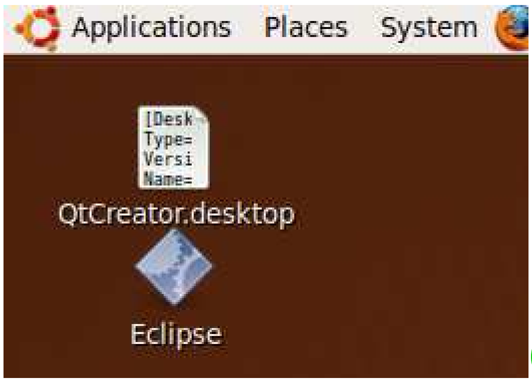
Getting Started with Qt



GUI Programming with Qt4

__The aim for this guide is to help you to__:

- Install Qt.
- Get pass the Hello World state and learn some basic Qt programming idioms.
- You can find everything in this guide on the Internet, but this guide gathers information from different sources and filter out irrelevant stuff with the purpose to give you a smooth start with Qt.

Warning: Qt Software update their software frequently so the version numbers show in this guide may differ for the current versions found on their web site. And under Linux installation all commands show are for Ubuntu Linux (other Linux dialects may require other commands).

Table of content

1 Introduction

1.1.1 What is Qt?

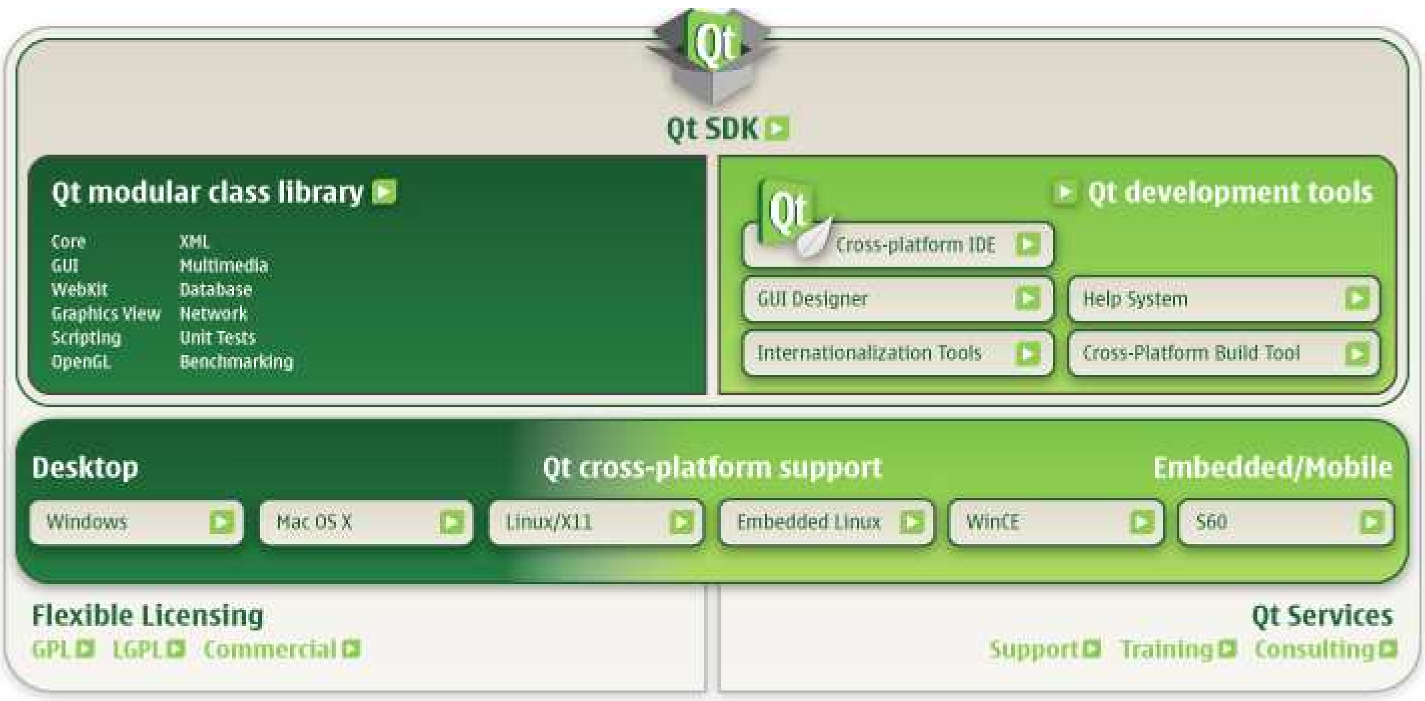
Qt is a cross-platform application and UI framework. It includes:

- a cross-platform class library
- integrated development tools
- a cross-platform IDE.

Using Qt, you can write applications once and deploy them across many desktop and embedded operating systems without rewriting the source code. But you must rebuild the sources on all platforms where you wants to run your application.

QT is coded in standard C++ so generally speaking you can use QT on every platform that has a C++ compiler. Application development is typical done in C++, but it is possible to use Java.

Notice: QT is for application development – not driver development or recompiling the Linux kernel.



1.1.2 Who makes Qt

Qt is primarily developed and maintained by Qt Software, a unit within Nokia.

The Qt framework was initially developed by Haavard Nord and Eirik Chambe-Eng. They started the development in 1991. In 1994 they started the company Trolltech and finally in 1995 they released the first public version of Qt. In 2008 Nokia acquired Trolltech ASA to enable the acceleration of their cross-platform software strategy for mobile devices and desktop applications. And later the same year Nokia fully integrated Trolltech and named the new Nokia entity **Qt Software.**

1.1.3 Licensing Model

Qt has been available under two licenses from day one: a commercial and an open source license - GPL. But starting from version 4.5 Qt Open Source is now available under LGPL.

- The GNU General Public License (“**GPL**”) version 3.0.
  GPL requires you to make application source code freely available.

- The GNU Lesser General Public License (“**LGPL**”) version 2.1.
  LGPL enables the development of closed source applications (under certain restrictions).

- The Qt **Commercial** Developer License.
  The LGPL carries some restrictions regarding the ability for users to relink libraries and other restrictions that may impose architectural requirements that some organizations might not be comfortable with. If this issue is important for your company, you must purchase a commercial license at the time you begin development.

1.1.4 Installations

You can develop QT applications on 3 different host platforms: Linux, Windows and Mac. Select the one appropriate for you. I don’t have a Mac so this option is not covered in this guide. If you run Linux on an X86 platform then you can simply download and unpack QT. If you run Linux on other platforms then you must download the sources and build QT.

If you will use Qt on a platform not presented here, or you need more detailed information then consult the official installation page here: http://doc.trolltech.com/4.6/installation.html

2 Installing Qt SDK on X86 Linux

Download the appropriate version (32 or 64 bit) for Linux from here:
http://qt.nokia.com/downloads (click Go LGPL)

You need to make the downloaded file executable in order to run it. You can either do this with your desktop's file manager or, at the command line, type (if you download a newer version your filename will be different):

```
$ chmod u+x qt-sdk-linux-x86-opensource-2010.04.bin
```

You should now be able to execute the file as normal. You can do this from the command line by typing:
(*Notice: the default setup, will install QT under your home directory, and this is normally a good choice for single user machines. But if different user accounts should be able to use QT, then you must run the installation program with root privileges (use sudo) and install QT under /usr/local/*)

```
$ ./qt-sdk-linux-x86-opensource-2010.04.bin
```

Apart from a C++ compiler, a number of development libraries need to be present to enable Qt Creator to build your Qt applications. On Debian and Ubuntu, use the following command in a terminal to make sure they are installed:

```
$ sudo apt-get install libglib2.0-dev libSM-dev libxrender-dev
libfontconfig1-dev libxext-dev
```

If you're using QtOpenGL, you'll also need:

```
$ sudo apt-get install libglui-dev
```

You should now be able to use the integrated development tool Creator, but if you also want to be able to use the command line tools, then you must add the path to Qt's bin directory to your path environment variable. To do so change directory to your home directory and open the file .bashrc in an editor:

```
$ cd ~
$ gedit .bashrc
```

Scroll to the bottom of the file and add the following line to the file:

```
PATH=$PATH:$HOME/apps/qtsdk-2010.04/qt/bin
```

You should now have working installation of Qt on your Linux box, and you are now able to continue with the chapter "**Using Qt on Linux** …".

3 Installing Qt Everywhere on a Linux host

If you want to develop embedded applications with Qt then you must install a special edition of the Qt framework, and it will also require some special configuration on the development platform (referred to as the host platform). Qt can be targeted different embedded platforms, but this guide only covers using a Linux host to develop embedded applications to run on an arm board with Linux as operating system (Devkit 8000).

If you want to use Qt on the Devkit 8000 then you will probably want to use the touch screen as well. To be able to do so you need to install a library with touch screen support called tslib before you configure Qt embedded.

3.1 Install and build `tslib`

If you want to use the touchscreen on Devkit8000 then you should download and install the `tslib` library before installation of Qt embedded.

We use autoconf to install `tslib` so unless you have installed `autoconf` already, you should start with the installation and build of `autoconf` on host.

```
sudo apt-get install autoconf
sudo apt-get install libtool
```

Download `tslib`:

```
$ cd ~/apps
$ mkdir tslib-arm
$ cd tslib-arm
$ apt-get source tslib
```

Configure `tslib`:

```
$ cd tslib-1.0
$ ./autogen.sh
$ ./configure --prefix=$HOME/apps/tslib_arm/ --host=arm-none-linux-gnueabi
```

In `config.h`, edit the line (approx. line 185)

```
#define malloc rpl_malloc --> /* #define malloc rpl_malloc */
```

Now cross-compile `tslib`:

```
$ make
$ sudo make install
```

`Tslib` is now ready to be used.

3.2 Install Qt Everywhere on a Linux Host

3.2.1 Download the Qt Everywhere 4.6.3 archive

Download Qt Everywhere 4.6.3 for Embedded Linux (or newer version) from here: http://qt.nokia.com/downloads (click Go LGPL)

Put the archive in your home directory and unpack the Qt Archive: Open a command prompt (terminal) and change directory to the directory with the downloaded archive if necessary. Uncompress the archive and then unpack it with the commands:

```
gunzip qt-everywhere-opensource-src-4.6.3.tar.gz
tar xf qt-everywhere-opensource-src-4.6.3.tar
```

This will give you a new directory with same name as the tar file.

3.2.2 Build the Qt Embedded 4.6.3 libraries for Embedded Linux

Change directory to the Qt Everywhere install dir

```
cd qt-everywhere-opensource-src-4.6.3
```

Add the Qt compiler path to your path: Open `~/.bashrc` and find the line that starts with "`PATH=`". At the end of the line, append ":`$HOME/apps/qt-everywhere-opensource-src-4.6.3`" (note the ':' at the beginning of the text!)

Edit the file `$HOME/apps/qt-everywhere-opensource-src-4.6.3/mkspecs/qws/linux-arm-g++/qmake.conf` as follows:

```
#
# qmake configuration for building with arm-linux-g++
#

include(../../common/g++.conf)
include(../../common/linux.conf)
include(../../common/qws.conf)

# modifications to g++.conf
QMAKE_CC        = arm-none-linux-gnueabi-gcc
QMAKE_CXX         = arm-none-linux-gnueabi-g++
QMAKE_LINK        = arm-none-linux-gnueabi-g++
QMAKE_LINK_SHLIB  = arm-none-linux-gnueabi-g++

# modifications to linux.conf
QMAKE_AR          = arm-none-linux-gnueabi-ar cqs
QMAKE_OBJCOPY     = arm-none-linux-gnueabi-objcopy
QMAKE_STRIP       = arm-none-linux-gnueabi-strip
QMAKE_INCDIR    += /home/stud/apps/tslib_arm/include
QMAKE_LIBDIR    += /home/stud/apps/tslib_arm/lib

load(qt_config)
```

Configure the Qt Everywhere 4.6.3 libraries for use on target (this takes some time)

```
$ cd
$ cd apps/qt-everywhere-opensource-src-4.6.3
$ ./configure -embedded arm -xplatform qws/linux-arm-g++ -qt-kbd-linuxinput -qt-mouse-
tslib -opensource -verbose -R /home/stud/apps/tslib_arm/lib/
```

Make the libraries:

```
# make
```

Switch to root account (note: If root account is not enabled, do so by typing "`sudo passwd root`")

```
$ su
Password: [enter 'root']
#
```

Install the libraries:

```
# make install
```

Switch to user account:

```
# su stud
$
```

3.3 Install Qt Everywhere 4.6.3 fonts and libraries on target

Installing Qt Everywhere on target means moving the built libraries and some fonts there.

Fonts: At least one font must be present on target to show text. Fonts can be found in `$HOME/apps/qt-everywhere-opensource-src-4.6.3/lib/fonts` on the host and must be copied to `/usr/local/Trolltech/QtEmbedded-4.6.3-arm/lib/fonts/` on target (create the directories as necessary). The font `DejaVuSans.ttf` is known to work on target, but the rest should too.

Libraries: (At least) the below libraries must be copied from `$HOME/apps/qt-everywhere-opensource-src-4.6.3/lib/` on host to `/usr/lib` on target

- `libQtCore.so.4`
- `libQtGui.so.4`
- `libQtNetwork.so.4`

3.4 Enable touch screen on DevKit8000

Allow Qt read/write access to the touch screen:

```
$ chmod a+rw /dev/input/event2
```

Add the touchscreen to input devices for Qt: On target, run the following command on target

```
$ echo export QWS_MOUSE_PROTO=\"Tslib:/dev/input/touchscreen0\" >> /etc/profile2
```

(note the '`\"`' escape characters and the double '>' used)

3.5 Building a target application

3.5.1 Creating a symbolic link to `tslib`

To have a succesful build, you must create a symbolic link on the host to the `Tslib` libraries in the lib path of `qmake`:

```
$ cd /usr/local/Trolltech/QtEmbedded-4.6.3-arm/lib/
$ ln -s /home/stud/apps/tslib_arm/lib/libts-0.0.so.0 libts-0.0.so.0
```

3.5.2 Specifying the correct `qmake`

Target applications are built by means of Qt's `qmake` tool. The Qt documentation states that you should run the following sequence:

```
myProjDir$ qmake -project
myProjDir$ qmake
myProjDir$ make
```

However, this uses the default `qmake` (for host) and produces an executable that is unsuitable for the target (ARM) architecture. The correct `qmake` is located in `~/apps/ls a        qt-everywhere-opensource-src-4.6.3/bin/`, so to create a target executable you should run the following sequence:

```
myProjDir$ ~/apps/qt-everywhere-opensource-src-4.6.3/bin/qmake -project
myProjDir$ ~/apps/qt-everywhere-opensource-src-4.6.3/bin/qmake
myProjDir$ make
```

(Note: The above sequence is generic and should be wrapped in a shell script)

The application can now be transferred to and executed on target. At least one Qt application must run as Qt Server and should be started using the command

```
# ./MyAppName –qws
```

Failure to start a Qt Server (using `–qws`) will result in a `Connection Refused`-error.

3.5.3 Calibrating the touchscreen

If your touchscreen misbehaves on target, try exiting your application, run `ts_calibrate` on target, go through the calibration routine and then re-run your application.

3.6 Installing Qt on Windows

Download the Qt SDK for Windows from here:

http://qt.nokia.com/downloads (click Go LGPL)

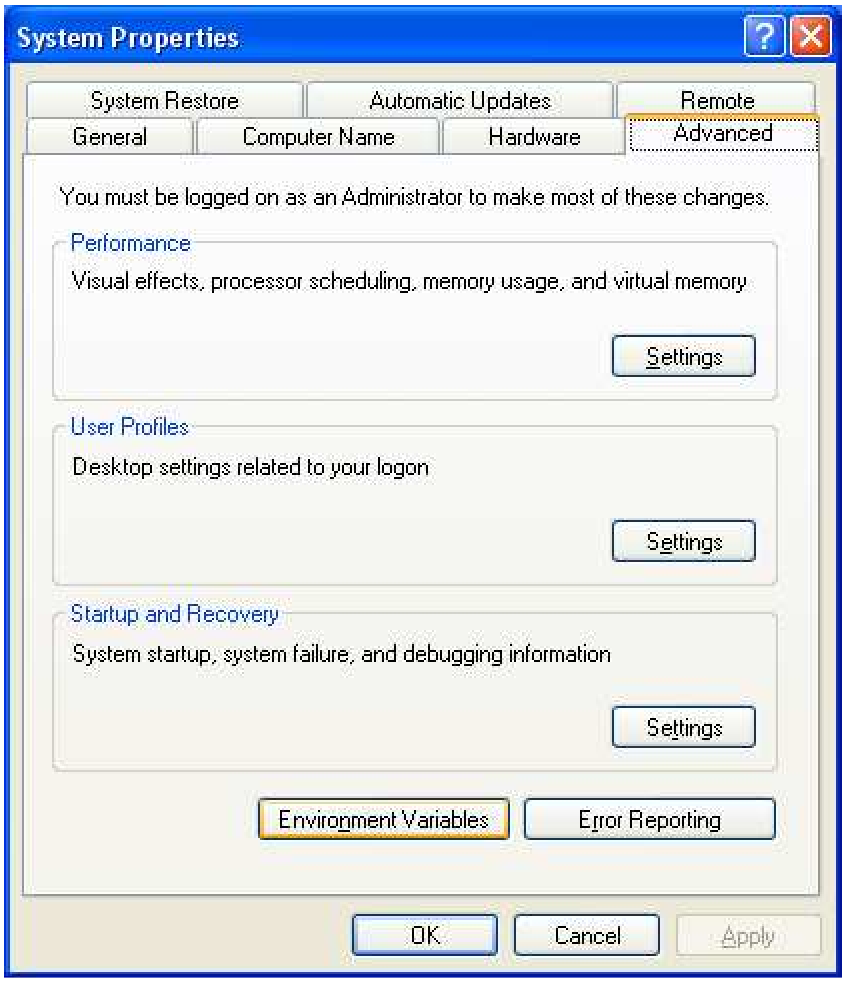
Click on **Download Qt SDK for Windows** to download the installation program. When the download completes, simply double click the intallation program and Qt for Windows will be installed on your machine.

Note:

1. The install path must not contain any spaces or Windows specific file system characters. The default path is: `C:\Qt\4.6.0`

2. The open source version of Qt on Windows cannot use Microsoft's C++ compiler! So (unless you use the commercial version) you must choose to install the MinGW C++ compiler when the installation program prompts you for that option (unless it's installed already).

When the installer finish you are ready to develop GUI applications with the command line tools. You will find them in the start-menu under Qt by Nokia v…
But if you will use the integrated development environment (IDE) **Qt Creator** (which you probable will), then you must first add the path to the Qt installation directory to your Path environment variable. To do so choose Control Panel in the Start-Menu and select System (*se the following figures)*


System Properties

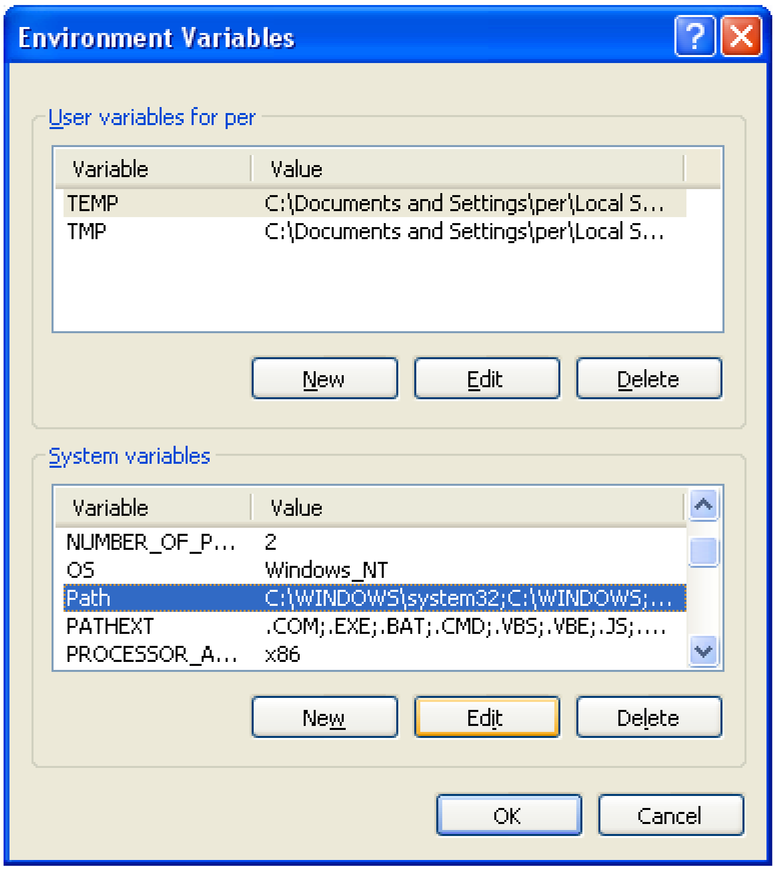
You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles
Desktop settings related to your logon

Startup and Recovery
System startup, system failure, and debugging information

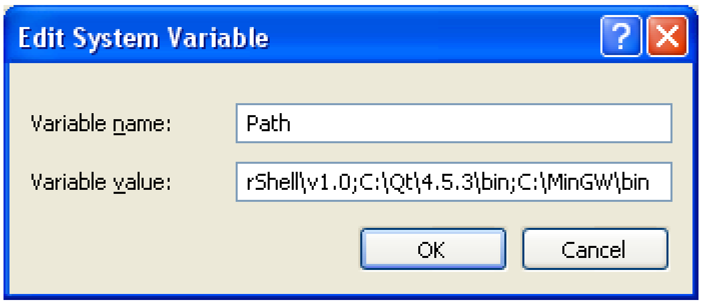
Click the Environment Variables buttom



Variable	Value
TEMP	C:\Documents and Settings\per\Local S...
TMP	C:\Documents and Settings\per\Local S...

Variable	Value
NUMBER_OF_P...	2
OS	Windows_NT
Path	C:\WINDOWS\system32;C:\WINDOWS;...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...	x86

Select Path and click Edit.



Variable name: Path

Variable value: rShell\v1.0;C:\Qt\4.5.3\bin;C:\MinGW\bin

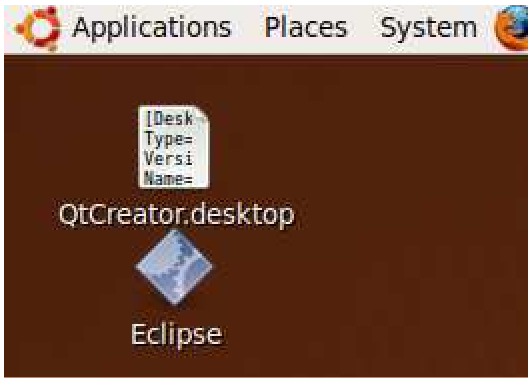
Add the path to Qt's executables on your machine and the path to the MinGW C++ compiler and click OK.

4 Using Qt

When you develop applications with Qt you can choose between several different programming environments: the Creator, command line tools, Eclipse and MS Visual Studio (only on Windows). In this guide only the first 2 options is explained. Personally I prefer to use the Creator, but if you are a skilled Emacs user you may prefer the command line tools.

4.1 Using Qt on Linux with the Creator

Start the Qt Creator by double clicking the icon on the desktop or select it in the Applications – Programming menu.



TEXT MISSING

4.2 Using Qt on Linux with command line tools

Open a terminal window and then create a subdirectory under home (or somewhere else where you have write access) named hello. Change directory to hello:

```
mkdir hello
cd hello
```

Create a C++ source file named hello.cpp with your favorite code editor – or you might just use gedit:

```
gedit hello.cpp
```

Type in the program shown below:

```
#include <QApplication>
#include <QLabel>

int main(int argc, char*argv[])
{
  QApplication app(argc, argv);
  QLabel *label = new QLabel("Hello Qt!");
  label->show();
  return app.exec();
}
```

Save the file and close the file editor. Create a Qt project file with the command (this command should be given from the directory with the source file(s)):

```
qmake –project
```

This will generate a Qt project file with the name hello.pro.

If you haven’t configured the PATH environment variable correctly, then you will get this error message:

```
The program 'qmake' can be found in the following packages:
 * qt3-dev-tools
 * qt4-qmake
Try: sudo apt-get install <selected package>
bash: qmake: command not found
per@per-laptop:~/QTdemos/hello$
```

If you have installed Qt and get this error message, then check your PATH environment variable – it must include a path to the directory containing qmake:

```
per@per-laptop:~/QTdemos/hello$ $PATH
bash:
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/us
r/games:/home/per/CodeSourcery/Sourcery_G++_Lite/bin/:/home/per/
qtsdk-2009.05/qt/bin: No such file or directory
per@per-laptop:~/QTdemos/hello$
```

The next step is to generate a makefile. This is done with the command:

```
qmake hello.pro
```

This builds a makefile which you run with make:

```
make
```

This will build the executable file hello which you run with the command:

```
./hello
```

This opens a new window that simply shows the text Hello Qt!

You have built your first GUI application with Qt’s command line tools, and you are ready to learn more about the Qt framework.

4.3 Integrating Qt development in Eclipse

If you wish to develop Qt applications in the familiar Eclipse environment, there is a Qt plugin for Eclipse available from http://qt.nokia.com/developer/eclipse-integration. Follow the instructions given there for installation and use.

4.4 Using Qt on Windows with the Creator

TEXT MISSING

4.5 Using Qt on Windows with command line tools

TEXT MISSING

5 Qt Basics

TEXT MISSING

6 2D Graphics

TEXT MISSING

7 Links

Installing:
http://doc.trolltech.com/4.6/installation.html

Installing Qt on Embedded Linux
http://doc.qt.nokia.com/4.6/qt-embedded-install.html

Getting started:
http://qt.nokia.com/developer/getting-started/getting-started

The official Qt book (free online):
http://www.informit.com/store/product.aspx?isbn=0132354160
Click on the tab Sample content to read the book.

Qt Reference Documentation
http://doc.qt.nokia.com/4.6/

The home page for Qt Embedded
http://doc.trolltech.com/4.6/qt-embedded-linux.html

Some very nice self-study material on Qt
http://qt.nokia.com/services-partners/qt-in-education/qt-in-education-course-material