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.

2016

РЕФЕРАТ

Курсовая работа содержит 20 страниц, 3 рисунка, 0 таблицы, 8 источников, 1 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 LaTeX.

Список исполнителей

ПРАВИТЬ!!!

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 доработку программного модуля для определения операционной системы и создание репозитория проекта.

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

Мейта М.В. – программист, документатор, ответственный за ... верстку необходимой документации в системе LaTeX.

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 «COEX».

Терещенко Ю.А. – программист, ответственный за написание части системы для сбора информации из систем мгновенного обмена сообщениями.

Шиповской В.В. – программист, ответственный за разработку графического интерфейса пользователя системы «COEX» и доработку программного модуля для нахождения медиа-файлов.

Содержание

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;
- 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 разработчикоа 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 мержится не разработчиком, а вторым человеком, который производит вычитку и тестирование изменения;

– все коммиты должны быть осмысленно подписаны/прокомментированы.

3.2 Система компьютерной вёрстки TEX

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

LATEX — это созданная Лэсли Лэмпортом издательская система на базе TEX'а [3] LATEX позволяет пользователю сконцентрировать свои услия на содержании и структуре текста, не заботясь о деталях его оформления.

Для подготовки отчётной и иной документации нами был выбран LATEX так как совместно с системой контроля версий Git он предоставляет возможность совместного создания и редактирования документов. Огромным достоинством системы LATEX то, что создаваемые с её помощью файлы обладают высокой степенью переносимости. [4]

Совместно с LATEX часто используется BibTEX — программное обеспечение для создания форматированных списков библиографии. Оно входит в состав дистрибутива LATEX и позволяет создавать удобную, универсальную и долговечную библиографию. BibTEX стал одной из причин, по которой нами был выбран LATEX 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 огромнейшее 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 фраемворка QT

– iostream
– QChar
– QCryptographicHash
– QDateTime
– QDir
– QDirIterator
– QFile
– QFileInfo
– QIODevice
– QList
– QRegExp
– QString
– QTextStream
– QtSql/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 QtSql/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-при-записи), 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 ”regexp”, 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

GitLab — это веб-менеджер для работы с Git-репозиторием, имеющий ряд преимуществ для упрощения взаимодействия, командной работы с кодом, отслеживания ошибок и задач среди коллектива разработчиков. GitLab не только предоставляет хост-аккаунты аналогично GitHub, но и позволяет использовать свой программный код на сторонних серверах.[7]

В GitLab существует механизм распредления задач между разработчиками, так называемых issues (англ. «задание, вопрос, проблема»), что позволяет отслеживать выполнение того или иного рода задач в течение всей работы над проектом.

Для работы над проектом «COEX» проектной группой был поднят собственный репозиторий на сервере git.keva.su.

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 лего 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 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 «COEX» параллельное выполнение программных модулей («плагинов»), осуществляющих поиск остаточных данных с образа системы, изображений, различных файлов и т.д.

5.1.1 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 нежёстко: 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 5.1.

тут ссылка на http://doc.crossplatform.ru/qt/4.3.2/signalsandslots.html

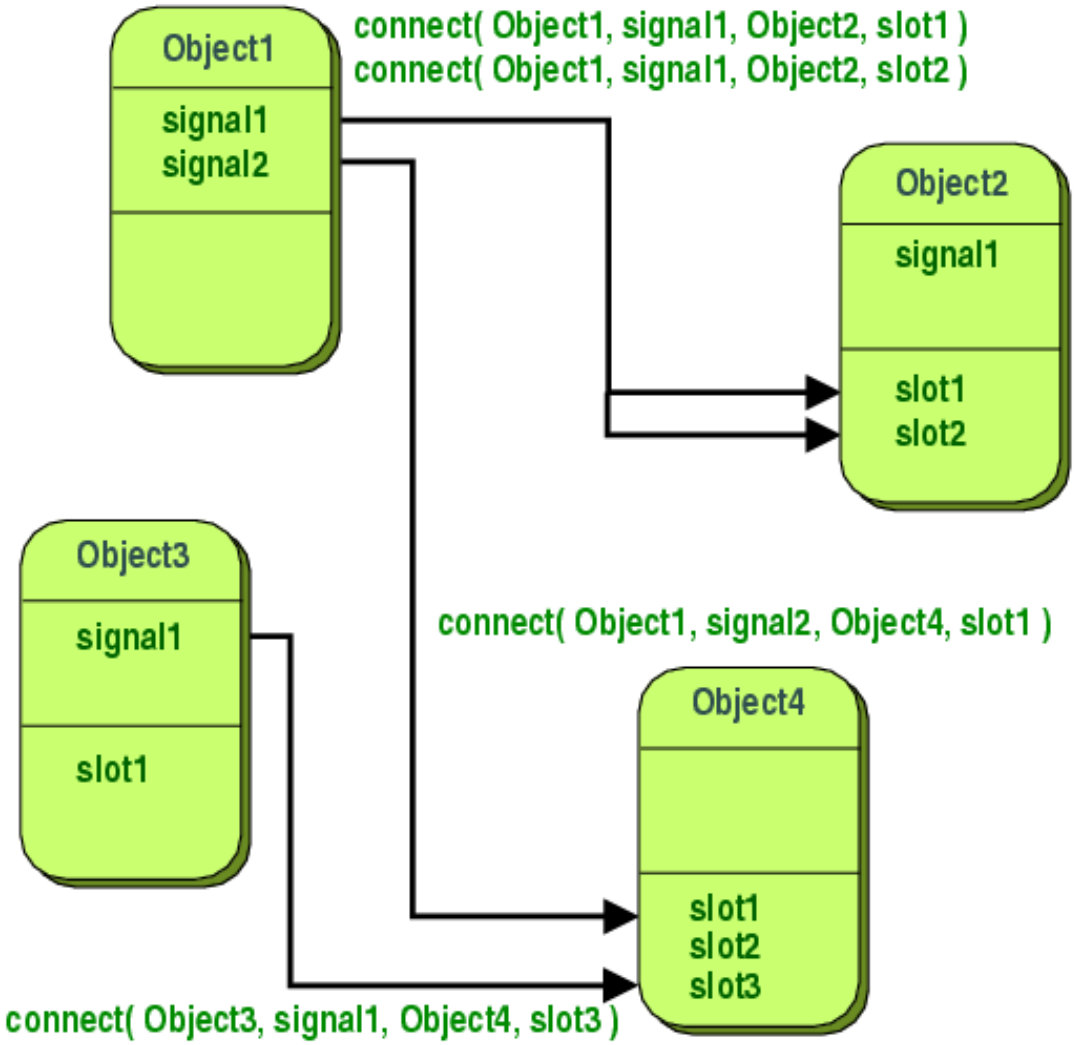

Рисунок 5.1 – Механизм сигналов и слотов для связи объектов в Qt

5.1.2 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.

http://qt-doc.ru/processy-i-potoki-v-qt.html

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. 5.2).

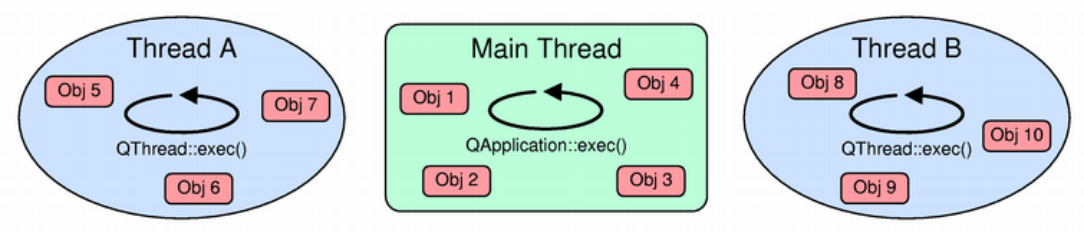

Рисунок 5.2 – Цикл обработки событий потоков в Qt

http://doc.crossplatform.ru/qt/4.4.3/threads.html

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.

Результат вывода программы представлен на рисунке 5.3.

Кроме написания потокового контроллера был доработан плагин ThreadTaskICQ таким образом, чтобы учитывались особенности работы с механизмом сигналов и слотов. Только после этого

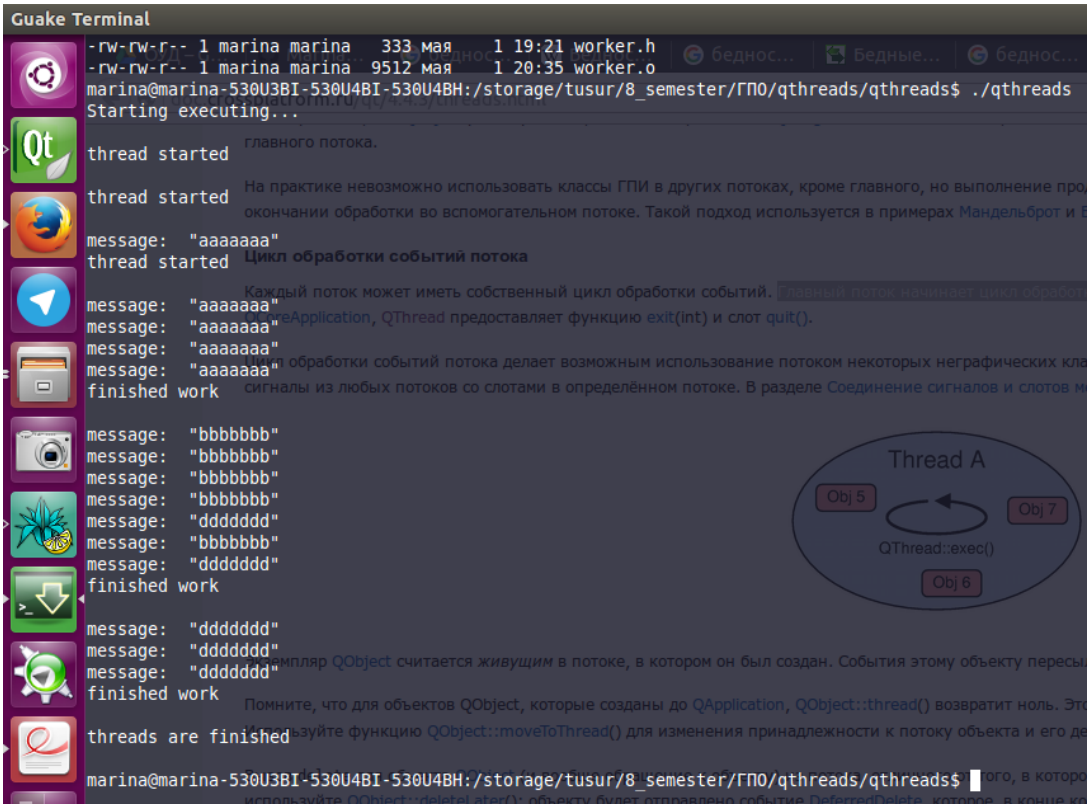

Рисунок 5.3 – Вывод программы qthreads

плагин может быть запущен в потоке с использованием класса QThreads.

5.1.3 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 «COEX», 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 «COEX» 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 «COEX».

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 LaTeX. МЦНМО, 2006. С. 448.

4 И. А. Чеботаев, П. З. Котельников. LaTeX 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 Code, test and deploy together [Электронный ресурс]. — Режим доступа: https://about.gitlab.com/ (дата обращения: 25.10.2015).

8 Справочник по XML-стандартам [Электронный ресурс]. — Режим доступа: http://msdn.microsoft.com/ru-ru/library/ms256177(v=vs.110).aspx.

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.