacj</field>
</doc>
-<doc>
  <field name="url">https://secure.last.fm</field>
  <field name="login">yuriy94</field>
  <field name="password">7GBUB^aaI5E8</field>
</doc>
-<doc>
  <field name="url">http://4pda.ru</field>
  <field name="login">qwerty94</field>
  <field name="password">WH9PboSO5W@Z</field>
</doc>
</add>
```

Рисунок 5.17 – Результат тестирования

#### 5.4 Графический интерфейс пользователя системы «COEX»

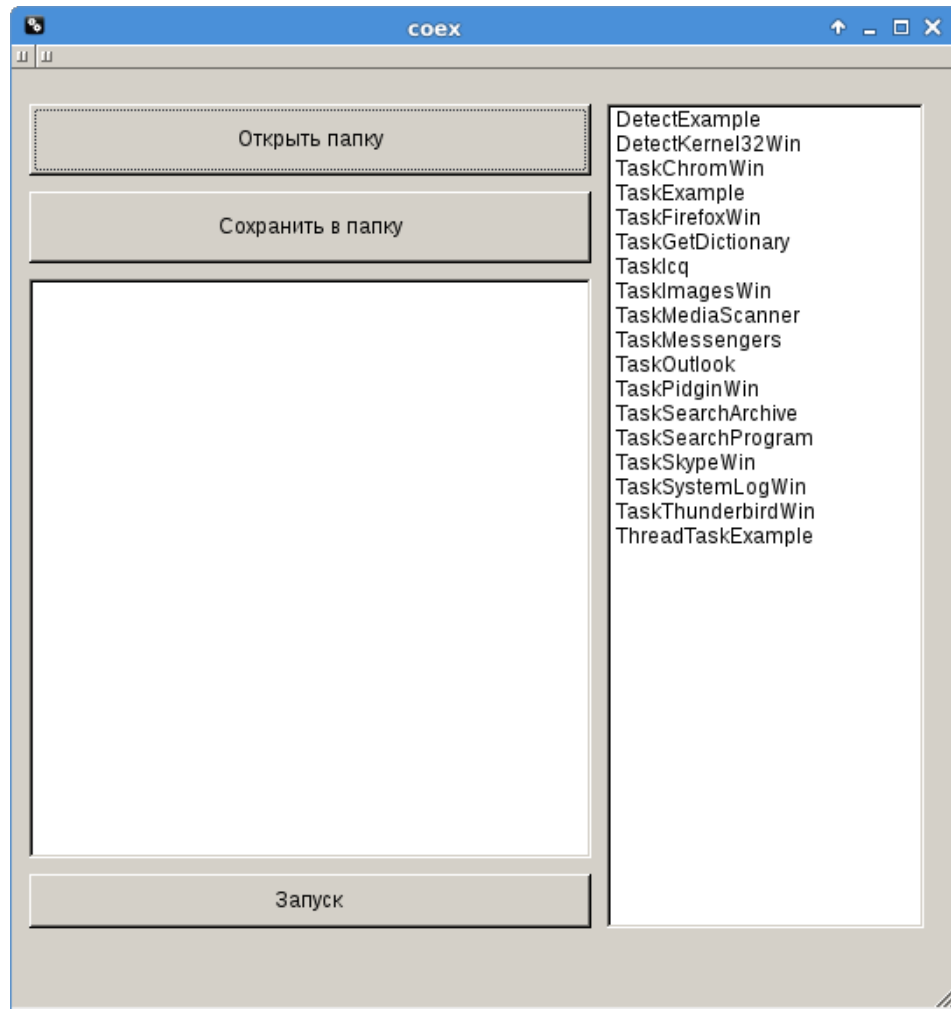


Рисунок 5.18 – Предыдущая версия интерфейса

Новый интерфейс состоит из следующих элементов:

- 1) элемент отвечающий за запуск coex;
- 2) элемент отвечающий за настройки;
- 3) сведения о программе;
- 4) кнопка закрытия приложения;
- 5) область вывода промежуточной информации.

Интерфейс окна настроек состоит из следующих элементов:

- 1) элемент «исходная папка» отвечает выбор папки, в которой будет производиться поиск;
- 2) элемент «папка назначения» отвечает за выбор папки для сохранения результатов;
- 3) данная область отвечает за выбор компонентов coex;
- 4) элемент «Сохранить» отвечает за сохранения выбранных настроек.

При следующем запуске приложения будут загружены сохраненные ранее настройки, а также геометрия главного приложения и его состояние т.е. оно откроется в том месте где его закрыли. Также данное окно является модальным т.е. оно прерывают работу главного окна и для продолжения его работы такое окно должно быть закрыто.

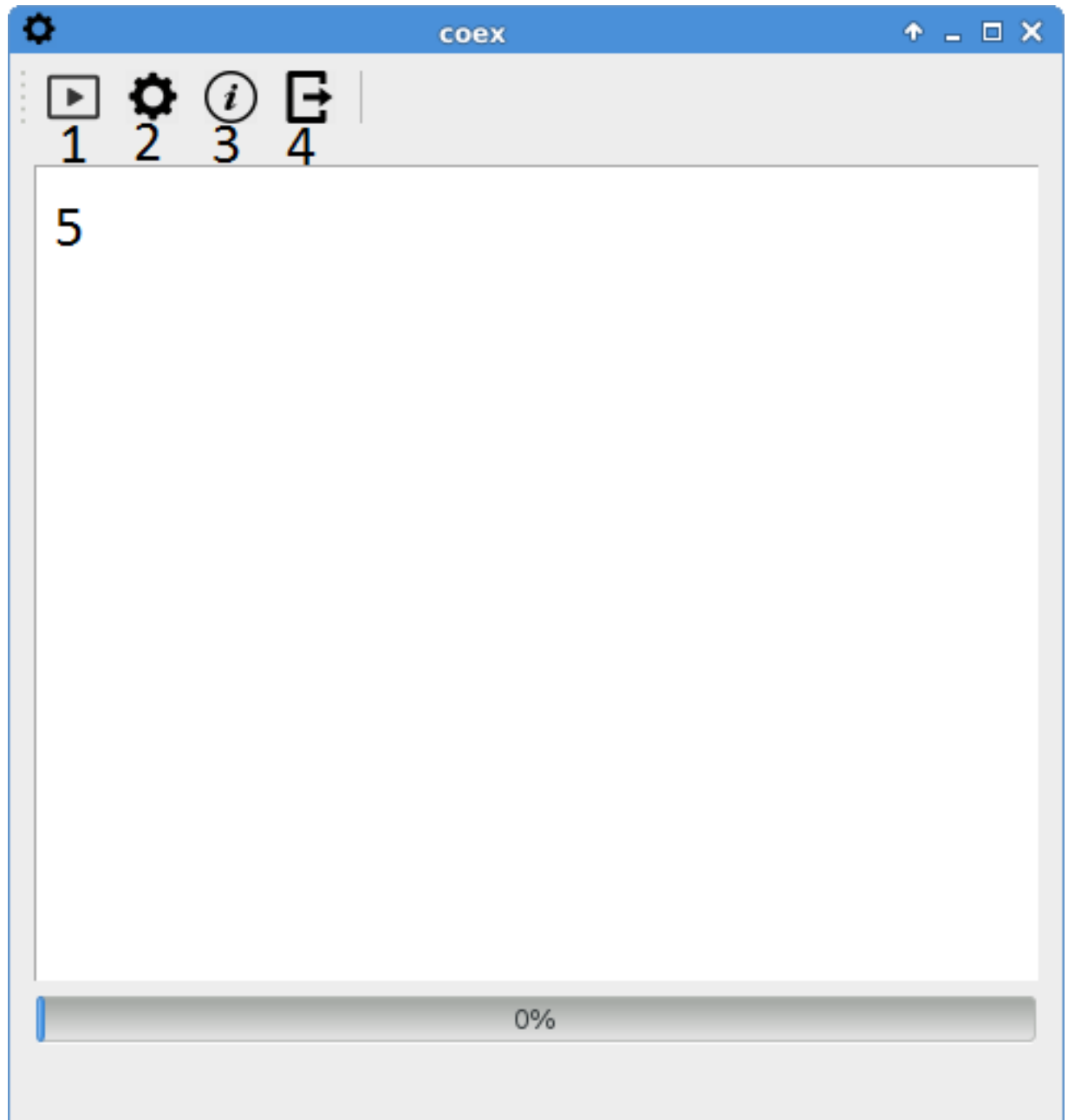


Рисунок 5.19 – Главное окно нового интерфейса

Для простоты мы предполагаем, что организация называется TUSUR, а приложение называется coex данные параметры прописываются в исходном коде приложения. Настройки будут храниться по-разному в зависимости от платформы.

В системах Unix:

- 1) HOME/.config/TUSUR/coex.conf;
- 2) HOME/.config/coex.conf;
- 3) /etc/xdg/TUSUR/coex.conf;
- 4) /etc/xdg/TUSUR/.conf.

В Mac OS

- 1) HOME/Library/Preferences/com.TUSUR.coex.plist;
- 2) HOME/Library/Preferences/com.TUSUR.plist;
- 3) /Library/Preferences/com.TUSUR.coex.plist;

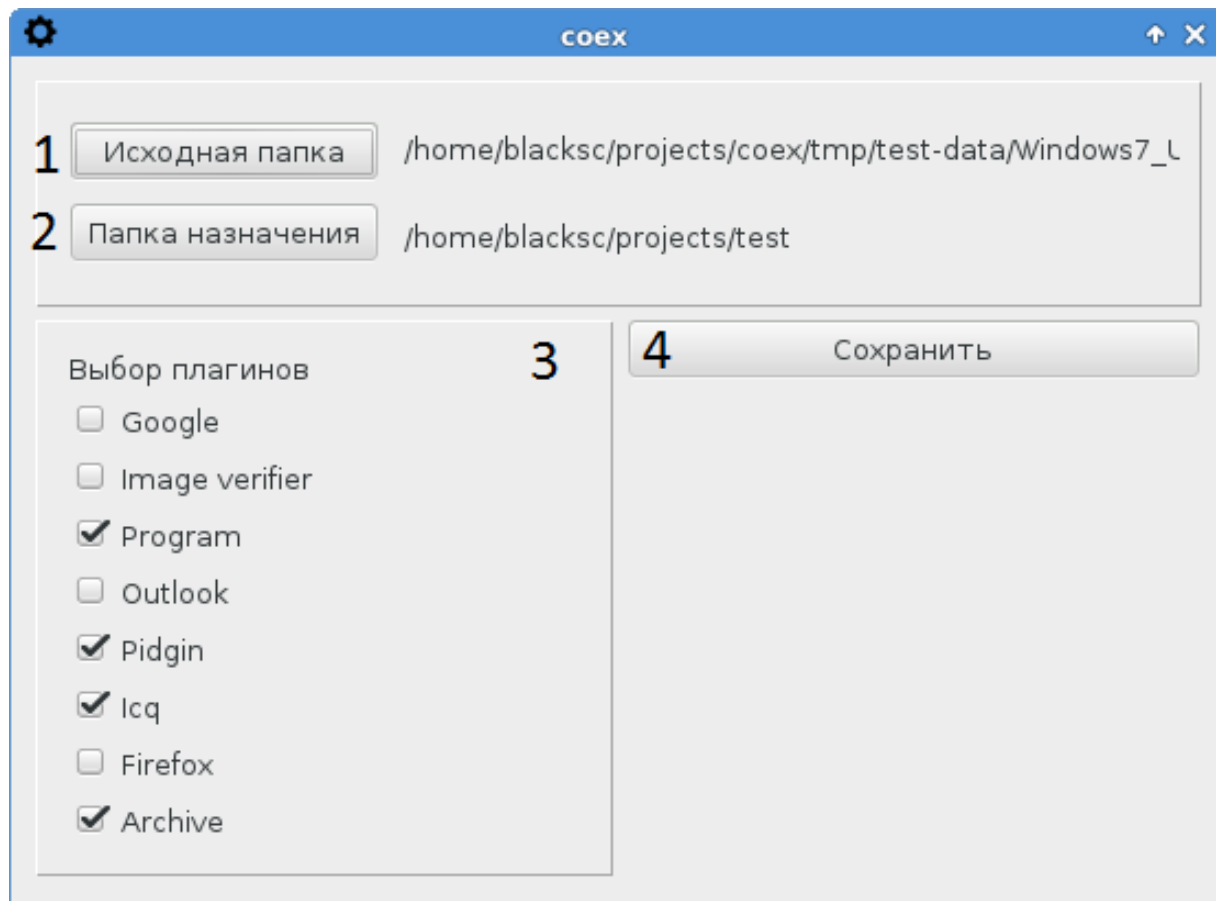


Рисунок 5.20 – Окно настроек

4) /Library/Preferences/com.TUSUR.plist.

В Windows настройки хранятся по следующим путям реестра:

- 1) HKEY\_CURRENT\_USER\Software\TUSUR\coex;
- 2) «HKEY\_CURRENT\_USER\Software\TUSUR»;
- 3) «HKEY\_LOCAL\_MACHINE\Software\TUSUR\coex»;
- 4) «HKEY\_LOCAL\_MACHINE\Software\TUSUR».

Содержание файла настроек представлено на рисунке 5.21.

Если не выбраны директории для работы, то при нажатии на кнопку “Запуск” отобразиться соответствующее сообщение (рисунок 5.24). Диалоговое окно выбора директории представлено на рисунке 5.23. По завершению работы системы отобразиться соответствующее сообщение (рисунок 5.25), при этом если нажать на кнопку “Результаты”, то откроется директория с результатами работы (рисунок 5.26).

```

[[General]
WindowGeometry=@ByteArray(\x1\xd9\xd0\xcb\x1\x0\x0\x0\xbe\x0\x0g\x0\x3\x17\x0
\x2\xea\x0\x0\xbf\x0\x0}\x0\x3\x16\x0\x2\xe6\x0\x0\x0\x0)
WindowState="@ByteArray(\x0\x0\xff\x0\x0\x0\xfd\x0\x0\x0\x0\x2X\x0\x2,\x0\x0\x4\
\x0\x0\x4\x0\x0\x0b\x0\x0\x0b\xfc\x0\x0\x1\x0\x0\x2\x0\x0\x1\x0\x0\x0e\x0t\x0o\x0l\x0x
42\x0\x61\x0r\x1\x0\x0\x0\xff\xff\xff\xff\x0\x0\x0\x0\x0\x0)"
archive%3A=true
firefox%3A=false
google%3A=false
icq%3A=true
image%3A=false
inputdir%3A=/home/blacksc/projects/coex/tmp/test-data/Windows7_Ult
outlook%3A=false
outputdir%3A=/home/blacksc/projects/test
pidgin%3A=true
programm%3A=true

```

Рисунок 5.21 – Содержание файла настроек

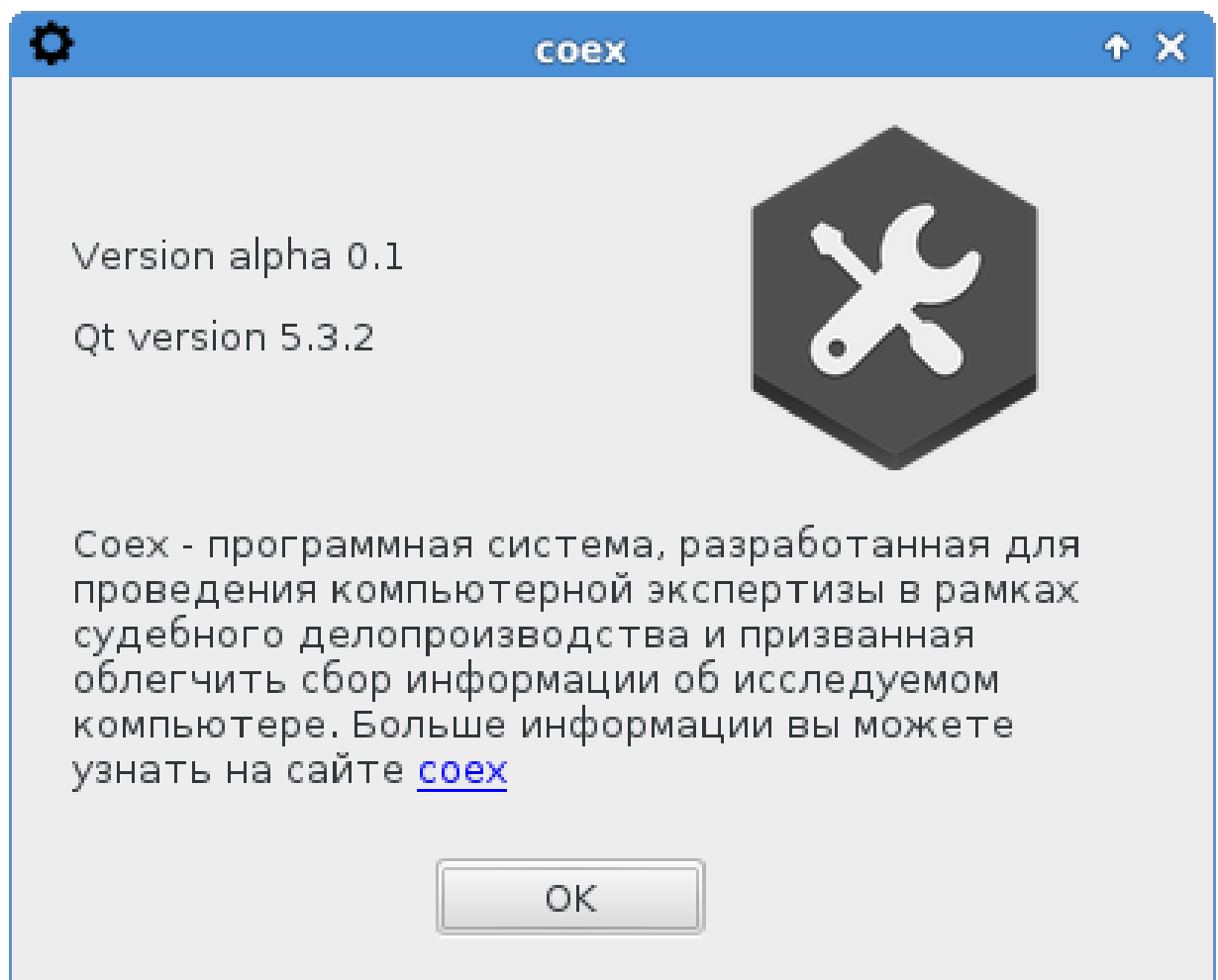


Рисунок 5.22 – О программе

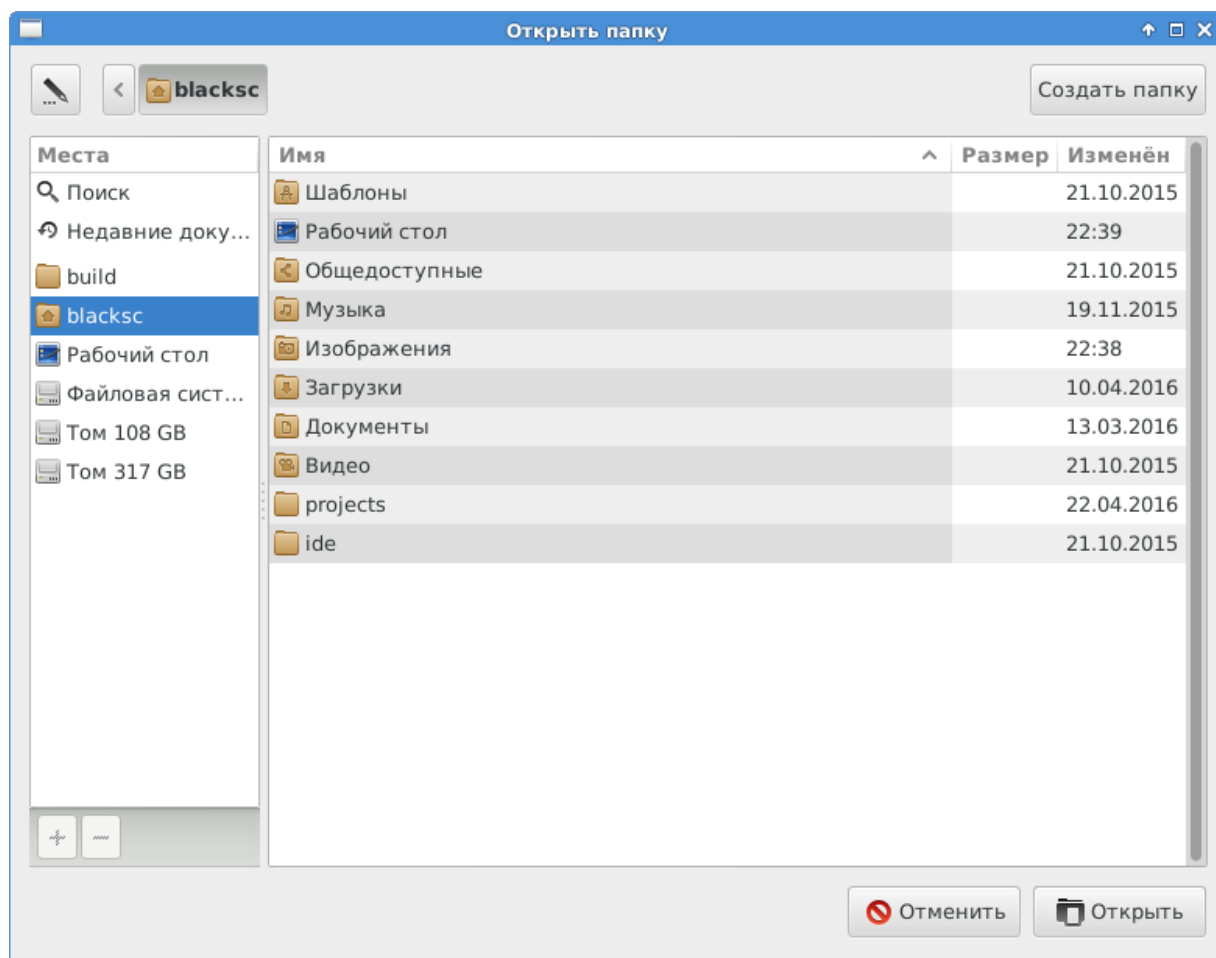


Рисунок 5.23 – Выбор директорий

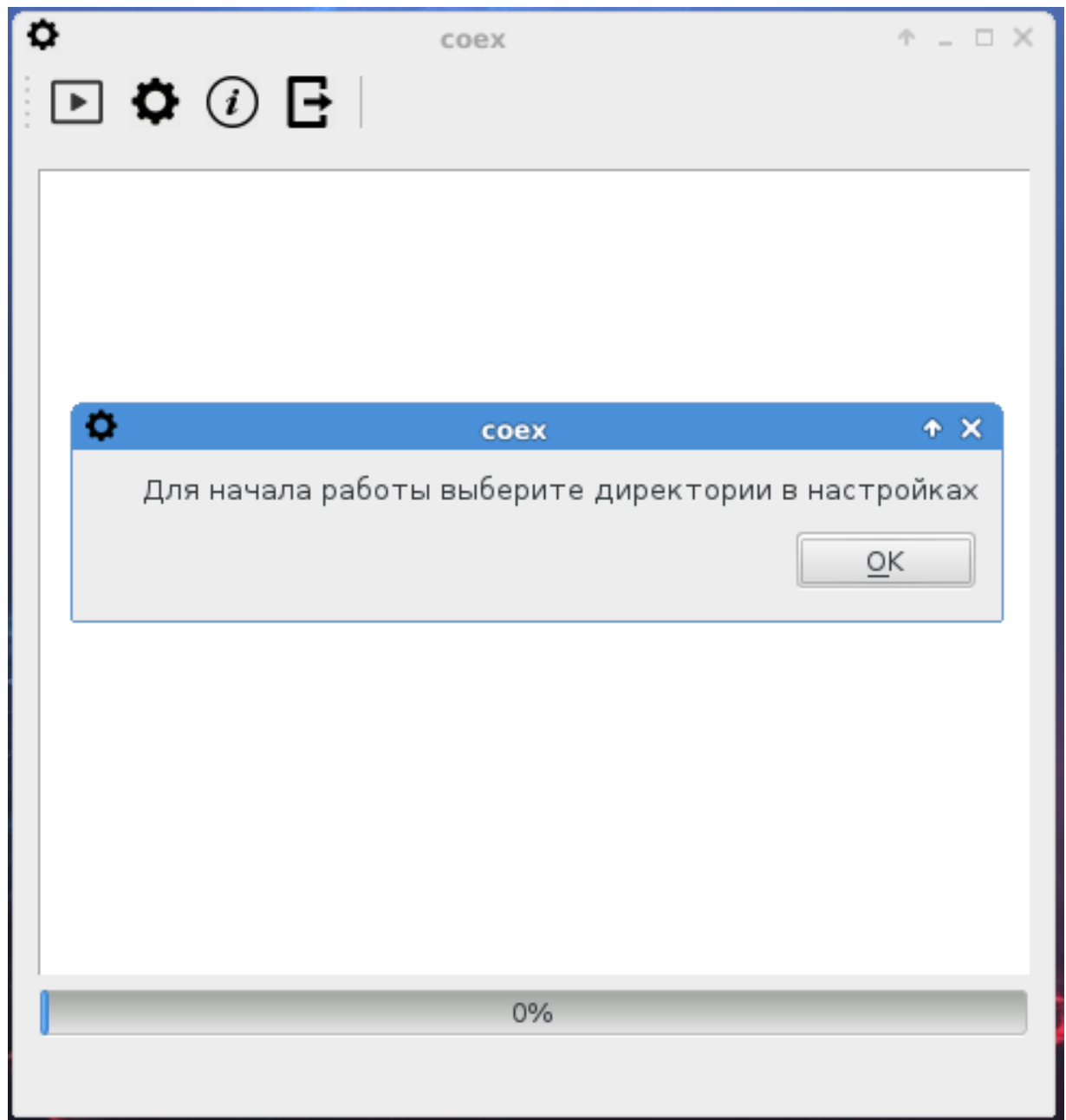


Рисунок 5.24 – Сообщение об ошибке



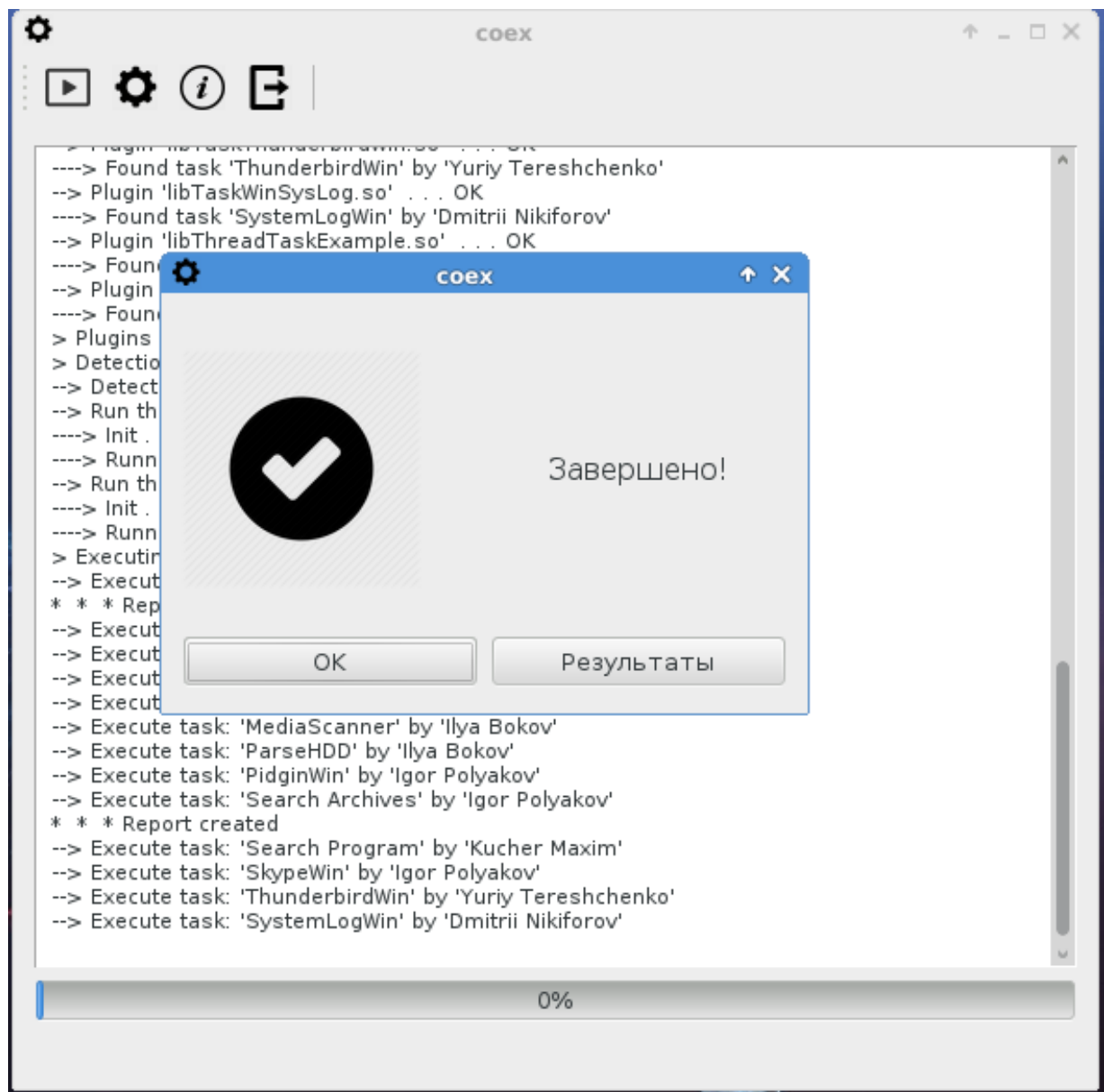


Рисунок 5.25 – Завершение работы

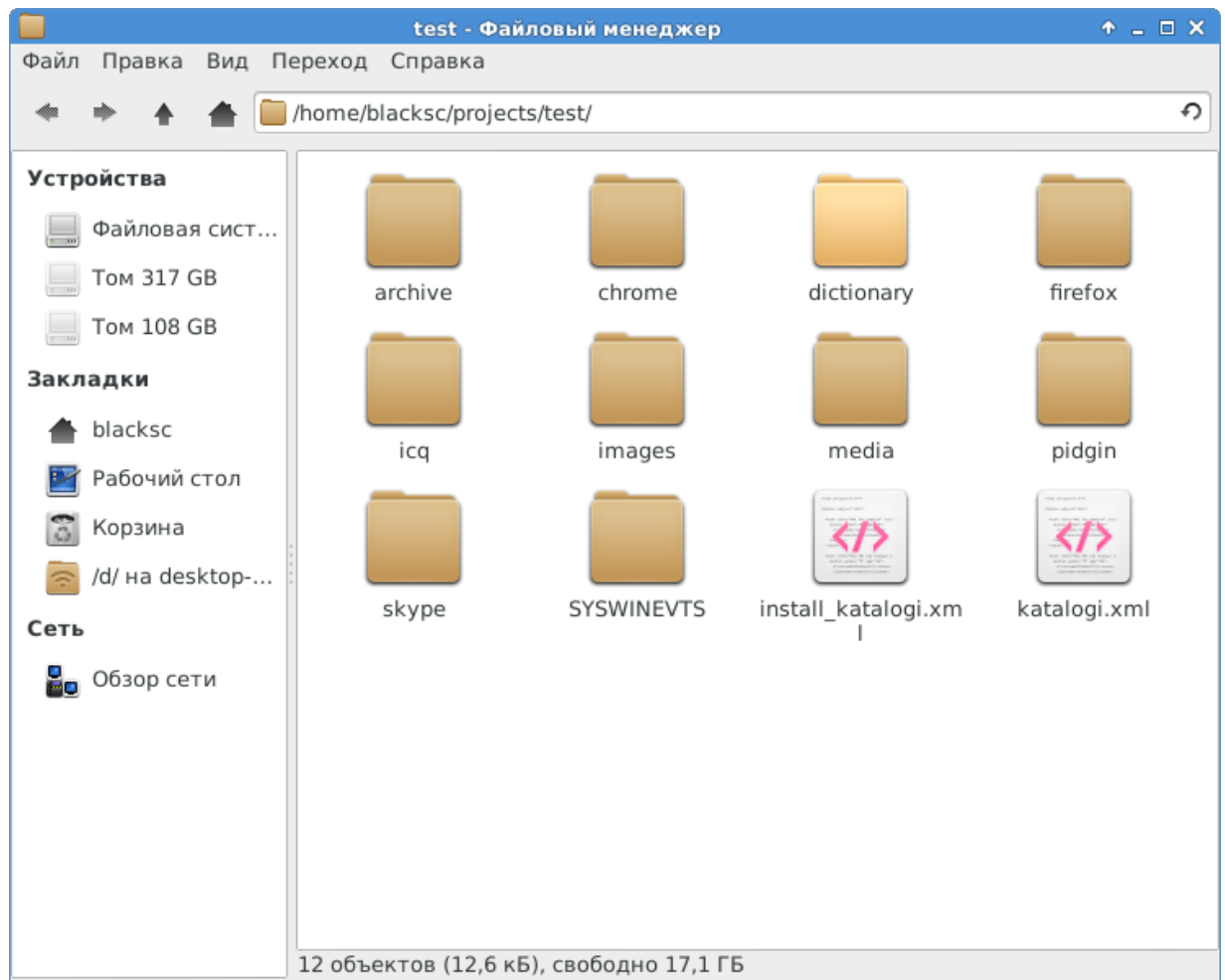


Рисунок 5.26 – Папка с результатами

## 5.5 Анализ данных с помощью Apache Solr

### 5.5.1 Общая информация об Apache Lucene, Solr

Apache Solr - это расширяемая поисковая платформа от Apache. Система основана на библиотеке Apache Lucene и разработана на Java. Особенности ее в том, что она представляет из себя не просто техническое решение для поиска, а именно платформу, поведение которой можно легко расширять/менять/настраивать под любые нужды - от обычного полнотекстового поиска на сайте до распределенной системы хранения/получения/аналитики текстовых и других данных с мощным языком запросов. Lucene — самый известный из поисковых движков, изначально ориентированный именно на встраивание в другие программы.

### 5.5.2 Подготовка окружения и установка Apache Lucene

Добавляем репозитории: `deb http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main`  
`deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main` (рис. 5.27).

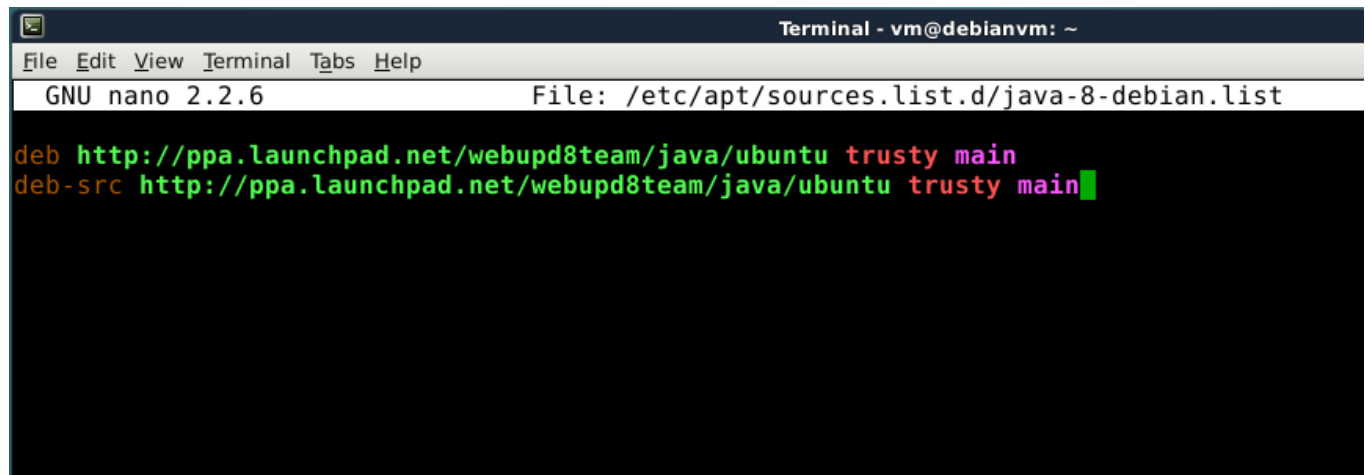


Рисунок 5.27 – Добавление репозиториев

Добавляем ключ: `apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys EEA14886` (рис. 5.28).

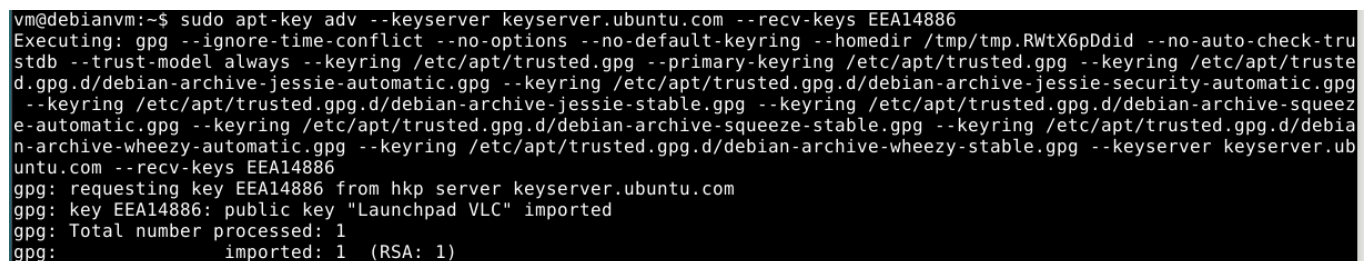


Рисунок 5.28 – Добавление ключа

Обновляем список пакетов: `apt-get update` (рис. 5.29).

Устанавливаем Java: `apt-get install oracle-java8-installer` (рис. 5.30).

```

vm@debianvm:~$ sudo apt-get update
Ign http://mirror.mephi.ru jessie InRelease
Get:1 http://mirror.mephi.ru jessie-updates InRelease [142 kB]
Get:2 http://ppa.launchpad.net trusty InRelease [15.5 kB]
Get:3 http://security.debian.org jessie/updates InRelease [63.1 kB]
Get:4 http://ppa.launchpad.net trusty/main Sources [1,607 B]
Get:5 http://security.debian.org jessie/updates/main Sources [133 kB]
Get:6 http://ppa.launchpad.net trusty/main amd64 Packages [3,375 B]
Hit http://mirror.mephi.ru jessie Release.gpg
Get:7 http://ppa.launchpad.net trusty/main Translation-en [1,556 B]
Get:8 http://security.debian.org jessie/updates/main amd64 Packages [241 kB]
Get:9 http://mirror.mephi.ru jessie-updates/main Sources [14.1 kB]
Get:10 http://security.debian.org jessie/updates/main Translation-en [131 kB]
Get:11 http://mirror.mephi.ru jessie-updates/main amd64 Packages/DiffIndex [3,472 B]
Get:12 http://mirror.mephi.ru jessie-updates/main Translation-en/DiffIndex [1,720 B]
Hit http://mirror.mephi.ru jessie Release
Get:13 http://mirror.mephi.ru jessie-updates/main amd64 2016-04-19-2053.08.pdiff [2,111 B]
Get:14 http://mirror.mephi.ru jessie-updates/main amd64 2016-04-23-1455.20.pdiff [378 B]
Get:15 http://mirror.mephi.ru jessie-updates/main amd64 2016-05-02-2123.23.pdiff [254 B]
Get:16 http://mirror.mephi.ru jessie-updates/main amd64 2016-05-02-2123.23.pdiff [254 B]
Get:17 http://mirror.mephi.ru jessie-updates/main 2016-04-19-2053.08.pdiff [2,245 B]
Get:18 http://mirror.mephi.ru jessie-updates/main 2016-04-19-2053.08.pdiff [2,245 B]
Hit http://mirror.mephi.ru jessie/main Sources
Hit http://mirror.mephi.ru jessie/main amd64 Packages
Hit http://mirror.mephi.ru jessie/main Translation-en
Fetched 757 kB in 5s (131 kB/s)
Reading package lists... Done

```

Рисунок 5.29 – Обновление списка пакетов

```

vm@debianvm:~$ sudo apt-get install oracle-java8-installer
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libasn1-8-heimdal libgssapi3-heimdal libhcrypto4-heimdal libheimbase1-heimdal libheimntlm0-heimdal libhx509-5-heimdal
  libkrb5-26-heimdal libroken18-heimdal libwind0-heimdal
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  gsfonts-x11
Suggested packages:
  bintfmt-support visualvm ttf-baekmuk ttf-unfonts ttf-unfonts-core ttf-kochi-gothic ttf-sazanami-gothic
  ttf-kochi-mincho ttf-sazanami-mincho ttf-arphic-uming
The following NEW packages will be installed:
  gsfonts-x11 oracle-java8-installer
0 upgraded, 2 newly installed, 0 to remove and 21 not upgraded.
Need to get 33.7 kB of archives.
After this operation, 162 kB of additional disk space will be used.
Do you want to continue? [Y/n]

```

Рисунок 5.30 – Установка java

Скачивание Apache Solr: `wget http://apache.mirror1.spango.com/lucene/solr/5.2.1/solr-5.2.1.tgz` (рис. 5.31).

Распаковка и установка: Распаковываем архив командой `tar xzf solr-5.2.1.tgz solr-5.2.1/bin/install_solr_service.sh --strip-components=2`. Устанавливаем Apache Solr командой `sudo bash ./install_solr_service.sh solr-5.2.1.tgz` (рис. 5.32).

Apache Solr по умолчанию работает на порту 8983. Проверяем работоспособность в браузере (рис. 5.33).

### 5.5.3 Добавление документов в поисковый индекс

Solr запущен, но на данный момент он не содержит каких-либо данных в поисковом индексе. Для отправки данных на сервер воспользуется shell скриптом, который будет брать содержимое

```

vm@debianvm:~$ wget http://apache.mirror1.spango.com/lucene/solr/5.2.1/solr-5.2.1.tgz
--2016-05-12 14:37:37-- http://apache.mirror1.spango.com/lucene/solr/5.2.1/solr-5.2.1.tgz
Resolving apache.mirror1.spango.com (apache.mirror1.spango.com)... 83.98.147.65
Connecting to apache.mirror1.spango.com (apache.mirror1.spango.com)|83.98.147.65|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 148849766 (142M) [application/x-gzip]
Saving to: 'solr-5.2.1.tgz'

solr-5.2.1.tgz          100%[=====>] 141.95M  1.55MB/s  in 78s

2016-05-12 14:38:56 (1.82 MB/s) - 'solr-5.2.1.tgz' saved [148849766/148849766]

vm@debianvm:~$ █

```

Рисунок 5.31 – Скачивание Apache Solr

```

Terminal - vm@debianvm: ~
File Edit View Terminal Tabs Help
Desktop Documents Downloads Music Pictures projects Public solr-5.2.1.tgz Templates Videos
vm@debianvm:~$ tar xzf solr-5.2.1.tgz solr-5.2.1/bin/install_solr_service.sh --strip-components=2
vm@debianvm:~$ sudo bash ./install_solr_service.sh solr-5.2.1.tgz
id: solr: no such user
Creating new user: solr
Adding system user `solr' (UID 120) ...
Adding new group `solr' (GID 127) ...
Adding new user `solr' (UID 120) with group `solr' ...
Creating home directory `/home/solr' ...
Extracting solr-5.2.1.tgz to /opt
Creating /etc/init.d/solr script ...
● solr.service - LSB: Controls Apache Solr as a Service
   Loaded: loaded (/etc/init.d/solr)
   Active: active (exited) since Thu 2016-05-12 14:40:10 EDT; 5s ago
   Process: 2301 ExecStart=/etc/init.d/solr start (code=exited, status=0/SUCCESS)

May 12 14:39:59 debianvm su[2303]: Successful su for solr by root
May 12 14:39:59 debianvm su[2303]: + ??? root:solr
May 12 14:39:59 debianvm su[2303]: pam_unix(su:session): session opened for user solr by (uid=0)
May 12 14:40:10 debianvm solr[2301]: [250B blob data]
May 12 14:40:10 debianvm solr[2301]: Started Solr server on port 8983 (pid=2359). Happy searching!
May 12 14:40:10 debianvm solr[2301]: [14B blob data]
Service solr installed.
vm@debianvm:~$ █

```

Рисунок 5.32 – Распаковка и установка

XML файлов из необходимой директории и отправлять их Solr. В результате мы добавили в Solr документы. Solr, в отличие от других систем хранит не документ целиком и выполняет поиск по нему, а разбивает XML-документ на поля и индексирует каждое из них (рис. 5.34).

#### 5.5.4 Формирование запросов

Так как документ в поисковом индексе представляет собой набор полей, то возможно формировать сложные поисковые запросы, которые при выполнении используют значения отдельных полей документа (рис. 5.35).

В области для ввода запросов присутствуют следующие поля:

- 1) q - основной запрос;
- 2) fq - фильтрующий запрос;
- 3) start - сдвиг в поиске;
- 4) rows - кол-во выводимых результатов;
- 5) fl - выводимые поля;
- 6) wt - формат вывода данных.

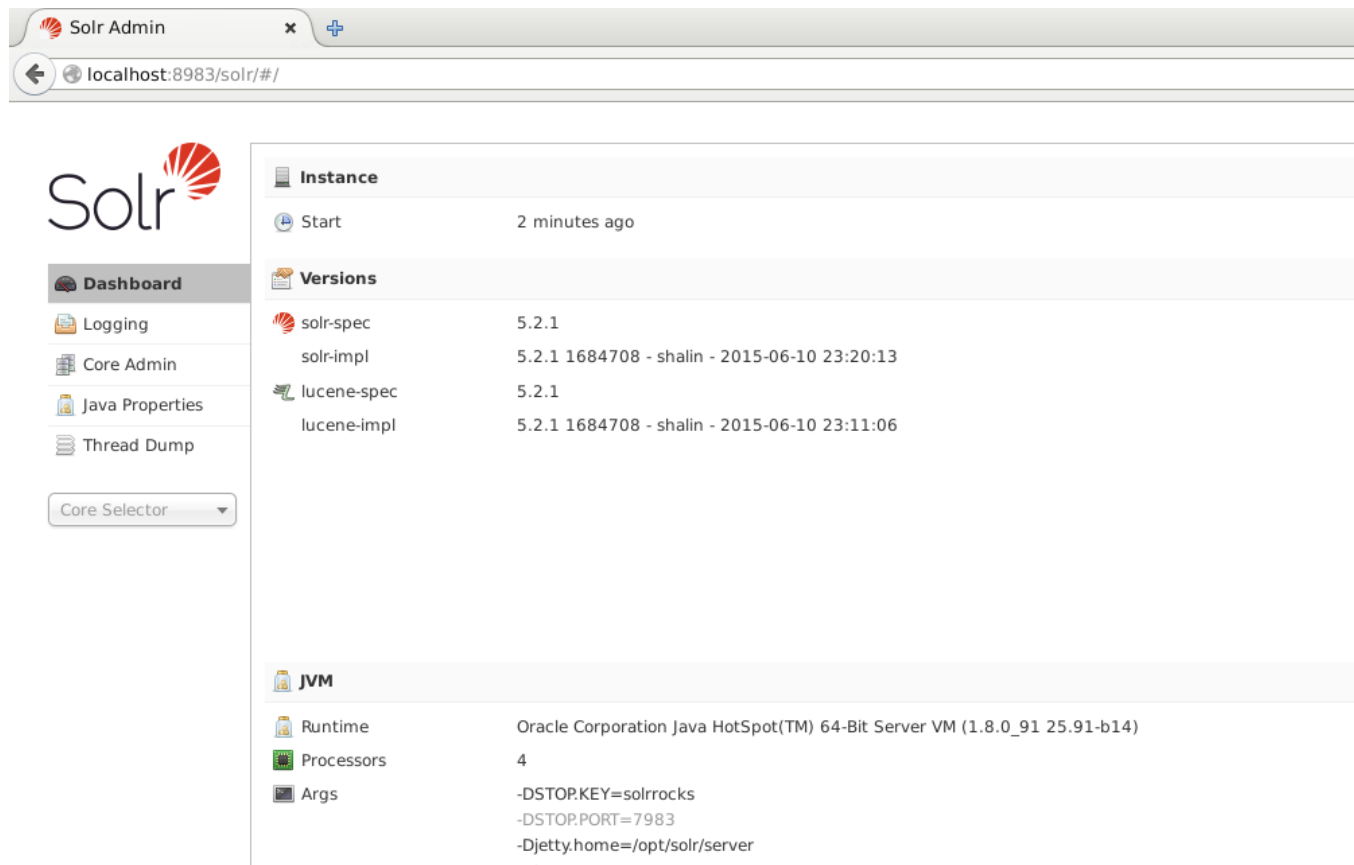


Рисунок 5.33 – Проверка работоспособности Apache Solr

```
vm@debianvm:~$ sudo su - solr -c "/opt/solr/bin/solr create -c getttingstarted -n data_driven_schema_configs"

Setup new core instance directory:
/var/solr/data/getttingstarted

Creating new core 'getttingstarted' using command:
http://localhost:8983/solr/admin/cores?action=CREATE&name=getttingstarted&instanceDir=getttingstarted

{
  "responseHeader":{
    "status":0,
    "QTime":1163},
  "core":"getttingstarted"}

vm@debianvm:~$
```

Рисунок 5.34 – Загрузка документов

Примеры запросов:

- 1) по содержанию значения в каком-либо поле документа (рисунок 5.36);
- 2) по содержанию в поле определенного значения (рисунок 5.37);
- 3) по значению поля, находящемуся в определенном интервале (рисунок 5.38, рисунок 5.39), с использованием вывода конкретных полей (рисунок 5.40);
- 4) с использованием булевых операторов (рисунок 5.41).

— common —

q

fq

-

+

sort

start, rows

0

5

fl

bookmarks\_param\_date\_added bookmarks\_param\_title

df

Raw Query Parameters

key1=val1&key2=val2

wt

json

Рисунок 5.35 – Область ввода запросов

Request-Handler (qt)

/select

— common —

q

Хабрахабр

fq

sort

start, rows

0 10

fl

df

Raw Query Parameters

key1=val1&key2=val2

wt

json

☒ indent

☐ debugQuery

☐ dismax

☐ edismax

☐ hl

☐ facet

☐ spatial

☐ spellcheck

Execute Query

http://localhost:8983/solr/gettingstarted/select?q=%D0%A5%D0%

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 1,
    "params": {
      "q": "Хабрахабр",
      "indent": "true",
      "wt": "json",
      "_": "1463134380250"
    }
  },
  "response": {
    "numFound": 38,
    "start": 0,
    "docs": [
      {
        "doc_type": [
          "bookmarks"
        ],
        "id": "chrome_10b019650483167b323ad339b28892d2",
        "application": [
          "chrome"
        ],
        "owner": [
          "blacksc"
        ],
        "bookmarks_param_date_added": [
          "вт мар 11 00:02:11 2014"
        ],
        "bookmarks_param_name": [
          "url"
        ],
        "bookmarks_param_value": [
          "http://habrahabr.ru/"
        ],
        "bookmarks_param_title": [
          "Хабрахабр"
        ]
      }
    ]
  }
}
```

Рисунок 5.36 – Запрос по содержанию в каком-либо поле документа



Request-Handler (qt)

/select

— common —

q

bookmarks\_param\_title:Хабрахабр

fq

sort

start, rows

0 10

fl

df

Raw Query Parameters

key1=val1&key2=val2

wt

json

☒ indent

☐ debugQuery

☐ dismax

☐ edismax

☐ hl

☐ facet

☐ spatial

☐ spellcheck

Execute Query

http://localhost:8983/solr/gettingstarted/select?q=bookmarks\_pa

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 0,
    "params": {
      "q": "bookmarks_param_title:Хабрахабр",
      "indent": "true",
      "wt": "json",
      "_": "1463134421983"
    }
  },
  "response": {
    "numFound": 1,
    "start": 0,
    "docs": [
      {
        "doc_type": [
          "bookmarks"
        ],
        "id": "chrome_10b019650483167b323ad339b28892d2",
        "application": [
          "chrome"
        ],
        "owner": [
          "blacksc"
        ],
        "bookmarks_param_date_added": [
          "вт мар 11 00:02:11 2014"
        ],
        "bookmarks_param_name": [
          "url"
        ],
        "bookmarks_param_value": [
          "http://habrahabr.ru/"
        ],
        "bookmarks_param_title": [
          "Хабрахабр"
        ],
        "..."
      }
    ]
  }
}
```

Рисунок 5.37 – Запрос по содержанию в поле определенного значения

q

download\_param\_time\_start:  
[2014-10-05T20:03:38Z TO NOW]

fq

sort

start, rows

0 10

fl

df

Raw Query Parameters

key1=val1&key2=val2

wt

json

☒ indent

☐ debugQuery

☐ dismax

☐ edismax

☐ hl

☐ facet

☐ spatial

☐ spellcheck

Execute Query

```

status : 0,
"QTime": 17,
"params": {
  "q": "download_param_time_start:[2014-10-05T20:03:38Z TO NOW]",
  "indent": "true",
  "wt": "json",
  "_": "1463134578798"
},
"response": {
  "numFound": 69,
  "start": 0,
  "docs": [
    {
      "doc_type": [
        "download"
      ],
      "id": "chrome_9c2bc2c5485ac851c06f1f07ad4e2bb1",
      "application": [
        "chrome"
      ],
      "owner": [
        "blacksc"
      ],
      "download_param_path": [
        "C:\\Users\\blacksc\\Downloads\\SmartGesture_Wi"
      ],
      "download_param_url": [
        "http://dlcdnet.asus.com/pub/ASUS/nb/DriversFor"
      ],
      "download_param_referrer": [
        "http://www.asus.com/ru/support/Download/3/411/"
      ],
      "download_param_size": [
        "23253_Kbytes"
      ],
      "download_param_time_start": [
        "2014-10-05T20:03:38Z"
      ],
      "download_param_time_end": [
        "2014-10-05T20:03:55Z"
      ],
      "_version_": 1534207421174513700
    }
  ]
}

```

Рисунок 5.38 – Запрос по значению поля даты, находящиеся в интервале от определенного значения до настоящего времени

q

download\_param\_time\_start:  
[2014-10-05T20:03:38Z TO  
2014-10-05T20:03:44Z]

fq

sort

start, rows  
0 10

fl

df

Raw Query Parameters  
key1=val1&key2=val2

wt  
json

☒ indent  
☐ debugQuery

☐ dismax  
☐ edismax  
☐ hl  
☐ facet  
☐ spatial  
☐ spellcheck

Execute Query

```

"status": 0,
"QTime": 1,
"params": {
  "q": "download_param_time_start:[2014-10-05T20:03:38Z TO 2014-10-05T20:03:44Z]",
  "indent": "true",
  "wt": "json",
  "_": "1463134632888"
},
"response": {
  "numFound": 1,
  "start": 0,
  "docs": [
    {
      "doc_type": [
        "download"
      ],
      "id": "chrome_9c2bc2c5485ac851c06f1f07ad4e2bb1",
      "application": [
        "chrome"
      ],
      "owner": [
        "blacksc"
      ],
      "download_param_path": [
        "C:\\Users\\blacksc\\Downloads\\SmartGesture_Win7_64_VI"
      ],
      "download_param_url": [
        "http://dlcdnet.asus.com/pub/ASUS/nb/DriversForWin8/Sm"
      ],
      "download_param_referrer": [
        "http://www.asus.com/ru/support/Download/3/411/0/2/Z9z"
      ],
      "download_param_size": [
        "23253_Kbytes"
      ],
      "download_param_time_start": [
        "2014-10-05T20:03:38Z"
      ],
      "download_param_time_end": [
        "2014-10-05T20:03:55Z"
      ],
      "_version_": 1534207421174513700
    }
  ]
}

```

Рисунок 5.39 – Запрос по значению поля даты, находящемуся в определенном интервале

Request-Handler (qt)

/select

— common —

q

download\_param\_time\_start:  
[2014-10-05T20:03:38Z TO  
2014-10-05T20:03:44Z]

fq

sort

start, rows

010

fl

id download\_param\_url download\_param\_path

df

Raw Query Parameters

key1=val1&key2=val2

wt

json

☒ indent

http://localhost:8983/solr/gettingstarted/select?q=download\_param\_time\_start%3A%5B2014-10-05T20%3A03%3A38Z+

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 4,
    "params": {
      "q": "download_param_time_start:[2014-10-05T20:03:38Z TO 2014-10-05T20:03:44Z]",
      "indent": "true",
      "fl": "id download_param_url download_param_path",
      "wt": "json",
      "_: "1463135995288"
    }
  },
  "response": {
    "numFound": 1,
    "start": 0,
    "docs": [
      {
        "id": "chrome_9c2bc2c5485ac851c06f1f07ad4e2bb1",
        "download_param_path": [
          "C:\\Users\\blacksc\\Downloads\\SmartGesture_Win7_64_VER201.zip"
        ],
        "download_param_url": [
          "http://dlcdnet.asus.com/pub/ASUS/nb/DriversForWin8/SmartGesture/SmartGesture_Win7_64_VER201.zip"
        ]
      }
    ]
  }
}
```

Рисунок 5.40 – Запрос с выводом определенных полей

Request-Handler (qt)

/select

— common —

q

id:chrome\_163b826ab5ba872c0445698d6e351899 OR bookmarks\_param\_title:Новости

fq

sort

start, rows

05

fl

bookmarks\_param\_date\_added bookmarks\_param\_title

df

Raw Query Parameters

key1=val1&key2=val2

wt

json

☒ indent

☐ debugQuery

☐ dismax

☐ edismax

http://localhost:8983/solr/gettingstarted/select?q=id%3Achrome\_163b826ab5ba872c0445698d6e351899 OR bookmarks\_param\_title:Новости

```

{
  "responseHeader": {
    "status": 0,
    "QTime": 1,
    "params": {
      "q": "id:chrome_163b826ab5ba872c0445698d6e351899 OR bookmarks_param_title:Новости",
      "indent": "true",
      "fl": "bookmarks_param_date_added bookmarks_param_title",
      "start": "0",
      "rows": "5",
      "wt": "json",
      "_": "1463139860111"
    }
  },
  "response": {
    "numFound": 2,
    "start": 0,
    "docs": [
      {
        "bookmarks_param_date_added": [
          "пн авг 4 17:55:35 2014"
        ],
        "bookmarks_param_title": [
          "Новости"
        ]
      },
      {
        "bookmarks_param_date_added": [
          "чт авг 11 20:45:23 2013"
        ],
        "bookmarks_param_title": [
          "BitTorrent индекс RuTracker.org (ex torrents.ru)"
        ]
      }
    ]
  }
}

```

Рисунок 5.41 – Запрос с использованием булевого оператора

## 5.6 Создание «бинарного» пакета DEB из исходных файлов программного комплекса «COEX»

Для распространения и установки программного комплекса «COEX» на электронные вычислительные машины был выбран формат двоичного пакета \*.DEB. Данный пакет включает в себя все необходимые файлы для работы программы, а также содержит список зависимостей. Он имеет строго типизированную структуру и используется операционными системами Unix. Строго типизированная структура позволяет операционной системе узнавать все нужные элементы для установки и работы с данным пакетом вне зависимости от программного обеспечения (далее ПО), находящейся в данном Deb-пакете, узнавать зависимости программных библиотек, необходимых для запуска программного продукта, содержащегося в двоичном пакете. Выбор данного формата пакета обусловлен возможностью его гибкой настройки для любого программного обеспечения (возможность использовать встроенные скрипты для настройки процесса установки ПО), поддержкой почти всеми операционными системами Unix. А также широкой распространенностью использования deb-пакетов в семействе операционных систем Unix в связи с высокой популярностью дистрибутивов операционных систем, в которой он распространяется. [9]

Для пакета \*.DEB содержащего программный комплекс «COEX» ранее использовалась ссылка на репозиторий содержащий данный пакет для его дальнейшей установки. При появлении новой версии программного комплекса «COEX» создается новый deb пакет, после чего пользователю нужно было зайти на сайт программного комплекса «COEX», и скачать новый deb пакет данного комплекса. Для автоматизации данного процесса было написано две программы, а также настроена операционная утилита CRON для запуска программ в определенный промежуток времени, определяемый в конфигурационном файле CRON. Первая программа запрашивает у пользователя права на внесение изменений в конфигурационный файл sources.list, затем находит данный файл и вносит изменения.

```
#!/bin/bash
echo "Install coex"
cd //
cd etc/apt
sudo su -c "echo 'deb http://rep.coex.su linux non-free' >> sources.list"
gpg --keyserver keyserver.ubuntu.com --recv F442F58696D6E45E
sudo su -c "gpg --export --armor F442F58696D6E45E" | sudo su -c "apt-key add --"
if [[ "$OSTYPE" -eq "Debian" ]]
then
    sudo su -c "aptitude update"
    sudo su -c "aptitude install coex"
    exit 1
elif [[ "$OSTYPE" -eq "Linux" ]]
then
    sudo su -c "apt-get update"
    sudo su -c "apt-get install coex"
fi
```

Рисунок 5.42 – Программа для внесения изменений в sources.list

Изменения включают в себя адрес до репозитория в сети Internet, где хранится deb пакет программного комплекса «COEX», а также запись ключа, которым подписывают скаченные файлы с репозитория, вносятся также команды по установке и обновлению пакета при скачивании через менеджера. Данные действия позволяют менеджеру обновлений операционной системы Linux следить за версиями deb пакета в репозитории, содержащего программный комплекс «COEX», при появлении новой версии менеджер сообщит пользователю о возможном обновлении. Первая программа

находится на сайте проекта «COEX», скачивается и запускается пользователем.

Вторая программа находится на сервере с репозиториями, и с помощью системной утилиты CRON запускается в определенное время, при наличии изменений из ветки master в системе контроля версий GIT.

```
#!/bin/bash
package_name="coex"
versionNewDeb=$(git describe --long)
#Eqvile old and new version *.deb package
valueOldContainVersionDeb=$(cat fileContainVersionCoex)
if [ "$valueOldContainVersionDeb" = "$versionNewDeb" ]
then
    echo "Version new package dont matches with old version package"
else
    echo "$versionNewDeb" > fileContainVersionCoex
    echo "$versionNewDeb"
    ./create_package.sh
    echo "Deb package full update"
fi
```

Рисунок 5.43 – Программа для запуска сборки по времени с помощью демона «CRON»

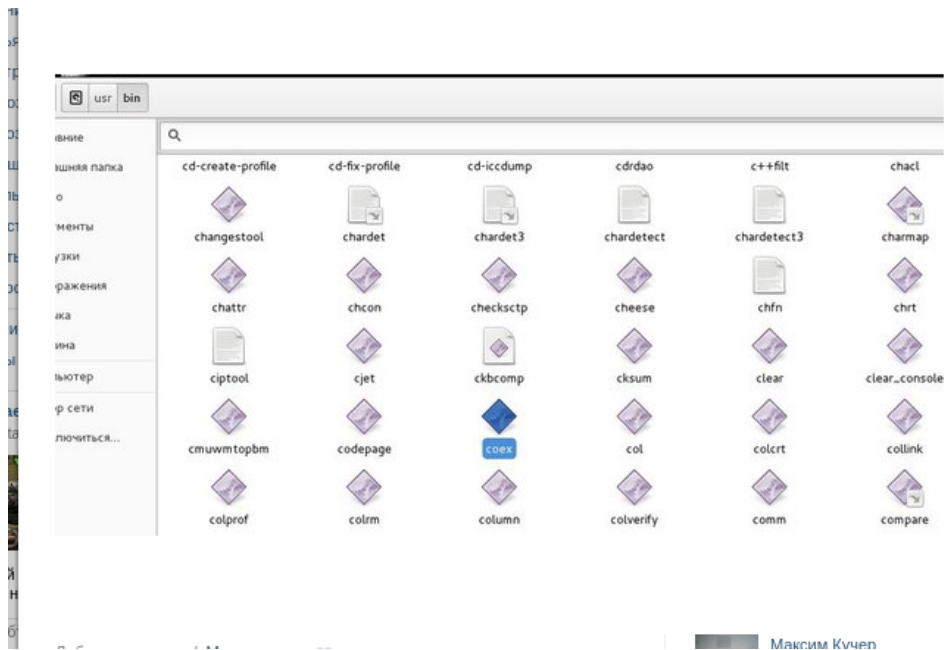
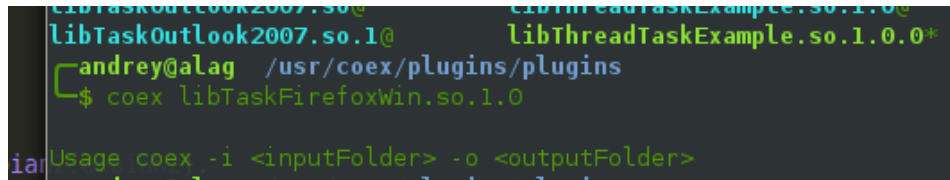
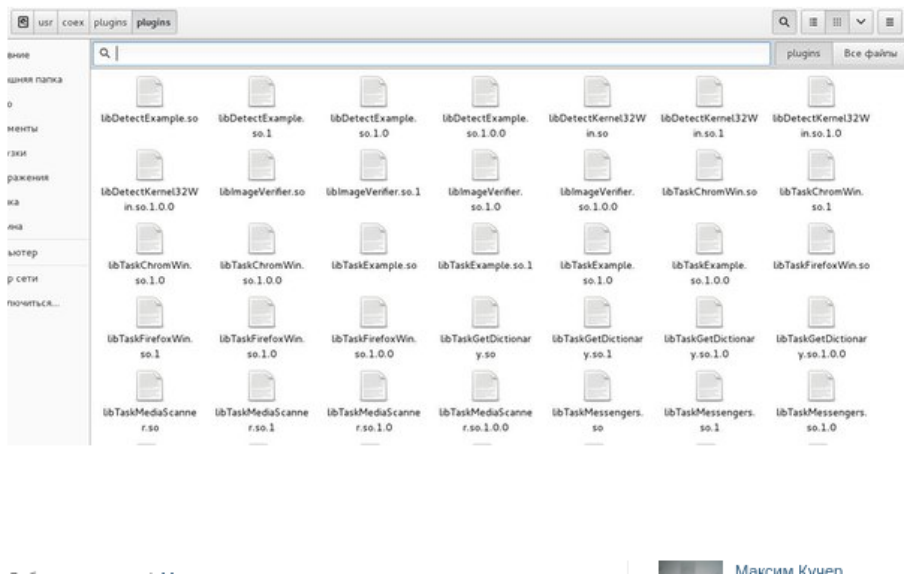
Скрипт-программа на основе сведений из системы контроля версий GIT проекта «COEX», создает версию для двоичного пакета формата deb, сам deb пакет создается при помощи скрипт-программы по созданию deb пакета из текста программ проекта.

```
#!/bin/bash
package_name="coex"
version=$(git describe --long)
maintainer_name=$(git config user.name)
maintainer_email=$(git config user.email)
echo "|-----"
echo "$ package_name ($version) ; urgency=low"
git tag -l | sort -u -r | while read TAG ; do
    echo
    if [ $NEXT ];then
        echo [$NEXT]
    else
        echo "[Current]"
    fi
    GIT_PAGER=cat git log --no-merges --format=" * %s" $TAG..$NEXT
    NEXT=$TAG
done
FIRST=$(git tag -l | head -1)
#echo
echo " -- $maintainer_name $maintainer_email $(date -R) --"
echo "|-----"
#GIT_PAGER=cat git log --no-merges --format=" * %s" $FIRST
```

Рисунок 5.44 – Программа для присвоения версии

Тестирование проводилось, на операционной системе «Debian 8.4», и ее наследниках(Ubuntu(16.04), Mint (17.3)).

В ходе тестирования проверялось корректность работы двоичного пакета после прохождения автоматизированной сборки, также проверялась функционирование программного комплекса «COEX», и корректность удаления проекта с компьютера пользователя. Результатом тестирования были выявлены проблемы с некоторыми зависимостями модулей, и связи модулей с программным ядром «COEX». После выявления проблем были перепроверены и исправлены зависимости, в модулях



где были найдены ошибки с зависимостями. А также переделана структура пакета, для исправления проблемы взаимодействия ядра «СОЕХ» и его модулей при установке через двоичный пакет.



## 5.7 Документирование плагинов

На данный момент основной репозиторий находится на ресурсе GitHub. Данный ресурс использует язык разметки Markdown (подробнее в разделе 3.3) и автоматически добавляет файл «Readme.md» к описанию программного модуля, если этот файл присутствует. В связи с этим было решено создать документацию плагинов, используя Markdown. Документация должна включать в себя:

- 1) Название плагина;
- 2) Версию плагина;
- 3) Автора плагина;
- 4) Описание плагина;
- 5) Требуемую операционную систему;
- 6) Версию ПО, с которым этот плагин работает;
- 7) Основные методы плагина с описанием входов и выходов.

Результат разработанной документации можно наблюдать на странице плагина в репозитории (рис. 5.48).

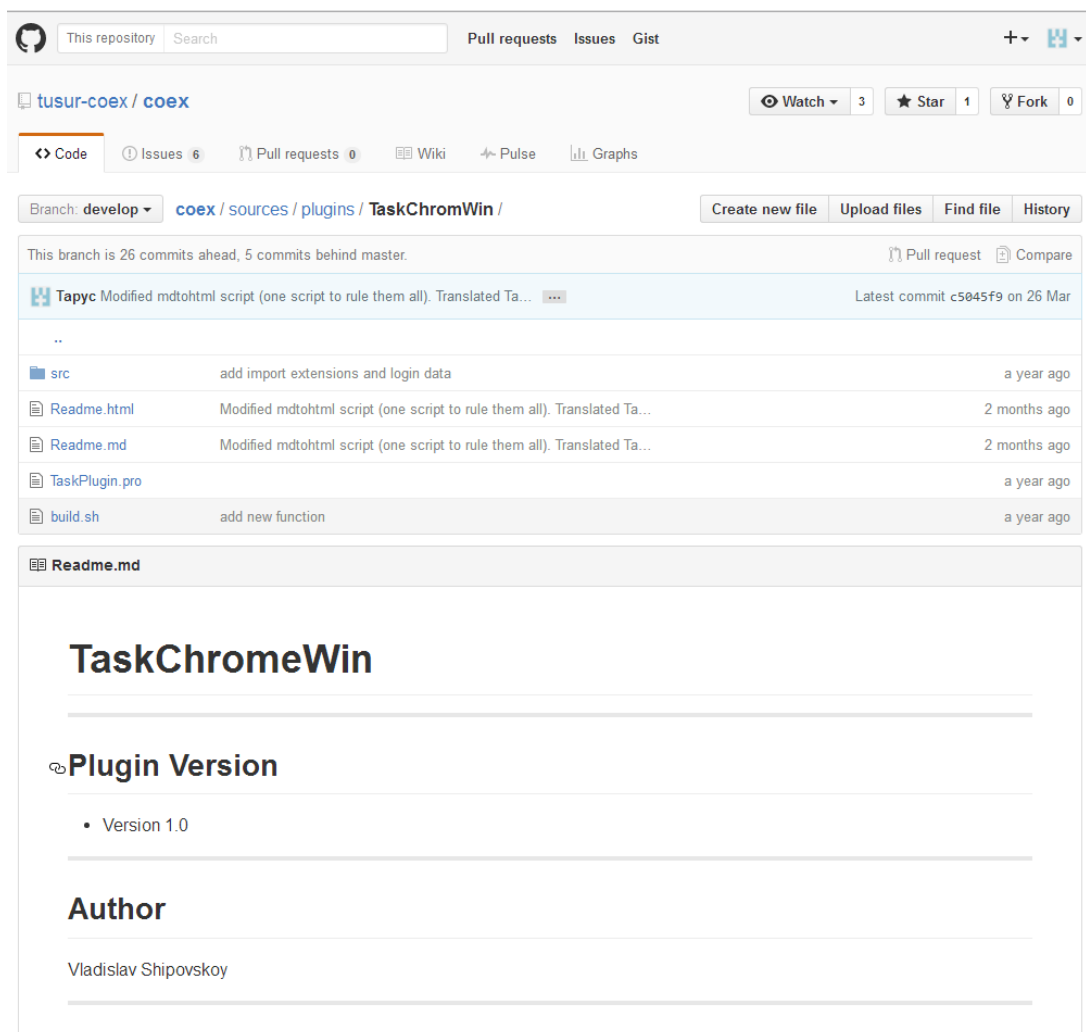


Рисунок 5.48 – Документация в веб-интерфейсе репозитория

Поскольку проект «COEX» имеет свою собственную веб-страницу, данную документацию также необходимо преобразовать в формат HTML, чтобы затем добавить на веб-страницу проекта. Для преобразования был разработан небольшой скрипт на языке Python (приложение Б). На рисунке 5.49 можно наблюдать ту же документацию, но в формате HTML.

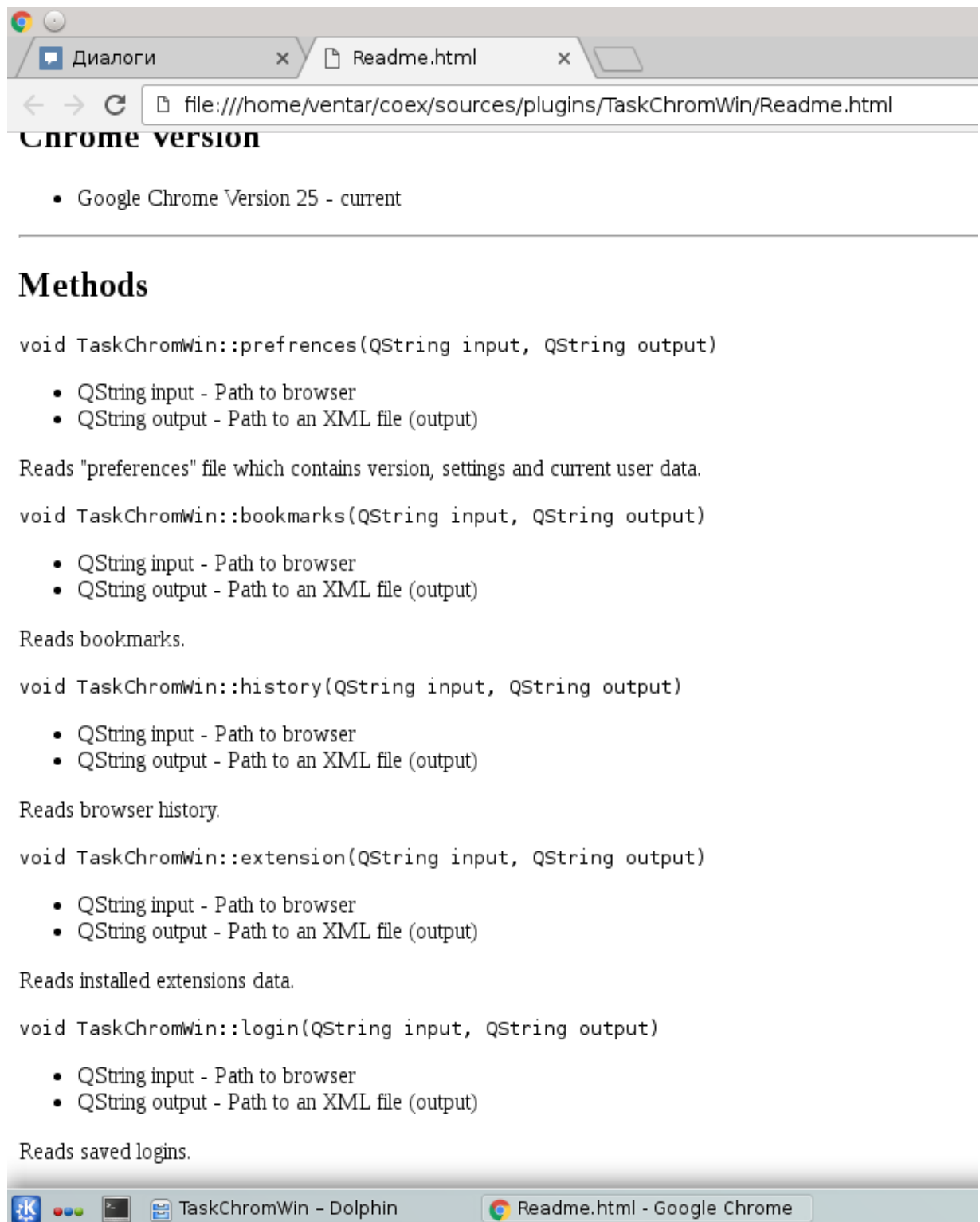


Рисунок 5.49 – Документация в формате HTML

## 5.8 Разработка и внедрение копии жесткого диска

Поскольку большое количество плагинов «СОЕХ» обращается к жесткому диску для поиска тех или иных файлов, что в свою очередь создает серьезную нагрузку на него, то было решено модифицировать архитектуру проекта с целью хранения копии информации о жестком диске. Данную информацию решено было хранить как поле объекта «config», к которому будут обращаться остальные плагины. Поле представляет из себя класс «Hdd» с атрибутом типа `QList<QDir>` (приложение В). Данный тип был выбран, поскольку он позволяет хранить данные о всех директориях и файлах внутри них, а также предоставляет удобные интерфейсы для доступа к ним. Методы класса «Hdd»:

- 1) `Hdd::Hdd(QString path);`
- 2) `Hdd::Hdd();`
- 3) `QFileInfoList getFiles(QStringList wildcardlist);`
- 4) `QFileInfoList getFiles(QString wildcard);`

Метод «`Hdd::Hdd(QString path)`» является конструктором класса. Переменная «path», подаваемая на вход метода является путем до начальной папки. Конструктор с помощью экземпляра класса «`QDirIterator`» посещает каждую папку в начальной папке и сохраняет данные о ней в переменную типа «`QDir`», после чего добавляет эту переменную к массиву «`QList<QDir>`», и наконец сохраняет полученный массив как поле класса. Алгоритм конструктора можно увидеть на рисунках 5.50 и 5.51.

Метод «`Hdd::Hdd()`» является деструктором класса.

Метод «`QFileInfoList getFiles(QStringList wildcardlist)`» возвращает объект «`QFileInfoList`» для всех файлов, которые соответствуют заданному массиву масок «`wildcardlist`». Алгоритм метода можно увидеть на рисунке 5.52.

Метод «`QFileInfoList getFiles(QString wildcard)`» выполняет ту же функцию, что и прошлый метод. Он является перегрузкой прошлого метода и принимает на вход одну маску вместо массива. Алгоритм метода можно увидеть на рисунке 5.53.

После разработки архитектуры класса, он был внедрен в «скелет» проекта. Класс конструируется перед работой плагинов, но после определения операционной системы.

Далее необходимо было изменить плагины, таким образом, чтобы они обращались к сохраненной копии диска вместо самого диска. Таким образом были изменены два плагина - «`TaskMediaScanner`» и «`TaskChromeWin`».

Теперь перед нами стояла задача сравнить нагрузку диска до и после внедрения класса «Hdd». Поскольку нами использовался удаленный репозиторий и система контроля версий git, то это не составило проблемы по причине того, что разработка класса велась в отдельной «ветке».

Было решено с помощью утилиты `iostat` замерить использование жесткого диска (в КБ/с) несколько раз до и после введения нового плагина и отфильтровать полученные результаты, чтобы учитывать исключительно нагрузку, создаваемую программой «СОЕХ». Для этого был разработан мультипоточный скрипт на языке Python, запускающий отдельно утилиту «`iostat`» и «СОЕХ» и фильтрующий результаты, сохраняемые утилитой «`iostat`» (приложение Г):

Далее результаты были обработаны, и на основании их был построен график, показывающий нагрузку на жесткий диск до и после внедрения класса «Hdd». Было решено оставить по 10 итераций на каждое измерение, поскольку после этого количества разница между итерациями была минималь-

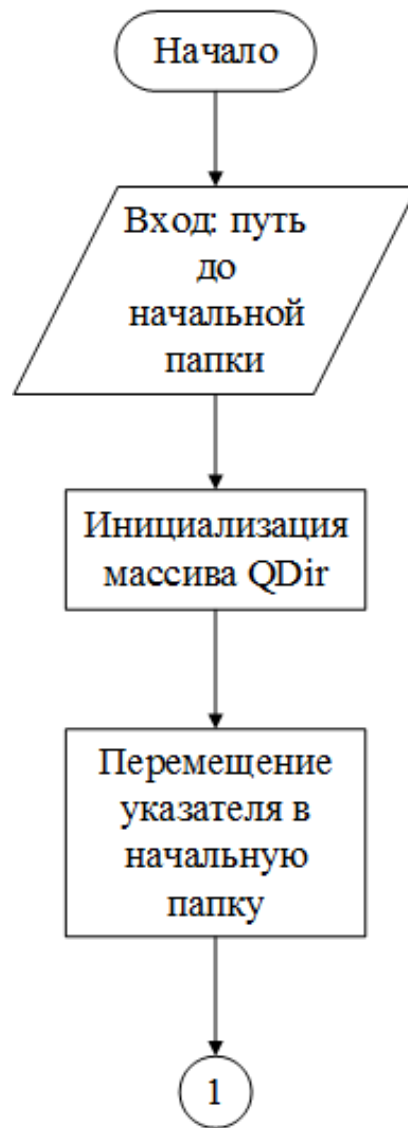


Рисунок 5.50 – Алгоритм конструктора класса «Hdd»

на и уже прослеживалась значимая разница между измерениями.

Из графика видно, что даже при изменении всего двух плагинов для использования новой архитектуры нагрузка на диск заметно снизилась. Так как на данный момент в проекте «СОЕХ» имеется 17 рабочих плагинов, преобразование каждого из них должно сильно сказаться на нагрузке жесткого диска в лучшую сторону.



Рисунок 5.51 – Продолжение алгоритма конструктора класса «Hdd»



Рисунок 5.52 – Продолжение алгоритма конструктора класса «Hdd»

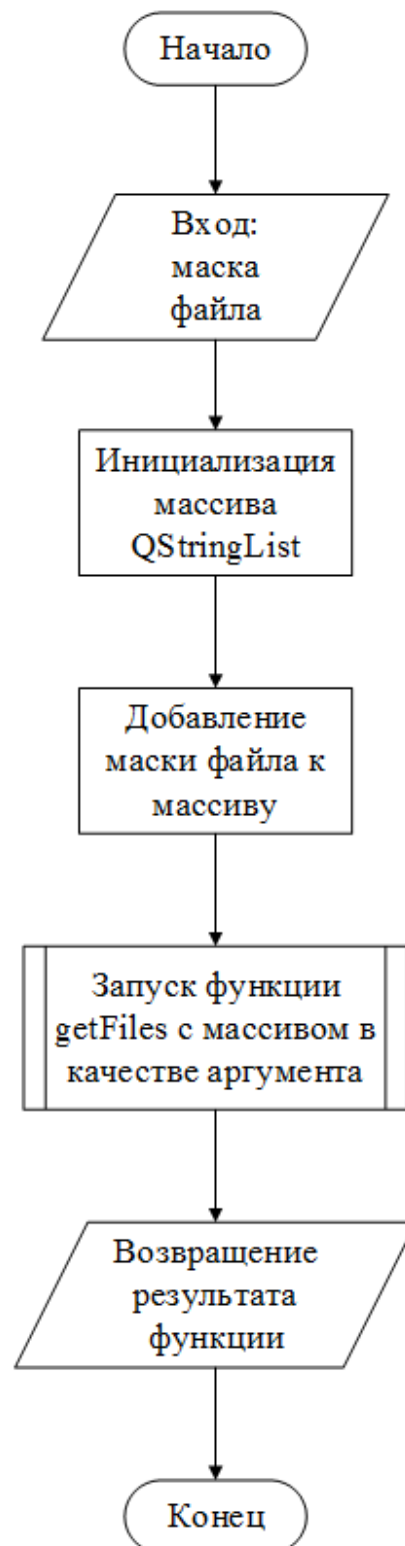


Рисунок 5.53 – Продолжение алгоритма конструктора класса «Hdd»

```

coex : bash - Konsole
File Edit View Bookmarks Settings Help
> Parsing is done
> Executing tasks . . .
--> Execute task: 'MediaScanner' by 'Ilya Bokov'
test folder 'WindowsXP_SP3_Pro'

> Loading plugins...
--> Plugin 'libDetectKernel32Win.so' . . . OK
----> Found detector 'detectByKernel32' by 'Dmitrii Nikiforov'
--> Plugin 'libTaskChromWin.so' . . . OK
----> Found task 'ChromWin' by 'Vlad Shipovskoy'
--> Plugin 'libTaskMediaScanner.so' . . . OK
----> Found task 'MediaScanner' by 'Ilya Bokov'
> Plugins loaded
> Detecting operation system . . .
--> Detected OS: 'Windows XP'
> Running hdd parser
> Parsing is done
> Executing tasks . . .
--> Execute task: 'ChromWin' by 'Vlad Shipovskoy'
--> Execute task: 'MediaScanner' by 'Ilya Bokov'
test folder 'Windows7_Ult'

> Loading plugins...
--> Plugin 'libDetectKernel32Win.so' . . . OK
----> Found detector 'detectByKernel32' by 'Dmitrii Nikiforov'
--> Plugin 'libTaskChromWin.so' . . . OK
----> Found task 'ChromWin' by 'Vlad Shipovskoy'
--> Plugin 'libTaskMediaScanner.so' . . . OK
----> Found task 'MediaScanner' by 'Ilya Bokov'
> Plugins loaded
> Detecting operation system . . .
--> Detected OS: 'Windows 7'
> Running hdd parser
> Parsing is done
> Executing tasks . . .
--> Execute task: 'ChromWin' by 'Vlad Shipovskoy'
--> Execute task: 'MediaScanner' by 'Ilya Bokov'
root@debian:/home/ventar/coex#

```

Рисунок 5.54 – Работа класса «Hdd» при запуске «COEX»



```

coex : bash - Konsole
File Edit View Bookmarks Settings Help
--> Execute task: 'MediaScanner' by 'Ilya Bokov'
test folder 'WindowsXP_SP3_Pro'

> Loading plugins...
--> Plugin 'libDetectKernel32Win.so' . . . OK
----> Found detector 'detectByKernel32' by 'Dmitrii Nikiforov'
--> Plugin 'libTaskChromWin.so' . . . OK
----> Found task 'ChromWin' by 'Vlad Shipovskoy'
--> Plugin 'libTaskMediaScanner.so' . . . OK
----> Found task 'MediaScanner' by 'Ilya Bokov'
> Plugins loaded
> Detecting operation system . . .
--> Detected OS: 'Windows XP'
> Executing tasks . . .
--> Execute task: 'ChromWin' by 'Vlad Shipovskoy'
--> Execute task: 'MediaScanner' by 'Ilya Bokov'
test folder 'Windows7_Ult'

> Loading plugins...
--> Plugin 'libDetectKernel32Win.so' . . . OK
----> Found detector 'detectByKernel32' by 'Dmitrii Nikiforov'
--> Plugin 'libTaskChromWin.so' . . . OK
----> Found task 'ChromWin' by 'Vlad Shipovskoy'
--> Plugin 'libTaskMediaScanner.so' . . . OK
----> Found task 'MediaScanner' by 'Ilya Bokov'
> Plugins loaded
> Detecting operation system . . .
--> Detected OS: 'Windows 7'
> Executing tasks . . .
--> Execute task: 'ChromWin' by 'Vlad Shipovskoy'
--> Execute task: 'MediaScanner' by 'Ilya Bokov'
15:38:49 3382 be/4 root 0.00 K/s 7.57 K/s 0.00 % 0.00 % ./coex -i ../tmp/test-data/WindowsXP_SP3_Pro -o ../tmp/tests/WindowsXP_SP3_Pro
15:38:50 3382 be/4 root 0.00 K/s 7.53 K/s 0.00 % 0.00 % ./coex -i ../tmp/test-data/WindowsXP_SP3_Pro -o ../tmp/tests/WindowsXP_SP3_Pro
15:38:54 3382 be/4 root 0.00 K/s 3.60 K/s 0.00 % 0.00 % ./coex -i ../tmp/test-data/WindowsXP_SP3_Pro -o ../tmp/tests/WindowsXP_SP3_Pro
15:39:12 3382 be/4 root 0.00 K/s 14.58 K/s 0.00 % 0.00 % ./coex -i ../tmp/test-data/WindowsXP_SP3_Pro -o ../tmp/tests/WindowsXP_SP3_Pro
15:39:14 3388 be/4 root 0.00 K/s 11.23 K/s 0.00 % 0.00 % ./coex -i ../tmp/test-data/Windows7_Ult -o ../tmp/tests/Windows7_Ult
15:39:42 3388 be/4 root 0.00 K/s 1255.03 K/s 0.00 % 0.00 % ./coex -i ../tmp/test-data/Windows7_Ult -o ../tmp/tests/Windows7_Ult
root@debian:/home/ventar/coex#

```

Рисунок 5.55 – Результат работы скрипта

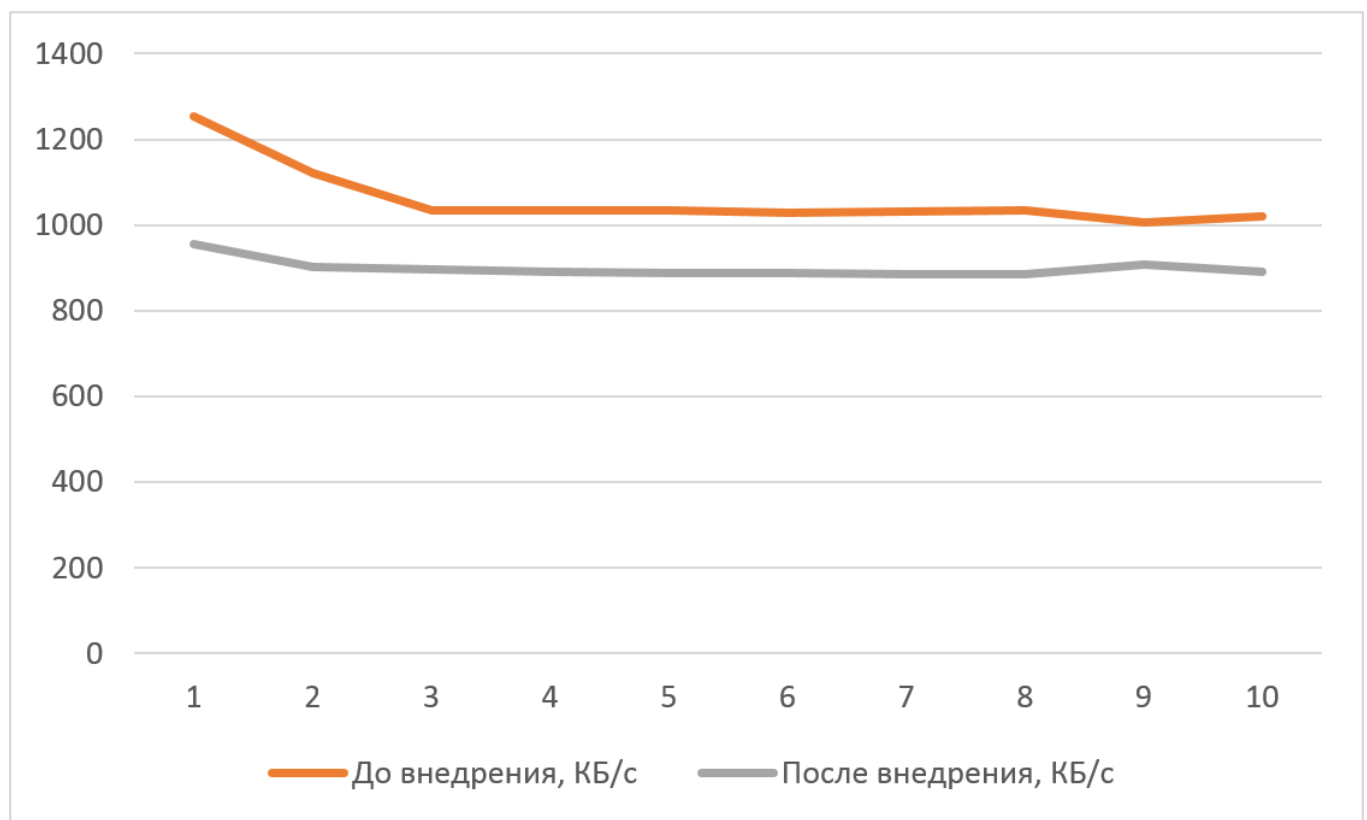


Рисунок 5.56 – Сравнение нагрузки на жесткий диск до и после изменения архитектуры

## 5.9 Многопоточное программирование

Одной из поставленных в данном семестре задач стало изучение возможностей многопоточного программирования с использованием программной библиотеки Qt. Программирование потоков осуществляется с помощью класса `QThreads`, а также механизма сигналов и слотов.

Все это необходимо для того, чтобы реализовать в системе «СОЕХ» параллельное выполнение программных модулей («плагинов»), осуществляющих поиск остаточных данных с образа системы, изображений, различных файлов и т.д.

### 5.9.1 Сигналы и слоты

Сигналы и слоты используются для связи между объектами. Механизм сигналов и слотов — это основная особенность Qt и, вероятно, основная часть Qt, которая больше всего отличается по функциональности от других библиотек.

Более старые инструментарии обеспечивают подобную связь с помощью функций обратного вызова. Обратный вызов — это указатель на функцию. Если необходимо, чтобы функция обработки уведомила о некотором событии, ей передается указатель на другую функцию (отзыв). Функция обработки вызовет функцию обратного вызова, когда это будет уместно. Но данный подход имеет два фундаментальных недостатка: во-первых, он не типобезопасен. Мы некогда не сможем проверить, что функция обработки вызывает отзыв с правильными аргументами. Во-вторых, этот метод жестко связан с функцией обработки, так как она должна знать, какой отзыв вызывать.

В Qt используется техника, альтернативная функциям обратного вызова: механизм сигналов и слотов. Сигнал испускается, когда происходит определенное событие. Слот — это функция, вызываемая в ответ на определенный сигнал.

Этот механизм типобезопасен: сигнатура сигнала должна соответствовать сигнатуре принимающего слота (фактически, слот может иметь более короткую сигнатуру, чем сигнал, который он получает, поскольку может игнорировать лишние аргументы). Сигналы и слоты связаны нежестко: класс, испускающий сигналы, не знает и не интересуется, который из слотов получит сигнал. Механизм сигналов и слотов Qt гарантирует, что, если сигнал соединен со слотом, слот будет вызываться с параметрами сигнала в нужный момент. Сигналы и слоты могут иметь любое количество аргументов любых типов. Они полностью типобезопасны.

Все классы, наследуемые от `QObject` или одного из его подклассов (например, `QWidget`) могут содержать сигналы и слоты. Сигналы испускаются при изменении объектом своего состояния, если это изменение может быть интересно другим объектам. Все объекты делают это для связи с другими объектами. Их не заботит, получает ли кто-нибудь испускаемые ими сигналы. Это является истинной инкапсуляцией информации, и она гарантирует, что объекты могут использоваться как отдельные компоненты программного обеспечения.

Слоты могут получать сигнал, но они также являются обыкновенными функциями-членами. Также, как объект не знает, получает ли кто-нибудь сигналы, испускаемые им, слоты не знают, существуют ли сигналы, с ними связанные. Это гарантирует, что можно создать полностью независимые Qt-компоненты.

Можно присоединять к одному слоту столько сигналов, сколько необходимо, и один сигнал может быть соединен со столькими слотами, сколько требуется. Также возможно соединение сиг-

нала непосредственно с другим сигналом (второй сигнал будет испускаться немедленно всякий раз, когда испускается первый).

Вместе сигналы и слоты представляют собой мощный механизм компонентного программирования. Графическое представление связи сигналов и слотов различных объектов можно увидеть на рисунке 5.57. [10]

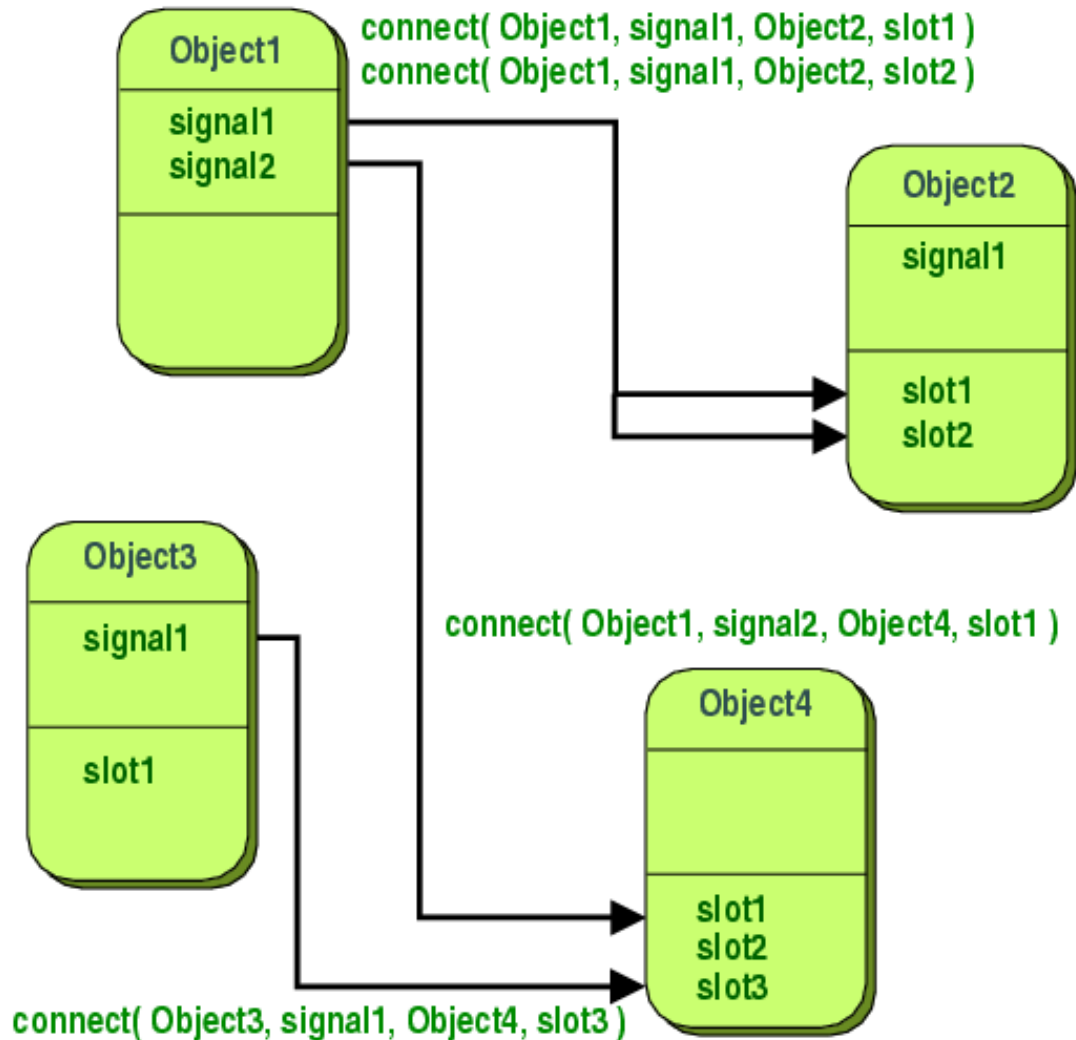


Рисунок 5.57 – Механизм сигналов и слотов для связи объектов в Qt

### 5.9.2 Потоки QThreads

В многопоточных приложениях, обслуживание интерфейса производится в отдельном потоке, а обработка данных – в другом (одном или нескольких) потоке. В результате приложение сохраняет возможность откликаться на действия пользователя даже во время интенсивной обработки данных. Еще одно преимущество многопоточности – на многопроцессорных системах различные потоки могут выполняться на различных процессорах одновременно, что несомненно увеличивает скорость исполнения.

Для реализации потоков Qt предоставляет класс QThread.

Поток — это независимая задача, которая выполняется внутри процесса и разделяет вместе с ним общее адресное пространство, код и глобальные данные.

Процесс, сам по себе, не является исполнительной частью программы, поэтому для исполнения программного кода он должен иметь хотя бы один поток (далее – основной поток). Конечно, можно создавать и более одного потока. Вновь созданные потоки начинают выполняться сразу же, параллельно с главным потоком, при этом их количество может изменяться — одни создаются, другие завершаются. Завершение основного потока приводит к завершению процесса, независимо от того, существуют другие потоки или нет. Создание нескольких потоков в процессе получило название многопоточность. [11]

Для использования многопоточности нужно унаследовать класс от `QThread`. Чтобы запустить поток, нужно вызвать метод `start()`.

Каждый поток может иметь собственный цикл обработки событий. Главный поток начинает цикл обработки событий, используя `QCoreApplication::exec()`; другие потоки могут начать свои циклы обработки событий, используя `QThread::exec()`.

Цикл обработки событий потока делает возможным использование потоком некоторых неграфических классов Qt, которые требуют наличия цикла обработки событий (такие как `QTimer`, `QTcpSocket` и `QProcess`). Это также даёт возможность соединить сигналы из любых потоков со слотами в определённом потоке (рис. 5.58). [12]

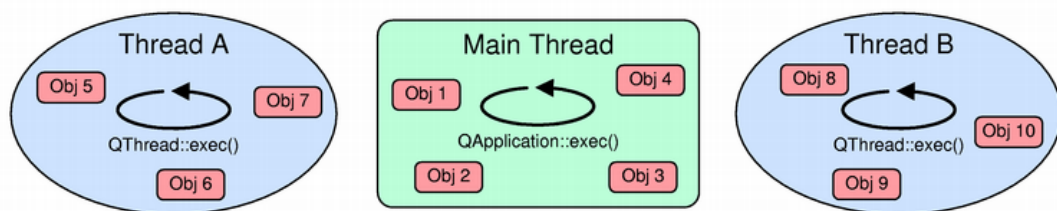


Рисунок 5.58 – Цикл обработки событий потоков в Qt

Для того, чтобы запускать программные модули на выполнение в нескольких потоках и должным образом завершать их выполнение, понадобилось написать контроллер — объект, который создает потоки для уже существующих объектов (самих плагинов), перемещает эти объекты в созданные потоки. Далее он запускает их выполнение при помощи метода `Controller::start_threads()`. После того, как каждый поток завершается, он посылает сигнал контроллеру о завершении `finished()` и переходит в режим ожидания. Когда все потоки завершаются, контроллер задействует слот `stop_threads()`, предназначенный для того, чтобы послать сигнал об успешном завершении работы как всех потоков, так и работы самого контроллера, основной программе. При этом для связи сигналов и слотов используется функция `connect(object_1, SIGNAL(signal_1), object_2, SLOT(slot_2))`.

Поскольку главный поток (основная программа) начинает цикл обработки событий, используя `QCoreApplication::exec()`, при получении сигнала `finished()`, сгенерированного контроллером, цикл обработки событий главного потока прерывается и главная программа успешно завершается.

Блок-схема алгоритма работы основной программы представлена на рисунке 5.59, блок-схема контроллера — на рисунке 5.60



Рисунок 5.59 – Блок-схема алгоритма работы основной программы qthreads



Рисунок 5.60 – Блок-схема алгоритма работы контроллера qthreads

Результат вывода программы представлен на рисунке 5.61.

```

Guake Terminal
-rw-rw-r-- 1 marina marina 333 мая 1 19:21 worker.h
-rw-rw-r-- 1 marina marina 9512 мая 1 20:35 worker.o
marina@marina-530U3BI-530U4BI-530U4BH: /storage/tusur/8_semester/ГП0/qthreads/qthreads$ ./qthreads
Starting executing...

thread started
thread started
message: "aaaaaaa"
thread started
message: "aaaaaaa"
message: "aaaaaaa"
message: "aaaaaaa"
finished work
message: "bbbbbbb"
message: "bbbbbbb"
message: "bbbbbbb"
message: "bbbbbbb"
message: "ddddd"
message: "bbbbbbb"
message: "ddddd"
finished work
message: "ddddd"
message: "ddddd"
message: "ddddd"
finished work
threads are finished
marina@marina-530U3BI-530U4BI-530U4BH: /storage/tusur/8_semester/ГП0/qthreads/qthreads$
  
```

Diagram of Thread A:

```

graph TD
    subgraph Thread_A [Thread A]
        direction TB
        Obj5[Obj 5] --> Obj7[Obj 7]
        Obj7 --> Obj6[Obj 6]
        Obj6 --> Obj5
    end
    QThread[QThread::exec()]
  
```

Рисунок 5.61 – Вывод программы qthreads

Кроме написания потокового контроллера был доработан плагин ThreadTaskICQ таким образом, чтобы учитывались особенности работы с механизмом сигналов и слотов. Только после этого плагин может быть запущен в потоке с использованием класса QThreads.

### 5.9.3 Итоги работы за семестр

Таким образом, в течение семестра была написана рабочая программа-реализация многопоточного программирования с использованием программной кроссплатформенной библиотеки Qt. Изучены некоторые особенности работы механизма слотов и сигналов. Доработан модуль ThreadTaskICQ. В дальнейшем планируется имплементировать написанный контроллер потоков под архитектуру системы «СОЕХ», а также дописать должным образом плагины для реализации возможности выполнения программных модулей в потоках с использованием слотов и сигналов.

### Заключение

В данном семестре нашей группой была выполнена часть работы по созданию автоматизированного программного комплекса для проведения компьютерной экспертизы. Основной целью в данном семестре стала подготовка проекта «СОЕХ» к релизу, для чего были разработаны веб-сайт и репозиторий проекта, графический интерфейс пользователя, доработаны некоторые из программных модулей, собран «бинарный» пакет для установки и распространения системы «СОЕХ».



## Список использованных источников

- 1 Федотов Николай Николаевич. Форензика - компьютерная криминалистика. Юрид. мир, 2007. 432 с.
- 2 Scott Chacon. Pro Git : professional version control. 2011. — Режим доступа: <http://progit.org/ebook/progit.pdf>.
- 3 С.М. Львовский. Набор и вёрстка в системе L<sup>A</sup>T<sub>E</sub>X. МЦНМО, 2006. С. 448.
- 4 И. А. Чеботаев, П. З. Котельников. L<sup>A</sup>T<sub>E</sub>X по-русски. Сибирский Хронограф, 2004. 489 с.
- 5 Qt Documentation [Электронный ресурс]. — Режим доступа: <http://qt-project.org/doc>.
- 6 Всё о кроссплатформенном программировании - Qt [Электронный ресурс]. — Режим доступа: <http://doc.crossplatform.ru/qt>.
- 7 Learn Git and GitHub without any code [Электронный ресурс]. — Режим доступа: <https://github.com/> (дата обращения: 25.04.2016).
- 8 Справочник по XML-стандартам [Электронный ресурс]. — Режим доступа: [http://msdn.microsoft.com/ru-ru/library/ms256177\(v=vs.110\).aspx](http://msdn.microsoft.com/ru-ru/library/ms256177(v=vs.110).aspx).
- 9 Tecmint: Linux Howtos, Tutorials and Guides [Электронный ресурс]. — Режим доступа: <http://www.tecmint.com/> (дата обращения: 15.03.2016).
- 10 Сигналы и слоты [Электронный ресурс]. — Режим доступа: <http://doc.crossplatform.ru/qt/4.3.2/signalsandslots.html> (дата обращения: 1.04.2016).
- 11 Процессы и потоки в Qt [Электронный ресурс]. — Режим доступа: <http://qt-doc.ru/processy-i-potoki-v-qt.html> (дата обращения: 1.04.2016).
- 12 Поддержка потоков в Qt [Электронный ресурс]. — Режим доступа: <http://doc.crossplatform.ru/qt/4.4.3/threads.html> (дата обращения: 1.04.2016).

Приложение А  
(Обязательное)  
Компакт-диск

Компакт-диск содержит:

- электронную версию пояснительной записки в форматах \*.tex и \*.pdf;
- актуальную версию программного комплекса для проведения компьютерной экспертизы;
- тестовые данные для работы с программным комплексом.

Приложение Б  
Md to html script

```
import markdown2
import os
import glob

def compose_anycase(string):
    result = ""
    for letter in string:
        result += "[%s%s]" % (letter.lower(), letter.upper())
    return result

path = os.path.join("sources", "plugins", "*")
readmes = glob.glob(os.path.join(path, compose_anycase("readme.md")))

for readme in readmes:
    outpath = "%s%s%s" % (os.path.dirname(readme), os.sep, "Readme.html")
    with open(outpath, 'w') as outfile, open(readme, 'r') as infile:
        file = infile.read()
        html = markdown2.markdown(file).encode('utf-8')
        outfile.write(html)
```

## Приложение В

## Hdd class

```

#include "hdd.h"
#include <QDebug>

Hdd::Hdd(QString path) {
    QDirIterator dirPath(path, QDir::Dirs | QDir::NoSymLinks
        | QDir::Hidden, QDirIterator::Subdirectories);
    QList<QDir> dirList;

    while (dirPath.hasNext())
    {
        QDir directory(dirPath.next());
        if (!dirList.contains(directory))
        {
            dirList.append(directory);
        }
    }

    this->infoList = dirList;

    //debug
    /*
    QFile file("/home/ventar/test/test.txt");
    QStringList wildcard = (QStringList() << "*.jpg");
    if (file.open(QIODevice::WriteOnly))
    {
        foreach(QDir directory, this->infoList)
        {
            QTextStream stream(&file);
            stream << directory.absolutePath() << endl;
            QFileInfoList list = directory.entryInfoList(QDir::Files
                | QDir::NoSymLinks | QDir::Hidden);
            foreach (QFileInfo fileInfo, list)
            {
                stream << fileInfo.absoluteFilePath() << endl;
            }
        }
    }
    */
}

```

```
Hdd::~Hdd() {
```

```
}
```

```
QFileInfoList Hdd::getFiles(QStringList wildcardlist) {
```

```
    QFileInfoList allists;
```

```
    foreach(QDir dir, Hdd::infoList) {
```

```
        allists.append(dir.entryInfoList( wildcardlist, QDir::Files  
            | QDir::NoSymLinks | QDir::Hidden ));
```

```
    }
```

```
    return allists;
```

```
}
```

```
QFileInfoList Hdd::getFiles(QString wildcard) {
```

```
    QStringList wildcardlist;
```

```
    wildcardlist.append(wildcard);
```

```
    return Hdd::getFiles(wildcardlist);
```

```
}
```

Приложение Г  
Disk usage logging script

```

import os
import sys
from threading import Thread

class IotopThread(Thread):
    def __init__(self):
        Thread.__init__(self, target=self.main)
        self.daemon = True
        self.time = 300
    def main(self):
        print "Starting 'iotop'"
        os.system("iotop -botqqqk --iter={0}
            >> /var/log/iotop".format(self.time))

        sys.exit()

class TestPlugin(Thread):
    def __init__(self):
        Thread.__init__(self, target=self.main)
        self.daemon = True
    def main(self):
        print "Starting 'testplugin.sh'"
        os.system("/home/ventar/coex/test.sh")

        sys.exit()

io_thread = IotopThread()
plugin_thread = TestPlugin()
io_thread.start()
plugin_thread.start()
while (io_thread.isAlive() or plugin_thread.isAlive()):
    pass
with open("/var/log/iotop", "r") as infile:
    file = infile.read()
    lines = file.split("\n")
    lines = [x for x in lines if "coex" in x]
with open("/home/ventar/result.txt", "w") as outfile:
    for line in lines:
        print line
        outfile.write(line + "\n")

```