

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

«_____» _____ 2015г.

КОМПЬЮТЕРНАЯ ЭКСПЕРТИЗА

Отчет по групповому проектному обучению

Группа КИБЭВС-1401

Ответственный исполнитель

студент гр. 722

_____ О.В. Лобанов

«_____» _____ 2015г.

Научный руководитель

аспирант каф. КИБЭВС

_____ А.И. Гуляев

«_____» _____ 2015г.

РЕФЕРАТ

Курсовая работа содержит 44 страниц, 44 рисунка, , 11 источников, 1 приложение.

КОМПЬЮТЕРНАЯ ЭКСПЕРТИЗА, ФОРЕНЗИКА, ЛОГИ, QT, XML, GIT, LATEX, MOZILLA THUNDERBIRD, MS OUTLOOK, WINDOWS, PST, MSG, RTF, HTML, БИБЛИОТЕКИ, РЕПОЗИТОРИЙ, ПОЧТОВЫЙ КЛИЕНТ, SQLLITE, РЕЕСТР, МЕТА-ДАННЫЕ, READPST, JPEG, PNG, ID3V1, JFIF, RIFF, CHUNK, DBX, C++.

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

Задачей, поставленной на данный семестр, стало написание программного комплекса, имеющего следующие возможности:

1) ПРАВИТЬ

Результаты работы в данном семестре:

– ПРАВИТЬ

Пояснительная записка выполнена при помощи системы компьютерной вёрстки L^AT_EX.

Список исполнителей

ПРАВИТЬ

Лобанов О.В. – программист, ответственный исполнитель, ответственный за разработку функций сбора информации из реестра.

Шиповской В.В. – программист, ответственный за написание части системы для сбора и обработки информации из браузера Google Chrome.

Серяков А.В. – программист, ответственный за написание части системы для сбора информации из почтового клиента MS Outlook.

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

Кучер М.В. – программист, ответственный за написание части системы для сбора информации об установленном ПО по остаточным файлам.

Терещенко Ю.А. – программист, ответственный за написание части системы для сбора информации из почтового клиента Mozilla Thunderbird.

Мейта М.В. – документатор, ответсвенный за верстку необходимой документации в системе L^AT_EX.

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 Qt - кроссплатформенный инструментарий разработки ПО	11
3.3.1 Автоматизация поиска журнальных файлов	13
3.3.2 Реализация сохранения результатов работы программного комплекса в XML	13
3.4 GitLab	13
4 Технические характеристики	13
4.1 Требования к аппаратному обеспечению	13
4.2 Требования к программному обеспечению	13
4.3 Выбор единого формата выходных файлов	13
5 Разработка программного обеспечения	14
5.1 Архитектура	14
5.1.1 Основной алгоритм	14
5.1.2 Описание основных функций модуля системы	15
5.2 Плагин SearchProgram	18
5.2.1 Исправление скриптов clone.sh и test.sh	18
5.2.2 Создание репозитория	19
5.3 мессенджеры	22
5.3.1 TaskFirefoxWin	22
5.3.2 TaskThunderbirdWin	22
5.4 Написание графического интерфейса пользователя для системы «COEX»	27
5.5 Доработка программного модуля Media Scanner	28
5.6 Сбор информации из почтового клиента MS Outlook	29
5.7 Создание «бинарного» пакета DEB из исходных файлов программного комплекса «COEX»	30
5.7.1 Подробное рассмотрение набора bash-скриптов для формирования *.deb пакета и формирования changelog	31
5.7.2 Тестирование созданного пакета	33
5.8 Разработка веб-сайта проекта «COEX»	37
5.8.1 Дизайн веб-сайта проекта	37
5.8.2 Вёрстка дизайна	38
5.8.3 Регистрация домена	38
5.8.4 Настройка WEB-сервера	39
5.8.5 Контент	39
6 Задачи на следующий семестр	41
Заключение	42

Приложение А Компакт-диск	44
-------------------------------------	----

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:

- определение индивидуальных задач для каждого участника проектной группы;
- исследование предметных областей в рамках индивидуальных задач;
- реализация новых программных модулей и доработка уже существующих.

Задачи по проектированию модулей: **ПРАВИТЬ**

- 1) сбор и анализ информации из реестра Windows;
- 2) сбор и анализ информации из браузера Google Chrome;
- 3) сбор и анализ информации из почтового клиента Mozilla Thunderbird;

- 4) сбор и анализ информации из почтового клиента MS Outlook;
- 5) поиск медиа-файлов (аудио, видео, изображение) и извлечение мета-данных из них;
- 6) сбор информации об установленном ПО по остаточным файлам.

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_EX

T_EX — это созданная американским математиком и программистом Дональдом Кнутом система для вёрстки текстов. Сам по себе T_EX представляет собой специализированный язык программирования. Каждая издательская система представляет собой пакет макроопределений этого языка.

L^AT_EX — это созданная Лэсли Лэмпортом издательская система на базе T_EX'a [3] L^AT_EX 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^AT_EX так как совместно с системой контроля версий Git он предоставляет возможность совместного создания и редактирования документов. Огромным достоинством системы L^AT_EX то, что создаваемые с её помощью файлы обладают высокой степенью переносимости. [4]

Совместно с L^AT_EX часто используется BibT_EX — программное обеспечение для создания форматированных списков библиографии. Оно входит в состав дистрибутива L^AT_EX и позволяет создавать удобную, универсальную и долговечную библиографию. BibT_EX стал одной из причин, по которой нами был выбран L^AT_EX для создания документации.

3.3 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 вы-

равнивания 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 использующихся 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 (например, 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 &, 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.

3.3.1 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.

3.3.2 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 реализованно в классе `WriteMessage`. В методе `WriteMessages`, структура которого представлена на UML диаграмме в разделе Архитектура.

3.4 GitLab

ПРАВИТЬ

Для работы над проектом проектной группой был поднят собственный репозиторий на сервере `git.keva.su`. Адреса репозитория следующие:

Исходные файлы проекта можно найти здесь:

`http://gitlab2.keva.su/gpo/coex`

Репозиторий для тестирования проекта:

`git clone git@gitlab2.keva.su:gpo/coex.git`

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 планиру-

ется 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. [7] 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.

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

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

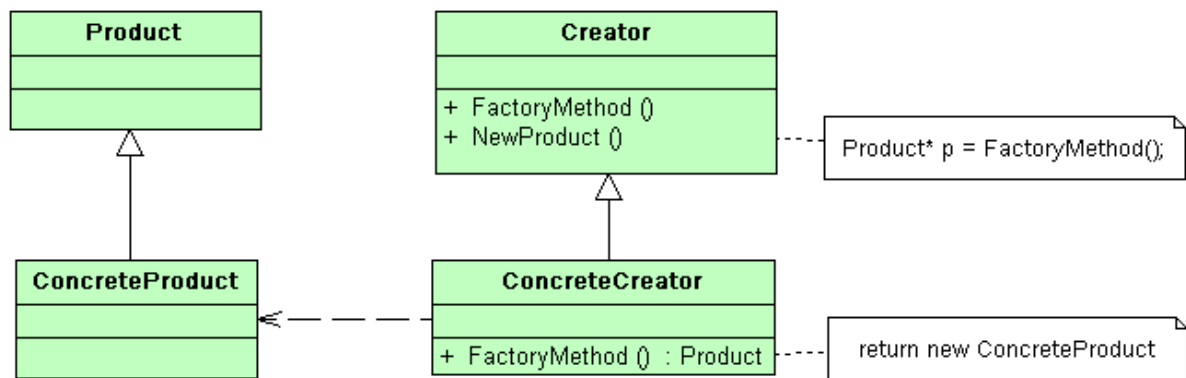


Рисунок 5.1 – Пример организации проекта при использовании шаблона проектирования Factory method

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

- 5) QString name() - возвращает имя данного таска;
- 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.

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 таска
- void setOption(QStringList list) - установка флагов для поданных на вход параметров
- QString command() - возвращает команду для инициализации таска вручную
- bool supportOS(const coex::typeOS &os) - возвращает флаг указывающий на возможность использования данного таска для конкретной операционной системы

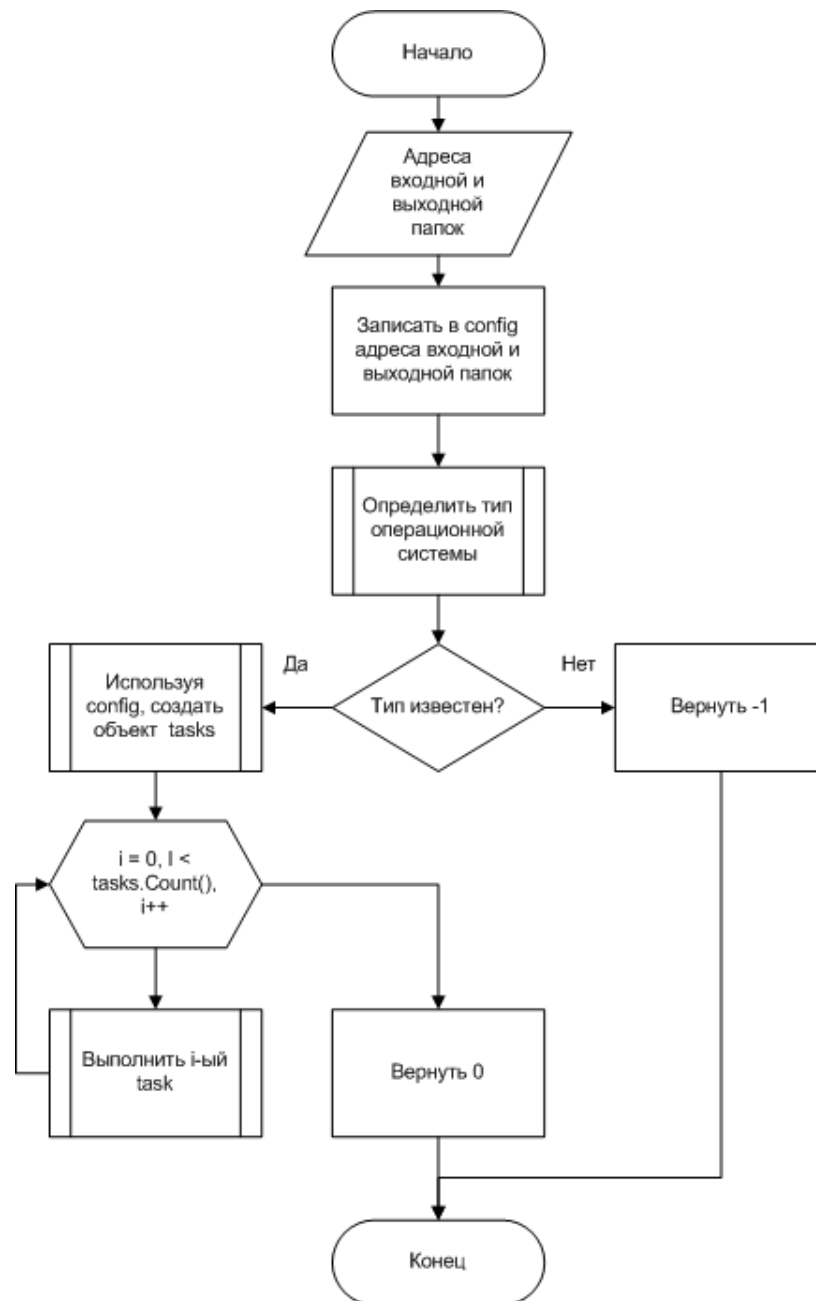


Рисунок 5.2 – Алгоритм работы с образом диска

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

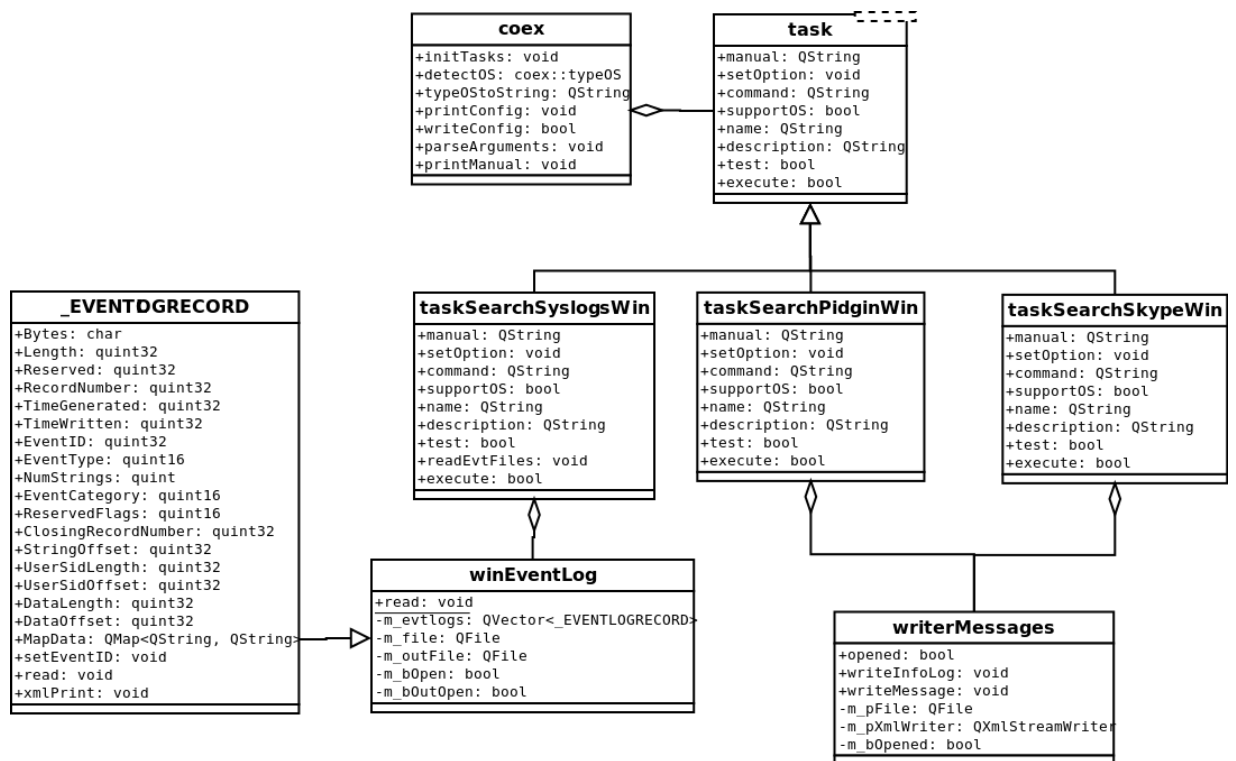


Рисунок 5.3 – UML-диаграмма классов

5.2 Плагин SearchProgram

Был доработан плагин «SearchProgram». Теперь при подаче образа с операционной системой не нужно указывать, какая ОС (Windows XP/7/8/8.1) на образе. Он использует написанный ранее плагин Дмитрия Никифорова, который определяет ОС. В итоге, плагин «SearchProgram» не требует каких-либо действий от пользователя.

Блок-схема плагина «SearchProgram»

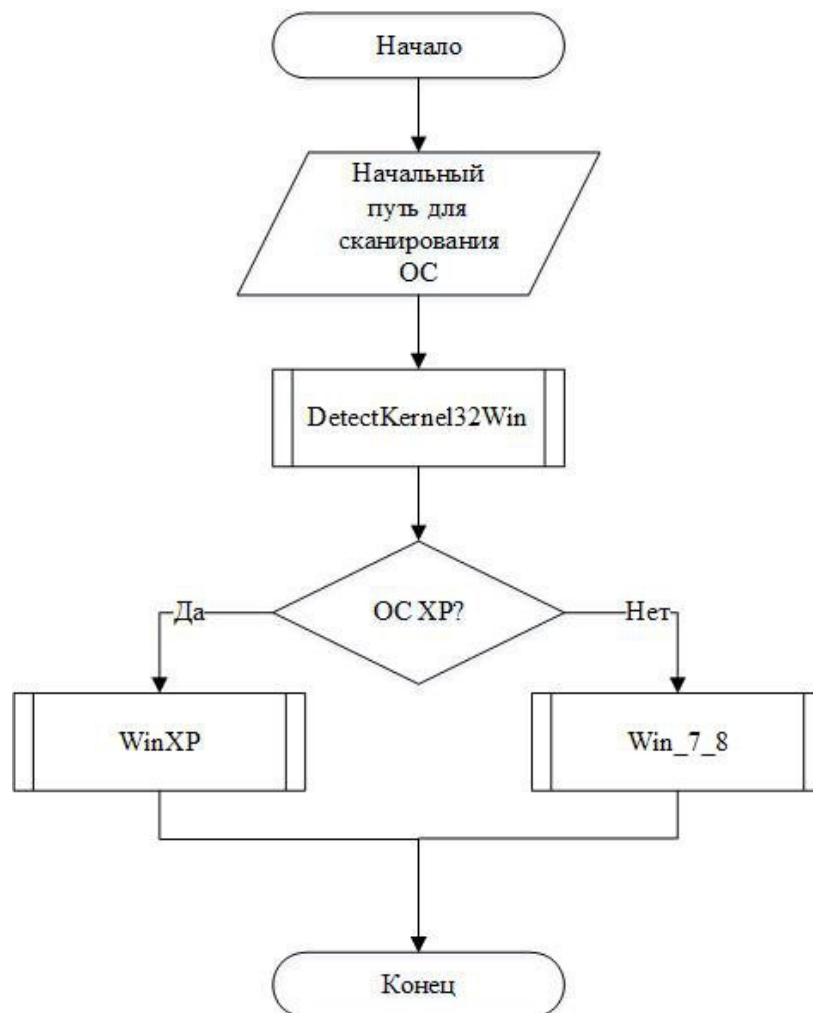


Рисунок 5.4 – Блок-схема плагина «SearchProgram»

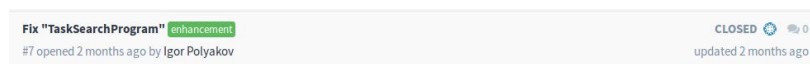


Рисунок 5.5 – Закрытый issue «Fix TaskSearchProgram»

5.2.1 Исправление скриптов clone.sh и test.sh

В течение семестра «gitlab» был перенесен на другой домен, поэтому надо было исправить скрипты «СОЕХ», чтобы они работали с новым доменом. Скрипты clone.sh и test.sh были исправлены.

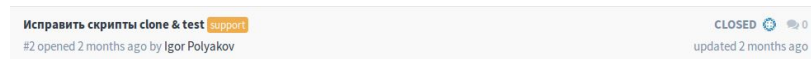


Рисунок 5.6 – Закрытый issue «Исправить скрипты clone test»

5.2.2 Создание репозитория

Цель: создать репозиторий для хранения deb – пакетов «COEX». Репозиторий разворачивался на виртуальной машине с ОС Debian 7.9. Для создания репозитория была выбрана программа «reprepro». Reprepro является инструментом для управления APT репозиториями. Он в состоянии управлять многократными репозиториями для многократных версий распределения и одного пула пакета. Reprepro может обработать обновления из входящего каталога, пакет копии (ссылки) между версиями распределения, перечислить все пакеты и/или версии пакета, доступные в репозитории и т.д. Reprepro поддерживает внутреннюю базу данных (.DBM файл) содержания репозитория, который делает его довольно быстрым и эффективным. В добавок, в Reprepro есть возможность подтверждения подлинности пакетов с помощью GPG – ключа. GNU Privacy Guard (GnuPG, GPG) — свободная программа для шифрования информации и создания электронных цифровых подписей. С помощью нее генерируем GPG – ключ. В качестве веб – сервера был выбран «Nginx», т.к. легко масштабируется на минимальном железе. Установка программы reprepro выполняется следующей командой: `aptitude install reprepro`. Далее, создаем каталог «repository» в `/var/www/`. Для настройки репозитория необходимо создать два конфигурационных файла «distributions» и «options» в папке «repository». Настроенный конфигурационный файл «distributions» программы «reprepro» выглядит следующим образом:

```
root@gitlab:/var/www/repository/conf# cat distributions
Origin: coex.su Deb-Repository
Label: AORDEB
Suite: stable
Codename: linux
Architectures: i386 amd64
Components: non-free
Description: Repository for web-development
Signwith: yes
root@gitlab:/var/www/repository/conf#
```

Рисунок 5.7 – Конфигурационный файл «distributions»

Параметр «SignWith: yes» указывает на то, что репозиторий будет использовать GPG – ключ для подтверждения подлинности пакетов. Настроенный конфигурационный файл «options»:

```
root@gitlab:/var/www/repository/conf# cat options
verbose
ask-passphrase
root@gitlab:/var/www/repository/conf#
```

Рисунок 5.8 – Конфигурационный файл «options»

«verbose» определяет, что всегда будет выводиться информация о ходе выполнения команды, а «ask-passphrase» заставляет «reprepro» спрашивать пароль для GPG-ключа при добавлении нового deb-пакета в репозиторий. Генерация GPG-ключа выполняется командой `gpg --gen-key`. Для до-

бавлении файлов в репозиторий используется следующая команда (при условии, что находимся в директории «repository»): `reprepro -b . includedeb linux /путь к deb-пакету`. После выполнения этих действий настройка репозитория закончена. Он доступен локально. Следует настроить веб-сервер, чтобы предоставить доступ к репозиторию через интернет. Олег Лобанов создал домен `repa.coex.su` для репозитория. Настроенный конфигурационный файл для веб-сервера «Nginx» выглядит следующим образом (рис. лалала):

```
root@gitlab:/etc/nginx/conf.d# cat repa.coex.su.conf
server {
    server_name repa.coex.su;
    listen 80;
    root /var/www/repository/;

    #access_log /var/log/nginx/80.89.147.35.log main;
    location / {
        autoindex on;
        #proxy_pass http://80.89.147.35:8080;
    }
    location /conf {
        deny all;
    }
    location /db {
        deny all;
    }
}
```

Рисунок 5.9 – Конфигурационный файл «repa.coex.su» веб-сервера «Nginx»

Так как созданный репозиторий — публичный, то нужно ограничить доступ к каталогам `/conf` и `/db`, содержащим сведения о настройках репозитория.

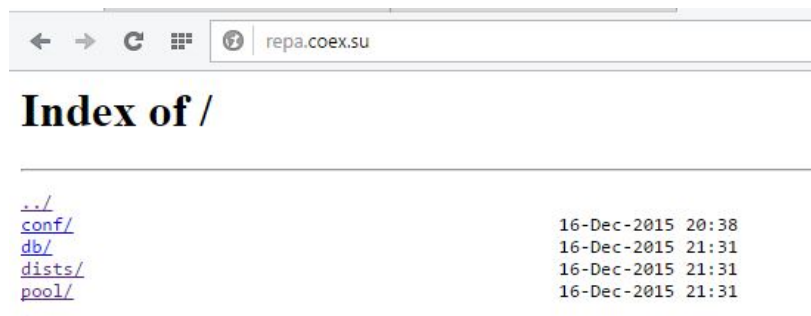


Рисунок 5.10 – Демонстрация страницы `repa.coex.su`



Рисунок 5.11 – Демонстрация страницы `repa.coex.su/conf/`

Нужно было предоставить пользователям репозитория возможность свободно получать открытый GPG – ключ. Для этого было принято решение воспользоваться публичным GPG – сервером `http://keyserver.ubuntu.com`. Но этот GPG – сервер принимает ключи в формате ASCII – armored.



Рисунок 5.12 – Демонстрация страницы repa.coex.su/db/

Поэтому сначала следует конвертировать файл pubring.gpg в pubring.asc следующей командой: `gpg --output pubring.asc --export --a $GPGKEY`. Полученный файл загружаем на публичный GPG – сервер. Теперь пользователям репозитория нужно установить открытый GPG – ключ репозитория repa.coex.su командами: `gpg --keyserver keyserver.ubuntu.com --recv «номер ключа, который запросит репозиторий»` и `gpg --export --armor «номер ключа, который запросит репозиторий» | apt-key add` – После успешного экспорта нужно добавить строчку `deb http://repa.coex.su linux non-free` в файл `/etc/apt/sources.list`. Обновить список пакетов командой `aptitude update` и установить «COEX» командой `aptitude install coex`.

[8] [9]

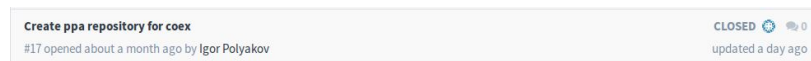


Рисунок 5.13 – Закрытый issue «Create ppa repository for coex»

5.3 мессенджеры

Цель работы:

- 1) исправить ошибки плагина TaskFirefoxWin;
- 2) переделать программный модуль TaskThunderbirdWin в плагин для проекта соех.

5.3.1 TaskFirefoxWin

Плагин TaskFirefoxWin предназначен для сбора истории посещений браузера Mozilla Firefox. История находится в файле базы данных places.sqlite, который расположен в C:\Users\User\AppData\Roaming\Mozilla\Firefox\Profiles\profilename.

Плагин выполняет рекурсивный обход директорий пока не найдет файл places.sqlite. Затем плагин подключается к базе данных, содержащейся в данном файле и выполняет sql-запрос на выборку данных из таблицы. 5.14

Программный модуль TaskFirefoxWin был некорректно переделан в плагин проекта соех. Вследствие чего, он не исполнялся. В результате тестирования было выявлено, что причиной некорректной работы плагина были служебные файлы TaskPlugin.pro, build.sh и taskFirefox.h. Были внесены исправления в эти служебные файлы. После этого плагин исполнялся. 5.15

Были добавлены тестовые данные для проверки работоспособности плагина. Также был добавлен вывод результатов работы плагина в XML. В результате тестирования было выявлено, что плагин не выполняет поставленную задачу.

В одном из обновлений Mozilla Firefox была изменена структура файла places.sqlite. Из-за этого не работал sql-запрос для выборки данных из таблицы с историей посещений. Был составлен новый запрос: `select * from moz_places.` 5.16

Было произведено тестирование исправленного плагина. Из представленных исходных данных был получен такой отчет в XML-файле. 5.17 5.18

5.3.2 TaskThunderbirdWin

TaskThunderbirdWin был выполнен в виде отдельного программного модуля. Модуль предназначен для сбора сообщений и представления их в формате XML. Сообщения хранятся в файле Inbox.mbox для входящих сообщений и Sent.mbox для исходящих сообщений. Путь, по которому находятся файлы: C:\Users\User\AppData\Roaming\Thunderbird\Profiles\profile_name\Mail\server_name.mbox представляет собой текстовый файл, в котором хранятся все сообщения почтового ящика. Начало почтового сообщения определяется строкой из 5 символов: словом «From» с последующим пробелом.

После открытия файл mbox разделяется на отдельные сообщения с помощью регулярного выражения «(From \\r\\n)|(From \\n\\r)|From \\r|From \\n». Затем к каждому сообщению применяются регулярные выражения:

- «\\nDate: ([^\\n]*)\\n» — время отправки/приема сообщения;
- «\\nFrom: .*([a-z][\\w\\.]*\\w@\\w[\\w\\.]*\\.\\w*).*\\nUser-Agent:» — кто отправил сообщения;
- «\\nTo: .*([a-z][\\w\\.]*\\w@\\w[\\w\\.]*\\.\\w*).*\\nSubject:» — кто получил сообщение;

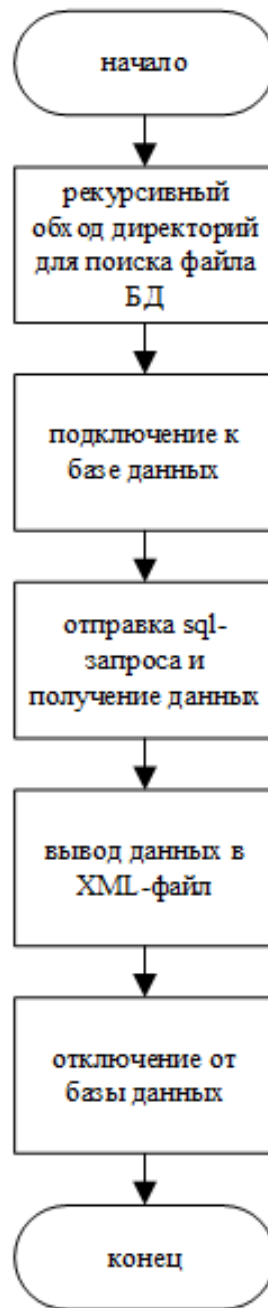


Рисунок 5.14 – Блок-схема алгоритма TaskFirefoxWin

– «\nContent-Transfer-Encoding: 8bit\s*(\S.*\S)\s*[0-3]\d\\. [01]\d\\. \d4 [0-2]\d:[0-5]\d, [^\n]*\n» — текст сообщения.

5.19

Как отдельный модуль, TaskThunderbirdWin выполнял поставленные перед ним задачи. 5.20

Но после того, как TaskThunderbirdWin был переписан под архитектуру соех, возникли проблемы в работе данного модуля. Ведется тестирование с целью нахождения ошибок, мешающих корректному выполнению данного модуля.

```

sources/plugins/TaskFirefoxWin/TaskPlugin.pro
...
9 9  @ -9,11 +9,10 @ INCLUDEPATH += ../../include/
10 10  OBJECTS_DIR = tmp/
11 11  QT -= gui
12 12  +QT += sql
13 13
14 14  CONFIG += dll
15 15
16 16  -SOURCES += \
17 17  - src/taskFirefox.cpp
18 18  +SOURCES += src/taskFirefox.cpp
19 19
20 20  -HEADERS += \
21 21  - src/taskFirefox.h
22 22  +HEADERS += src/taskFirefox.h
...

sources/plugins/TaskFirefoxWin/build.sh
1 1  #!/bin/bash
2 2
3 3  -qmake-qt4 TaskPlugin.pro
4 4  +qmake-qt4
5 5  make
...

sources/plugins/TaskFirefoxWin/src/taskFirefox.h
...
19 19  @ -19,7 +19,7 @ class TaskExample : coex::ITask
20 20  virtual QString description();
21 21
22 22  virtual bool isSupportOS(const coex::ITypeOperationSystem *os);
23 23  virtual void setOption(QStringList);
24 24  + virtual void setOption(QStringList options);
25 25  virtual bool execute(const coex::IConfig *config);
26 26
27 27  private:
28 28  bool m_bDebug;
29 29
30 30  @ -30,4 +30,4 @ extern &quot;C&quot;;
31 31  coex::ITask* createTask();
32 32
33 33  -#endif // __TASK_FIREFOX__
34 34  +#endif // __TASK_FIREFOX_H__

```

Рисунок 5.15 – Изменения в служебных файлах

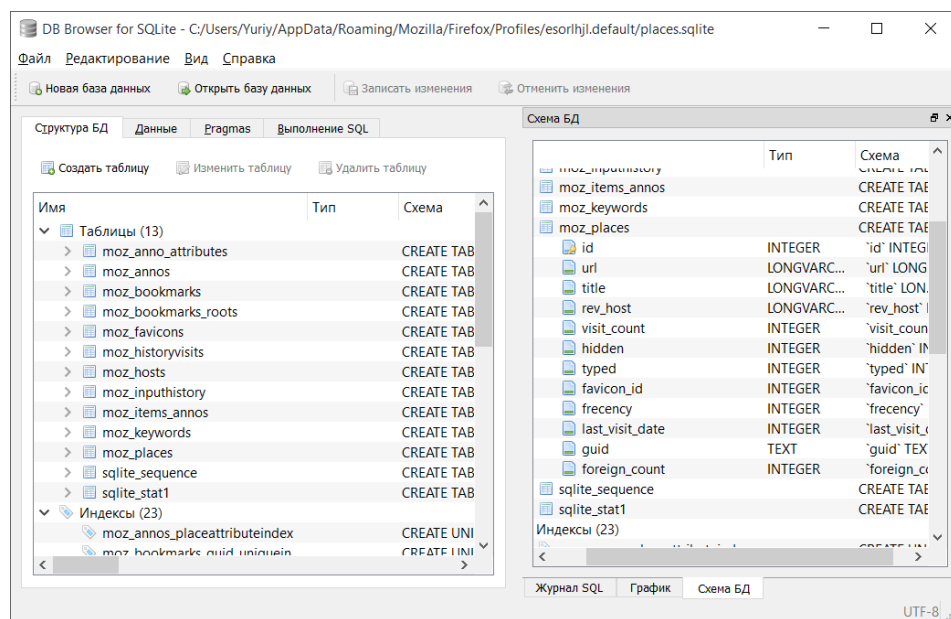


Рисунок 5.16 – Структура базы данных и нужной таблицы

Структура БД Данные Pragma Выполнение SQL				
Таблица: moz_places				
	id	url	title	rev
	Filter	Filter	Filter	Filter
1	1	https://www.mozilla.org/ru/fire...	NULL	gro.al
2	2	https://www.mozilla.org/ru/fire...	NULL	gro.al
3	3	https://www.mozilla.org/ru/fire...	NULL	gro.al
4	4	https://www.mozilla.org/ru/con...	NULL	gro.al
5	5	https://www.mozilla.org/ru/abo...	NULL	gro.al
6	6	place:sort=8&maxResults=10	NULL	.
7	7	https://www.youtube.com/	NULL	moc.e
8	8	place:folder=BOOKMARKS_ME...	NULL	.
9	9	http://stopgame.ru/	NULL	ur.em
10	10	place:type=6&sort=14&maxRe...	NULL	.
11	11	http://4pda.ru/news/	NULL	ur.adp
12	12	http://geektimes.ru/	NULL	ur.sen
13	13	https://vk.com/yurec1994	NULL	moc.k
14	14	http://pikabu.ru/hot	NULL	ur.uba
15	15	http://lostfilm.tv/	NULL	vt.mlit

Рисунок 5.17 – Тестовые данные

```

- <add>
  <field name="1">https://www.mozilla.org/ru/firefox/central/</field>
  <field name="2">https://www.mozilla.org/ru/firefox/help/</field>
  <field name="3">https://www.mozilla.org/ru/firefox/customize/</field>
  <field name="4">https://www.mozilla.org/ru/contribute/</field>
  <field name="5">https://www.mozilla.org/ru/about/</field>
  <field name="6">place:sort=8&maxResults=10</field>
  <field name="7">https://www.youtube.com/</field>
- <field name="8">
  place:folder=BOOKMARKS_MENU&folder=UNFILED_BOOKMARKS&
  folder=TOOLBAR&queryType=1&sort=12&maxResults=10&excludeQueries=1
  </field>
  <field name="9">http://stopgame.ru/</field>
  <field name="10">place:type=6&sort=14&maxResults=10</field>
  <field name="11">http://4pda.ru/news/</field>
  <field name="12">http://geektimes.ru/</field>
  <field name="13">https://vk.com/yurec1994</field>
  <field name="14">http://pikabu.ru/hot</field>
  <field name="15">http://lostfilm.tv/</field>

```

Рисунок 5.18 – Результат тестирования TaskFirefoxWin

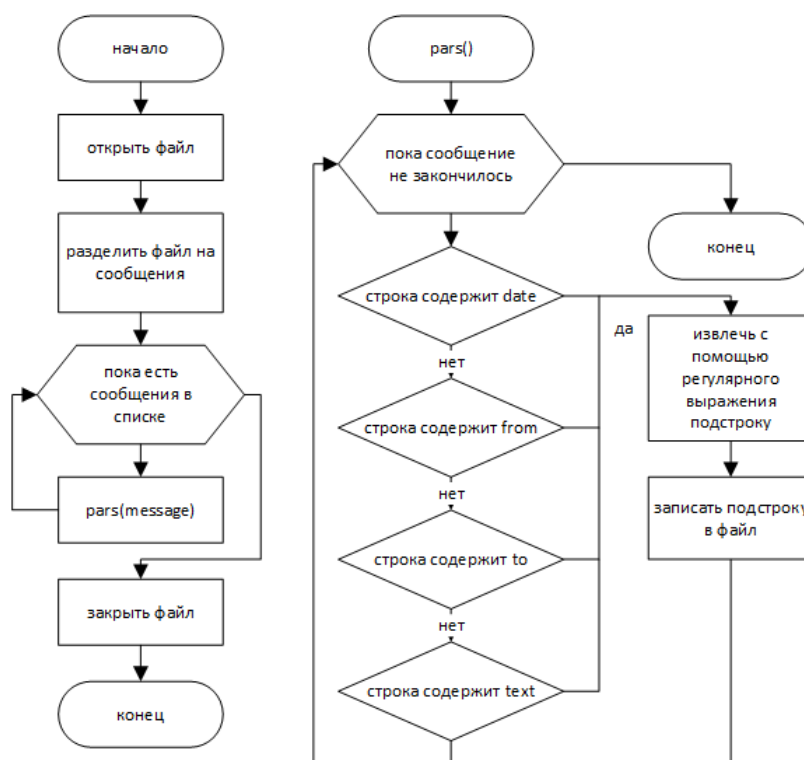


Рисунок 5.19 – Блок-схема алгоритма TaskThunderbirdWin

```

▼<add>
  ▼<file>
    ▼<field name="name">
      C:\Users\ghost\AppData\Roaming\Thunderbird\Profiles\g5bq66yo.default\ImapMail\imap.yandex.com\&BB4EQgQ, BEAEMAQyBDsENQq9BD0ESwQ1-
    </field>
    ▼<message>
      <field name="date">Thu, 23 Apr 2015 12:15:55 +0600</field>
      <field name="from">art0rias@yandex.ru</field>
      <field name="to">yuriy9494@gmail.com</field>
      <field name="text">пак за руку цап</field>
    </message>
    ▼<message>
      <field name="date">Thu, 23 Apr 2015 12:46:35 +0600</field>
      <field name="from">art0rias@yandex.ru</field>
      <field name="to">yuriy9494@gmail.com</field>
      <field name="text">I've been to see grandmother Over the green.</field>
    </message>
    ▼<message>
      <field name="date">Thu, 23 Apr 2015 12:47:49 +0600</field>
      <field name="from">art0rias@yandex.ru</field>
      <field name="to">yuriy94@hotmail.com</field>
      <field name="text">What did you say for it? Thank you, Grandam.</field>
    </message>
  </file>
</add>

```

Рисунок 5.20 – Результат тестирования TaskThunderbirdWin

5.4 Написание графического интерфейса пользователя для системы «COEX»

5.5 Доработка программного модуля Media Scanner

5.6 Сбор информации из почтового клиента MS Outlook

В начале семестра был переписан модуль «Outlook» под новую архитектуру «COEX». Добавлены наследуемые функции для описания плагина и функции для контроля выполнения плагина в программном комплексе «COEX» из главного класса «task».

Функции для описания плагина и функции для контроля выполнения плагина представлены на рисунке 5.21, функция для контроля выполнения плагина в программном комплексе «COEX» — на рисунке 5.22.

```

1  #include "taskOutlook.h"
2  #include "writerAddress.h"
3
4  taskOutlook::taskOutlook() {
5      m_bDebug = false;
6  };
7
8  QString taskOutlook::help() {
9      return "\t--debug - viewing debug messages";
10 };
11
12 QString taskOutlook::name() {
13     return "Outlook";
14 };
15
16 QString taskOutlook::author() {
17     return "Serakov Andrey";
18 };
19
20 QString taskOutlook::description() {
21     return "TaskOutlook2007 task";
22 };
23
24 bool taskOutlook::isSupportOS(const coex::ITypeOperationSystem *os) {
25     return (os->platform() == "Windows");
26 };
27
28 void taskOutlook::setOption(QStringList options) {
29
30     if(options.contains("--debug"))
31         m_bDebug = true;
32 };
33
34 QStringList Prov(QStringList list, QStringList oflist)
35 {
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Рисунок 5.21 – Функции для описания плагина и функции для контроля выполнения плагина

```

200
201     writerAddressOutlook OutlookMessage(config->outputFolder()+"/Outlook/Message.xml", oflistAddress0tp[i]);
202     //OutlookMessage.writerAddressOutlook;
203     OutlookMessage.writeMessage(oflistAddress0tp[i], oflistDate[i], oflistText[i], oflistTheme[i]);
204 }
205
206     return true;
207 }
208
209 coex::ITask* createTask()
210 {
211     return (coex::ITask*)(new taskOutlook());
212 }
213

```

Рисунок 5.22 – Функция для контроля выполнения плагина в программном комплексе

5.7 Создание «бинарного» пакета DEB из исходных файлов программного комплекса «СОЕХ»

Формат *.deb — расширение имён файлов «бинарных» пакетов для распространения и установки программного обеспечения в ОС проекта Debian и других, использующих систему управления пакетами dpkg. Пакеты Debian содержат исполняемые и конфигурационные файлы, номер версии формата, информацию об авторских правах, а также другую документацию, необходимую для установки программы из пакета.

Основные составляющие *.deb-пакета, необходимые для его создания:

- control;
- md5sums;
- changelog;
- rules;
- README;
- conffiles;
- dirs;
- watch.

Control — центральный файл пакета (обязательный), описывающего все основные свойства. Файл — текстовый, состоящий из пар «Атрибут: значение».

Md5sums — содержит md5-хеши для всех файлов, кроме файлов, находящихся в каталоге DEBIAN/. Данный файл необязателен для deb-пакета, однако программы верификации пакетов считают пакеты, не содержащие этот файл, ошибочными. Может использоваться некоторыми программами администрирования системы для верификации изменений в файловой системе. Осуществляет контроль целостности файлов.

Changelog — обязательный файл, его специальный формат описан в руководстве по политике Debian, раздел 4.4 «debian/changelog». Этот формат используется программой dpkg и другими для получения информации о номере версии, редакции, разделе и срочности пакета. [10]

Rules — используется для управления компиляцией пакета.

README — в этот файл записывается любая дополнительная информация, а также различия между программой из пакета Debian и исходной программой.

Conffiles — Обычно пакеты содержат «болванки» конфигурационных файлов, например, размещаемых в /etc. Очевидно, что если конфигурационный файл в пакете обновляется, пользователь потеряет свой отредактированный конфигурационный файл. Эта проблема легко решается использованием папок типа «config.d», содержимое которых включается в основной конфигурационный файл, заменяя собой повторяющиеся опции. [11]

Dirs — в этом файле указываются каталоги, которые необходимы для обычной установки (make install DESTDIR=..., вызываемая dh_auto_install), но которые автоматически не создаются. Обычно, это указывает на проблему в Makefile.

Watch — формат файла watch описан в справочной странице uscan(1). Файл watch настраивает программу uscan (предоставляется пакетом devscripts) для слежения за сайтами, с которых вы скачали исходный код. Он также используется службой Debian для слежения за состоянием внешних источников (DEHS). [10]

Для решения данной задачи было сделано:

- изучен *.deb формат, особенности файлов стандартов необходимых для создания данного пакета и установки его в операционные системы семейства Unix;
- изучены программные продукты для работы с данным форматом (dpkg, build-essential, quilt, fakeroot, devscripts, debhelper и dh-make, lintian);
- изучены особенности написания скриптов bash для операционных систем Unix;
- ознакомление и изучение программы make, набора инструкцией Makefile и программной утилиты для QT C++ qmake;
- углубленное изучение системы построения программного комплекса «COEX» и зависимых библиотек для компиляции данного программного комплекса;
- создание двух программ bash-скриптов для формирования *.deb пакета и формирования changelog на основе истории коммитов системы контроля версий GIT;
- тестирование на «чистой» операционной системе Linux Mint 17.2;
- изучен формат набора макрорасширений системы компьютерной вёрстки LaTeX, личный отчет по групповому проектному обучению был оформлен согласно данному формату, и отдан документатору для редакции и включения в общий отчет.

5.7.1 Подробное рассмотрение набора bash-скриптов для формирования *.deb пакета и формирования changelog

В начале программы с помощью скрипта build.sh компилируется проект и осуществляется сборка бинарного файла «coex», а также динамических библиотек, используемых программным комплексом «COEX». Исполняемые файлы плагинов данного проекта имеют формат *.so.

В корне пакета создается папка «DEBIAN». Эта папка содержит управляющую генерацией пакета информацию и не копируется на диск при установке пакета. Также корневая папка пакета содержит будущий «корень диска»: при установке пакета все файлы (кроме папки «debian») распаковываются в корень /. Поэтому бинарный файл должен лежать по пути относительно корня пакета «usr/bin/coex», что и было сделано в скрипте create_package.sh (рис. 5.23).

```
echo -e "Script for creating .deb pack
mkdir -p debian/DEBIAN
mkdir -p debian/usr/bin
cp bin/coex ./debian/usr/bin
```

Рисунок 5.23 – Копирование основного бинарного файла проекта

Исполняемые файлы плагинов и программные библиотеки программного комплекса «COEX» представлены на рисунке 5.24.

Установка иконки продукта нужна для графического интерфейса пользователя программного комплекса «COEX» (рис. 5.25). Файл coex.desktop служит для запуска приложения, содержит основную информацию о программном комплексе «COEX» (рис. 5.26).

Лицензия на программный комплекс «COEX» защищает права разработчиков данного программного обеспечения. Руководство по использованию программного комплекса «COEX» находится в стадии разработки и будет содержать основные команды для использования программного комплекса «COEX», информацию о плагинах и об особенностях работы с ними (рис. 5.27).

```
##Plugins and libs for COEX Binary
mkdir -p debian/usr/coex/plugins
mkdir -p debian/usr/coex/libs
cp -R bin/plugins/ ./debian/usr/coex
cp -R bin/libs/ ./debian/usr/coex
```

Рисунок 5.24 – Исполняемые файлы плагинов и программные библиотеки программного комплекса «COEX»

```
##Image for coex
mkdir -p debian/usr/share/doc/$package_name
mkdir -p debian/usr/share/man
mkdir -p debian/usr/share/applications
mkdir -p debian/usr/share/icons/hicolor/64x64/apps
cp avatar-63.jpg ./debian/usr/share/icons/hicolor/64x64/apps
```

Рисунок 5.25 – Иконка для GUI приложения «COEX»

```
## The Applications
echo "Name=$package_name">debian/usr/share/applications/coex.desktop
echo "Comment=$package_name for forensics">debian/usr/share/applications/coex.desktop
echo "GenericName= tool of forensics">debian/usr/share/applications/coex.desktop
echo "Exec=$package_name">debian/usr/share/applications/coex.desktop
echo "Icon=$package_name">debian/usr/share/applications/coex.desktop
echo "Categories=tool for forensics">debian/usr/share/applications/coex.desktop
echo "Terminal=true">debian/usr/share/applications/coex.desktop
echo "Type=Application">debian/usr/share/applications/coex.desktop
echo "Version=$version">debian/usr/share/applications/coex.desktop
```

Рисунок 5.26 – Файл coex.desktop «COEX»

```
## Dock and Manual
touch debian/usr/share/doc/$package_name/copyright
touch debian/usr/share/man/$package_name/Manual
```

Рисунок 5.27 – Лицензия и мануал по «COEX»

Файл control (рис. 5.28) — обязательный файл, в котором прописана версия программного комплекса «COEX». Также указаны зависимости от библиотек, нужных для использования «COEX».

```
## The control file
platform="all"
maintainer_name="$(git config user.name)"
maintainer_email="$(git config user.email)"
#size="$(du -s -k debian/)"
echo "Package: $package_name" > debian/DEBIAN/control
echo "Version: $version" >> debian/DEBIAN/control
echo "Section: admin" >> debian/DEBIAN/control
echo "Priority: optional" >> debian/DEBIAN/control
echo "Architecture: $platform" >> debian/DEBIAN/control
echo "Depends: g++ (>= 4.8.4), libqt4-opengl (>= 4:4.7.2), libqtcore4 (>= 4:4.8.0), libqt4"
echo "Maintainer: $maintainer_name <$maintainer_email>" >> debian/DEBIAN/control
#echo "Installed-Size: $size" >> debian/DEBIAN/control
echo "Description: Project KIBEVS-1401: Computer forensics" >> debian/DEBIAN/control
```

Рисунок 5.28 – Файл control

Файл changelog (рис. 5.29) — в данном файле находятся сведения о всех изменениях программного комплекса «COEX» и версий пакета. Данный файл генерируется в отдельном скрипте gen_change_log.sh рисунок 10

```
#!/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 5.29 – Файл changelog

Файл md5sums (рис. 5.30) — в данном файле записываются хеш-значения всех файлов, находящихся в «debian/usr».

```
#find . -type f -exec md5sum {} \;
md5deep -r debian/usr |> debian/DEBIAN/md5sums
```

Рисунок 5.30 – Файл md5sums

Далее, чтобы распаковать *.deb-пакет для установки на компьютер пользователя, необходимо всем файлам задать права доступа root, иначе пакет не установится, после чего утилитой dpkg собрать из всех созданных файлов пакет формата *.deb.

5.7.2 Тестирование созданного пакета

После создания *.deb-пакета для его тестирования выбраны были три образа операционных систем семейства Unix, а именно: Linux Mint 17.2, Ubuntu 14.04, Debian 8.2. На каждой из операционных систем был установлен пакет coex_0.1-43-g59b4cc1_all.deb. Программное обеспечение было успешно установлено, все плагины работают (рис. 5.31).

Библиотеки и исполняемые файлы плагинов coex на debian — рисунок 5.32.

Бинарный файл coex на Linux Mint — рисунок 5.33.

Плагины на Linux Mint — рисунок 5.34.

Запуск плагина на Linux Mint — рисунок 5.35.

Бинарный файл coex на Ubuntu — рисунок 5.36.

Запуск coex на Ubuntu — рисунок 5.37.

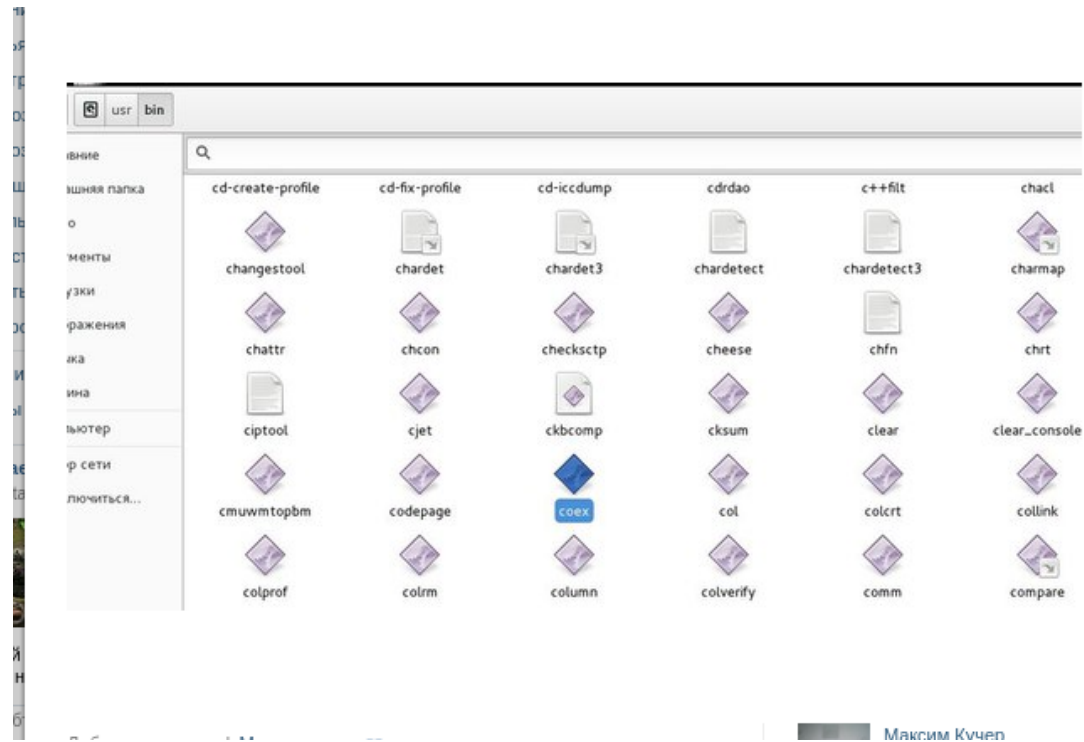


Рисунок 5.31 – Бинарный файл coex на debian

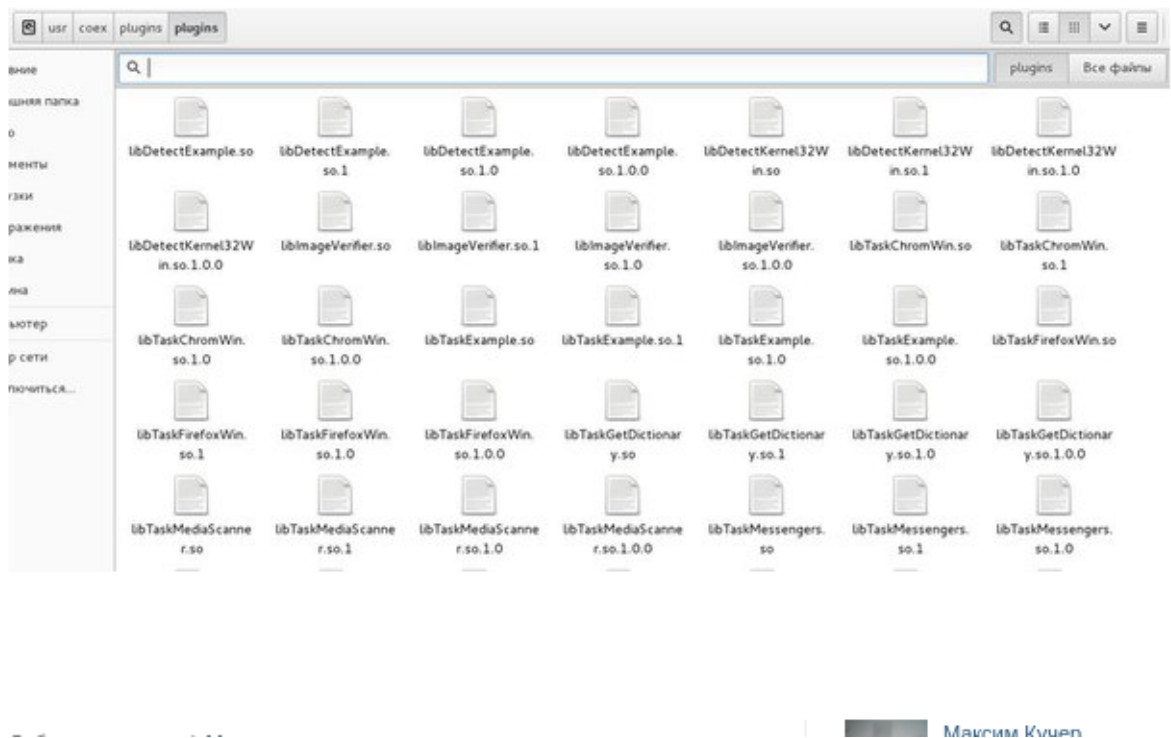


Рисунок 5.32 – Библиотеки и исполняемый файлы плагинов coex на debian

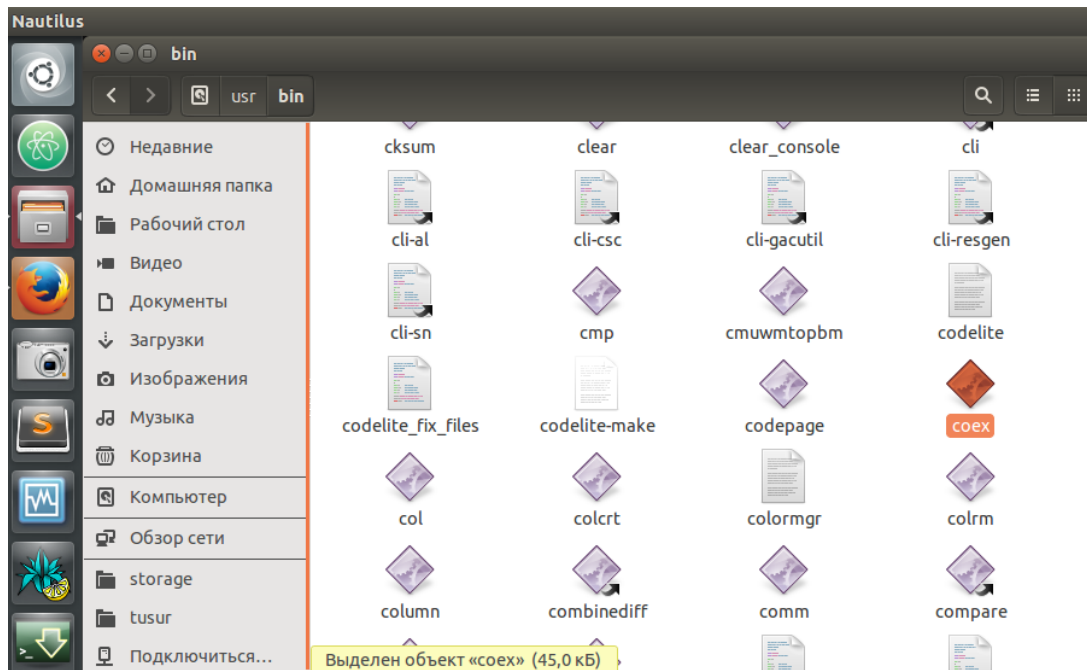


Рисунок 5.36 – Бинарный файл coex на Ubuntu 15.04

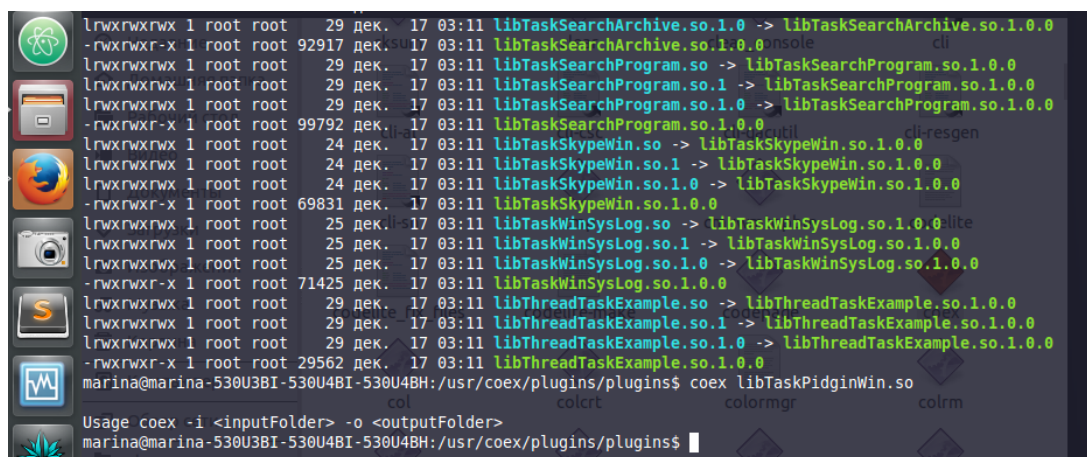


Рисунок 5.37 – Запуск coex на Ubuntu 15.04

5.8 Разработка веб-сайта проекта «СОЕХ»

5.8.1 Дизайн веб-сайта проекта

Трудно себе представить в современном мире компанию или программный продукт без персональной страницы в сети Интернет. В связи с этим возникла необходимость создания веб-сайта для проекта «СОЕХ».

Современные тенденции в веб-дизайне, как и в приложениях: минимализм и простота функционала. Крупнейшие корпорации в течение последних лет выпустили гайдлайны для своих продуктов: у Microsoft — ModerUI, у Google — Flat Design. В качестве основного направления для дизайна выбран плоский стиль с минимум сторонней графики. Используемые технологии: HTML5, CSS3, JavaScript. Не менее важным является структура будущего макета, для этого необходимо выделить основные блоки информации, которые будут на сайте:

- 1) Информация о системе. Здесь необходимо описать, что представляет из себя система «СОЕХ» и ее необходимость.
- 2) Возможности системы, описание плагинов.
- 3) Установка. Как установить систему, ссылки на скачивание.
- 4) Архитектура. Информация о структуре системы.
- 5) Контакты.

Исходя из вышесказанного был выбран первый вариант макета: сайт-презентация (рис. 5.38).

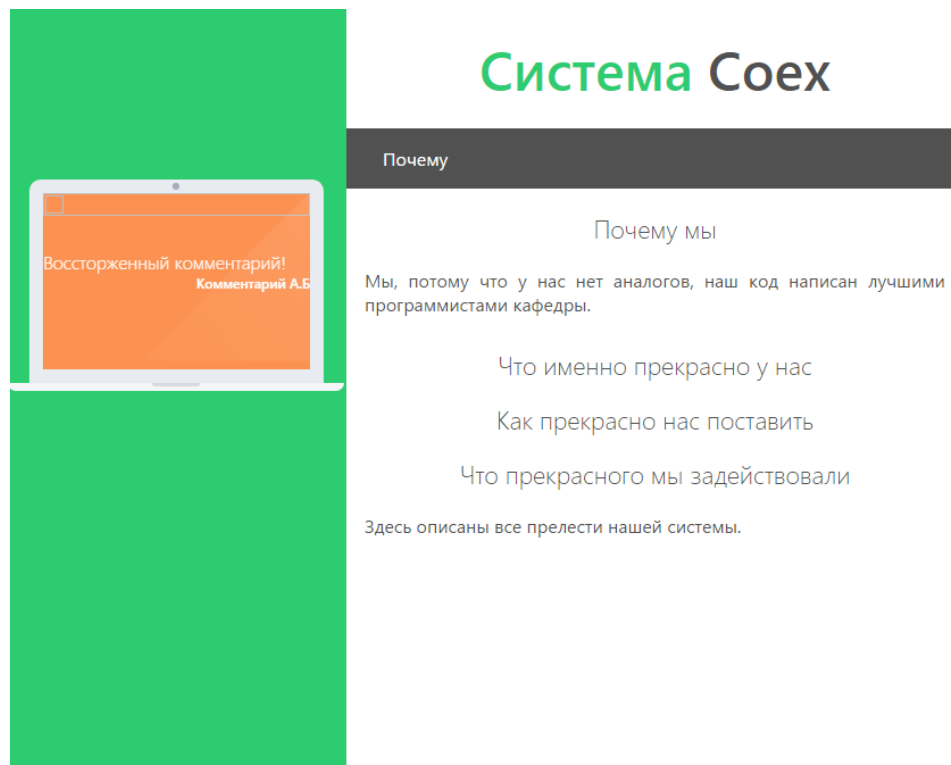


Рисунок 5.38 – Первый вариант дизайна

Позднее данный дизайн был отклонён из-за нерационального использования пространства веб-страницы и сложности расположения контента в пользу нового варианта (рис. 5.39).

Был выбран вариант с одним блоком информации на одну страницу «прокрутки» — так мож-

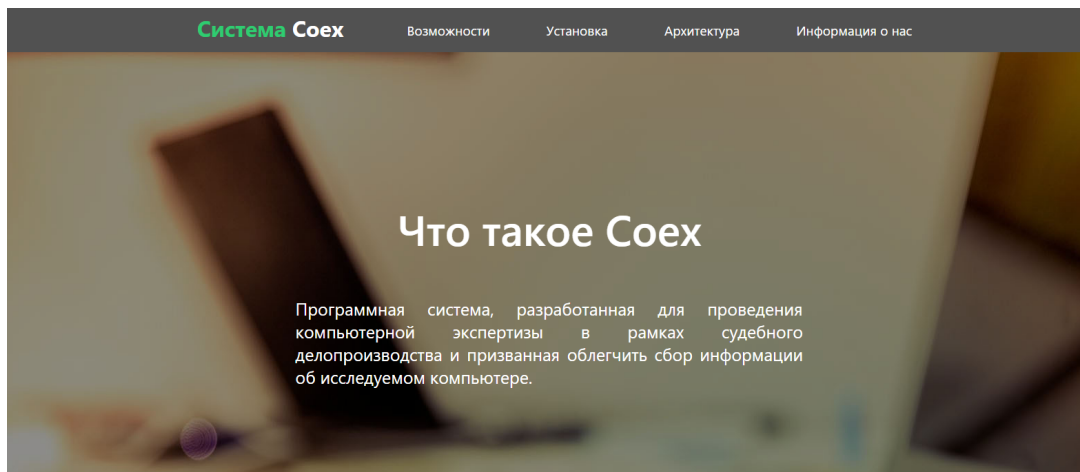


Рисунок 5.39 – Новый вариант дизайна

но акцентировать внимание посетителя на необходимых деталях.

5.8.2 Вёрстка дизайна

Как было сказано ранее, для верстки использовались технологии HTML5 и CSS3, что позволило существенно сократить код стилей и нагляднее писать код страницы. Минусом данного подхода является требование к наличию у посетителей последних версий интернет-браузеров, но, как показывает статистика, количество таких пользователей превалирует (рис. 5.40).

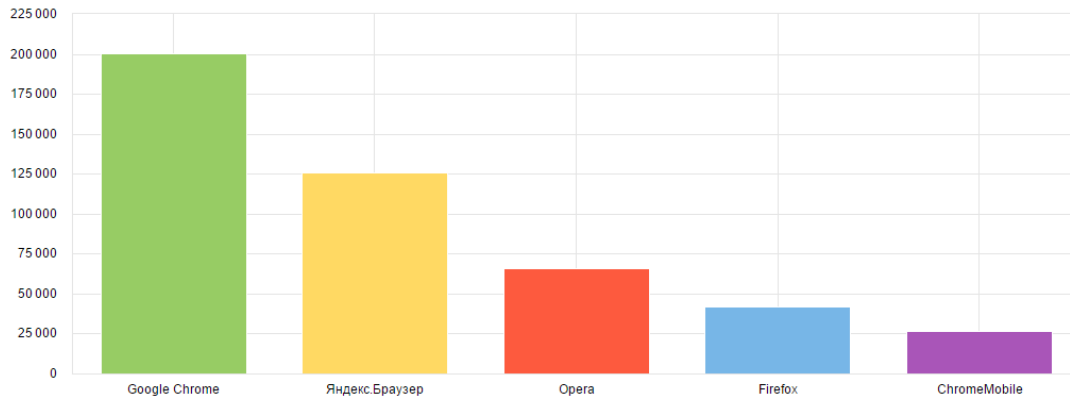


Рисунок 5.40 – Количество посетителей с различными браузерами

Для обеспечения поддержки мобильных устройств и устройств с нестандартным разрешением экрана в вёрстке максимально использованы новые возможности CSS3, такие как flex-box, box-sizing и другие (рис. 5.41).

5.8.3 Регистрация домена

Был подобран и зарегистрирован домен соех.su, параллельно с этим изучены принципы построения DNS серверов и NX записи (рис. 5.42).

```

251 #info .sect-block{
252     padding: 10% 0 0;
253     color: #fff;
254 }
255
256 #capa, #consist{
257     background-color: #f2f2f6;
258 }
259 .modules{
260 }
261
262
263 .module{display: inline-block; padding: 20px; width: 200px}
264 .module .icon{width: 150px; height: auto; display: block; margin: 0 auto; }
265 .module h3{ font-weight: 300; text-align: center; }
266 .module .decr-mod{ text-align: justify; }
267
268 .install-block{
269     min-width: 320px;
270 }
271
272 .git-url{
273     display: block;
274 }
275 .git-url img{ display: block; }
276 .git-url span{ display: block; text-align: center;}
277
278 .site-footer{
279     background-color: #515151;
280     color: #fff;
281     min-width: 320px;
282     overflow: hidden;
283 }
284 #consist .sect-block .description{ padding-top: 0.5em; }
285 .site-footer .copyrights{ width: 100%; padding: 10px; }
286 .site-footer .copyrights a { color: #6F6F6F; display: block; text-decoration: none; }
287 .site-footer .copyrights a:hover{ color: #fff; }
288 .site-footer .copyrights span{ display: block; color: #6F6F6F; }

```

Рисунок 5.41 – Часть кода CSS-стилей

Имя	Тип	Адрес
coex.su.	NS (сервер имён)	ns1.flynet.pro.
coex.su.	NS (сервер имён)	ns2.flynet.pro.
coex.su.	NS (сервер имён)	ns3.flynet.pro.
coex.su.	A (адрес Internet v4)	91.221.36.216
coex.su.	TXT (текстовая запись)	v=spf1 ip4:91.221.36.156 a mx ~all
coex.su.	MX (почтовый сервер)	mail
coex.su.	MX (почтовый сервер)	mail
ftp	A (адрес Internet v4)	91.221.36.216
mail	A (адрес Internet v4)	91.221.36.216
pop	A (адрес Internet v4)	91.221.36.216
smtp	A (адрес Internet v4)	91.221.36.216
www	A (адрес Internet v4)	91.221.36.216

Рисунок 5.42 – Пример NX-записей для домена coex.su

5.8.4 Настройка WEB-сервера

Для сайта был запущен и настроен веб-сервер apache в связке с nginx. В дальнейшем apache будет «отдавать» посетителям динамический контент, такой как новости, а nginx – статический: исходные коды программы с репозитория, картинки, видео и прочее (рис. 5.43).

5.8.5 Контент

Для наполнения сайта были написаны тексты, описывающие систему, ее части и возможности. Для каждого плагина была подготовлена своя иконка (рис. 5.44) и описание.

```
#
# The configuration files in the etc/apache22/extra/ directory can be
# included to add extra features or to modify the default configuration of
# the server, or you may simply copy their contents here and change as
# necessary.

# Server-pool management (MPM specific)
<VirtualHost 91.221.37.68:8080>
    ServerName coex.su
    DocumentRoot /home/b4el/data/www/coex.su
    AssignUserID b4el b4el
    CustomLog /dev/null combined
    ErrorLog /dev/null
    php_admin_value open_basedir "/home/b4el/data:."
    php_admin_value sendmail_path "/usr/sbin/sendmail -t -i -f servers@mine$
    php_admin_value upload_tmp_dir "/home/b4el/data/mod-tmp"
    php_admin_value session.save_path "/home/b4el/data/mod-tmp"
    AddType application/x-httpd-php .php .php3 .php4 .php5 .phtml
    AddType application/x-httpd-php-source .phps
</VirtualHost>
```

Рисунок 5.43 – Часть конфига Apache

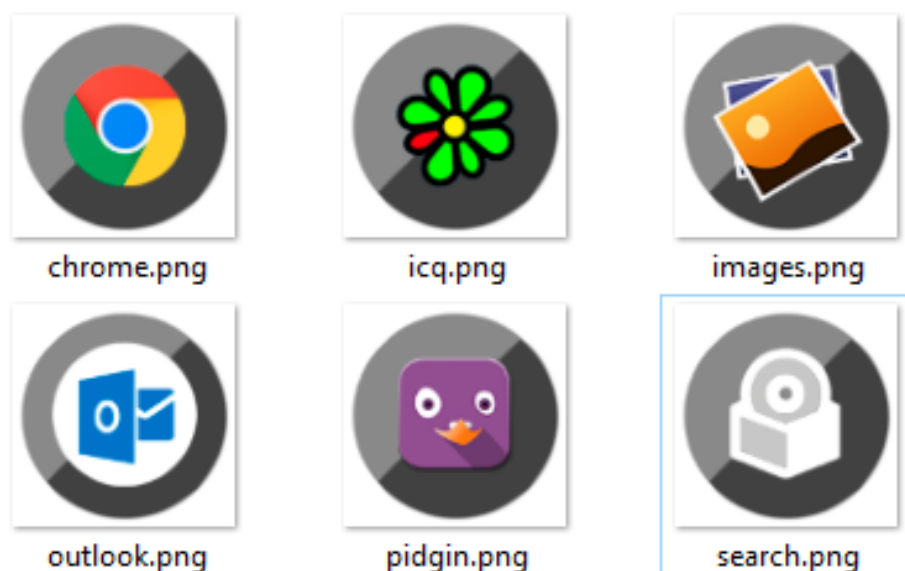


Рисунок 5.44 – Иконки для плагинов

6 Задачи на следующий семестр

Задачи на следующий семестр:

- под другие архитектуры
- создать скрипты
- написать демон
- наполнить контентом сайт
- документирование кода
- портирование на Windows В планах на будущее стоит завершение дизайна сайта и перенос его на CMS, наполнение контентом, проведение SEO разметки.

Заключение

В данном семестре нашей группой была выполнена часть работы по созданию автоматизированного программного комплекса для проведения компьютерной экспертизы, проанализированы дальнейшие перспективы и поставлены цели для дальнейшего развития проекта.

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^AT_EX. МЦНМО, 2006. С. 448.
- 4 И. А. Чеботаев, П. З. Котельников. L^AT_EX 2_ε 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 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).
- 8 Проблема с NO_PUBKEY: как получить GPG-ключ и добавить его в базу apt? [Электронный ресурс]. — Режим доступа: <http://www.nixp.ru/recipes/4.html> (дата обращения: 25.11.2015).
- 9 Создаем собственный deb-репозиторий (на примере создания репозитория для Komodo Edit) [Электронный ресурс]. — Режим доступа: <http://anosov.me/2011/07/make-own-deb-repository/> (дата обращения: 20.11.2015).
- 10 Как собрать бинарный deb пакет: подробное HowTo [Электронный ресурс]. — Режим доступа: <http://habrahabr.ru/post/78094/> (дата обращения: 12.10.2015).
- 11 Руководство начинающего разработчика Debian [Электронный ресурс]. — Режим доступа: <https://www.debian.org/doc/manuals/maint-guide/dreq.ru.html> (дата обращения: 30.10.2015).

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.