第 11 章　界面程序设计——Qt

一个成功的应用程序，必须要有友好美观的界面支持，这一点也正是 Windows 应用程序大行其道的一个重要原因。在 Linux 系统下，传统的应用程序展现给用户的是字符界面，操作界面不够美观，对各种外设的支持也不够全面。为提供更为美观、可操作性更强的程序界面，Linux 提供了多种功能强大的界面程序设计工具包，Qt 就是其中最为著名的一个。

本章从 Qt 开发包的安装讲起，介绍了 Qt 开发的基本过程。通过本章的学习，读者应重点掌握如下内容。

- 熟悉 Qt 集成开发环境的使用。
- 掌握用 Qt 开发桌面应用程序的基本过程。

11.1　Qt 简述

Qt 是一个支持多平台（Windows、Linux 等）的面向对象（使用 C++语言）的图形用户界面应用程序框架，它为应用程序开发人员提供了建立图形界面程序所需要的全部功能。Qt 的开发过程是完全面向对象的。本书假设读者已对面向对象的 C++语言已有初步了解。如果读者对面向对象的开发过程没有概念，可以在阅读本章过程中参照一些面向对象方面的书籍。

11.1.1　Qt 的组成

Qt 有多个发布版本，主要包括企业版（专业版）、自由版。其中自由版是为开发开放源码软件而免费提供的。Qt 的企业版主要包括以下模块。

- 基本模块：包括核心、窗口组件、对话框、图形用户界面工具包及应用类库等。
- 集成开发环境：即 IDE。
- 工作区模块：提供了对多文档界面（MDI）的支持。
- OpenGL 三维图形模块：提供对 OpenGL 开发的支持。
- 网络模块：提供对套接口、TCP/UDP、DNS 等网络操作的支持。
- 表格模块：提供对电子表格进行编辑的功能。
- XML 模块：提供对 XML 进行解析的功能。
- 数据库模块：提供对数据库访问的支持。

11.1.2　Qt 的优点

Qt 被广大程序员认可的一个重要原因，就是其支持多种操作系统平台。如果读者在

Windows 平台下从事开发工作，可能会使用 Microsoft Foundation Classes（MFC）或者 C#.Net 等面向对象的编程工具。但是，使用 MFC 或者 C#开发完成的产品只能在 Windows 系统下运行，要将这些程序移植至 Linux 系统下代价是非常昂贵的。要编写在 Windows 系统下或者 Linux 系统下可移植性更强的工具就是 Qt。目前 Qt 所支持的主要操作系统包括以下几种。

- Windows 系列，如 Windows95/98、WindowsXP、Windows2003 等。
- UNIX 系列，如 HP-UX、IBM AIX、Sun Solaris 等。
- Linux 系列，如 RedHat Linux、SuseLinux、DebianLinux 等。

除 Qt 良好的可移植性之外，Qt 的另外一个优点就是生成的程序运行速度快。使用过 GUI 从事界面开发的程序员应当了解，GUI 程序运行速度通常不尽如人意。Qt 在运行速度方面具有很好的表现。这一方面是由于 Qt 的设计人员对 Qt 进行了长时间优化的结果。另外，Qt 运行速度快也是基于其实现方式。Qt 是一个 GUI 仿真开发工具包，不依赖于任何其他工具包。它通过调用不同硬件平台上的低级绘图函数实现图像处理，这使得 Qt 生成的可执行程序具有更高的运行速度。

另外，Qt 也是完全面向对象的，它具有一切面向对象编程环境的优点。同时，Qt 也是易用的，它采用完全可视化的开发环境，支持“所见即所得”的软件开发方式。

11.2 Qt 开发包的安装

要使用 Qt 开发程序，首先应该在系统中安装该开发工具包。在缺省情况下，大多系统不会安装 Qt 开发包，需要手工进行安装工作。通常情况下，Qt 安装包都包含在 Linux 的安装介质中，可以通过 yast 进行安装，输入代码如下所示。

```
#yast
```

（1）程序中出现 yast 的安装界面，如图 11-1 所示。

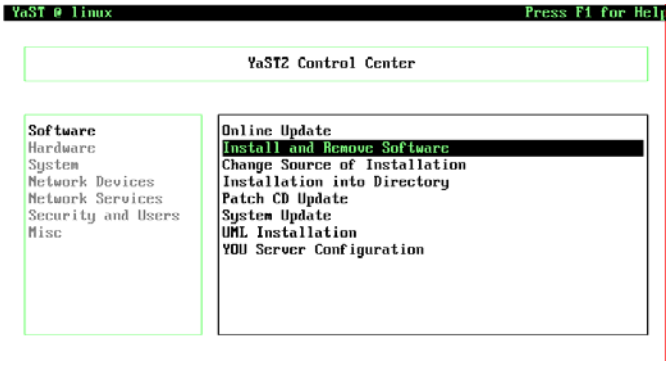

图 11-1 yast 安装界面

（2）在上述界面中，用<Tab>键选择“Install and Remove Software”选项，单击<Enter>键后出现的界面如图 11-2 所示。

（3）在界面中选择“search”选项，然后指定搜索的名称为“Qt”，可以看到结果，如图 11-3 所示。

图 11-2　安装/卸载软件界面

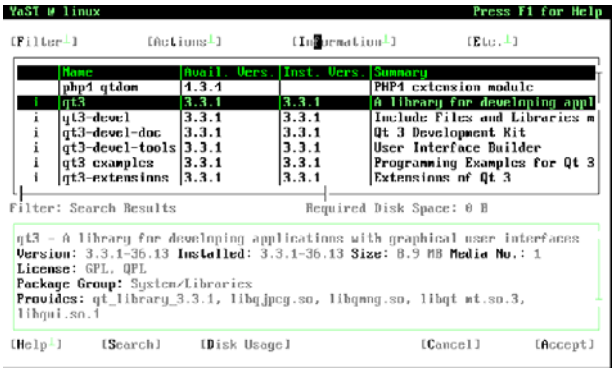

图 11-3　搜索 Qt 安装包界面

Qt 安装包包含若干模块，如开发库、开发文档、开发工具等。为简便起见，建议用户选择所有 Qt 开发的安装包。选中的方式是用方向键将光标移动到相应行，然后按<Space>键选择。选中后，相应行前面应该出现一个加号。如果在未进行选择的情况下，相应行前面已经出现“i”符号，表明该包已安装过，无需再安装。选中安装包后，按<Tab>键移动光标至“Accept”，然后按<Enter>键。系统将提示插入光盘，后续操作按照提示一步步完成即可。

提示：本书是以 SuseLinux 9 为例进行描述，其他版本的 Linux 系统下 Qt 的安装过程大同小异，可以参照相应操作系统的说明文档。

11.3　Qt 集成开发环境介绍

Qt 开发工具包提供了功能强大的集成开发环境：Qt 设计器。如果读者在 Windows 系统下用 VisualStudio.Net 或者其他面向对象的开发工具，对于该环境一定不会陌生。

11.3.1　启动设计器

在命令提示符下运行 designer 即可启动设计器，其主界面如图 11-4 所示。

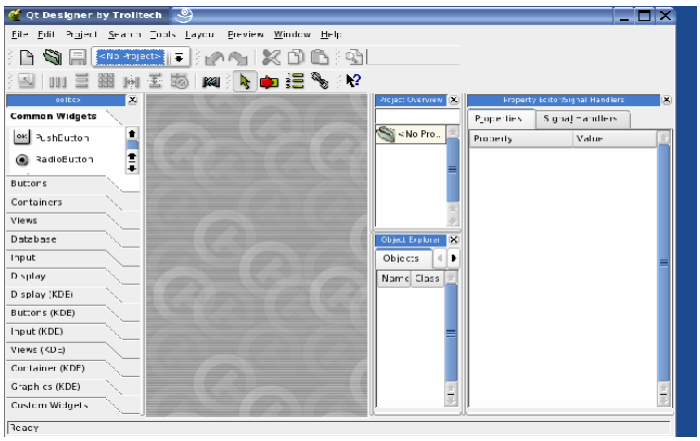

图 11-4　Qt 设计器主界面

11.3.2　设计器界面元素介绍

整个设计器可以划分为菜单栏、工具栏、标题栏、组件面板、窗体设计窗口、对象属性/事件编辑器、对象浏览器等部分，如图 11-5 所示，主要部分的功能如下。

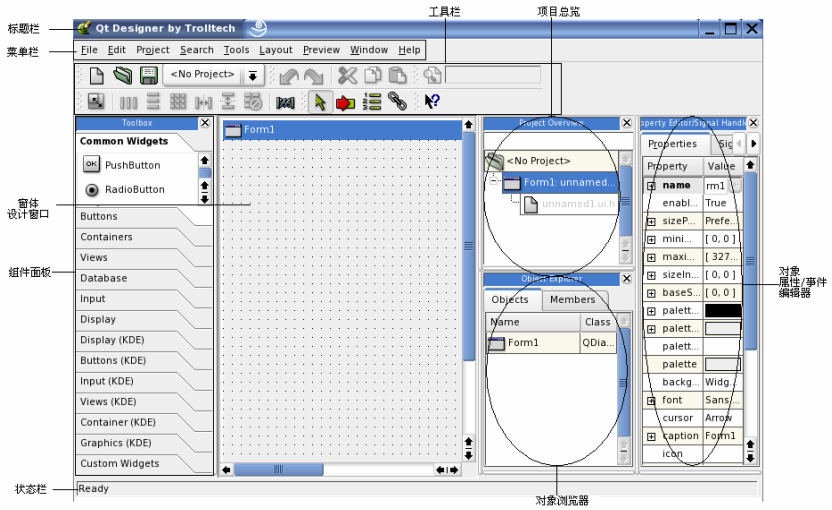

图 11-5　设计器界面元素

- 菜单栏：通过菜单栏可以执行设计器中大部分的命令，如创建新项目、打开已有项目等。菜单栏中包含的菜单组包括文件、编辑、项目、搜索、工具、设计、预览、窗口及帮助，各个菜单组下都包含了相应的菜单项。
- 工具栏：工具栏提供了快速执行部分功能的方法。通过单击工具栏中的按钮，可以达到与执行菜单项相同的效果。当光标悬停在相应工具栏时，将弹出工具栏的提示信息。
- 组件面板：组件面板提供了开发应用程序常用的组件，如按钮、文本框等。这些组

件可以拖拉至窗体设计窗口中，达到“所见即所得”的效果。

- 窗体设计窗口：窗体设计窗口是可视化设计的核心部分。可以通过拖拉的方式，将组件面板中的相应组件拖拉到窗体设计器中。当建立新的项目时，窗体中将产生默认的窗口，其标题为“Form1”，可以通过属性编辑器修改其标题。
- 对象属性或事件编辑器：该部分定义了目标对象（窗口及各种组件等）的属性及组件的事件处理。

11.4　Qt 程序开发

Qt 程序的开发是一个非常复杂的过程，其采用的是面向对象的开发方式。Qt 设计器工具支持“所见即所得”的开发方式，支持各种窗口组件。其功能之强大，可以与 Windows 系统下的 Visual Studio 相媲美。受篇幅所限，本书并不对其内容进行详解，而是以一个简单的 helloworld 示例进行介绍。读者如果有兴趣，可以参考 Qt 专门开发手册了解更详细的内容。

11.4.1　建立新项目

在 Qt 设计器中选择菜单栏中的【file】→【new】命令，然后选择 C++项目文件，在弹出的新建项目窗口中输入项目名称“helloworld”，单击“OK”按钮建立新项目，如图 11-6 所示。

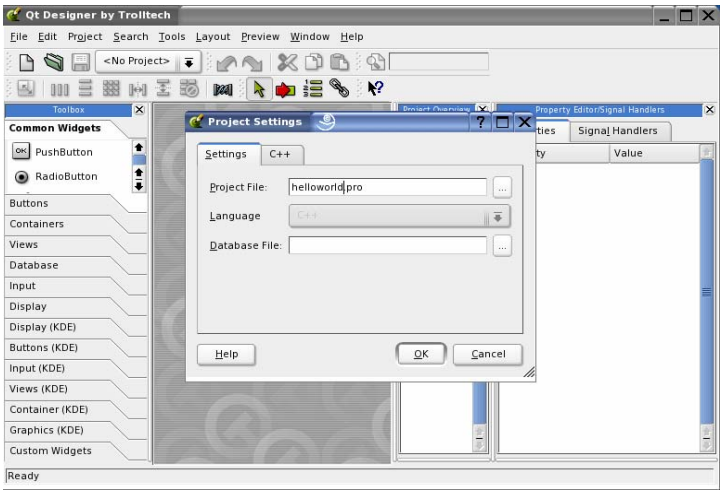

图 11-6　新建项目窗口

11.4.2　设计窗口

在 Qt 程序设计中，用户所需要做的第一件事就是创建一个主窗口，这个主窗口可以是

从 QWidget 或者 QDialog 派生的。在一个 Qt 程序中，可以创建多个窗口，但是只能有一个主窗口。在 Qt 设计器中选择菜单栏中的【file】→【new】命令，然后选择“Dialog”，创建一个新的对话框窗口。新建窗口的标题为 Form1，在对象属性或事件窗口中将其修改为 HelloWorld，如图 11-7 所示。

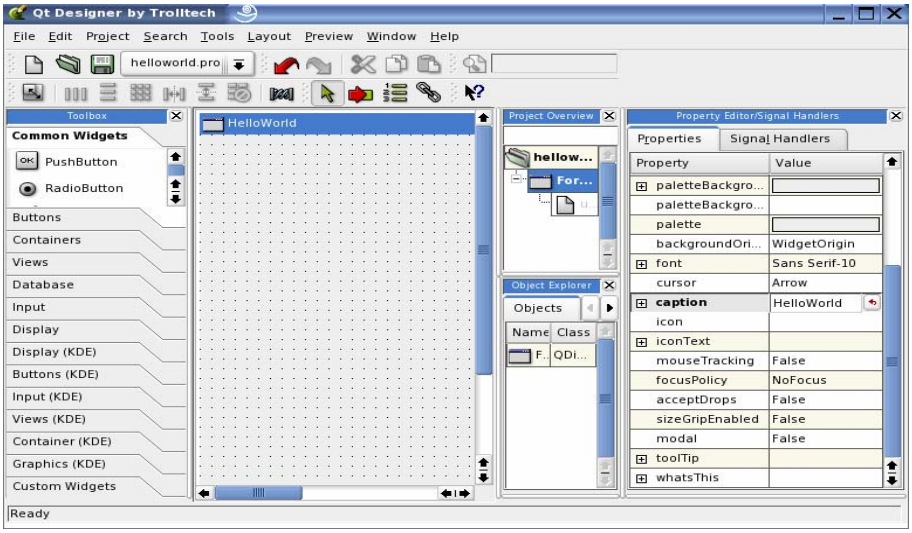

图 11-7　新建窗口

窗口建立完成后，可以从组件面板中拖动组件至窗口，并设置组件的相应属性。本例中，需要一个“确定”按钮。首先在组件面板中选中 PushButton，然后移动光标至新建的 dialog 窗口，在窗口中单击，便在窗口上建立了一个新的按钮。在对象的属性窗口中，修改其 text 属性为“确定”，如图 11-8 所示。

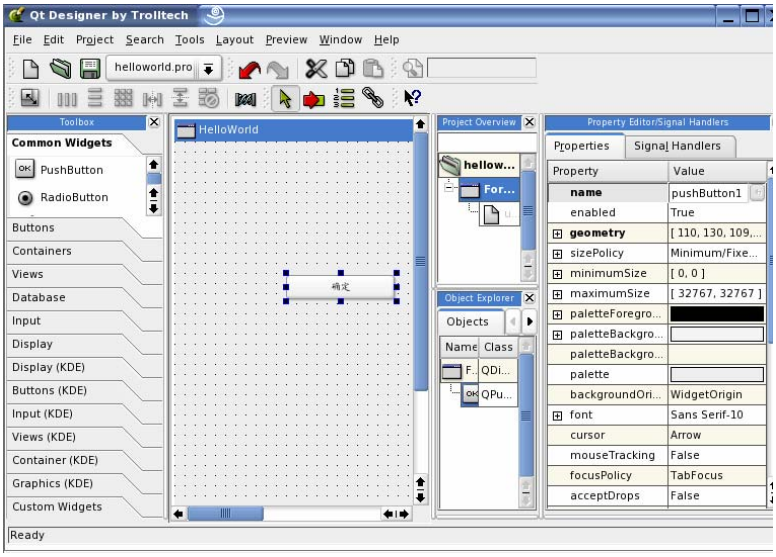

图 11-8　增加组件

经过上述操作后，系统将自动生成一个类 Form1，其实现代码位于当前工作目录的.ui 子

目录下，自动生成的代码文件分别为 form1.cpp 和 form1.h。其中 form1.cpp 的内容如下所示。

```
1    #include "form1.h"
2    #include <qvariant.h>
3    #include <qpushbutton.h>
4    #include <qlayout.h>
5    #include <Qtooltip.h>
6    #include <qwhatsthis.h>
7    #include <qimage.h>
8    #include <qpixmap.h>
9    #include "../form1.ui.h"
10   Form1::Form1( QWidget* parent, const char* name, bool modal, WFlags fl )
11         : QDialog( parent, name, modal, fl )
12   {
13         if ( !name )
14               setName( "Form1" );
15
16         pushButton1 = new QPushButton( this, "pushButton1" );
17         pushButton1->setGeometry( QRect( 90, 100, 109, 28 ) );
18         languageChange();
19         resize( QSize(287, 235).expandedTo(minimumSizeHint()) );
20         clearWState( WState_Polished );
21         // signals and slots connections
22         connect( pushButton1, SIGNAL( clicked() ), this, SLOT( pushButton1_clicked()
23    ) );
24   }
25   Form1::~Form1()
26   {
27         // no need to delete child widgets, Qt does it all for us
28   }
```

从该程序中可以看到，通过 Qt 设计器设计完成的主窗口，实际上是自动产生了一个派生自 QDialog 的类 Form1。程序运行时，将根据 Form1 类的定义产生一个主窗口和一个按钮，并将它们展现到屏幕上。

提示：在面向对象的程序设计过程中，一个类包含两个特殊的方法，称为构造函数、析构函数。构造函数的作用是在生成对象时，调用构造函数完成一些初始化的操作。析构函数的作用是在释放对象时，完成一些清理性的操作。在上述例子中，第 10 行即是 Form1 类的构造函数，第 25 行即是类 Form1 的析构函数。

11.4.3　添加事件处理程序

窗口及组件添加完成后，还需要对组件和窗口的事件进行处理。本例中，需要在单击“确定”按钮时弹出提示信息“HelloWorld”。因此，首先选中刚刚添加的按钮，然后在对象的事件处理窗口中选择“signal handlers”选项面板，在其中的 click 事件条上双击，将自动生成 click 的事件处理函数。不过，该函数是空的，需要用户根据需要在其中进行处理。本例中，只是需要弹出一个提示信息窗口。因此，在系统自动生成的事件处理程序中加入代码，如下所示。

```
#include <qmessagebox.h> /*在事件处理程序所在文件的头部*/
Void Form1::pushButton1_clicked()
{
```

```
    QMessageBox::information(NULL, "Information", "HelloWorld");
}
```

在 Qt 的事件处理机制中，使用了名为“槽”和“信号”的一种元素。所谓“槽”，可以通俗地理解为类中的一个函数。与普通函数不同的是，这个函数可以定义为在接收到某个“信号”时，触发该函数的执行。“信号”可以这样理解，当用户单击某个控件（按钮）时，将会产生一个信号。如果该信号被连接到某个槽，则槽所对应的函数将会执行。槽与信号的关系如图 11-9 所示。

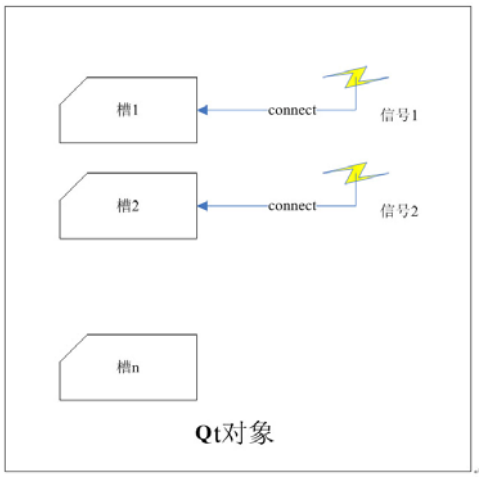

图 11-9　槽与信号的关系

11.4.4　添加主程序

要最终生成可执行的程序，还需要添加一个 main 主程序。在 Qt 设计器中选择菜单栏中的【file】→【new】命令，然后选择“C++Main File”选项，如图 11-10 所示。

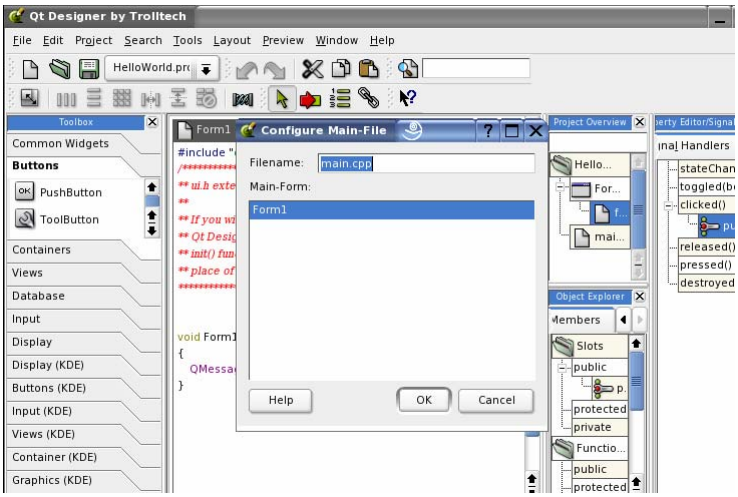

图 11-10　增加主程序

上图所示的窗口中选择当前应用的主窗口。本例中只有一个窗口，所以选择 Form1 为主窗口。单击“OK”按钮，系统将自动生成 main 函数。代码如下所示。

```
1    #include <qapplication.h>                                          /*系统头文件*/
2    #include "form1.h"                                                 /*自定义头文件*/
3    int main( int argc, char ** argv )                                 /*主函数*/
4    {
5        QApplication a( argc, argv );                                  /*初始化应用*/
6        Form1 w;                                                       /*定义 Form 类型变量*/
7        w.show();                                                      /*显示窗口*/
8        a.connect( &a, SIGNAL( lastWindowClosed() ), &a, SLOT( quit() ) );   /*安装事件处理*/
9        return a.exec();
10   }
```

在上述程序中，QApplication 是 Qt 程序所必需的全局对象。在 Qt 中，每个程序只能包含一个 QApplication 对象。

- 第 7 行调用 Form 对象的 show 方法，将 Form 对象显示在屏幕上。
- 第 8 行的 connect 调用是将主窗口关闭这一信号与函数 quit 进行了关联。在该函数调用成功后，一旦主窗口关闭这一事件发生，系统将自动调用函数 quit。这一机制将在后续章节中继续加以介绍。
- 第 9 行通过调用 QApplication 对象的 exec 方法，将程序的控制权由用户交给 Qt。Qt 将负责接收和处理用户产生的或者系统产生的事件，并把这些事件传递到相应的窗口。当应用程序关闭时，exec 函数将返回。exec 函数的功能类似于 Visual C++中的 run 函数。

在本例中，通过调用 QApplication 对象的 connect 方法，将主窗口关闭这一信号与槽所对应的函数 quit 进行关联。一旦 connect 调用成功，在主窗口关闭时，将会产生一个信号，该信号将触发函数 quit 的执行。

11.5　Qt 程序的生成

通过前面章节中对 gcc 编译器、makefile 等内容的介绍，读者应该知道要编译一个应用程序，需要编写 makefile 文件，然后通过执行 make 命令进行编译。Qt 程序的生成也有一个类似的过程。Qt 提供了专门的工具用于生成一个项目的 makefile 文件，这个工具就是 qmake。使用该工具的方法如下所示。

```
qmake  项目名称.pro
```

项目名称.pro 是 Qt 设计器生成的项目文件。执行完成上述命令后，即在当前目录下生成一个 makefile 文件。然后调用 make 工具根据该 makefile 进行编译链接，生成最终的可执行的应用程序。在命令提示符下运行该程序的结果如图 11-11 所示。

图 11-11　运行结果

11.6 小结

Qt 是 Linux 系统下非常强大的界面开发工具，使用它可以开发出非常专业的界面应用。本章仅向读者介绍了使用 Qt 的基础知识。其中，首先详细介绍 Qt 开发环境的使用方法，然后重点介绍了使用 Qt 进行程序开发的典型步骤，通过这部分的内容，用户可以了解 Qt 程序开发的一般过程。同时，读者可以根据实际应用开发的需要，开发出功能更为强大的桌面应用。