Работа с библиотекой Qt

Некоторые понятия

В ОС **Linux Mandrake** и **Red Hat** изначально закладывается возможность создавать компьютерные программы с помощью среды программирования **Qt**. Дело в том, что оконная среда (оболочка) **KDE** как раз и изготовлена на базе компонентов из библиотеки **Qt**. Если вы помните, в **Kylix** также задействованы подключаемые заголовочные файлы, названия которых начинаются с буквы **Q** – главного признака этой широкоприменяемой среды программирования, созданной фирмой **Trolltech**.

> Существует две версии **Qt** – коммерческая и бесплатная. Бесплатная версия входит в состав практически всех известных дистрибутивов **Linux**.

Существует два возможных варианта создания компьютерных программ в среде **Qt**.

- Первый из этих вариантов заключается в создании исходных кодов проекта программы полностью в тестовом редакторе. При этом вы должны знать заранее о том, как написать строку исходного кода для выполнения того или иного действия, т.е. вы должны, приступая к программированию, знать основы **Qt**. Этот вариант обычно используют профессиональные программисты и советуют всем начинать создание программ именно этим методом.
- Второй метод основан на том, что интерфейсная часть программы создается с помощью входящего в состав **KDE** так называемого «Qt дизайнера» – **Qt Designer**. Интерфейсной частью программы я назвал рабочее окно программы с установленными на нем различными кнопками, окнами редактирования и другими компонентами, необходимыми для управления работой программы, её «невидимой», основной частью. После создания в графической среде изображения рабочего окна программы, в текстовом редакторе следует на языке программирования C++ написать исходные коды функций, которые будут служить в качестве реакций на действия того или иного компонента.

Сразу следует запомнить новые названия, специфические для **Qt**. Все графические примитивы, называемые в **Kylix** и **C++ Builder** компонентами, здесь называются *виджиты*, функции, являющиеся реакциями на действия компонентов называются *слоты*. Также существуют *сигналы*, которые передаются от виджитов к слотам. Если быть точнее, то *слоты* – это те программные формы, в которые «вставляются» функции – обработчики событий.

Чтобы ваш **Linux** был настроен на создание компьютерных программ в среде **Qt**, при инсталляции этой ОС нужно выбрать режим работы компьютера

«Разработка» и в качестве главной оконной среды выбрать **KDE**. Но это не всегда гарантирует создание необходимых условий. Например, когда я начинал осваивать программирование в **Linux**, у меня была установлена версия **Mandrake 9.1**, в которой постоянно возникали самые разнообразные проблемы, порой неразрешимые. Версия **Mandrake 10.1** никаких проблем не создавала. Это говорит о том, на мой взгляд, что в **Mandrake 9.1** или не полностью укомплектована среда программирования **Qt**, или она плохо «прописана».

И еще. Начиная с версии **qt-x11-free-3.2** в эту среду программирования внесены большие изменения, которые делают все последующие их версии не совместимыми с более ранними версиями, например, **qt-x11-free-3.1.1** и еще более ранними, в то же время версии **Qt** начиная с **3.2** способны создавать так называемые «кроссплатформенные» (или межплатформенные) приложения, которые могут работать в некоторых различных операционных системах. В Интернете на сайте фирмы **Trolltech**, а также на большом числе «зеркальных» сайтов можно найти и скачать различные версии **Qt**. На момент написания этой статьи самой последней была версия **qt-x11-free-3.3.4**.

Программируем в Qt

В этом разделе рассмотрим процесс создания компьютерной программы с помощью **Qt Designer**. Мною при этом использовалась ОС **Linux Mandrake 10.1**, на которой установлена **qt-x11-free-3.3.3**.

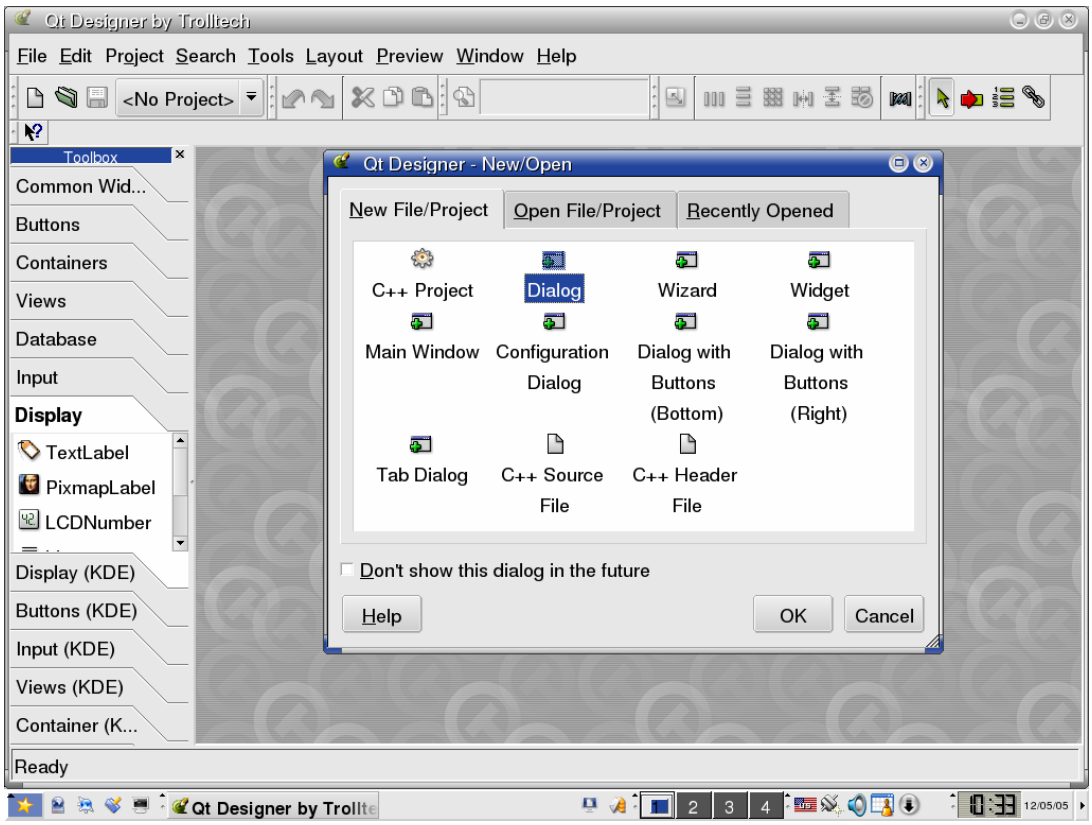

Рис. 1. Рабочее окно **Qt Designer**

На рис. 1 показано рабочее окно интегрированной среды разработки **Qt Designer**.

По центру рабочего окна размещается диалоговое окно выбора типа создаваемого проекта. Нами будет создаваться проект типа **Dialog**.

Слева на рабочем окне показан перечень разделов библиотеки виджитов (компонентов). В самом верхнем разделе располагаются наиболее часто употребляемые виджиты.

Следующая строка – **Buttons** – это самые различные кнопки.

И так далее. Не буду останавливаться подробно на описании – все это запросто осваивается самостоятельно, а для детального рассмотрения потребуется слишком много места. Желающим более подробно разобраться в процессе создания компьютерных программ в среде **Qt** советую в Интернете с сайта, расположенного по адресу http://www.linuxcenter.ru/, скачать книгу Ж. Бланшет и М. Саммерфилда «Разработка графического интерфейса с помощью библиотеки Qt3». В книге очень подробно описан весь процесс создания компьютерных программ в среде **Qt**, в том числе и с использованием **Qt Designer**.

Перед началом создания проекта **Qt** приложения следует создать директорию, в которой будут находиться все файлы проекта. Названием директории (папки) должно быть название нашего будущего приложения. Дадим название для директории `/prob3_qt/`. Под этим именем в дальнейшем автоматически средой программирования будет создан файл проекта и исполняемый файл программы.

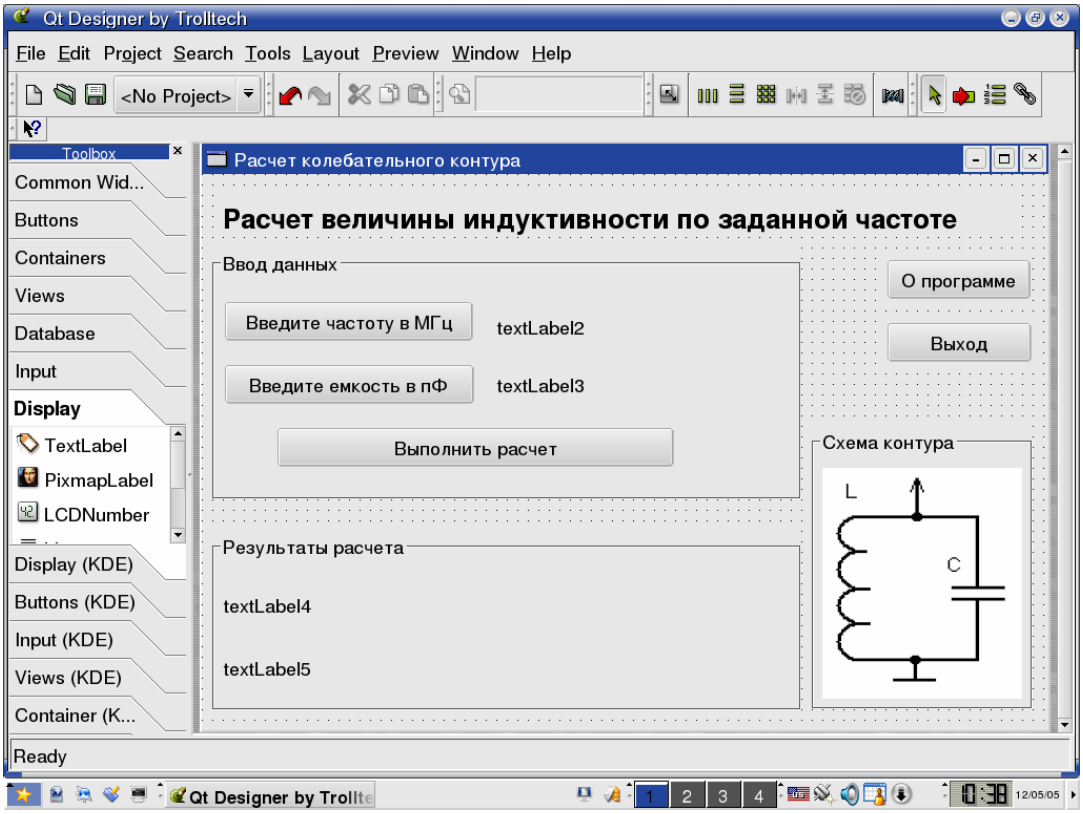

Рис. 2. Рабочее окно с заполненной формой

Первым делом необходимо рабочей форме проекта дать имя, которое будет служить именем основного класса и файлов создаваемых автоматически средой программирования. В данном случае устанавливаем для формы имя **Proba3**. Затем приступаем к установке на форму виджитов.

На рис. 2 показано рабочее окно **Qt Designer,** с установленными на форме виджитами, которые будут составлять интерфейс будущей программы.

> Дело в том, что мною принято решение показать вам решение одной и той же задачи различными методами, в двух различных средах программирования – в среде Kylix (рассмотрено в книге «Программирование в Borland Kylix») и с использованием компонентов QT, рассматриваемых в данной книге.

Для того, чтобы установленные на форме виджиты не портили облик рабочего окна программы при изменении его размера, существуют специальные приемы и компоненты, позволяющие стабилизировать размеры и места расположения виджитов на форме вне зависимости от размеров самой формы. Команды для точного расположения виджитов на форме находятся в разделе **Layout** главного меню. Однако в данном приложении, для простоты, мною применяется другой прием для стабилизации размеров и места расположения виджитов на форме – все виджиты устанавливаются внутри специальных *контейнеров*. В данном случае мною используется контейнер типа **groupBox**, который представляет собой рамку с текстом надписи.

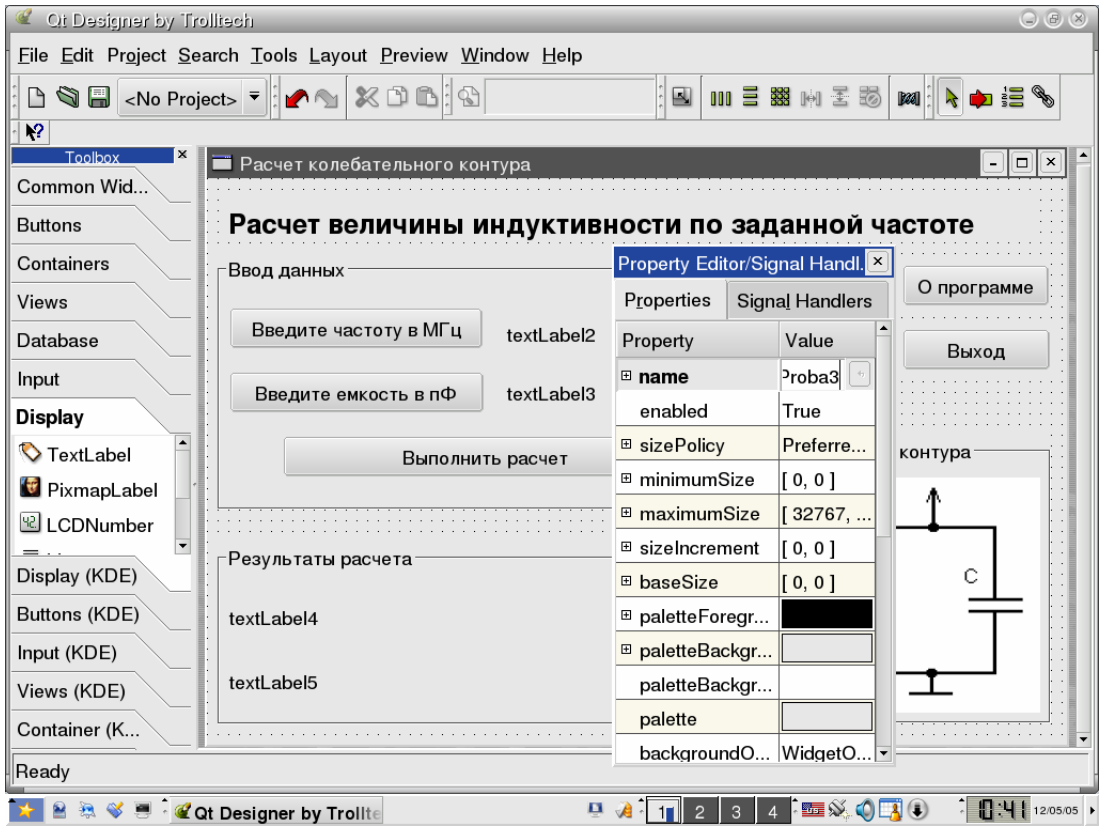

Рис. 3. Окно **Property Editor/Signal handlers**

Создаваемая нами в этом разделе программа называется **Proba3** и предназначена для расчета величины индуктивности контурной катушки по заданным величинам частоты и емкости.

Свойства установленных на форму виджитов выбираем в окне **Property Editor**, которое можно вызвать из главного меню командой **Window=>Views=>Property Editor/Signal handlers.** На рис. 3 показано окно **Property Editor/Signal handlers** на фоне формы.

После установки на форму всех виджитов следует назначить примерный порядок включения каждого из виджитов в работу, т.е. установить так называемый «порядок навигации». Для этого выбираем команду **Tools=>Tab Order**, в результате чего на каждом из виджитов, которые могут принимать фокус, появляются цифры в синих кружочках, как это показано на рис. 4.

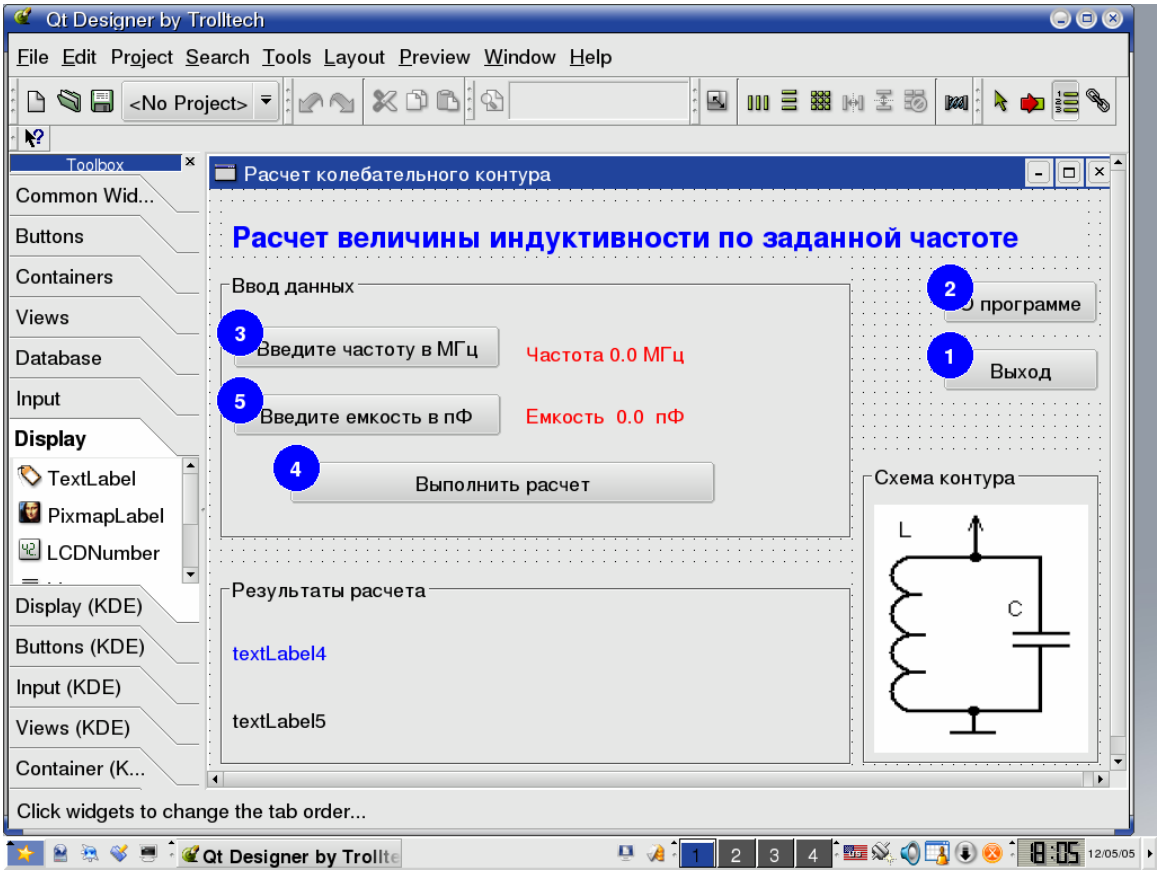

Рис. 4. Установка порядка навигации среди виджитов

Желаемый порядок очередности срабатывания виджитов устанавливается щелчками мышкой и клавишей **<Tab>.** Для выхода следует нажать клавишу **<Esc>.**

После этих процедур следует сохранить созданную форму в заданной директории. При этом средой программирования автоматически будет создан файл **proba3.ui**, в котором будет сохранена форма с установленными на ней виджитами.

Теперь наступила очередь создания файла с описанием исходных кодов для каждого из виджитов. Эта работа выполняется в специальном текстовом

редакторе, который вызывается двойным щелчком левой кнопки мышки на любом участке формы, свободном от виджитов. После этого среда запрашивает о желании вами создать файл **proba3.ui.h**, в котором будут находиться исходные коды слотов.

На рис. 5 показано рабочее окно IDE с текстовым редактором.

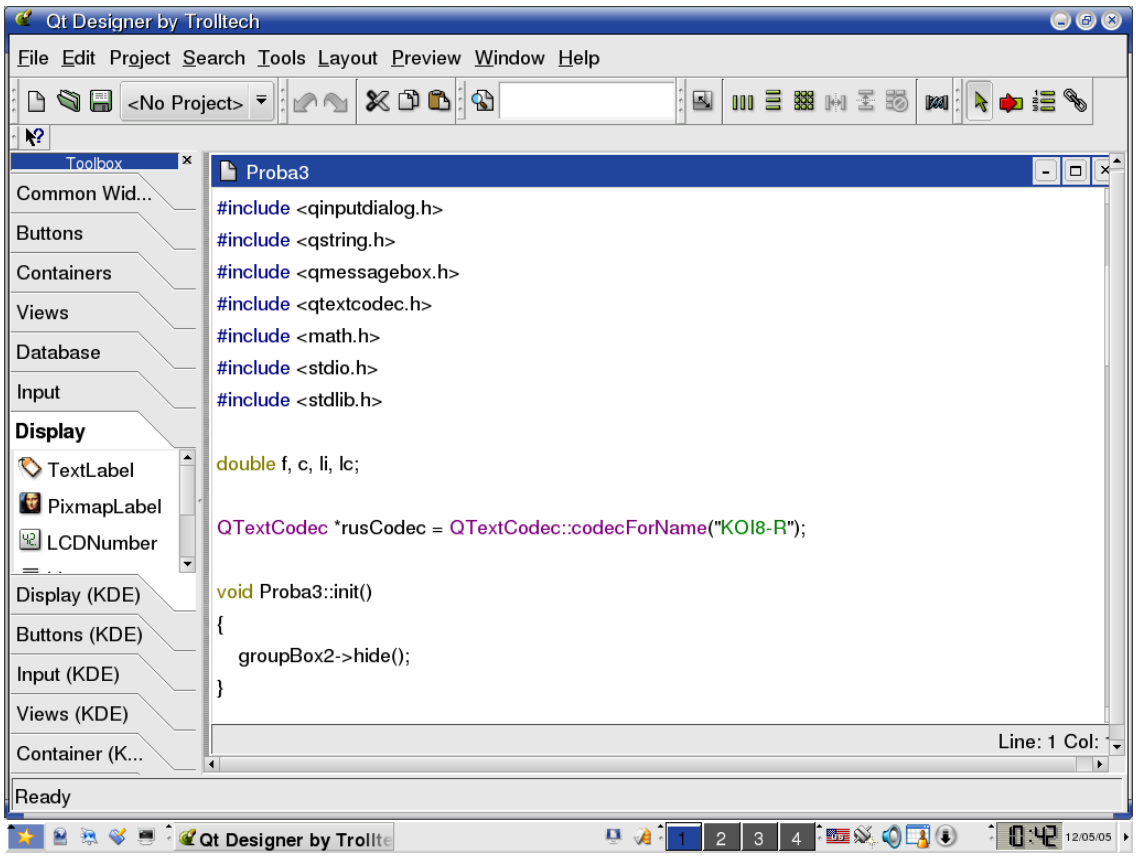

Рис. 5. Окно с редактором текста

В текстовом редакторе должны быть созданы функции (слоты), которые должны обрабатывать сигналы, поступающие в программу от виджитов. В листинге 1. приведен текст исходных кодов создаваемых нами слотов.

Листинг 1. Текст файла *proba3.ui.h*

```
/****************************************************************************
** ui.h extension file, included from the uic-generated form
**implementation.
**
** If you want to add, delete, or rename functions or slots, use
** Qt Designer to update this file, preserving your code.
**
** You should not define a constructor or destructor in this file.
** Instead, write your code in functions called init() and destroy().
** These will automatically be called by the form's constructor and
** destructor.
****************************************************************************/
#include <qinputdialog.h>
#include <qstring.h>
```

```
#include <qmessagebox.h>
#include <qtextcodec.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

double f, c, li, lc;
   // Следующая строка разрешает в строках типа QString использовать
   //   русские буквы
QTextCodec *rusCodec = QTextCodec::codecForName("KOI8-R");
//-------------------------------------------------------------------------
void Proba3::init()                                          // 1
{
    groupBox2->hide(); // скрывает нижний контейнер groupBox2
}
//-------------------------------------------------------------------------
void Proba3::about()                                         // 2
{
   // Следующая строка разрешает в функциях tr() использовать
   //   русские буквы
    QTextCodec::setCodecForTr(QTextCodec::codecForName("KOI8-R"));
    QMessageBox::about(this, tr(" О программе PROBA3"),
              tr(" <p> Программа  <h2>PROBA3</h2>"
              "<p>Автор:  Геннадий - <b>R3XB</b>  &copy;   2005г "
              "<p>http://ra3xb.narod.ru;  mailto:r3xb@kaluga.ru"));
}
//-------------------------------------------------------------------------
void Proba3::buton1()                                        // 3
{
    bool ok;
 QString ss01 = rusCodec->toUnicode("Введите частоту в МГц");
    f=QInputDialog::getDouble("Proba3", ss01, 0,
                      0, 10000, 2, &ok, this);
      QString ss1;
    ss1 = rusCodec->toUnicode("Величина частоты f =  %1  МГц ").arg(f,0,
'g', 10);
    textLabel2->setText(ss1);
}
//-------------------------------------------------------------------------
void Proba3::buton2()                                         // 4
{
    bool ok;
 QString ss02 = rusCodec->toUnicode("Введите емкость в пФ ");

    c=QInputDialog::getDouble("Proba3", ss02, 0,
                      0, 10000, 2, &ok, this);
    QString ss2;
    ss2 = rusCodec->toUnicode("Величина емкости  C =  %1 пФ").arg(c,0, 'g',
10);
    textLabel3->setText(ss2);
```

```
}
//--------------------------------------------------------------------------
void Proba3::buton3()                                          // 5
{
    lc = pow((159/f), 2);
    li = lc/c;
    QString ss3, ss4;
ss3 = rusCodec->toUnicode("Величина индуктивности L =  %1 мкГн").arg(li, 0,
'g', 10);
 ss4 = rusCodec->toUnicode("Вспомогательная величина LC = %1 ").arg(lc, 0,
'g', 10);
    textLabel4->setText(ss3);
    textLabel5->setText(ss4);
    groupBox2->show();
}
//--------------------------------------------------------------------------
```

В начале листинга 1 приведен текст разъяснений на английском языке, который создается средой программирования. В тексте сказано, что в этот файл можно включать свои функции, которые обрабатываются **Qt Designer'**ом совместно с файлом описания установленных на форме виджитов, что можно создавать функции под названиями `init()` и `destroy()`, которые будут обрабатываться соответственно конструктором и деструктором класса формы.

После этих разъяснений описаны подключаемые заголовочные файлы, в которых описываются функции, задействованные в этом файле.

Первая функция (№1) `void Proba3::init()` вызывается и срабатывает при создании рабочего окна программы. В тело этой функции запишем команду, которая скрывает контейнер **groupBox2** вместе с размещенными внутри его виджитами сразу при создании рабочего окна программы.

Функция 2 носит название `about()` и служит для вывода на экран диалогового окна с информацией о данной программе по нажатию клавиши <О программе>.

Функция 3 является реакцией на нажатие клавиши <Введите частоту в МГц>, а следующая за ней функция 4 – реакция на нажатие клавиши <Введите емкость в пФ>.

Функция 5 является реакцией на нажатие клавиши <Выполнить расчет>. Здесь выполняются математические вычисления, а в последней строке делается видимым скрытый в функции 1 контейнер **groupBox2.**

После того, как нами написаны исходные тексты функций, необходимо все эти функции (кроме первой, которая изначально подключена к конструктору) превратить в слоты, т.е. подключить к соответствующим виджитам, реакциями которых они являются. Для выполнения этой процедуры в главном меню выбираем **Edit=>Edit Slots…,** после чего появляется окно **Edit Functions**, показанное на рис. 6.

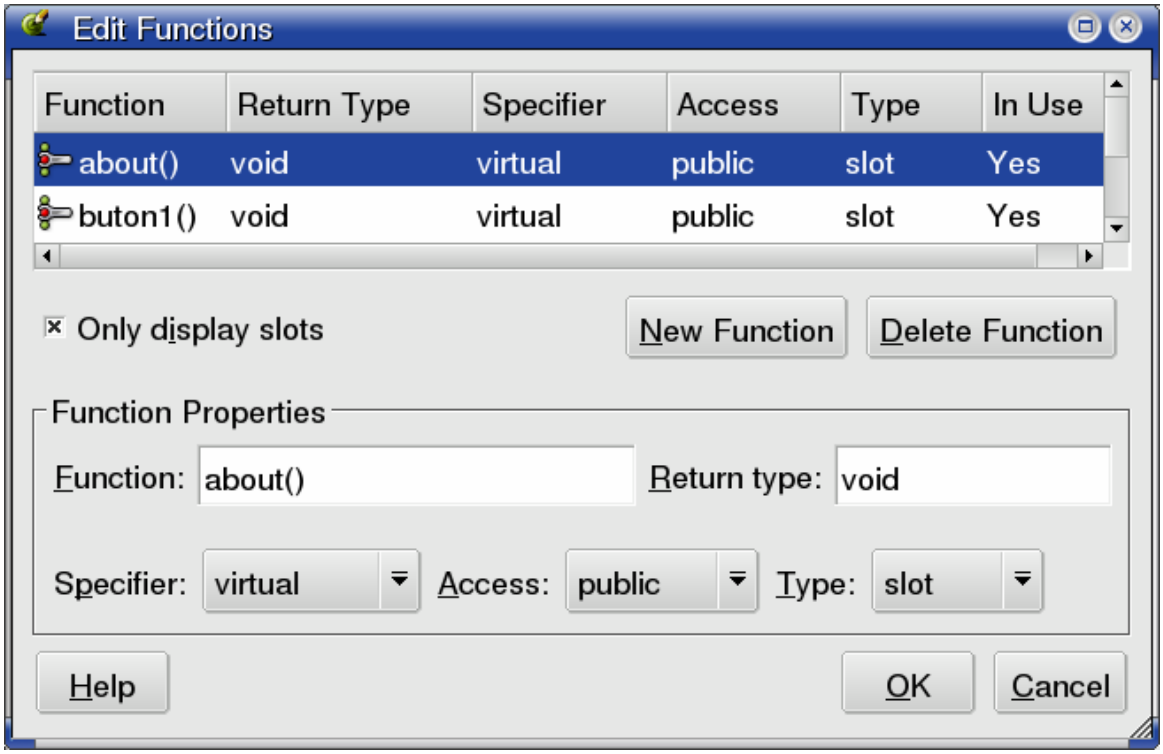

Рис. 6. Окно Edit Functions

Нажимаем на этом окне клавишу <New Function> и водим в строку первую функцию – `about()`. Затем точно таким же путем вводим все последующие созданные нами функции, которые являются реакциями на действия имеющихся на форме виджитов. Таким путем мы обозначаем эти функции как слоты.

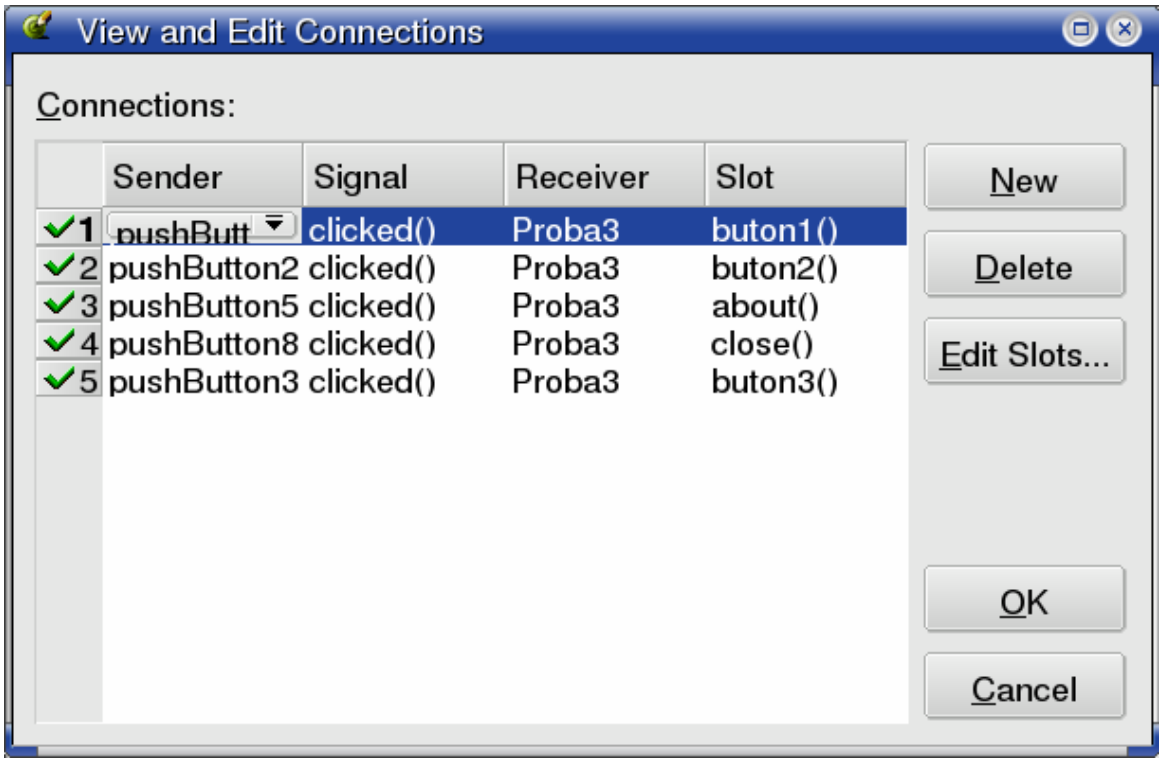

Рис. 7. Окно редактора связей

Следующим этапом необходимо соединить сигналами виджиты с соответствующими слотами. Для этого выбираем в главном меню Edit=>Connections, в результате чего появляется диалоговое окно **View and Edit Connections**, изображенное на рис. 7.

Нажимаем на клавишу **New** и вводим сначала в колонке **Sender** (Передатчик) название виджита, к которому нужно подключить слот. В колонке **Signal** во всех строках пишем **clicked(),** в колонке **Receiver** (Приемник) пишем во всех строках имя формы – **Proba3**. В последней колонке **Slot** вводим имя функции, соответствующей данному виджиту.

Теперь снова сохраняем все в заданной директории **prob3_qt**. Если заглянуть в эту директорию, то обнаружим там только два файла – файл **proba3.ui** и файл **proba3.ui.h**.

Для полного комплекта необходимо еще создать в текстовом редакторе файл main.cpp и разместить его в этой же директории **prob3_qt.**

Листинг 2. Файл main.cpp

```
#include <qapplication.h>
#include "proba3.h"
#include <qtextcodec.h>

int main(int argc, char *argv[])
 {
   QApplication a(argc, argv);
   Proba3 *dialog = new Proba3;
   a.setMainWidget(dialog);
   dialog->show();
   return a.exec();
 }
//---------------------------------------------------------------------------
```

Этот файл объединяет в единую рабочую среду все три файла, находящиеся в заданной директории проекта.

Заключительным этапом программирования является создание исполняемого файла проекта.

Для этого, находясь в директории проекта, вводим в командной строке команду **qmake –project**, которая создает файл проекта **prob3_qt.pro**. Затем вводим команду **qmake prob3_qt.pro**, в результате выполнения которой должен быть создан файл **Makefile.**

После того, как будет создан файл **Makefile** следует ввести команду **make**, по которой **Makefile** начинает свою работу, результатом которой должно быть создание исполняемого файла программы **prob3_qt**, который создается только в том случае, если не допущено ни единой ошибки.

Исполняемый файл запускается в работу обычным образом, т.е. либо из командной строки, либо из файлового менеджера **KDE**. На рис. 8 показано рабочее окно программы сразу после запуска в работу.

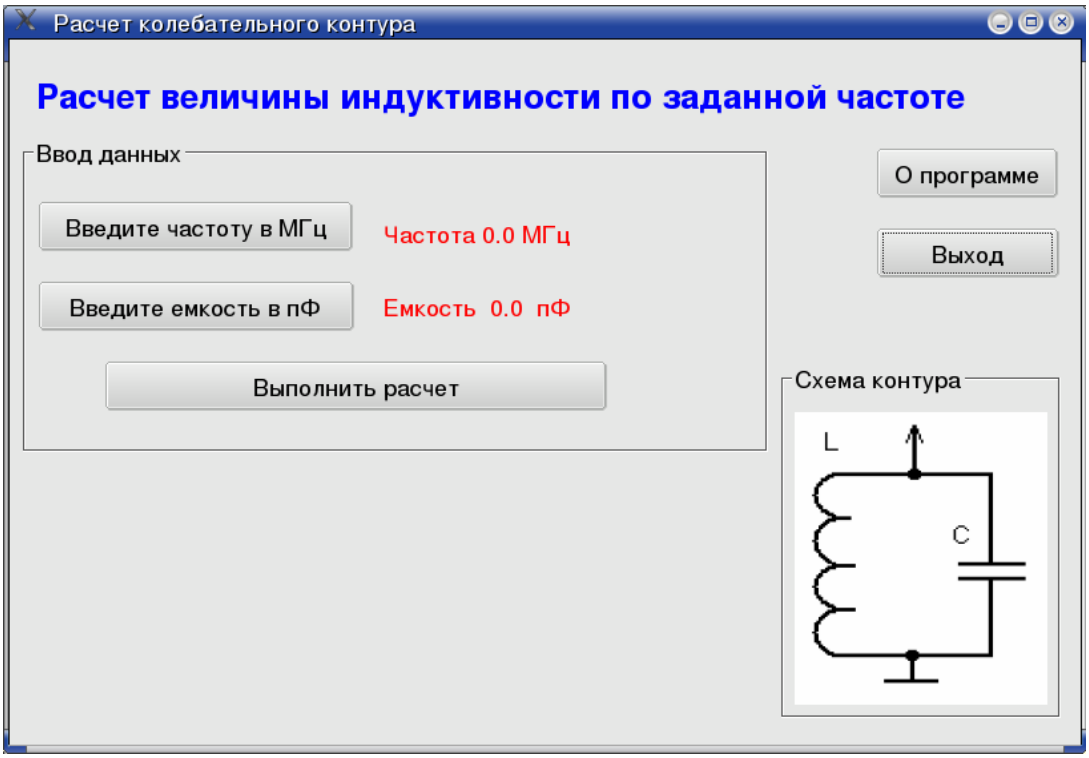

Рис. 8. Начальный вид рабочего окна

На рис. 9 на фоне главного рабочего окна показано диалоговое окно, предназначенное для ввода в программу частоты.

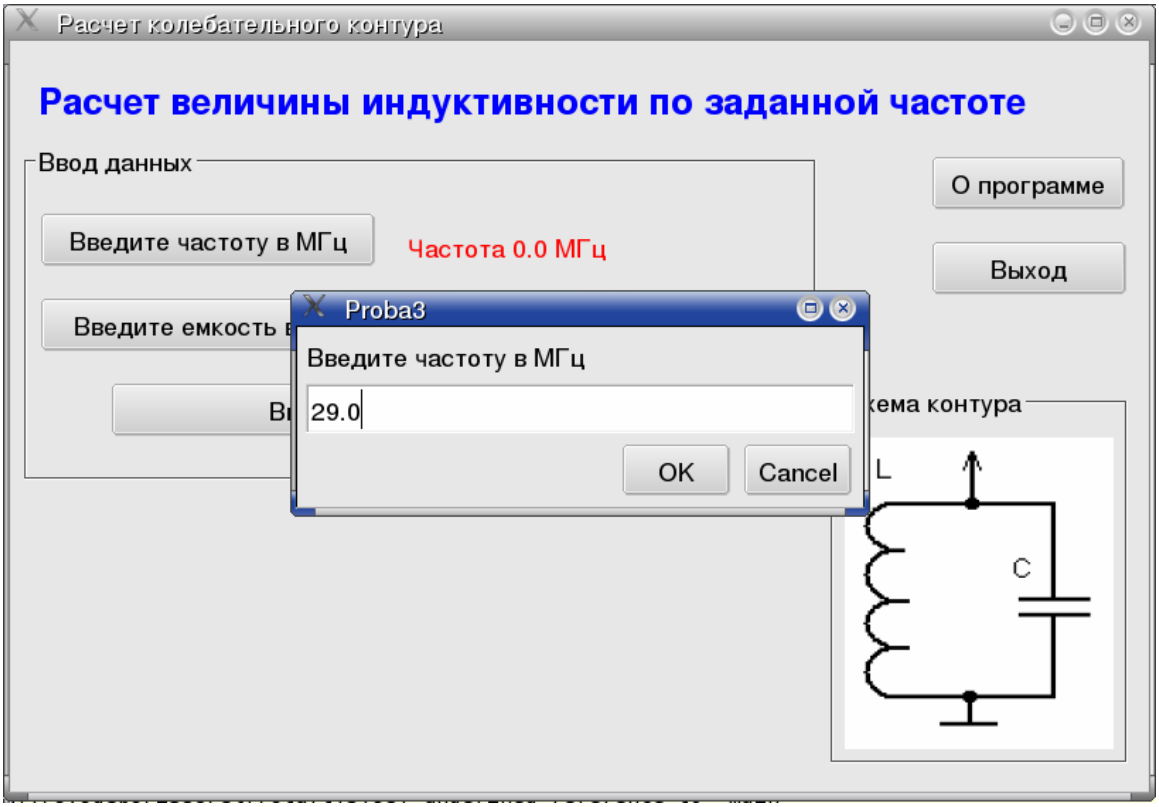

Рис. 9. Диалоговое окно для ввода частоты

На рис. 10 изображено рабочее окно программы после нажатия клавиши <Выполнить расчет>.

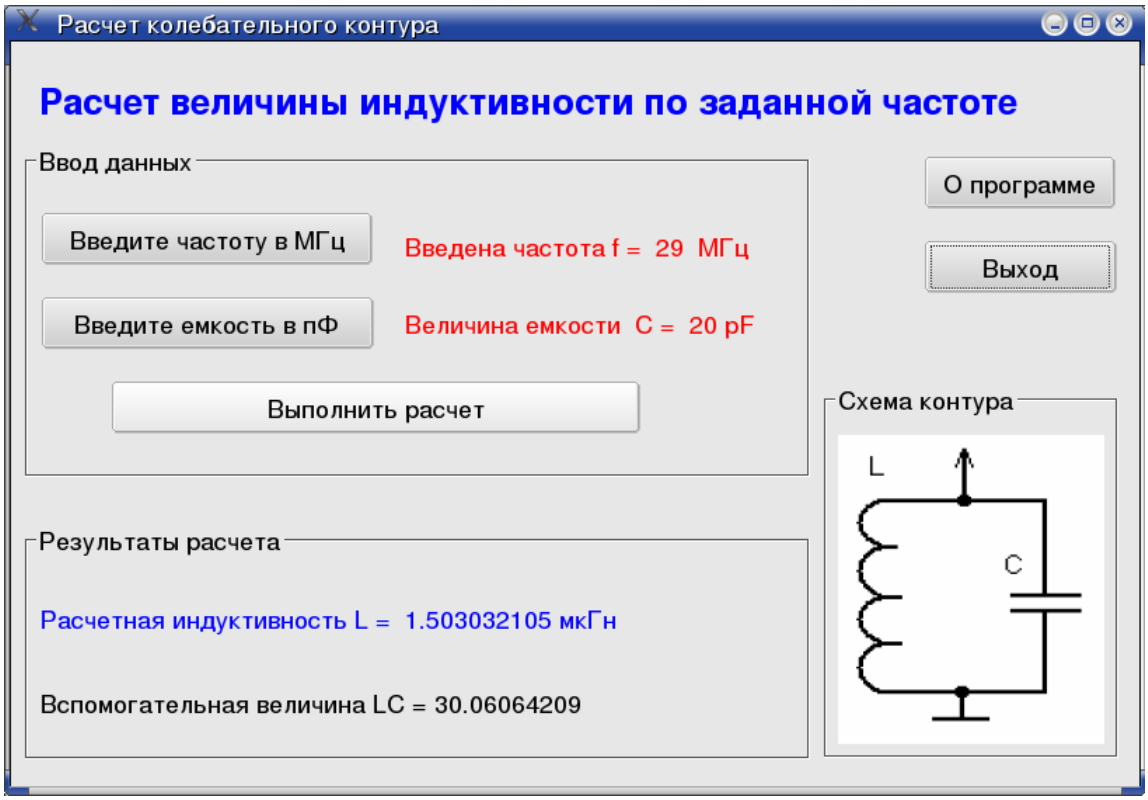

Рис. 10. Главное окно программы

На рис. 11 показано диалоговое окно с информацией о программе и её авторе.

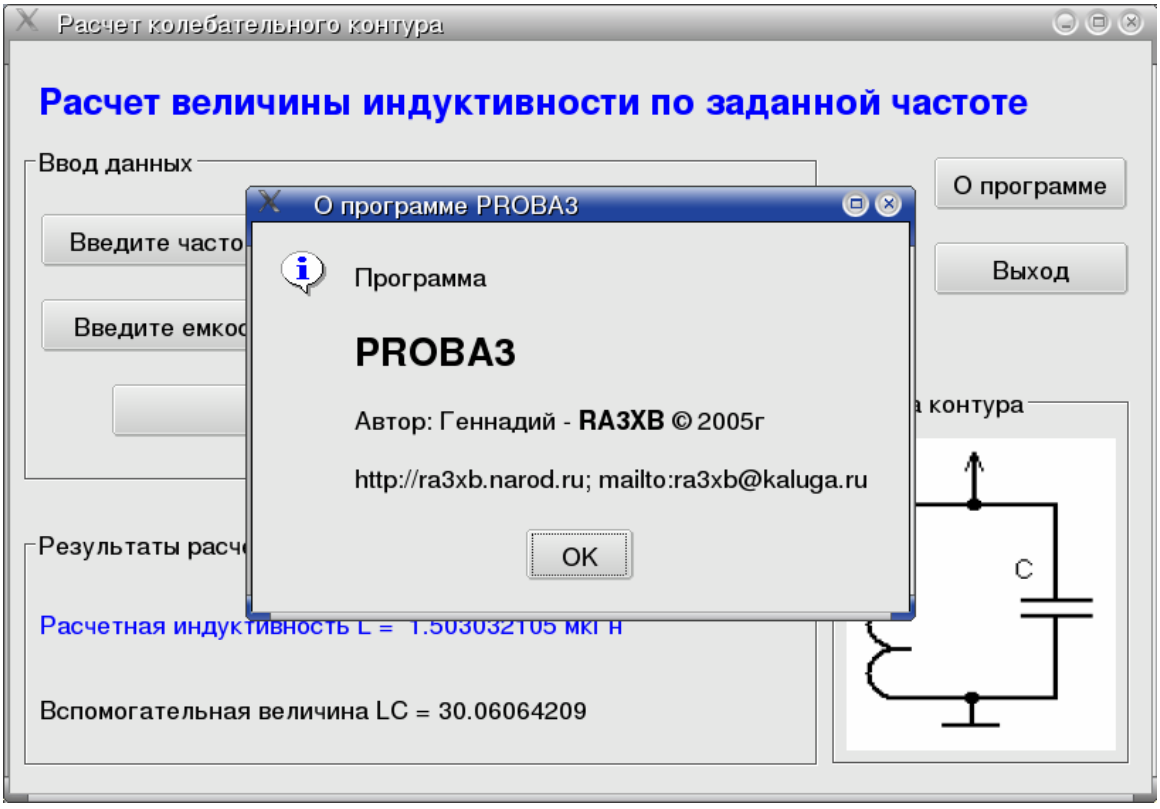

Рис. 11. Диалоговое окно «О программе»

На этом процесс создания программы в среде программирования **Qt** закончен. Далее необходимо создать текстовые файлы **INSTALL** и **README**. Если программа довольно сложная, то нужно также создать справочник с описанием всех особенностей программы.

Кроме того, следует знать, что оконная система **KDE** также может служить средой для создания компьютерных программ, в том числе и совместных программ **KDE + Qt.**

Как запустить в работу Qt-приложение

Созданный выше проект **Qt** приложения можно запустить в работу, согласно заверению разработчиков, на самых разных компьютерах под управлением самых разных операционных систем. Необходимым условием является только наличие на этом компьютере работоспособной **Qt** библиотеки.

Проверить наличие такой библиотеки можно путем ввода в командной строке команды: `locate qmake | grep bin`

Если все обстоит нормально, то в ответ на ввод этой команды система выдаст ответ:

`/usr/lib/qt3/bin/qmake` или что-то в этом роде.

Например, в **Red Hat 9.0** в ответ мною получено:

```
/usr/bin/qmake
/usr/bin/qmake3
/usr/lib/qt-3.1/bin/qmake
```

Если система выдала сообщение о том, что эта команда для оболочки ОС является неизвестной, то необходимо обновить (или создать заново) базу данных задействованных в системе команд. Это делается следующим образом.

Сначала в командной строке вводится команда: `su`

В ответ на которую система запрашивает у вас пароль суперпользователя. После ввода пароля вводим команду для создания базы данных: `updatedb`

Если после создания (или обновления) базы данных система снова отказывается признавать команду `qmake`, то **Qt** библиотека на вашем компьютере либо не установлена, либо установлена очень старая версия, в которой `qmake` отсутствует. В этом случае либо нужно установить на компьютер более новый вариант **Linux**'а, либо скачать **Qt** из Интернета и установить на свой компьютер. Мне думается, что первый вариант более удобный и надежный.

В лучшем случае, когда все в порядке и система признает команду `qmake`, запустить в работу новое **Qt** приложение совсем не трудно.

Рассмотрим вариант ваших действий, когда вы скачали с моего сайта исходные коды программы **proba3.**

1. Создаем директорию под названием prob3_qt. Это название предлагаю вам для того, чтобы приблизить условия к моим.
2. В эту директорию скопировать три основных файла, взятых вами из пакета той программы, исполняемый файл которой вы хотите иметь на своем компьютере. Первый из этих файлов – файл `main.cpp`, второй – файл с расширением `*.ui`, третий – файл с расширением `*.ui.h`. В рассматриваемом случае это файлы `main.cpp`, `proba3.ui` и `proba3.ui.h`.
3. Находясь в заданной директории, вводим в командной строке команду `qmake –project`, которая создает здесь же файл проекта **prob3_qt.pro**.
4. Затем вводим команду `qmake prob3_qt.pro`, в результате выполнения которой должен быть создан файл **Makefile.**
5. После того, как будет создан файл **Makefile** следует ввести команду `make`, по которой **Makefile** начинает свою работу, результатом которой должно быть создание исполняемого файла программы **prob3_qt**, который запросто создается, если не было допущено ни единой ошибки при написании текстов исходных кодов функций – слотов.

Полученный исполняемый файл запускается в работу обычным образом, т.е. из командной строки по команде `./prob3_qt` или в файловом менеджере двойным щелчком левой кнопкой мышки на картинке файла.

Коротко о Qt

Qt – мультиплатформенная C++ **GUI** (Graphical User Interface – Графический интерфейс пользователя (на русском языке это будет как **ГИП**)) прикладная структура. Другими словами можно сказать, что **Qt** – это среда для создания интерфейса пользователя к компьютерным программам, способная работать на разных компьютерах и под управлением различных операционных систем.

Qt 3.3 вводит новые особенности и много усовершенствований по сравнению с рядом **Qt 3.2.x**. Приложения **Qt** ряда версии **3.3** в двоичном исполнении совместимы с приложениями ряда **3.2.x**, при этом созданные приложения для **3.2** продолжат работу с **Qt 3.3**.

Эти утверждения были проверены на следующих платформах:

```
win32-borland
win32-g++
win32-icc
win32-msvc
win32-msvc.net
```

```
aix-g++
aix-xlc
aix-xlc-64
freebsd-g++
freebsd-icc
hpux-acc
hpux-g++
irix-cc
irix-cc-64
irix-g++
linux-ecc-64
linux-g++
linux-icc
solaris-cc
solaris-cc-64
solaris-g++
solaris-g++-64
tru64-g++

macx-g++
macx-pbuilder
```

Если вы хотите использовать **Qt 3** на неподдерживаемой версии **Unix**, попробуйте войти в контакт с разработчиками по адресу qt-bugs@trolltech.com.

Как получить пакет **Qt**:

Qt Open Source Edition: Скачайте архив `.tar.gz` с сайта **ftp.trolltech.com.** Для более быстрого скачивания, используйте *ftpsearch*, и ищите `qt-x11-free-3.3.4`, чтобы найти этот пакет Qt на более близким от вашего местонахождения зеркале **ftp**.

Qt Professional Edition или **Qt Enterprise Edition**: чтобы приобрести эти коммерческие пакеты, необходимо сначала получить по электронной почте инструкции о том, как получить новое исполнение **Qt**. Для контакта обращайтесь на sales@trolltech.com.

О любых проблемах, с которыми вы сталкиваетесь при работе с **Qt 3.3** нужно сообщать по адресу qt-bugs@trolltech.com.

Qt – это торговая марка фирмы **Trolltech AS**.

Как установить Qt на компьютере с ОС Linux

Прежде, чем вы начнете устанавливать Qt библиотеку и программы примера на свой компьютер, необходимо запустить сценарий "configure", чтобы установить информацию о платформе и другие особенности. Вы можете использовать выбор платформы, чтобы определить операционную систему и компилятор, который должен будет использоваться.

Поддерживаемые платформы и компиляторы:

```
Aix-g++  hpux-g++  linux-g++ solaris-cc-64  win32-g++
aix-xlc hpux-g++-64  linux-g++-64  solaris-g++  win32-
icc aix-xlc-64 irix-cc linux-icc  solaris-g++-64
win32-msvc freebsd-g++  irix-cc-64 macx-g++  tru64-cxx
win32-msvc.net  freebsd-icc irix-g++  macx-pbuilder
tru64-g++  hpux-acc  linux-ecc-64  solaris-cc  win32-
borland
```

Если вы испытываете проблемы при установке и компиляции **Qt 3.x**, ищите ответы и замечания по различным платформам платформы в Интернете на сайте по адресу: **http://www.trolltech.com/developer/platforms/,** куда фирма отправляет по почте информацию о всех известных проблемах, как только они будут определены.

Пример задействования сценария "configure" :

```
./configure -platform irix-cc-64 -shared -debug
```

Настройка:

Вы можете создавать вашу собственную конфигурацию, прибавляя новые файлы в mkspecs директорию. Используйте существующие указанные конфигурации как отправные точки.

Инсталляция Qt/X11 версии 3.3.4

В зависимости от того, в какой из директорий вы хотите установить **Qt/X11**, вполне возможно, что необходимо зарегистрироваться как root.

В большинстве случаев этого не требуется.

Далее будет рассказано о том, как установить библиотеку **Qt/X11** в рекомендуемую директорию /usr/local/.

Скачать файл библиотеки **qt-x11-free-3.3.4.tar.gz** из Интернета и скопировать его в директорию `/usr/local/.` Войти в эту директорию по команде:

```
cd  /usr/local
```

Далее следует распаковать файл библиотеки:

```
gunzip qt-x11-free-3.3.4.tar.gz
tar xvf qt-x11-free-3.3.4.tar
```

Таким путем нами был создан готовый к работе пакет библиотеки **Qt/X11**, содержащий в директории `/usr/local/qt-x11-free-3.3.4`. В этом пакете все файлы находятся в исходных кодах, которые далее необходимо привести в рабочее состояние

Для удобства дальнейшего обращения переименуем `qt-x11-free-3.3.4` в более краткое обозначение `qt` (или в `qt3`):

```
mv qt-x11-free-3.3.4 qt
```

Теперь следует помнить, что все файлы библиотеки **Qt/X11** находятся в директории `/usr/local/qt`.

2. Вполне возможно, что вам потребуется прописать основные переменные среды окружения **Qt/X11** в файле `.profile` (или `.login`, в зависимости от вашей оболочки) в вашем домашней директории. Если такого файла в вашей домашней директории нет, то его следует создать в текстовом редакторе. В новом файле должны быть прописаны следующие пути:

```
QTDIR     - это основная директория Qt
PATH      - это путь к программе moc и другим
инструментам Qt
MANPATH   - это путь к Qt man страницам
LD_LIBRARY_PATH   - путь к Qt подключаемым
библиотекам
```

Замечание. При работе под IRIX нужно дополнить пути цифрами LD_LIBRARYN32_PATH или LD_LIBRARY64_PATH. Чтобы точно установить переменную, которая соответствует вашей конфигурации, смотрите `rld (5)` man page для получения большего количества информации.

Если ваша оболочка – **bash, ksh, zsh** или **sh**, то необходимо в файле `.profile` добавить следующие строки:

```
QTDIR=/usr/local/qt
PATH=$QTDIR/bin:$PATH
MANPATH=$QTDIR/doc/man:$MANPATH
LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH
export QTDIR PATH MANPATH LD_LIBRARY_PATH
```

Если у вас используется оболочка **csh** или **tcsh,** то необходимо в файл `.login` добавить следующие строки:

```
setenv QTDIR /usr/local/qt
setenv PATH $QTDIR/bin:$PATH
setenv MANPATH $QTDIR/doc/man:$MANPATH
setenv LD_LIBRARY_PATH $QTDIR/lib:$LD_LIBRARY_PATH
```

После выполнения приведенных выше операций необходимо перезапустить консоль для того, чтобы система знала, что по крайней мере пути $QTDIR и $PATH для дальнейших действий уже установлены. Без них установки может появиться сообщение об ошибке.

3. Если вы используете лицензионную копию, то следует установить ваш файл лицензии как `$HOME/.qt-license`.

Для **free** версии файл лицензии не нужен.

4. Создание собственно **Qt** библиотеки.

Следующим этапом создается собственно **Qt** библиотека, создаются программы примеров, обучающая программа, и инструменты (например **Qt Designer**).

Первая команда:

```
./configure
```

По этой команде производится конфигурирование (`configure`) **Qt** библиотеки для вашей машины. Для получения справки по команде `./configure` следует записать команду как `./configure –help`.

При установке **Qt** на разные платформы необходимо выполнять команду с различными опциями, для этого смотрите файл PLATFORMS.

Команда `./configure` выполняется в течение нескольких минут, в зависимости от быстродействия установленного на вашем компьютере процессора.

Создавать библиотеку, все примеры и обучающую программу следует по команде:

```
make
```

Если ваша платформа или компилятор не поддерживаются операционной системой, пожалуйста смотрите описания в Интернете на сайте

http://www.trolltech.com/platforms/ для информации относительно известных выпусков и версий.

Команда `make` выполняется довольно долго. Например, на компьютере с процессором **Celeron 2700 MHz** работа по выполнению этой команды продолжалась в течение 120 минут, на компьютере с **Celeron 800** – порядка шести часов подряд.

Если выполнение двух предыдущих команд произошло без инцидентов, то следующей командой должна быть:

```
make install
```

По этой команде выполняется установка всех файлов из пакета **Qt** именно в те директории на диске вашей машины, для работы в которых эти файлы предназначены.

Команда `make install` выполняется на компьютере с прцессором **Celeron 2700 MHz** в течение примерно 20 минут, на компьютере с **Celeron 800** – порядка 70 минут подряд.

(Вводите `./configure -help` для получения большего количества информации).

5. В очень немногих случаях вы будете должны запустить `/sbin/ldconfig` или кое-что подобное в этом роде, если вы используете разделяемые библиотеки.

Такая ситуация может быть, если система выдает сообщение вроде:

```
can't load library 'libqt.so.3'
(Не может прочитать библиотеку 'libqt.so.3')
```

Чтобы работать с файлом `/sbin/ldconfig` вы должны зарегистрироваться как `root` на вашей системе.

И не забудьте устанавливать LD_LIBRARY_PATH как объяснено выше, в пункте 2.

6. Диалоговая HTML документация установлена в `/usr/local/qt/doc/html/`, главная страница – `/usr/local/qt/doc/html/index.html` рабочие `man` страницы установлены в `/usr/local/qt/doc/man/`.

Желаю успехов в освоении программирования в ОС Linux!

Замечания, возражения и пожелания можете направлять автору по электронной почте mailto:r3xb@kaluga.ru