Qt/Embedded Features — My Perspective

Jharana Mehta
jharana@mahindrabt.com

Jan 02, 2002

Abstract

In this article we shall see how to work with **Qt/Embedded**, an application framework for embedded systems, and examine and review its features one by one. I came across Qt/Embedded while working on the porting of Konqueror, an open source web browser, while working for an embedded [GNU/Linux] project for my employers, and here's what I found.

1 What is Qt/Embedded?

Qt/Embedded is a cross-platform C++ GUI application framework, which provides application developers with the functionality needed to build GUI based applications for embedded systems. It is the embedded version of Qt/X11, which is a part of any standard GNU/Linux distribution these days, and using which the **K Desktop Environment** has been developed.

It is a product of Trolltech, is completely open source, and is available in both commercial and free editions. In this article we will be talking about the Qt/Embedded Free Edition which comes with the GNU GPL license.

2 Getting Started

First step is to download the gzipped tar file of the latest version from the *Trolltech* website. The latest release is 3.0.1, but the more stable version for embedded from my point of view is 2.3.1. the 3.x.x are not optimised for memory consumption yet, and currently not suitable for building embedded applications.

Next step, untar the `tar.gz` file. It will occupy around 150MB, its a huge package as it comes with all source code included, but its worth keeping that much disk space free for it, believe me.

The `README` and `INSTALL` file contain instructions on how to configure and compile Qt/Embedded for your target CPU. Variables `QTDIR` and `LD_LIBRARY_PATH` need to be defined properly, apart from that configuration and compilation couldn't be easier. The compilation process is long though and can take up more than half an hour! Successful compilation results in the `libqte.so.x.x.x` file in the `QTDIR/lib` directory, which are shared object libraries, and all that an application needs to link with in order to run on an embedded device!

To run the demo,

```
>cd QTDIR/examples/launcher/
>./start_demo
```

If you run into problems during installation, refer to the installation and troubleshooting page of the Qt online documentation — *http://doc.trolltech.com/2.3/install-qws.html*

Note that you require a working Linux Framebuffer to use Qt/Embedded. In order to enable Framebuffer support in your kernel, take a look at *http://www.linuxdoc.org/HOWTO/Framebuffer-HOWTO.html*

To get started on programming with Qt, go through the tutorial page. This tutorial starts with a simple 10 line program and ends with a 650 line game in 14 steps, introducing new features and concepts at each step. The signal slot inter object communication mechanism, might seem a bit complicated at first, but a walk through of this example will give you the hang of it!

In order to compile Qt/Embedded programs, it is better to use the ready made free utilities like `qmake` or `tmake`. You simply have to enter the names of your header and `.cpp` files in a template `.pro` file, and then this utility will generate the Makefile for you. The qmake binary is presnt in the `QTDIR/bin` directory. Usage:

```
>qmake -o Makefile hello.pro
>make
>./hello
```

3 Qtopia

Qtopia, formerly known as QPE (Qt Palmtop Environment) is a window environment and application suite designed for PDAs, palmtop computers, Internet appliances, and similar devices. It is fully based on Qt/Embedded.

Qtopia includes a full set of **Personal Information Management** (PIM) applications, like calendar, address book, to do-list, etc. Also available are E-mail client, games, configuration utilities, and more.

You might want to check this out too; it has a number of interesting applications like email client, mpegplayer etc meant for PDAs, set top boxes and other embedded devices. It can be downloaded from the same web site, and it compiles easily too, though check here for compilation instructions or if patches need to be applied for the specific release.

4 Qt/Embedded Features — My Perspective

4.1 Object Oriented Design

Qt/Embedded is definitely object oriented, so creating a widget class of your own and componentising it is really easy. Also there is this signal-slot mechanism which I have mentioned before, for communicating between objects. These signal-slot mechanisms are cleaner than the callback method, which uses pointers to functions, and thus this eliminates callback-related core dumps. The signals are similar to the events `(keyPressed, valueChanged)` associated with an object, and slots are similar to the actions

`(setValue, changeStatus)` to be performed on an object. These signals and slots can be user defined and associated between any two or more than two objects.

4.2 Internationalization

Qt provides integrated support for making localized applications, where all user interface texts are translated into the local language, based on message translation tables. Qt also has full support for Unicode 16-bit international character set.

4.3 Scalability

Qt scales from a smallest footprint of 800K to 3MB. There are different configuration files for minimal, small, medium, large, everything and custom configuration. These files define macros to disable features. A description of the macros and how to disable unneeded features is given here. If your embedded application doesn't require sound support, you can leave it out. If you don't want support for languages other than English, just incorporate the line `# define QT_NO_TRANSLATION` in the `qconfig.h` file.

Compiling it with all features included creates a stripped shared object library of around 3.7MB. Also options can be given to support different pixel depths, different image formats like PNG, JPEG etc, which are more memory efficient than other image formats like `bmp`, `gif`, etc.

Qt graphics directly access the framebuffer, so the huge X11 libs are done away with altogether, the reason for the dramatically small size of the library.

It has a 15MB font directory, thus supporting many different types of fonts, but in order to reduce memory footprint, you can choose to retain only one `*.qpf` font file (< 100k), and it will still work.

4.4 Portability

Qt/Embedded supports all platforms supported by GNU/Linux. This includes CPUs based on the ARM, i386, Motorola-68000, MIPS, PowerPC chips. I have ported the Qt/Embedded library and Qt Palmtop Environment (Qtopia) to Strong ARM assabet board, and the look and feel was just the same as on an Intel machine. Cross compilation is a matter of giving the right configuration options, e.g. to build on GNU/Linux x86 for the GNU/Linux Strong ARM target, you would use:

```
./configure -platform linux-x86-g++ -xplatform linux-arm-\
g++
```

Simply defining a variable `QWS_SIZE` the display size of the application window can be scaled from 320×240 upto 1024×768. So the same application can be seen in a small palmtop size LCD display, a VGA monitor and a TV with no change in code!

4.5 Performance

Since the Qt libs directly access the framebuffer, the rendering is faster than X11. A green screen comes up first, and then the application.

But, a few warnings. If starting a Qt application for the first time, don't panic if your graphics get destroyed as soon as you move the mouse. Stop your console mouse daemon

"/etc/rc.d/init.d/gpm stop" and start again.

Also, sometimes, while running a Qt application, I got a segmentation fault, which caused the keyboard to behave weirdly, and I had no option but to reboot the system. I think other people too have come across this error. Also I suspect there are some memory leaks somewhere too, as running the same application on an embedded board repeatedly caused it to fail after 3-4 times.

If, while running a qt application you run a memory check program like top, you might see that the qt application is occupying 10MB or more. But that is due to the fact that the memory of the graphics card is included in the memory usage reported for a qt Embedded application.

4.6 Tools

The Qt/Embedded package comes with a number of useful tools like *Qt designer*, *Qt linguist* etc.

Qt designer is a visual aid for building qt widgets and applications and Qt linguist is a translator tool for translating GUI text into a language of your choice.

Qt Assistant is a tool for providing on line help for your application.

The *QEmbed tool* is provided for including image files and other resources directly into your application rather than loading the data from external files, thus saving precious memory resources.

The *Qvfb* (Qt virtual framebuffer) allows Qt/Embedded programs to be developed on your desktop machine, without switching between consoles and X11, and thus facilitates rapid application development, as you need to test your application on the embedded device only once its complete.

4.7 Online Documentation

Qt does have a very well structured on-line Documentation that outlines in detail all the different Qt classes and their methods. The documentation also contains a trouble shooting FAQ, Installation Instructions, tutorials and help on the various tools provided with the Qt/Embedded package.

There is an examples directory in `QTDIR` which contains a number of sample applications. Whatever kind of application you want to develop, be sure to look into this directory, you will definitely find something that will take you halfway there or at least serve as a starting point if not more. These examples are also explained in the documentation, so understanding them becomes easier. There are a number of online tutorials and code walk throughs for familiarizing oneself with Qt. A section on compiling Qt based applications could be useful though, if added.

4.8 Support

There is a very active mailing list for developers of qt applications. You can subscribe to it by send the word "subscribe" (without the quotes) to qt-interest-request@trolltech.com to be added to qt-interest list. There is a separate mailing list for Qt/Embedded, textttqt-embedded-interest-request@trolltech.com, but it is not so active, you are better off writing a query about Qt/Embedded to the general Qt mailing list. Mostly the archive will contain a similar query to yours, and if you don't find an answer, drop a mail to the list and you will get a reply within 24

hours. Don't worry about your question sounding silly or ignorant, people on the list are pretty helpful.

5 A Final Word

I would say that if you are looking to develop attractive GUI based applications for your embedded device, want multi language support, rapid application development and deployment, leverage the advantage of [free] open source community and GPL, then look no further, Qt/Embedded is definitely the right toolkit for you!

About the Author Jharana Mehta is working as a software designer with Mahindra British Telecom. Her focus areas are GNU/Linux based Application Development for Pervasive Computing and is currently working for the Embedded Systems COE. She has an active interest in the Open Source software and the Free Software Movement. She is a regular contributor to various research publications and an active member of IEEE. She can be reached at `jharna_4@yahoo.com`

FreeBSD 4.5 Released

Murray Stokely <`murray@FreeBSD.org`> announced on Jan 29, 2002 that FreeBSD 4.5-RELEASE came out. Since FreeBSD 4.4 was released in September 2001, hundreds of fixes were made, the FreeBSD team also updated many system components, made several substantial performance improvements, and addressed a wide variety of security issues. FreeBSD can be freely downloadable from *ftp://ftp.FreeBSD.org/pub/FreeBSD/*.

February 21, 2002, Mandrake [GNU/Linux] 8.2 β3 got available. Please test it again and report bugs for Mandrake. The URL for free download is *http://www.mandrake.com*.

Upcoming in the Next Issue:

- Journaling File System on GNU/Linux.

  JFS is an advanced file system, in which combine the database concept, the most two important features of JFS are: guaranteed consistency of metadata barring hardware failures and quick recovery in the case of failure. Steve Best, the researcher at IBM Linux Technology Center, will bring you an introduction to JFS for GNU/Linux.

  So far, JFS is not yet a default part of the Linux kernel, so you have to build it by yourself, and Steve will tell you the conception of JFS, and how to make it available on your computer step by step.

- Survey of FOSDEM

  Free and Open Source DEvelopment Meeting was successfully held during Feb 16-17, 2002 in Brussels, Belgium. Several thousands free software hackers, programmers, supporters attended the meeting. Raphäl Bauduin, one of the organizers for FOSDEM will bring you an survey about this interesting event of our community.

- Miguel on Mono & GNOME

  Miguel de Icaza will talk about the relation in depth between the Mono and GNOME — Don't miss it!