

---

## **C++ Qt5 范例开发大全**

作者：Longki

# 目 录

## 第1章 开发环境

- 1.1 Qt 简介 ..... 5
- 1.2 下载安装 Qt Creator ..... 6
- 1.3 第一个程序 Hello World ..... 7

## 第2章 窗体应用

- 1.1 窗体基类说明 ..... 12
- 1.2 控制窗体大小 ..... 13
- 1.3 窗体初始位置及背景色 ..... 13
- 1.4 修改标题栏图标 ..... 14
- 1.5 移动无边框窗体 ..... 16
- 1.6 去掉标题栏中最大化、最小化按钮 ..... 17
- 1.7 多窗体调用 ..... 18
- 1.8 字体形状窗体 ..... 20

## 第3章 控件应用

- 1.1 QPushButton 按钮 ..... 23
- 1.2 QLabel 标签 ..... 23
- 1.3 QLineEdit 单行文本 ..... 23

24	
1.4 QTextEdit 多行文本 .....	25
1.5 QPlainTextEdit 多行文本 .....	26
1.6 QComboBox 下拉列表框 .....	26
1.7 QFontComboBox 字体下拉列表框 .....	27
1.8 QSpinBox 控件 .....	28
1.9 QTimeEdit 时间控件 .....	29
1.10 QDateEdit 日期控件 .....	30
1.11 QScrollBar 控件 .....	30
1.12 QRadioButton 单选按钮 .....	31
1.13 QCheckBox 复选框 .....	32
1.14 QListView 列表控件 .....	34
1.15 QTreeView 树控件 .....	34
1.16 QTableView 表格控件 .....	35
1.17 QHBoxLayout 横向布局 .....	36
1.18 QGridLayout 网格布局 .....	37
1.19 QGroupBox 控件 .....	38
1.20 QTabWidget 控件 .....	39
1.21 QMenu、QToolBar 控件 .....	41
1.22 任务栏托盘菜单 .....	43
<b>第4章 组件应用</b>	
1.1 日历组件 .....	47
1.2 登录窗口 .....	48

1.3 文件浏览对话框 .....	50
1.4 颜色选择对话框 .....	51
1.5 进度条实例.....	53
1.6Timer 实时更新时间 .....	54
<b>第5章 文件操作</b>	
1.1 创建文件夹 .....	57
1.2 写入文件 .....	58
1.3 修改文件内容 .....	60
1.4 删除文件 .....	62
1.5 修改文件名 .....	63
1.6 INI 文件写入操作 .....	65
1.7 INI 文件读取操作 .....	68
1.8 创建 XML 文件 .....	71
1.9 读取 XML 文件 .....	72
<b>第6章 图形图像操作</b>	
1.1 绘制文字 .....	75
1.2 绘制线条 .....	75
1.3 绘制椭圆 .....	77
1.4 显示静态图像 .....	78
1.5 显示动态图像 .....	78
1.6 图片水平移动 .....	79
1.7 图片翻转 .....	80
1.8 图片缩放 .....	82

	1.9 图片中加文字 .....	84
	1.10 图像扭曲 .....	85
	1.11 模糊效果 .....	85
	1.12 着色效果 .....	86
	1.13 阴影效果 .....	87
	1.14 透明效果 .....	87
<b>第7章</b>	<b>多媒体应用</b>	
	1.1 音频、视频播放器 .....	90
	1.2 播放 Flash 动画 .....	94
	1.3 播放图片动画 .....	95
<b>第8章</b>	<b>系统操作</b>	
	1.1 获取屏幕分辨率 .....	98
	1.2 获取本机名、IP 地址 .....	98
	1.3 根据网址获取 IP 地址 .....	99
	1.4 判断键盘按下键值 .....	100
	1.5 获取系统环境变量 .....	101
	1.6 执行系统命令 .....	102
<b>第9章</b>	<b>注册表</b>	
	1.0 简要说明注册表 .....	105
	1.1 写入注册表 .....	105
	1.2 查找注册表 .....	106
	1.3 修改 IE 浏览器的默认主页 .....	107
<b>第10章</b>	<b>数据库基础</b>	
	1.1 查询数据库驱动 .....	109

1.2Qodbc 连接 Access 数据库 .....	109
1.3 插入数据 .....	111
1.4 数据列表 .....	112
1.5 操作 SQLite 数据库 .....	113
1.6SQLite 数据库视图管理器 .....	115
<b>第十一章 网络开发</b>	
1.1 点对点聊天服务端 .....	119
1.2 点对点聊天客户端 .....	123
1.3 局域网广播聊天 .....	128
1.4SMTP 协议发送邮件 .....	148
1.5 调用系统 DLL 判断网络连接状态 .....	152
<b>第十二章 进程与线程</b>	
1.1 进程管理器 .....	155
1.2 线程 QThread 应用 .....	158
1.3 线程 QRunnable 应用 .....	159
<b>第十三章 数据安全</b>	
1.1 QByteArray 加密数据 .....	163
1.2 AES 加密数据 .....	164
1.3 MD5 加密数据 .....	165
1.4 生成随机数 .....	166
<b>第十四章 打包部署</b>	
1.1 FilePacker 打包 .....	169
1.2 Inno Setup 打包 .....	174

# 第一章 开发环境

---

---

## 1.1 Qt 简介

Qt 是 1991 年由奇趣科技开发的跨平台 C++ 图形用户界面应用程序开发框架。它既可以开发 GUI 程式，也可用于开发非 GUI 程式，比如控制台工具和服务器。Qt 是面向对象的框架，使用特殊的代码生成扩展（称为元对象编译器(Meta Object Compiler, moc)）以及一些宏，易于扩展，允许组件编程。2008 年，奇趣科技被诺基亚公司收购，QT 也因此成为诺基亚旗下的编程语言工具。2012 年，Qt 被 Digia 收购。

### 1、优良的跨平台特性：

Qt 支持下列操作系统：Microsoft Windows 95/98， Microsoft Windows NT， Linux， Solaris， SunOS， HP-UX， Digital UNIX (OSF/1， Tru64)， Irix， FreeBSD， BSD/OS， SCO， AIX， OS390， QNX 等等。

### 2、面向对象

Qt 的良好封装机制使得 Qt 的模块化程度非常高，可重用性较好，对于用户开发来说是非常方便的。Qt 提供了一种称为 signals/slots 的安全类型来替代 callback，这使得各个元件之间的协同工作变得十分简单。

### 3、丰富的 API

Qt 包括多达 250 个以上的 C++ 类，还提供基于模板的 collections， serialization， file， I/O device， directory management， date/time 类。甚至还包括正则表达式的处理功能，支持 2D/3D 图形渲染，支持 OpenGL。





## 1.2 下载安装 Qt Creator

要进行 Qt 应用开发，开发环境是必备条件，Qt Creator 是跨平台的 Qt IDE，Qt Creator 是 Qt 被 Nokia 收购后推出的一款新的轻量级集成开发环境（IDE）。此 IDE 能够跨平台运行，支持的系统包括 Linux（32 位及 64 位）、Mac OS X 以及 Windows。根据官方描述，Qt Creator 的设计目标是使开发人员能够利用 Qt 这个应用程序框架更加快速及轻易的完成开发任务。

### 1、下载 Qt Creator

官网地址：<http://qt-project.org/downloads>，本书针对 Windows 环境，下载的软件是 Qt 5.1.1 for Windows 32-bit (MinGW 4.8, OpenGL, 666 MB)。

### 2、安装

单击 qt-windows-opensource-5.1.1-mingw48\_opengl-x86-offline.exe 可执行文件，如图 1.1 看到欢迎界面，参照图片点击下一步完成。

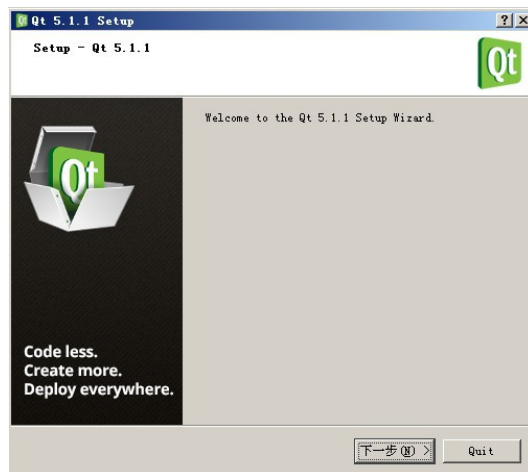


图 1.1



图 1.2

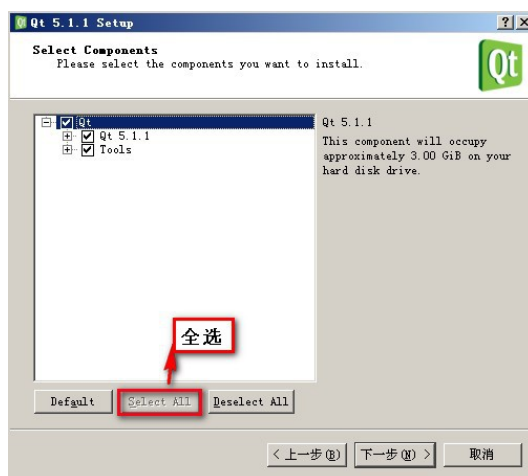


图 1.3

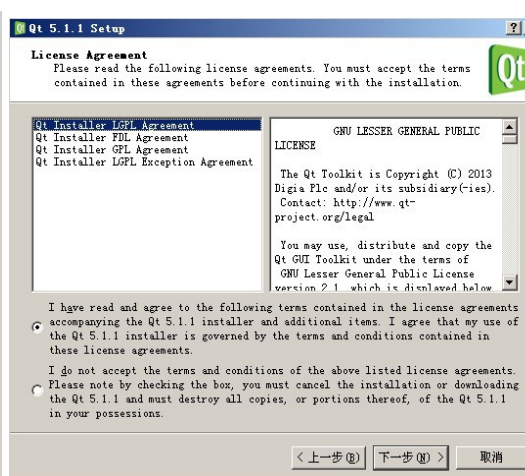


图 1.4

单击[下一步] 图 1.5 Qt Creator 就开始安装了。

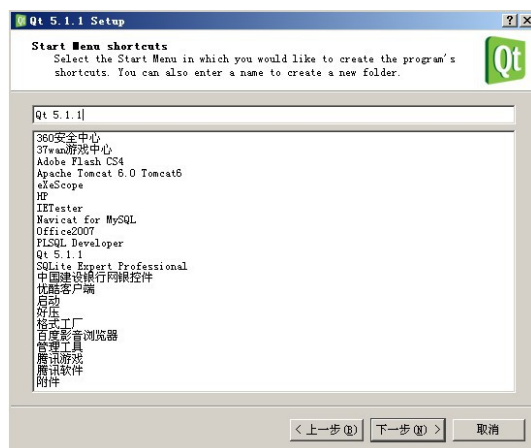


图 1.5

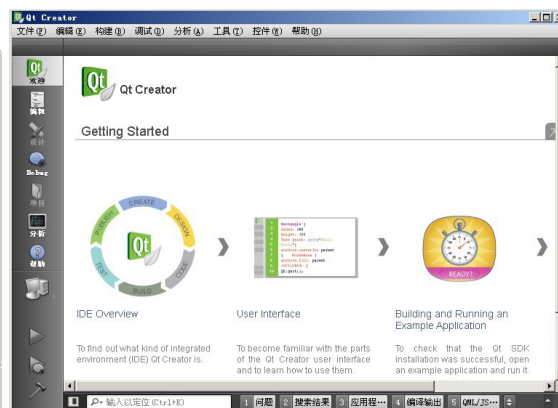


图 1.6

等几分钟之后如果看到图 1.6，那么恭喜你安装成功了。

### 3、配置环境变量

配置环境变量，以便于在以后项目中直接点击发布 exe 程序，可以直接打开程序，配置如下：

在桌面上选择“我的电脑”，单击右键，在弹出菜单中依次选择[属性]-[高级]-[环境变量]选项，在弹出窗口的系统变量中，找到 path 变量名，双击变量，在弹出框中添加 ;D:\Qt\Qt5.1.1\5.1.1\mingw48\_32\bin 注意前面的“；”号，

如图 1.7 单击确定。



图 1.7

## 1.3 第一个程序 Hello World

### 1、实例需求

在窗口输出“Hello World”文字。

### 2、实例参照

光盘/QtCode/QtOne/QtOne/ QtOne.pro

### 3、实例实现

打开 Qt Creator - 文件-新建文件或项目-应用程序-Qt Gui 应用-选择，如图 1.1

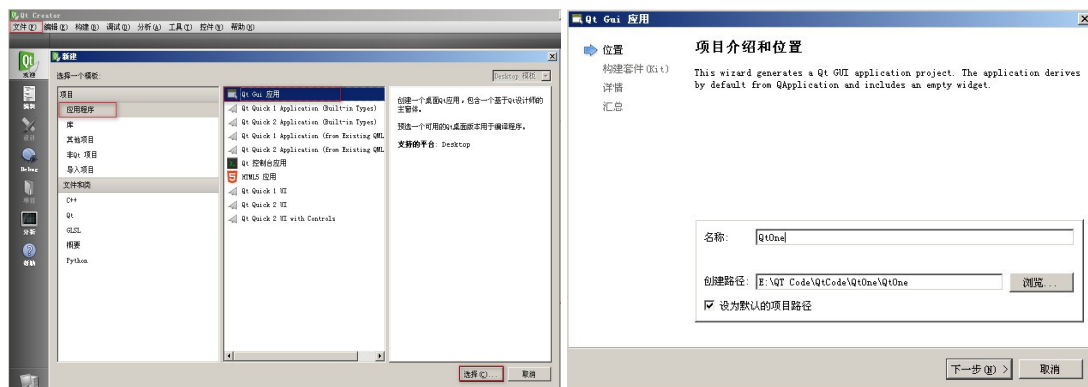


图 1.1

图 1.2

图 1.2 将项目放到指定目录，便于管理。单击[下一步]

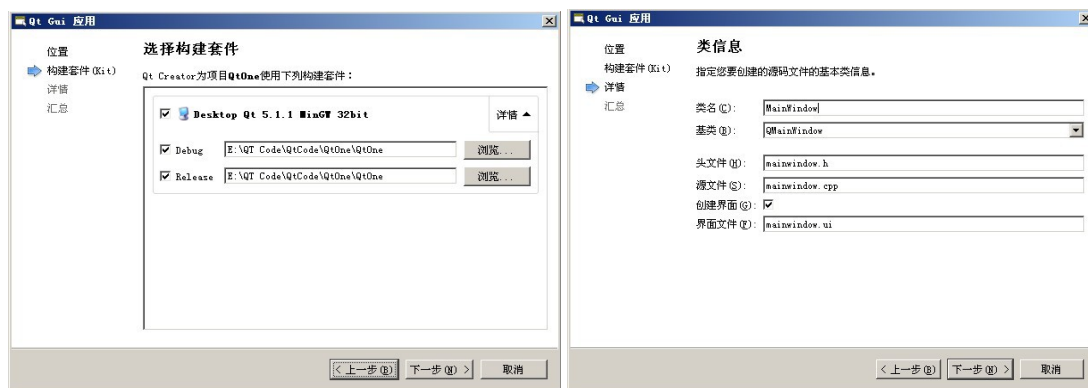


图 1.3

图 1.4

图 1.3 默认编译文件存放目录,图 1.4 直接点击[下一步], 后面详细说明三种基类用途。

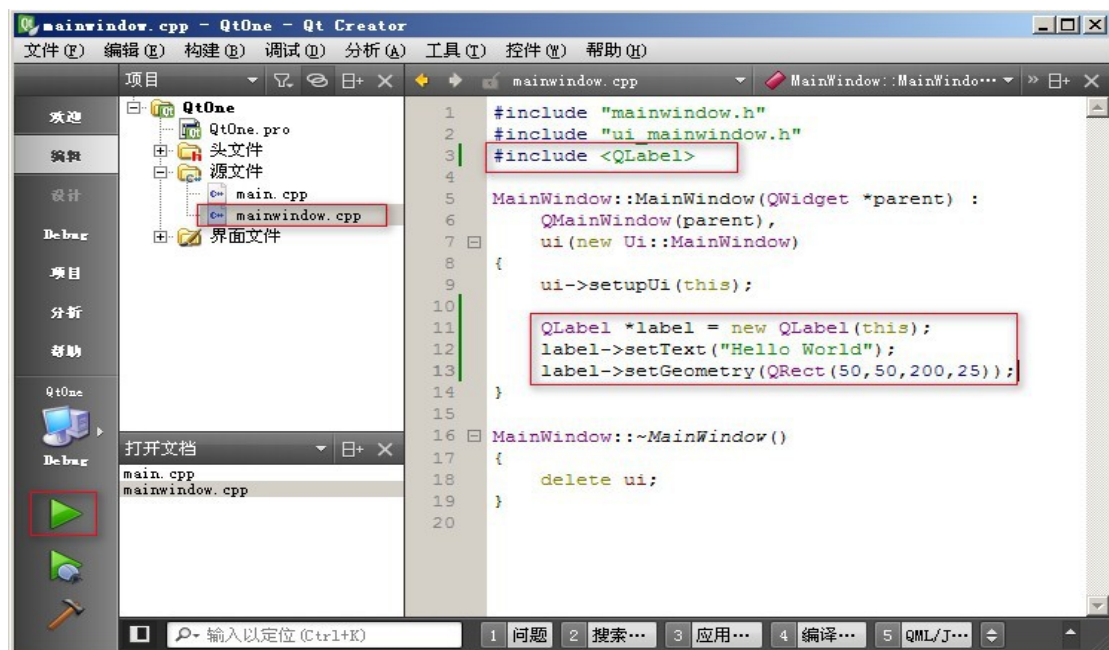


图 1.5

如图 1.5 编写完代码后点击 [绿色箭头] 执行程序，如图 1.6



图 1.6

mainwindow.cpp 源代码说明

```
1  #include "mainwindow.h"
2  #include "ui_mainwindow.h"
3  #include <QLabel>    //引用QLabel类
4
5  MainWindow::MainWindow(QWidget *parent) :
6      QMainWindow(parent),
7      ui(new Ui::MainWindow)
8  {
9      ui->setupUi(this);
10
11      //创建一个QLabel控件
12      QLabel *label = new QLabel(this);
13      //QLabel控件显示文字内容
14      label->setText("Hello World");
15      //显示位置
16      label->setGeometry(QRect(50,50,200,25));
17  }
18
19  MainWindow::~MainWindow()
20  {
21      delete ui;
22  }
```

代码说明：

1~3 行包含引用类，1~2 行是根据所选基类自动生成，3 行引用 QLabel 控件类。

5~9 行自动生成，功能绘制窗体并显示。

12 行实例一个 QLabel 控件以便显示内容。

14 行为 QLabel 控件添加内容。

16 行控件在窗体中显示位置，QRect(参数 1，参数 2，参数 3，参数 4)

参数 1：在窗体中 X 轴参数

参数 2：在窗体中 Y 轴参数

参数 3：QLabel 控件宽度

参数 4：QLabel 控件高度

19~22 行自动生成，回收机制，用完实例后释放内存。

## Main.cpp 源代码说明

```

1  #include "mainwindow.h"
2  #include <QApplication>
3
4  int main(int argc, char *argv[])
5  {
6      QApplication a(argc, argv);
7      MainWindow w;
8      w.show();
9
10     return a.exec();
11 }
12

```

代码说明：

- 1~2 行包含引用头文件。
- 4 行入口函数，即程序执行入口。
  - 参数 1: argc 命令行总的参数个数。
  - 参数 2: argv[] 是 argc 的参数。
- 6 行创建 QApplication 对象 a 参数为 argc,argv。
- 7 行实例对象 MainWindow 对象。
- 8 行显示对象。
- 10 行循环执行 QApplication 对象实例。

## 4、目录结构说明

打开项目目录如图 1.2，debug 文件夹与 release 文件夹内部文件是根据编辑器如图 1.1 选择而自动生成，文件夹内是编译后的可执行文件等。



图 1.1

图 1.2

进入 QtOne 文件夹，文件功能说明。如图 1.3

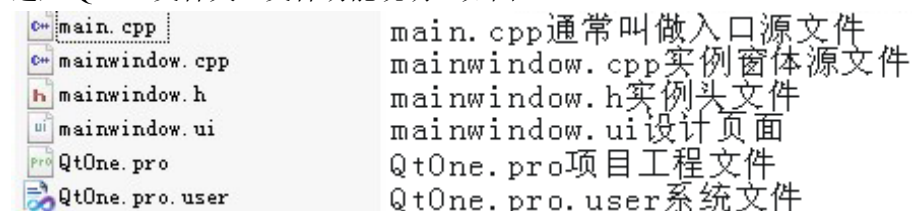


图 1.3

# 第二章 窗体应用

---

---

## 1.1 窗体基类说明



图 1.1

当创建项目到图 1.1 时，会发现编辑器提供三个基类，分别为：  
QMainWindow、QWidget、QDialog，三个基类的区别说明如下。

### 1、QMainWindow

QMainWindow 类提供一个有菜单条、锚接窗口（例如工具条）和一个状态条的主应用程序窗口。主窗口通常用在提供一个大的中央窗口部件（例如文本编辑或者绘制画布）以及周围菜单、工具条和一个状态条。QMainWindow 常常被继承，因为这使得封装中央部件、菜单和工具条以及窗口状态条变得更容易，当用户点击菜单项或者工具条按钮时，槽会被调用。

### 2、QWidget

QWidget 类是所有用户界面对象的基类。窗口部件是用户界面的一个基本单元：它从窗口系统接收鼠标、键盘和其它事件，并且在屏幕上绘制自己。每一个窗口部件都是矩形的，并且它们按 Z 轴顺序排列。一个窗口部件可以被它的父窗口部件或者它前面的窗口部件盖住一部分。

### 3、QDialog

QDialog 类是对话框窗口的基类。对话框窗口是主要用于短期任务以及和用户进行简要通讯的顶级窗口。QDialog 可以是模态对话框也可以是非模态对话框。QDialog 支持扩展性并且可以提供返回值。它们可以有默认按钮。



## 1.2 控制窗体大小

### 1、实例需求

控制窗体不可更改大，最大、最小为 300x300。



### 2、实例参照

- 光盘/QtCode/QtTwo/Qt01/ Qt01.pro

### 3、实例实现

```
//窗体标题
this->setWindowTitle("Qt5.1 窗体应用");
//窗体最大 300*300
this->setMaximumSize(300,300);
//窗体最小 300*300
this->setMinimumSize(300,300);
```

## 1.3 窗体初始位置及背景色

### 1、实例需求

控制窗体在屏幕右上角 X 轴 100，Y 轴 100 显示。

背景色为红色。



## 2、实例参照

- 光盘/QtCode/QtTwo/Qt02/ Qt02.pro

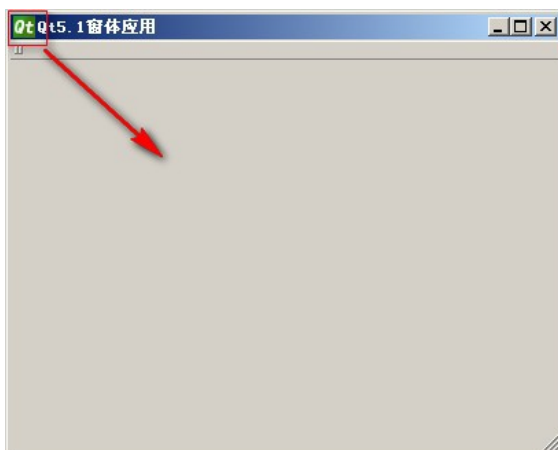
## 3、实例实现

```
//默认窗体居中显示，如果想要更改用 move 或 setGeometry  
this->move(100, 100);  
//背景红色  
this->setStyleSheet("background:red");
```

# 1.4 修改标题栏图标

## 1、实例需求

修改标题栏图标。



## 2、实例参照

- 光盘/QtCode/QtTwo/Qt03/ Qt03.pro

## 3、实例实现

要实现修改标题栏图标功能，首先需要有一个图片，最好是 16x16 像素的 ICO 图片，

这里在实例参照中已提供图片供使用。

第一步：

创建 Qt03 项目。

第二步：

在 Qt03 项目中创建文件夹 resource，文件夹名字可以自己定义，将准备好的 ICO 图片放入文件夹。

第三部：

Qt Creator – 文件 – 新建文件或项目 – 文件和类中选择 Qt – Qt 资源文件，如图 1.1

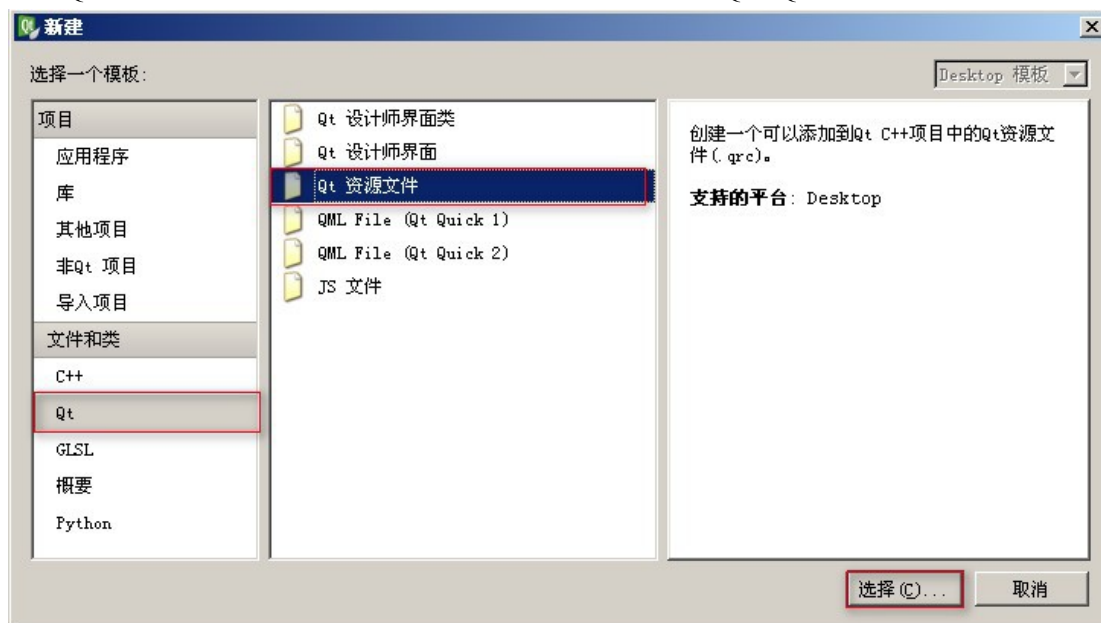


图 1.1

单击选择-弹出图 1.2 给资源文件起个名字，可自己定义。



图 1.2

图 1.3

单击 image.qrc 文件 – 单击[添加]按钮 [添加前缀] –在单击 [添加文件]选择刚刚项目中准备的 ICO 图片，单击 resource/QT.ico 给图片起一个别名，这里命名为 ico。

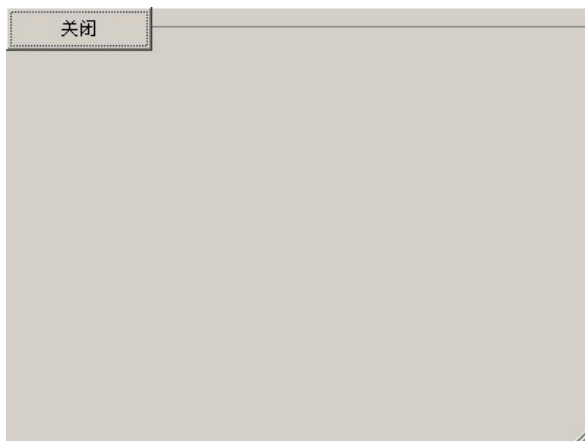
第四部：在 mainwindow.cpp 中添加代码。

```
//窗体标题
this->setWindowTitle("Qt5.1 窗体应用");
//窗体 ICO 图片，如图不起别名，后缀直接写图片全名。
this->setWindowIcon(QIcon(":/new/prefix1/ico"));
```

## 1.5 移动无边框窗体

### 1、实例需求

移动无边框窗体。



### 2、实例参照

- 光盘/QtCode/QtTwo/Qt04/ Qt04.pro

### 3、实例实现

第一步：打开 mainwindow.h 头文件，添加代码。

```
#include <QMouseEvent> //引用鼠标类头文件
#include <QPushButton> //引用按钮类头文件
```

```
//定义鼠标三种状态方法
protected:
    //鼠标按下
    void mousePressEvent(QMouseEvent *e);
    //鼠标移动
    void mouseMoveEvent(QMouseEvent *e);
    //鼠标释放
    void mouseReleaseEvent(QMouseEvent *e);

//定义 QPoint 对象
private:
    QPushButton *btClose;
```

```
QPoint last;
```

第二部：代开 mainwindow.cpp 源代码文件，添加代码。

```
//标题名
this->setWindowTitle("移动无边框窗体");
//去掉标题栏
this->setWindowFlags(Qt::FramelessWindowHint);

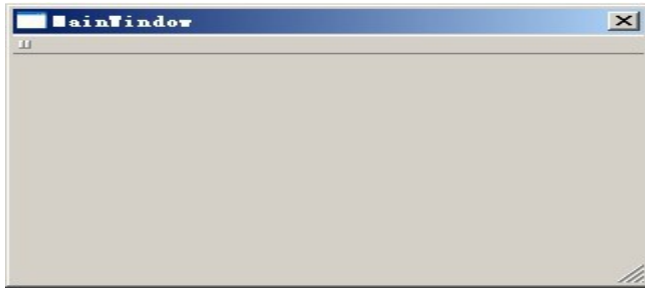
//实例一个按钮控件，因为去掉标题栏后，窗体没有关闭按钮了。
//所以自己添加一个按钮实现关闭功能。
btClose = new QPushButton(this);
btClose->setText("关闭");
//按钮点击事件
connect(btClose, SIGNAL(clicked()), this, SLOT(close()));
```

```
//获取鼠标点定位窗体位置
void MainWindow::mousePressEvent(QMouseEvent *e)
{
    last = e->globalPos();
}
void MainWindow::mouseMoveEvent(QMouseEvent *e)
{
    int dx = e->globalX() - last.x();
    int dy = e->globalY() - last.y();

    last = e->globalPos();
    move(x()+dx, y()+dy);
}
void MainWindow::mouseReleaseEvent(QMouseEvent *e)
{
    int dx = e->globalX() - last.x();
    int dy = e->globalY() - last.y();
    move(x()+dx, y()+dy);
}
```

## 1.6 去掉标题栏中最大化、最小化按钮

### 1、实例如图



## 2、实例参照

- 光盘/QtCode/QtTwo/Qt05/ Qt05.pro

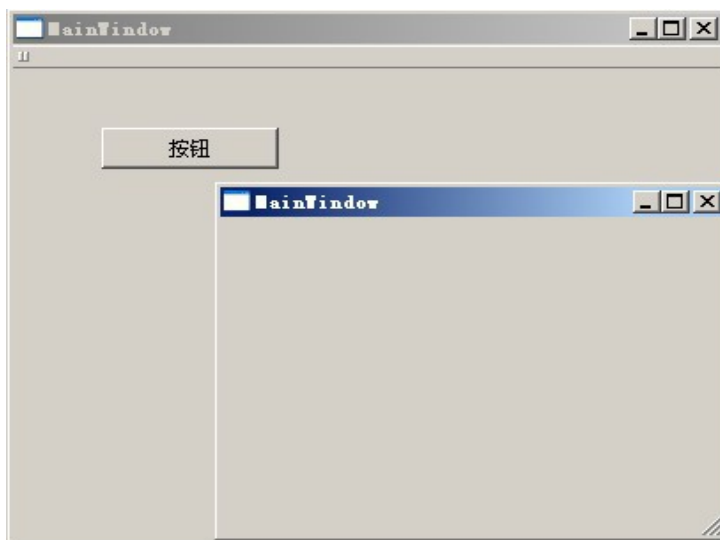
## 3、实例实现

```
//关闭按钮失效  
//this->setWindowFlags(Qt::WindowMinMaxButtonsHint);  
//去掉最大化、最小化按钮，保留关闭按钮  
this->setWindowFlags(Qt::WindowCloseButtonHint);
```

# 1.7 多窗体调用

## 1、实例需求

从 **MainWindow** 窗体点击按钮打开 **MainWindow2** 窗体。



## 2、实例参照

- 光盘/QtCode/QtTwo/Qt06/ Qt06.pro

## 3、实例实现

第一步：创建完项目之后，在点击 文件-新建文件或项目-文件和类-Qt-Qt 设计师界面类 如图 1.1

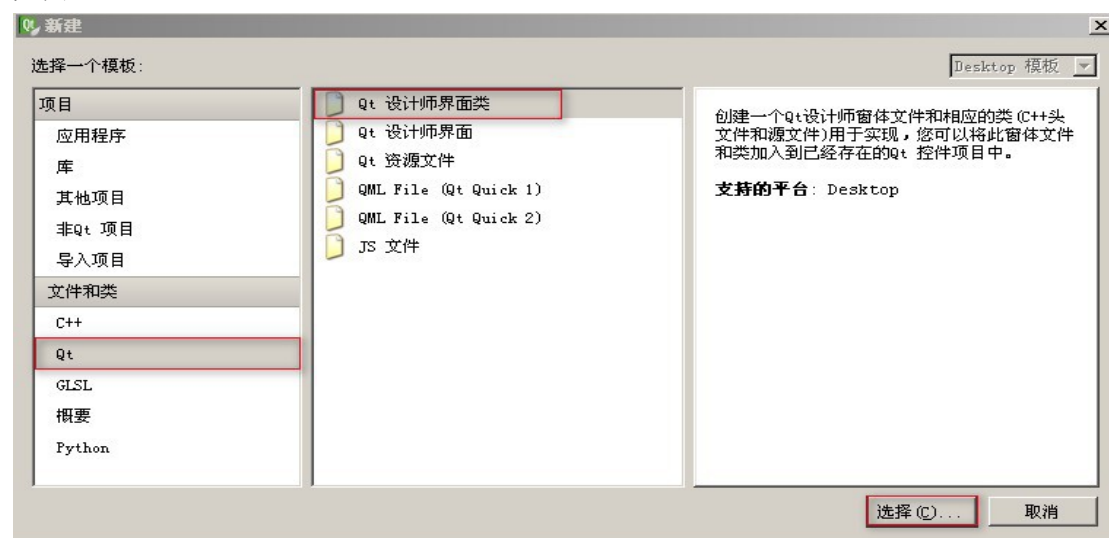


图 1.1

单击选择弹出图 1.2

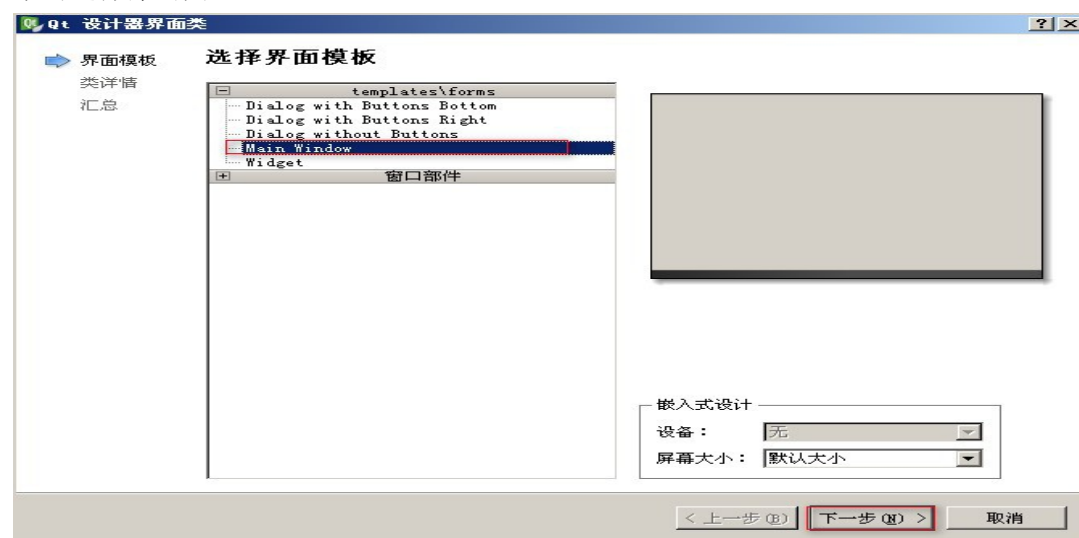


图 1.2

单击下一步，弹出图 1.3

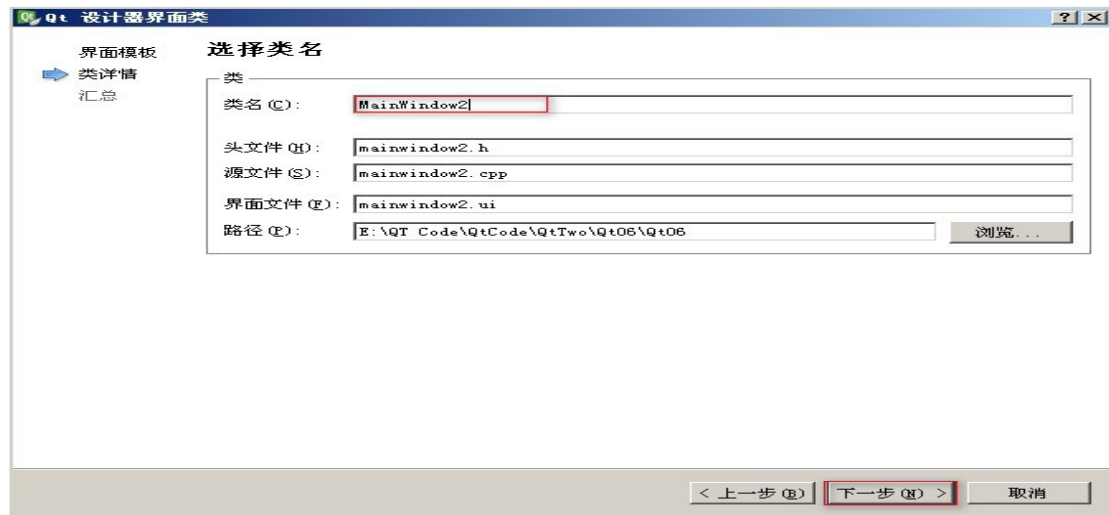


图 1.3

单击下一步，此时项目中又多了一个 MainWindow2 窗体。  
打开 MainWindow.h 头文件，编写代码。

```
#include <QPushButton> //引用 QPushButton 类
#include <mainwindow2.h> //引用 mainwindow2 类
```

```
//实例按钮
private:
    QPushButton *button;
    MainWindow2 w2;
//实例方法
private slots:
    void showMainWindow2();
```

打开 MainWindow.cpp 文件，编写代码。

```
//实例 QPushButton 控件
button = new QPushButton(this);
//按钮显示位置
button->setGeometry(QRect(50, 50, 100, 25));
//按钮值
button->setText("按钮");
//点击事件
connect(button, SIGNAL(clicked()), this, SLOT(showMainWindow2()));
```

```
//调用方法
void MainWindow::showMainWindow2()
```



```
{  
    //显示MainWindow2 窗体  
    w2.show();  
}
```

## 1.8 字体形状窗体

### 1、实例需求

将窗体背景色透明，根据图片形状显示窗体。



### 2、实例参照

- 光盘/QtCode/QtTwo/Qt07/ Qt07.pro

### 3、实例实现

将透明图片文字拷贝到项目中，前面已经讲述过怎么添加资源文件，这里就不在赘述了。

添加图片之后，打开 MainWindow.cpp 文件编写如下代码。

```
//去掉标题栏  
this->setWindowFlags(Qt::FramelessWindowHint);  
//设置背景透明  
this->setAttribute(Qt::WA_TranslucentBackground, true);  
//窗体添加样式，样式为CSS样式表  
// background-image:url 添加图片  
// background-repeat:no-repeat 不平铺  
this->setStyleSheet("background-image:url(/new/prefix1/touming);background-repeat:no-repeat;");
```

# 第三章 控件应用

---

---

## 1.1 QPushButton 按钮

### 1、实例需求

在窗体中创建按钮 A，点击按钮 A，改变文字为按钮 B。

### 2、实例参照

- 光盘/QtCode/QtThree/Qt01/ Qt01.pro

### 3、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QPushButton>
```

```
//声明 QPushButton  
private:  
    QPushButton *button;  
  
//声明 QPushButton 方法  
private slots:  
    void txtButton();
```

打开 MainWindow.cpp 文件编写如下代码。

```
//创建按钮  
button = new QPushButton("按钮 A", this);  
//定义按钮 X 轴, Y 轴, W 宽, H 高  
button->setGeometry(QRect(100, 100, 100, 25));  
//给按钮添加插槽事件(点击事件)  
connect(button, SIGNAL(released()), this, SLOT(txtButton()));
```

```
//点击方法  
void MainWindow::txtButton()  
{
```

```
//改变按钮文字
button->setText("按钮 B");
}
```

## 1.2 QLabel 标签

### 1、实例需求

在窗体中创建 QLabel 标签显示“我是 QLabel”字样，红色加粗倾斜字体。

### 2、实例参照

- 光盘/QtCode/QtThree/Qt02/ Qt02.pro

### 3、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QLabel>
```

```
//声明 QLabel
private:
    QLabel *label;
```

打开 MainWindow.cpp 文件编写如下代码。

```
//实例 QLabel 控件
label = new QLabel("我是 QLabel", this);
//QLabel 位置
//label->move(100, 100);
label->setGeometry(QRect(100, 100, 200, 30));
//label 样式(CSS 样式表)
//font-size 字号
//color 字体颜色
//font-weight 字宽
//font-style 字体样式
label->setStyleSheet("font-size:20px;color:red;font-weight:bold;font-style:italic");
```

## 1.3 QLineEdit 单行文本

## 1、实例参照

- 光盘/QtCode/QtThree/Qt03/ Qt03.pro

## 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QLineEdit>
```

```
//声明 QLineEdit 控件  
private:  
    QLineEdit *lineEdit;
```

打开 MainWindow.cpp 文件编写如下代码。

```
//创建 QLineEdit 控件  
lineEdit = new QLineEdit(this);  
//位置大小  
lineEdit->setGeometry(QRect(100, 100, 200, 25));  
//样式  
//border 边框线大小  
//border-style 边框样式 solid 实线  
//border-color:blue red 上下蓝色 左右红色 lineEdit->  
>setStyleSheet("border:1px;border-style:solid;color:red;border-color: blue  
red;");  
//限制最长输入 12 位  
lineEdit->setMaxLength(12);  
//不可写入  
//lineEdit->setEchoMode(QLineEdit::NoEcho);  
//密码*号输入  
lineEdit->setEchoMode(QLineEdit::Password);
```

## 1.4 QTextEdit 多行文本

### 1、实例参照

- 光盘/QtCode/QtThree/Qt04/ Qt04.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QTextEdit>
```

```
private:
    QTextEdit *textEdit;
```

打开 MainWindow.cpp 文件编写如下代码。

```
//实例 QTextEdit 控件
textEdit = new QTextEdit(this);
//控件位置大小
textEdit->setGeometry(QRect(50, 50, 200, 150));
//内容
textEdit->setText("我是第一行<br/>我是第二行");
```

## 1.5 QPlainTextEdit 多行文本

### 1、实例参照

- 光盘/QtCode/QtThree/Qt05/ Qt05.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QPlainTextEdit>
```

```
private:
    QPlainTextEdit *plainTextEdit;
```

打开 MainWindow.cpp 文件编写如下代码。

```
//实例
plainTextEdit = new QPlainTextEdit(this);
//位置
plainTextEdit->setGeometry(QRect(50, 50, 200, 100));
//添加内容
plainTextEdit->setPlainText("第一行");
```

## 1.6 QComboBox 下拉列表框

### 1、实例参照

- 光盘/QtCode/QtThree/Qt06/ Qt06.pro

## 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QComboBox>
```

```
private:  
    QComboBox *comboBox;
```

打开 MainWindow.cpp 文件编写如下代码。

```
//实例 QComboBox  
comboBox = new QComboBox(this);  
//控件显示位置大小  
comboBox->setGeometry(QRect(50, 50, 120, 25));  
//定义字符串列表  
QStringList str;  
str << "数学" << "语文" << "地理";  
//将字符串列表绑定 QComboBox 控件  
comboBox->addItem(str);
```

## 1.7 QFontComboBox 字体下拉列表框

### 1、实例参照

- 光盘/QtCode/QtThree/Qt07/ Qt07.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QFontComboBox>  
#include <QPushButton>  
#include <QLabel>
```

```
//实例控件  
private:  
    QFontComboBox *fontComboBox;  
    QPushButton *button;  
    QLabel *label;
```

```
//按钮点击方法
private slots:
    void txtButton();
```

打开 MainWindow.cpp 文件编写如下代码。

```
//实例 QFontComboBox
fontComboBox = new QFontComboBox(this);
//实例 QPushButton
button = new QPushButton(this);
//实例 QLabel
label = new QLabel(this);
label->setGeometry(QRect(50, 150, 300, 25));
//按钮名
button->setText("按钮");
//位置
button->move(180, 50);
//事件
connect(button, SIGNAL(released()), this, SLOT(txtButton()));
//QFontComboBox 控件位置
fontComboBox->setGeometry(QRect(50, 50, 120, 25));
```

```
//方法
void MainWindow::txtButton()
{
    label->setText("选择字体: "+fontComboBox->currentText());
}
```

## 1.8 QSpinBox 控件

### 1、实例参照

- 光盘/QtCode/QtThree/Qt08/ Qt08.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QSpinBox>
```

```
private:
    QSpinBox *spinBox;
```



打开 MainWindow.cpp 文件编写如下代码。

```
//实例 QSpinBox
spinBox = new QSpinBox(this);
//位置
spinBox->setGeometry(QRect(50, 50, 100, 25));
//值范围
spinBox->setRange(0, 200);
//初始值
spinBox->setValue(10);
//后缀
spinBox->setSuffix("元");
//前缀
spinBox->setPrefix("$");
```

## 1.9 QTimeEdit 时间控件

### 1、实例参照

- 光盘/QtCode/QtThree/Qt09/ Qt09.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QTimeEdit>
```

```
private:
    QTimeEdit *timeEdit;
```

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QDateTime>
```

```
//实例
timeEdit = new QTimeEdit(this);
//位置
timeEdit->setGeometry(QRect(50, 50, 120, 25));
//获取系统时间
QDateTime sysTime = QDateTime::currentDateTime();
```

```
//获取时分秒以“:”号拆分赋予list数组
QStringList list = sysTime.toString("hh:mm:ss").split(':');
//将时分秒绑定控件 timeEdit-
>setTime(QTime(list[0].toInt(),list[1].toInt(),list[2].toInt()));
```

## 1.10 QDateTime 日期控件

### 1、实例参照

- 光盘/QtCode/QtThree/Qt10/ Qt10.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QDateEdit>
```

```
private:
    QDateEdit *dateEdit;
```

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QDateTime>
```

```
//实例
dateEdit = new QDateEdit(this);
//位置
dateEdit->setGeometry(QRect(50, 50, 120, 25));
//获取系统时间
QDateTime sysTime = QDateTime::currentDateTime();
//获取时分秒以“-”号拆分赋予list数组
QStringList list = sysTime.toString("yyyy-MM-dd").split('-');
//将年月日绑定控件 dateEdit-
>setDate(QDate(list[0].toInt(),list[1].toInt(),list[2].toInt()));
```

## 1.11 QScrollBar 控件

### 1、实例参照

- 光盘/QtCode/QtThree/Qt11/ Qt11.pro

## 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QScrollBar>
#include <QSpinBox>
```

```
private:
    QScrollBar *scrollBar;
    QSpinBox *spinBox;
```

打开 MainWindow.cpp 文件编写如下代码。

```
//实例
scrollBar = new QScrollBar(this);
spinBox = new QSpinBox(this);
//横显/竖显
scrollBar->setOrientation(Qt::Horizontal);
//位置
scrollBar->setGeometry(QRect(50, 50, 180, 20));
spinBox->setGeometry(QRect(50, 90, 100, 25));
//控制条宽度
scrollBar->setPageStep(10);
//scrollBar 事件
connect(scrollBar, SIGNAL(valueChanged(int)), spinBox, SLOT(setValue(int)));
//spinBox 事件
connect(spinBox, SIGNAL(valueChanged(int)), scrollBar, SLOT(setValue(int)));
//初始值
scrollBar->setValue(50);
```

## 1.12 QRadioButton 单选按钮

### 1、实例参照

- 光盘/QtCode/QtThree/Qt12/ Qt12.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QRadioButton>
```

```
#include <QLabel>
```

```
private slots:
    void radioChange();
private:
    QRadioButton *radioM;
    QRadioButton *radioW;
    QLabel *label;
```

打开 MainWindow.cpp 文件编写如下代码。

```
//实例
radioM = new QRadioButton(this);
radioW = new QRadioButton(this);
label = new QLabel(this);
//位置
radioM->setGeometry(QRect(50, 50, 50, 50));
radioW->setGeometry(QRect(100, 50, 50, 50));
label->setGeometry(QRect(50, 100, 100, 25));
radioM->setText("男");
radioW->setText("女");
//默认选择
radioM->setChecked(true);
label->setText("男");
//事件
connect(radioM, SIGNAL(clicked()), this, SLOT(radioChange()));
connect(radioW, SIGNAL(clicked()), this, SLOT(radioChange()));
```

```
//Radio 点击方法
void MainWindow::radioChange()
{
    if(sender() == radioM)
    {
        label->setText("男");
    }else if(sender() == radioW)
    {
        label->setText("女");
    }
}
```

## 1.13 QCheckBox 复选框

## 1、实例参照

- 光盘/QtCode/QtThree/Qt13/ Qt13.pro

## 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QCheckBox>
#include <QLabel>
```

```
private slots:
    void checkChange();

private:
    QCheckBox *checkBox01;
    QCheckBox *checkBox02;
    QCheckBox *checkBox03;
    QLabel *label;
```

打开 MainWindow.cpp 文件编写如下代码。

```
//实例化 checkBox 控件
checkBox01 = new QCheckBox(this);
checkBox02 = new QCheckBox(this);
checkBox03 = new QCheckBox(this);
//实例 label 控件显示结果
label = new QLabel(this);
//控件位置
checkBox01->setGeometry(QRect(50, 50, 50, 50));
checkBox02->setGeometry(QRect(100, 50, 50, 50));
checkBox03->setGeometry(QRect(150, 50, 50, 50));
label->setGeometry(QRect(50, 100, 200, 30));
//控件值
checkBox01->setText("数学");
checkBox02->setText("语文");
checkBox03->setText("地理");
//checkbox 控件点击事件
connect(checkBox01, SIGNAL(clicked(bool)), this, SLOT(checkChange()));
connect(checkBox02, SIGNAL(clicked(bool)), this, SLOT(checkChange()));
connect(checkBox03, SIGNAL(clicked(bool)), this, SLOT(checkChange()));
```

```
//定义接收变量
QString str;

void MainWindow::checkChange()
{
    if(sender() == checkBox01)
    {
        //判断是否被选中
        if(checkBox01->checkState() == Qt::Checked)
        {
            str += "数学";
        }else
        {
            str = str.replace(QString("数学"),QString(""));
        }

    }else if(sender() == checkBox02)
    {
        if(checkBox02->checkState() == Qt::Checked)
        {
            str += "语文";
        }else
        {
            str = str.replace(QString("语文"),QString(""));
        }

    }else if(sender() == checkBox03)
    {
        if(checkBox03->checkState() == Qt::Checked)
        {
            str += "地理";
        }else
        {
            str = str.replace(QString("地理"),QString(""));
        }

    }

    //绑定值
    label->setText(str);
}
```

## 1.14 QListView 列表控件

### 1、实例参照

- 光盘/QtCode/QtThree/Qt14/ Qt14.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QListView>           //QListView 类
#include <QStringListModel> //数据模型类
```

```
private:
    QListView *listView;
    QStringListModel *model;
```

打开 MainWindow.cpp 文件编写如下代码。

```
//实例 listView 控件
listView = new QListView(this);
//定义位置宽高
listView->setGeometry(QRect(50, 50, 100, 100));
//StringList 数组
QStringList string;
string << "数学" << "语文" << "外语" << "地理";
//添加数据
model = new QStringListModel(string);
//将数据绑定 listView 控件
listView->setModel(model);
```

## 1.15 QTreeView 树控件

### 1、实例参照

- 光盘/QtCode/QtThree/Qt15/ Qt15.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QTreeView>
#include <QStandardItemModel>
```

```
private:
```

```
QTreeView *treeView;  
QStandardItemModel *model;
```

打开 MainWindow.cpp 文件编写如下代码。

```
//实例 QTreeView 控件  
treeView = new QTreeView(this);  
//位置  
treeView->setGeometry(QRect(50, 50, 150, 200));  
//实例数据类型 4 个节点, 2 列  
model = new QStandardItemModel(3, 2);  
//列名称  
model->setHeaderData(0, Qt::Horizontal, "第一列");  
model->setHeaderData(1, Qt::Horizontal, "第二列");  
//定义节点  
QStandardItem *item1 = new QStandardItem("数学");  
item1->setIcon(QIcon(":/new/prefix1/folder"));  
  
QStandardItem *item2 = new QStandardItem("语文");  
item2->setIcon(QIcon(":/new/prefix1/folder"));  
  
QStandardItem *item3 = new QStandardItem("外语");  
item3->setIcon(QIcon(":/new/prefix1/folder"));  
  
//外语子项  
QStandardItem *item4 = new QStandardItem("外语 A");  
item4->setIcon(QIcon(":/new/prefix1/file"));  
item3->appendRow(item4);  
  
//将节点添加至 QStandardItemModel  
model->setItem(0, 0, item1);  
model->setItem(1, 0, item2);  
model->setItem(2, 0, item3);  
  
//将 QStandardItemModel 数据绑定 QTreeView 控件  
treeView->setModel(model);
```

## 1.16 QTableView 表格控件

### 1、实例参照

- 光盘/QtCode/QtThree/Qt16/ Qt16.pro



## 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QTableView>
#include <QStandardItemModel>
```

```
private:
    QTableView *tableView;
    QStandardItemModel *model;
```

打开 MainWindow.cpp 文件编写如下代码。

```
//实例
tableView = new QTableView(this);
//位置
tableView->setGeometry(QRect(50, 50, 310, 200));
//实例数据模型
model = new QStandardItemModel();
//定义列
model->setHorizontalHeaderItem(0, new QStandardItem("数学"));
model->setHorizontalHeaderItem(1, new QStandardItem("语文"));
model->setHorizontalHeaderItem(2, new QStandardItem("外语"));

//行数据 0 行,0 列
model->setItem(0, 0, new QStandardItem("数学 A"));
model->setItem(0, 1, new QStandardItem("语文 A"));
model->setItem(0, 2, new QStandardItem("外语 A"));

model->setItem(1, 0, new QStandardItem("数学 B"));
model->setItem(1, 1, new QStandardItem("语文 B"));
model->setItem(1, 2, new QStandardItem("外语 B"));

//将数据模型绑定控件
tableView->setModel(model);
```

## 1.17 QHBoxLayout 横向布局

### 1、实例参照

- 光盘/QtCode/QtThree/Qt17/ Qt17.pro

## 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QHBoxLayout>
#include <QPushButton>
#include <QWidget>
```

```
private:
    QHBoxLayout *hboxLayout;
    QPushButton *button1;
    QPushButton *button2;
    QPushButton *button3;
    QWidget *widget;
```

打开 MainWindow.cpp 文件编写如下代码。

```
//创建横向布局
hboxLayout = new QHBoxLayout();
button1 = new QPushButton("按钮 1");
button2 = new QPushButton("按钮 2");
button3 = new QPushButton("按钮 3");

//向布局中添加控件
hboxLayout->addWidget(button1);
hboxLayout->addWidget(button2);
hboxLayout->addWidget(button3);

//间隔
hboxLayout->setSpacing(60);

//实例 QWidget
widget = new QWidget();
//绑定布局
widget->setLayout(hboxLayout);
//绑定界面
this->setCentralWidget(widget);
```

## 1.18 QGridLayout 网格布局

## 1、实例参照

- 光盘/QtCode/QtThree/Qt18/ Qt18.pro

## 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QGridLayout>
#include <QPushButton>
#include <QWidget>
```

```
private:
    QGridLayout *gridLayout;
    QPushButton *button1;
    QPushButton *button2;
    QPushButton *button3;
    QWidget *widget;
```

打开 MainWindow.cpp 文件编写如下代码。

```
//创建纵向布局
gridLayout = new QGridLayout();

button1 = new QPushButton("按钮 1");
button2 = new QPushButton("按钮 2");
button3 = new QPushButton("按钮 3");

//向布局中添加控件
gridLayout->addWidget(button1, 0, 0, 1, 1);
gridLayout->addWidget(button2, 0, 1, 1, 1);
gridLayout->addWidget(button3, 1, 0, 1, 1);

//实例 QWidget
widget = new QWidget();

//绑定布局
widget->setLayout(gridLayout);
//绑定界面
this->setCentralWidget(widget);
```

## 1.19 QGroupBox 控件

### 1、实例参照

- 光盘/QtCode/QtThree/Qt19/ Qt19.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QGroupBox>
#include <QPushButton>
#include <QVBoxLayout>
```

```
private:
    QGroupBox *box;
    QPushButton *button;
    QVBoxLayout *vbox;
```

打开 MainWindow.cpp 文件编写如下代码。

```
//实例 QGroupBox
box = new QGroupBox(this);
//位置、大小
box->setGeometry(QRect(30, 30, 340, 100));
//标题
box->setTitle("语音栏目");
//实例按钮
button = new QPushButton();
button->setText("按钮");
//实例布局
vbox = new QVBoxLayout;
//将按钮加入布局
vbox->addWidget(button);
//将布局加入 QGroupBox 控件
box->setLayout(vbox);
```

## 1.20 QTabWidget 控件

### 1、实例参照

- 光盘/QtCode/QtThree/Qt20/ Qt20.pro

## 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QTabWidget>
```

```
private:
    QTabWidget *tabWidget;
//tabA 类
class tabA:public QWidget
{
    Q_OBJECT
public:
    tabA(QWidget *parent=0);
};
//tabB 类
class tabB:public QWidget
{
    Q_OBJECT
public:
    tabB(QWidget *parent=0);
};
```

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QWidget>
#include <QPushButton>
```

```
//实例 QTabWidget
tabWidget = new QTabWidget(this);

tabWidget->setGeometry(QRect(30, 30, 340, 140));
tabWidget->addTab(new tabA, "A 栏目");
tabWidget->addTab(new tabB, "B 栏目");
```

```
tabA::tabA(QWidget *parent):QWidget(parent)
{
    QPushButton *buttonA = new QPushButton(this);
    buttonA->setText("页面 A");
}
```

```
tabB::tabB(QWidget *parent):QWidget(parent)
{
    QPushButton *buttonB = new QPushButton(this);
    buttonB->setText("页面B");
}
```

## 1.21 QMenu、QToolBar 控件

### 1、实例参照

- 光盘/QtCode/QtThree/Qt21/ Qt21.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QMenu>
#include <QMenuBar>
#include <QAction>
#include <QToolBar>
```

```
private:
    QMenu *fileMenu, *editMenu, *helpMenu;
    QToolBar *fileToolBar, *editToolBar;
    QAction *newAct;
    QAction *cutAct;
    QAction *copyAct;
    QAction *pasteAct;
    QAction *aboutQtAct;
private slots:
    void newFile();
```

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QMessageBox>
```

```
//实例菜单
fileMenu = new QMenu(this);
editMenu = new QMenu(this);
helpMenu = new QMenu(this);
```

```
//填充菜单子节点
newAct = new QAction(QIcon( ":/images/new" ), tr( "新建" ), this );
newAct->setShortcut(tr("Ctrl+N" ));
newAct->setStatusTip(tr("新建文件" ));
connect(newAct, SIGNAL(triggered()), this, SLOT(newFile()));

cutAct = new QAction(QIcon( ":/images/cut" ), tr( "剪切" ), this );
cutAct->setShortcut(tr("Ctrl+X" ));
cutAct->setStatusTip(tr("剪切内容" ));

copyAct = new QAction(QIcon( ":/images/copy" ), tr( "复制" ), this );
copyAct->setShortcut(tr("Ctrl+C" ));
copyAct->setStatusTip(tr("复制内容" ));

pasteAct = new QAction(QIcon( ":/images/paste" ), tr( "粘贴" ), this );
pasteAct->setShortcut(tr("Ctrl+V" ));
pasteAct->setStatusTip(tr("粘贴内容" ));

aboutQtAct = new QAction(tr( "关于 Qt" ), this );
aboutQtAct->setStatusTip(tr("关于 Qt 信息" ));
connect(aboutQtAct, SIGNAL(triggered()), qApp, SLOT(aboutQt()));

//填充菜单
fileMenu = menuBar()->addMenu(tr( "文件" ));
fileMenu->addAction(newAct);
fileMenu->addSeparator();

editMenu = menuBar()->addMenu(tr("编辑" ));
editMenu->addAction(cutAct);
editMenu->addAction(copyAct);
editMenu->addAction(pasteAct);
menuBar()->addSeparator();

helpMenu = menuBar()->addMenu(tr("帮助" ));
helpMenu->addAction(aboutQtAct);

//toolBar 工具条
fileToolBar = addToolBar(tr("新建"));
fileToolBar->addAction(newAct);

editToolBar = addToolBar(tr("修改"));
editToolBar->addAction(cutAct);
editToolBar->addAction(copyAct);
```

```
editToolBar->addAction(pasteAct);
```

```
//子菜单事件
void MainWindow::newFile() {
    QMessageBox::warning(this, tr("Warning"),
        tr("创建新文件? "),
        QMessageBox::Yes | QMessageBox::Default,
        QMessageBox::No);
}
```

## 1.22 任务栏托盘菜单

### 1、实例参照

- 光盘/QtCode/QtThree/Qt22/ Qt22.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QSystemTrayIcon> //任务栏类
#include <QMenu> //菜单类
```

```
private:
    QSystemTrayIcon *myTrayIcon;
    QMenu *myMenu;
    QAction *restoreWinAction;
    QAction *quitAction;
    void createMenu();

private slots:
    void showNormal();
```

打开 MainWindow.cpp 文件编写如下代码。

```
//菜单
createMenu();
//判断系统是否支持托盘图标
if(!QSystemTrayIcon::isSystemTrayAvailable())
{
    return;
}
```



```
//实例 QSystemTrayIcon
myTrayIcon = new QSystemTrayIcon(this);
//设置图标
myTrayIcon->setIcon(QIcon(":/new/prefix1/ico"));
//鼠标放托盘图标上提示信息
myTrayIcon->setToolTip("Qt 托盘图标功能");
//设置消息
myTrayIcon->showMessage("托盘", "托盘管理", QSystemTrayIcon::Information, 10000);
//托盘菜单
myTrayIcon->setContextMenu(myMenu);
//显示
myTrayIcon->show();
```

```
//绘制菜单
void MainWindow::createMenu()
{
    restoreWinAction = new QAction("恢复(&R)", this);
    quitAction = new QAction("退出(&Q)", this);
    //恢复
    connect(restoreWinAction, SIGNAL(triggered()), this, SLOT(showNormal()));
    //退出
    connect(quitAction, SIGNAL(triggered()), qApp, SLOT(quit()));

    myMenu = new QMenu((QWidget*)QApplication::desktop());
    //添加菜单
    myMenu->addAction(restoreWinAction);
    //分隔符
    myMenu->addSeparator();
    myMenu->addAction(quitAction);
}

//恢复
void MainWindow::showNormal()
{
    this->show();
}

//点击最小化按钮隐藏界面
void QWidget::changeEvent(QEvent *e)
{
    if((e->type() == QEvent::WindowStateChange) && this->isMinimized())
    {

```

```
        //QTimer::singleShot(100, this, SLOT(hide()));  
        this->hide();  
    }  
}
```

## 第四章 组件应用

---

## 1.1 日历组件

### 1、实例参照

- 光盘/QtCode/QtFour/Qt01/ Qt01.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QLabel>
#include <QLineEdit>
#include <QCalendarWidget>
```

```
private slots:
    void showTime();
    void setData();
private:
```

```
QLabel *label;  
QLineEdit *lineEdit;  
QCalendarWidget *calendarWidget;
```

打开 MainWindow.cpp 文件编写如下代码。

```
//实例 QLabel 控件  
label = new QLabel(this);  
label->setText("选择日期: ");  
//位置  
label->setGeometry(QRect(50, 50, 100, 25));  
lineEdit = new QLineEdit(this);  
lineEdit->setGeometry(QRect(110, 50, 150, 22));  
//事件  
connect(lineEdit, SIGNAL(cursorPositionChanged(int, int)), this, SLOT(showTime()));  
//实例时间控件  
calendarWidget = new QCalendarWidget(this);  
//位置  
calendarWidget->setGeometry(20, 75, 350, 180);  
//隐藏时间控件  
calendarWidget->setHidden(true);  
//时间控件点击事件  
connect(calendarWidget, SIGNAL(clicked(QDate)), this, SLOT(setData()));
```

```
void MainWindow::showTime()  
{  
    calendarWidget->setHidden(false);  
}  
  
void MainWindow::setData()  
{  
    //接收选择时间  
    QDate date = calendarWidget->selectedDate();  
    //时间格式化  
    QString str = date.toString("yyyy-MM-dd");  
    //赋值  
    lineEdit->setText(str);  
    //日期控件隐藏  
    calendarWidget->setHidden(true);  
}
```

## 1.2 登录窗口

### 1、实例参照

- 光盘/QtCode/QtFour/Qt02/ Qt02.pro

2、说明：这里需要注意在创建项目时：基类选择：**QWidget 类**，不选 **QMainWindow 类**。

打开 MainWindow.h 文件编写如下代码。

```
#include <QLabel>
#include <QGridLayout>
#include <QLineEdit>
#include <QPushButton>
#include <QHBoxLayout>
#include <QVBoxLayout>
#include <QMessageBox>
```

```
private:
    QLineEdit *usrLineEdit;
    QLineEdit *pwdLineEdit;
    QLabel *usrLabel;
    QLabel *pwdLabel;
    QGridLayout *gridlayout;
    QPushButton *okBtn;
    QPushButton *cancelBtn;
    QVBoxLayout *vboxLayout;

public slots:
    virtual void accept();
```

打开 MainWindow.cpp 文件编写如下代码。

```
//窗体最小宽度高度
setMinimumSize(280, 150);
//窗体最大宽度高度
setMaximumSize(280, 150);

//实例用户名密码控件
usrLabel = new QLabel(tr("用户名: "));
pwdLabel = new QLabel(tr("密 码: "));
usrLineEdit = new QLineEdit;
```

```
pwdLineEdit = new QLineEdit;
//密码*号输入
pwdLineEdit->setEchoMode(QLineEdit::Password);
//限制输入 12 位
usrLineEdit->setMaxLength(12);
pwdLineEdit->setMaxLength(12);

//实例网格布局控件
gridlayout = new QGridLayout;

//添加布局 0,0,1,1 行 列 行间距 列间距
gridlayout->addWidget(usrLabel, 0, 0, 1, 1);
gridlayout->addWidget(usrLineEdit, 0, 1, 1, 1);

//间隔
gridlayout->setSpacing(20);

gridlayout->addWidget(pwdLabel, 1, 0, 1, 1);
gridlayout->addWidget(pwdLineEdit, 1, 1, 1, 1);

//实例按钮控件
okBtn = new QPushButton(tr("确定"));
cancelBtn = new QPushButton(tr("取消"));

connect(okBtn, SIGNAL(clicked()), this, SLOT(accept()));
connect(cancelBtn, SIGNAL(clicked()), this, SLOT(reject()));

QHBoxLayout *hboxLayout = new QHBoxLayout;

hboxLayout->setSpacing(60);

//横向布局填充控件
hboxLayout->addWidget(okBtn);
hboxLayout->addWidget(cancelBtn);

//实例纵向布局填充刚刚实例的两个布局
vboxLayout = new QVBoxLayout;
vboxLayout->addLayout(gridlayout);
vboxLayout->addLayout(hboxLayout);

//显示内容
this->setLayout(vboxLayout);
```

```
//确定方法
void Dialog::accept()
{
    if(usrLineEdit->text().trimmed() == "admin" && pwdLineEdit->text().trimmed() == "admin")
    {
        QDialog::accept();
    }else
    {
        QMessageBox::warning(this, "警告", "用户名或密码错误!!!",
        QMessageBox::Yes);
        usrLineEdit->setFocus();
    }
}
```

## 1.3 文件浏览对话框

### 1、实例参照

- 光盘/QtCode/QtFour/Qt03/ Qt03.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QLineEdit>      //单行文本控件类
#include <QPushButton>    //按钮类
```

```
private:
    QLineEdit *filename;
    QPushButton *button;

private slots:
    void showFiles();
```

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QFileDialog>    //引用文件浏览对话框类
```

```
//实例单行文本控件
filename = new QLineEdit(this);
//位置
```

```
filename->setGeometry(QRect(50, 50, 230, 25));  
//实例按钮  
button = new QPushButton(this);  
//按钮位置  
button->setGeometry(QRect(280, 50, 80, 25));  
//按钮名  
button->setText("浏览");  
//按钮点击事件  
connect(button, SIGNAL(clicked()), this, SLOT(showFiles()));
```

```
//按钮点击方法  
void MainWindow::showFiles()  
{  
    //定义变量 str 接收 QFileDialog 对话框获取的文件路径  
    QString str = QFileDialog::getOpenFileName(this, "open file", "/", "text  
file(*.txt);;C file(*.cpp);;All file(*.*)");  
    //将变量绑定 QLineEdit 控件  
    filename->setText(str.toUtf8());  
}
```

## 1.4 颜色选择对话框

### 1、实例参照

- 光盘/QtCode/QtFour/Qt04/ Qt04.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QPushButton>  
#include <QLabel>
```

```
private:  
    QPushButton *button;  
    QLabel *label;  
private slots:  
    void editText();
```

打开 MainWindow.cpp 文件编写如下代码。



```
#include <QColorDialog> //引用 QColorDialog 类
```

```
//实例按钮
button = new QPushButton(this);
//位置
button->setGeometry(QRect(200, 50, 80, 25));
//按钮值
button->setText("选择颜色");
//按钮事件
connect(button, SIGNAL(clicked()), this, SLOT(editText()));
//实例 QLabel
label = new QLabel(this);
//label 位置
label->setGeometry(QRect(50, 50, 100, 25));
//label 值
label->setText("我的颜色可以改变");
```

```
//修改方法
void MainWindow::editText()
{
    //颜色对话框设置
    QColorDialog::setCustomColor(0, QRgb(0x0000FF));
    //定义 QColor 接收值
    QColor color = QColorDialog::getColor(QColor(0, 255, 0));
    //定义 QPalette(调色板类)
    QPalette p = palette();
    //调色板接收颜色
    p.setColor(QPalette::WindowText, QColor(color));
    //给 label 绑定颜色
    label->setPalette(p);
}
```

## 1.5 进度条实例

### 1、实例参照

- 光盘/QtCode/QtFour/Qt05/ Qt05.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QPushButton> //按钮类
#include <QProgressBar> //进度条类
```

```
private:
    QPushButton *button;
    QProgressBar *bar;
private slots:
    void startBar();
```

打开 MainWindow.cpp 文件编写如下代码。

```
//实例 QProgressBar 控件
bar = new QProgressBar(this);
//位置
bar->setGeometry(QRect(50, 50, 200, 20));
//范围值
bar->setRange(0, 100000-1);
//初始值
bar->setValue(0);
//实例 Button
button = new QPushButton(this);
//位置
button->setGeometry(QRect(270, 50, 80, 25));
//值
button->setText("开始");
//事件
connect(button, SIGNAL(clicked()), this, SLOT(startBar()));
```

```
void MainWindow::startBar()
{
    for(int i=0;i<100000;i++)
    {
        //100%结束
        if(i == 99999)
        {
            button->setText("结束");
        }
        //赋值
        bar->setValue(i);
    }
}
```

## 1.6Timer 实时更新时间

### 1、实例参照

- 光盘/QtCode/QtFour/Qt06/ Qt06.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QLabel>
#include <QTimer>
```

```
private:
    QLabel *label;
    QTimer *timer;
private slots:
    void timerTime();
```

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QDateTime>
```

```
//实例 QLabel 控件
label = new QLabel(this);
//位置
label->setGeometry(QRect(50, 50, 200, 25));
//实例 QTimer 控件
timer = new QTimer;
//timer 时间
connect(timer, SIGNAL(timeout()), this, SLOT(timerTime()));
//执行
timer->start(1000);
```

```
void MainWindow::timerTime()
{
    //获取系统时间
    QDateTime sysTime = QDateTime::currentDateTime();
    label->setText(sysTime.toString());
}
```

# 第五章 文件操作

---

## 1.1 创建文件夹

### 1、实例参照

- 光盘/QtCode/QtFive/Qt01/Qt01.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QLineEdit>
#include <QPushButton>
```

```
private:
    QLineEdit *edit;
    QPushButton *button;
private slots:
    void createFolder();
```

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QDir>
#include <QMessageBox>
```

```
//实例 QLineEdit 控件
edit = new QLineEdit(this);
//位置
edit->setGeometry(QRect(50, 50, 200, 25));
//值
edit->setText("D://1234Folder");
//实例按钮
button = new QPushButton(this);
//按钮位置、大小
button->setGeometry(QRect(280, 50, 80, 25));
//值
button->setText("创建");
//按钮事件
connect(button, SIGNAL(clicked()), this, SLOT(createFolder()));
```

```
void MainWindow::createFolder()
{
    //实例 QDir
    QDir *folder = new QDir;
    //判断创建文件夹是否存在
    bool exist = folder->exists(edit->text());
    if(exist)
    {
        QMessageBox::warning(this, tr("创建文件夹"), tr("文件夹已经存在!"));
    }else
    {
        //创建文件夹
        bool ok = folder->mkdir(edit->text());
    }
}
```

```
        //判断是否成功
        if(ok)
        {
            QMessageBox::warning(this, tr("创建文件夹"), tr("文件夹创建成功！"));
        }else
        {
            QMessageBox::warning(this, tr("创建文件夹"), tr("文件夹创建失败！"));
        }
    }
}
```

## 1.2 写入文件

### 1、实例参照

- 光盘/QtCode/QtFive/Qt02/Qt02.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QPushButton>
#include <QLineEdit>
```

```
private:
    QPushButton *button;
    QLineEdit *edit;
    QLineEdit *content;
private slots:
    void createFile();
```

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QFile>
#include <QMessageBox>
```

```
//实例
edit = new QLineEdit(this);
//位置
edit->setGeometry(QRect(50, 50, 200, 25));
//值
edit->setText("D:/Qt02.txt");
```

```
//实例
content = new QLineEdit(this);
//位置
content->setGeometry(QRect(50, 100, 200, 25));
//值
content->setText("写入文件的内容");

//实例 button
button = new QPushButton(this);
//位置
button->setGeometry(QRect(270, 50, 80, 25));
//值
button->setText("创建");
//事件
connect(button, SIGNAL(clicked()), this, SLOT(createFile()));
```

```
void MainWindow::createFile()
{
    //实例 QFile
    QFile file(edit->text());
    //判断文件是否存在
    if(file.exists())
    {
        QMessageBox::warning(this, "创建文件", "文件已经存在!");
    } else
    {
        //存在打开，不存在创建
        file.open(QIODevice::ReadWrite | QIODevice::Text);
        //写入内容, 这里需要转码，否则报错。
        QByteArray str = content->text().toUtf8();
        //写入 QByteArray 格式字符串
        file.write(str);
        //提示成功
        QMessageBox::warning(this, "创建文件", "文件创建成功!");
    }
    //关闭文件
    file.close();
}
```

### 1.3 修改文件内容



## 1、实例参照

- 光盘/QtCode/QtFive/Qt03/Qt03.pro

## 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QPushButton>
#include <QTextEdit>
```

```
private:
    QPushButton *browseBt;
    QPushButton *saveBt;
    QTextEdit *edit;
    QTextEdit *content;
private slots:
    void saveFile();
    void browseFile();
```

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QFileDialog>
#include <QMessageBox>
#include <QTextStream>
#include <QTextCodec>
```

```
//文件路径
edit = new QTextEdit(this);
edit->setGeometry(QRect(50, 50, 240, 26));

//浏览文件按钮
browseBt = new QPushButton(this);
browseBt->setGeometry(QRect(290, 50, 80, 25));
browseBt->setText("浏览文件");
connect(browseBt, SIGNAL(clicked()), this, SLOT(browseFile()));

//显示文件内容
content = new QTextEdit(this);
content->setGeometry(QRect(50, 80, 320, 150));

//修改完毕后，保存文件
saveBt = new QPushButton(this);
saveBt->setGeometry(QRect(290, 240, 80, 25));
```

```
saveBt->setText("保存");  
connect(saveBt, SIGNAL(clicked()), this, SLOT(saveFile()));
```

```
//浏览文件  
void MainWindow::browseFile()  
{  
    //定义变量 str 接收 QFileDialog 对话框获取的文件路径  
    QString str = QFileDialog::getOpenFileName(this, "open file", "/", "text  
file(*.txt);;C file(*.cpp);;All file(*.*)");  
    //将变量绑定 QTextEdit 控件  
    edit->setText(str.toUtf8());  
    //判断是否选择文件  
    if(edit->toPlainText().isEmpty())  
    {  
        return;  
    }  
    QFile file(edit->toPlainText());  
    //判断文件是否打开成功  
    if(!file.open(QIODevice::ReadOnly|QIODevice::Text))  
    {  
        QMessageBox::warning(this, "打开文件", "打开文件失败!");  
        return;  
    }  
    QTextStream ts(&file);  
    //循环文档数据至结尾  
    while(!ts.atEnd())  
    {  
        //将全部数据绑定至 content 控件  
        content->setPlainText(ts.readAll());  
    }  
    //关闭文档  
    file.close();  
}
```

```
//保存文件  
void MainWindow::saveFile()  
{  
    QFile file(edit->toPlainText());  
    file.open(QIODevice::ReadWrite | QIODevice::Text);  
    //写入内容,这里需要转码,否则报错。  
    QByteArray str = content->toPlainText().toUtf8();  
    //写入 QByteArray 格式字符串  
    file.write(str);  
}
```

```
//提示成功
MessageBox::warning(this, "修改文件", "文件修改成功!");
file.close();
}
```

## 1.4 删除文件

### 1、实例参照

- 光盘/QtCode/QtFive/Qt04/Qt04.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QLineEdit>
#include <QPushButton>
```

```
private:
    QLineEdit *filePath;
    QPushButton *browseBt;
    QPushButton *deleteBt;
private slots:
    void browseFile();
    void deleteFile();
```

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QFileDialog>
```

```
//显示文件路径
filePath = new QLineEdit(this);
filePath->setGeometry(QRect(50, 50, 200, 25));

//浏览文件按钮
browseBt = new QPushButton(this);
browseBt->setText("浏览文件");
browseBt->setGeometry(QRect(270, 50, 80, 25));
connect(browseBt, SIGNAL(clicked()), this, SLOT(browseFile()));

//删除文件按钮
```

```
deleteBt = new QPushButton(this);
deleteBt->setText("删除文件");
deleteBt->setGeometry(QRect(50, 100, 80, 25));
connect(deleteBt, SIGNAL(clicked()), this, SLOT(deleteFile()));
```

```
//浏览文件
void MainWindow::browseFile()
{
    //定义变量 str 接收 QFileDialog 对话框获取的文件路径
    QString str = QFileDialog::getOpenFileName(this, "open file", "/", "text
file(*.txt);;C file(*.cpp);;All file(*.*)");
    //将变量绑定 QTextEdit 控件
    filePath->setText(str.toUtf8());
}
```

```
//删除文件
void MainWindow::deleteFile()
{
    //删除文件
    QFile::remove(filePath->text());
}
```

## 1.5 修改文件名

### 1、实例参照

- 光盘/QtCode/QtFive/Qt05/Qt05.pro

### 2、实例实现



打开 MainWindow.h 文件编写如下代码。

```
#include <QLineEdit>
#include <QPushButton>
```

```
private:
    QLineEdit *filePath;
    QLineEdit *newName;
    QPushButton *browseBt;
    QPushButton *saveBt;
private slots:
    void browseFile();
    void saveFile();
```

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QFileDialog>
#include <QMessageBox>
```

```
//文件路径
filePath = new QLineEdit(this);
filePath->setGeometry(QRect(50, 50, 200, 25));

//浏览按钮
browseBt = new QPushButton(this);
browseBt->setGeometry(QRect(270, 50, 80, 25));
browseBt->setText("浏览");
connect(browseBt, SIGNAL(clicked()), this, SLOT(browseFile()));
```

```
//新名字
newName = new QLineEdit(this);
newName->setGeometry(QRect(50, 90, 200, 25));
newName->setText("新名字.txt");

//保存按钮
saveBt = new QPushButton(this);
saveBt->setGeometry(QRect(270, 90, 80, 25));
saveBt->setText("保存");
connect(saveBt, SIGNAL(clicked()), this, SLOT(saveFile()));
```

```
//浏览
void MainWindow::browseFile()
{
    //定义变量 str 接收 QFileDialog 对话框获取的文件路径
    QString str = QFileDialog::getOpenFileName(this, "open file", "/", "text
file(*.txt);;C file(*.cpp);;All file(*.*)");
    //将变量绑定 QTextEdit 控件
    filePath->setText(str.toUtf8());
}
```

```
//保存
void MainWindow::saveFile()
{
    //实例 QFileInfo 函数
    QFileInfo file(filePath->text());
    //获取文件路径
    QString path = file.absolutePath();
    //bool 型变量接收是否修改成功成功 true, 不成功 false。
    bool x = QFile::rename(filePath->text(), path + "/" + newName->text());
    if(x)
    {
        QMessageBox::warning(this, "修改文件名", "文件修改成功!");
    }else
    {
        QMessageBox::warning(this, "修改文件名", "文件修改失败!");
    }
}
```

## 1.6 INI 文件写入操作

## 1、实例参照

- 光盘/QtCode/QtFive/Qt06/Qt06.pro

## 2、实例实现



打开 MainWindow.h 文件编写如下代码。

```
#include <QLineEdit>
#include <QPushButton>
#include <QLabel>
#include <QGridLayout>
#include <QWidget>
#include <QSettings>
```

```
private:
    QGridLayout *gLayout;
    QWidget *widget;
    QLabel *filePathL;
    QLineEdit *filePath;
    QLabel *nodeL;
    QLineEdit *nodeEdit;
    QLabel *keyL;
    QLineEdit *keyEdit;
    QLabel *valL;
    QLineEdit *valEdit;
    QPushButton *writeBt;
    QSettings *writeIni;
```

```
private slots:
```

```
void writeFile();
```

打开 MainWindow.cpp 文件编写如下代码。

```
//ini 文件路径 Label
filePathL = new QLabel;
filePathL->setText("ini 文件完整路径");

//ini 文件路径输入框
filePath = new QLineEdit;
filePath->setText("D:/a001.ini");

//节点 Label
nodeL = new QLabel;
nodeL->setText("节点名");

//节点输入框
nodeEdit = new QLineEdit;
nodeEdit->setText("node");

//键 Label
keyL = new QLabel;
keyL->setText("键值");

//键输入框
keyEdit = new QLineEdit;
keyEdit->setText("ip");

//值 Label
valL = new QLabel;
valL->setText("值");

//值输入框
valEdit = new QLineEdit;
valEdit->setText("192.168.1.1");

//按钮
writeBt = new QPushButton;
writeBt->setText("写入文件");
connect(writeBt, SIGNAL(clicked()), this, SLOT(writeFile()));

gLayout = new QGridLayout;
```



```
gLayout->addWidget(filePathL, 0, 0, 1, 1);
gLayout->addWidget(filePath, 0, 1, 1, 4);
gLayout->addWidget(nodeL, 1, 0, 1, 1);
gLayout->addWidget(nodeEdit, 1, 1, 1, 1);
gLayout->addWidget(keyL, 2, 0, 1, 1);
gLayout->addWidget(keyEdit, 2, 1, 1, 1);
gLayout->addWidget(valL, 2, 3, 1, 1);
gLayout->addWidget(valEdit, 2, 4, 1, 1);
gLayout->addWidget(writeBt, 3, 4, 1, 1);
```

```
//实例 QWidget
```

```
widget = new QWidget();
```

```
//绑定布局
```

```
widget->setLayout(gLayout);
```

```
this->setCentralWidget(widget);
```

```
//写入文件
```

```
void MainWindow::writeFile()
```

```
{
```

```
    //QSettings 构造函数的第一个参数是 ini 文件的路径, 第二个参数表示针对 ini 文件
```

```
    writeIni = new QSettings(filePath->text(), QSettings::IniFormat);
```

```
    //写入键、值
```

```
    writeIni->setValue(nodeEdit->text() + "/" + keyEdit->text(), valEdit->text());
```

```
    //写入完成删除指针
```

```
    delete writeIni;
```

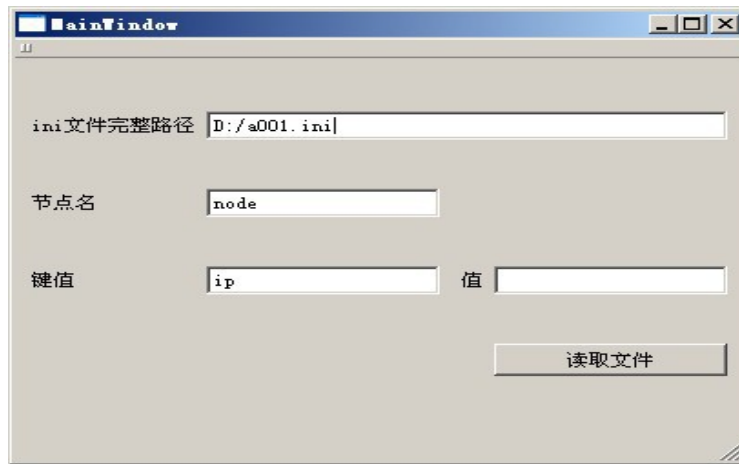
```
}
```

## 1.7 INI 文件读取操作

### 1、实例参照

- 光盘/QtCode/QtFive/Qt07/Qt07.pro

### 2、实例实现



打开 MainWindow.h 文件编写如下代码。

```
#include <QLineEdit>
#include <QPushButton>
#include <QLabel>
#include <QGridLayout>
#include <QWidget>
#include <QSettings>
```

```
private:
    QGridLayout *gLayout;
    QWidget *widget;
    QLabel *filePathL;
    QLineEdit *filePath;
    QLabel *nodeL;
    QLineEdit *nodeEdit;
    QLabel *keyL;
    QLineEdit *keyEdit;
    QLabel *valL;
    QLineEdit *valEdit;
    QPushButton *readBt;
    QSettings *readIni;

private slots:
    void readFile();
```

打开 MainWindow.cpp 文件编写如下代码。

```
//ini 文件路径 Label
filePathL = new QLabel;
filePathL->setText("ini 文件完整路径");
```

```
//ini 文件路径输入框
filePath = new QLineEdit;
filePath->setText("D:/a001.ini");

//节点 Label
nodeL = new QLabel;
nodeL->setText("节点名");

//节点输入框
nodeEdit = new QLineEdit;
nodeEdit->setText("node");

//键 Label
keyL = new QLabel;
keyL->setText("键值");

//键输入框
keyEdit = new QLineEdit;
keyEdit->setText("ip");

//值 Label
valL = new QLabel;
valL->setText("值");

//值输入框
valEdit = new QLineEdit;

//按钮
readBt = new QPushButton;
readBt->setText("读取文件");
connect(readBt, SIGNAL(clicked()), this, SLOT(readFile()));

gLayout = new QGridLayout;
gLayout->addWidget(filePathL, 0, 0, 1, 1);
gLayout->addWidget(filePath, 0, 1, 1, 4);
gLayout->addWidget(nodeL, 1, 0, 1, 1);
gLayout->addWidget(nodeEdit, 1, 1, 1, 1);
gLayout->addWidget(keyL, 2, 0, 1, 1);
gLayout->addWidget(keyEdit, 2, 1, 1, 1);
gLayout->addWidget(valL, 2, 3, 1, 1);
gLayout->addWidget(valEdit, 2, 4, 1, 1);
gLayout->addWidget(readBt, 3, 4, 1, 1);
```

```
//实例 QWidget
widget = new QWidget();
//绑定布局
widget->setLayout(gLayout);
this->setCentralWidget(widget);
```

```
//读取文件
void MainWindow::readFile()
{
    //QSettings 构造函数的第一个参数是 ini 文件的路径, 第二个参数表示针对 ini 文件
    readIni = new QSettings(filePath->text(), QSettings::IniFormat);
    //将读取到的 ini 文件保存在 QString 中, 先取值, 然后通过 toString() 函数转换成
    QString 类型
    QString ipResult = readIni->value(nodeEdit->text()+"/"+keyEdit->
    >text()).toString();
    //将结果绑定 IP 值控件上
    valEdit->setText(ipResult);
    //写入完成删除指针
    delete readIni;
}
```

## 1.8 创建 XML 文件

### 1、实例参照

- 光盘/QtCode/QtFive/Qt08/Qt08.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QLabel>
#include <QPushButton>
```

```
private:
    QLabel *explainL;
    QPushButton *createBt;

private slots:
    void createFile();
```

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QFile>
#include <QXmlStreamWriter>
```

```
//实例 QLabel
explainL = new QLabel(this);
explainL->setText("这里就不重复写类似 txt 读写的布局了，直接在代码中\r\n 填写节点等操作。 \r\n 文件路径： D:/a001.xml\r\n 根节点： Root");
explainL->setGeometry(QRect(50, 50, 300, 100));

//实例按钮
createBt = new QPushButton(this);
createBt->setText("创建 XML 文件");
createBt->setGeometry(QRect(50, 180, 100, 25));
connect(createBt, SIGNAL(clicked()), this, SLOT(createFile()));
```

```
//创建文件
void MainWindow::createFile()
{
    //文件路径
    QString xmlPath = "D:/a001.xml";
    QFile file(xmlPath);
    if(file.open(QIODevice::WriteOnly | QIODevice::Text))
    {
        //实例 QXmlStreamWriter
        QXmlStreamWriter stream(&file);
        stream.setAutoFormatting(true);
        //文档头
        stream.writeStartDocument();
        //根节点
        stream.writeStartElement("Root");
        //元素、值
        stream.writeAttribute("href", "http://qt.nokia.com/");
        //节点内容
        stream.writeTextElement("title", "Qt Home");
        stream.writeEndElement();
        stream.writeEndDocument();
        //关闭文件
        file.close();
    }
}
```

```
}  
}
```

## 1.9 读取 XML 文件

### 1、实例参照

- 光盘/QtCode/QtFive/Qt09/Qt09.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QLabel>  
#include <QPushButton>
```

```
private:  
    QLabel *resultL;  
    QPushButton *readBt;  
  
private slots:  
    void readNode();
```

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QFile>  
#include <QXmlStreamReader>
```

```
//实例 QLabel  
resultL = new QLabel(this);  
resultL->setGeometry(QRect(50, 50, 300, 100));  
  
//实例按钮  
readBt = new QPushButton(this);  
readBt->setText("读取 title 节点");  
readBt->setGeometry(QRect(50, 180, 150, 25));  
connect(readBt, SIGNAL(clicked()), this, SLOT(readNode()));
```

```
//读取节点  
void MainWindow::readNode()  
{  
    //文件路径
```

```
QString xmlPath = "D:/a001.xml";
QFile file(xmlPath);
//定义变量接收信息
QString str;
//判断文件是否存在
if(file.exists())
{
    if(file.open(QIODevice::ReadOnly | QIODevice::Text))
    {
        //实例 QXmlStreamReader 对象读取文件
        QXmlStreamReader xmlRead(&file);

        //循环节点
        while (!xmlRead.atEnd())
        {
            //指针下移
            xmlRead.readNext();
            if(xmlRead.isStartElement())
            {
                //如果节点有等于 title 的
                if(xmlRead.name() == "title")
                {
                    //取 title 值赋予变量 str
                    str = xmlRead.readElementText();
                }else
                {
                    str = "没找到节点";
                }
            }
        }
        //将值绑定 QLabel 控件显示
        resultL->setText(str);
    }else
    {
        resultL->setText("文件打开失败");
    }
    //关闭文件
    file.close();
}else
{
    resultL->setText("文件不存在");
}
}
```

## 第六章 图形图像操作

---



## 1.1 绘制文字

### 1、实例参照

- 光盘/QtCode/QtSix/Qt01/Qt01.pro

### 2、实例实现

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QGraphicsScene>
#include <QGraphicsView>
```

```
//实例 QGraphicsScene
QGraphicsScene *scene = new QGraphicsScene;
//背景色
scene->setForegroundBrush(QColor(0, 255, 255, 127));
//字体设置
QFont font("黑体", 60);
//添加文字
scene->addSimpleText("图形图像", font);
```

```
//网格
//scene->setForegroundBrush(QBrush(Qt::lightGray, Qt::CrossPattern));
//实例 QGraphicsView
QGraphicsView *view = new QGraphicsView(scene);
//显示
this->setCentralWidget(view);
```

## 1.2 绘制线条

### 1、实例参照

- 光盘/QtCode/QtSix/Qt02/Qt02.pro

### 2、实例实现

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QGraphicsScene>
#include <QGraphicsView>
```

```
//实例 QGraphicsScene
QGraphicsScene *scene = new QGraphicsScene;

//QPen 属性
QPen pen;
pen.setStyle(Qt::DashDotLine);
pen.setWidth(2);
pen.setBrush(Qt::black);
pen.setCapStyle(Qt::RoundCap);
pen.setJoinStyle(Qt::RoundJoin);

//绘制线条
scene->addLine(30, 30, 200, 30, pen);
//实例 QGraphicsView
QGraphicsView *view = new QGraphicsView(scene);
//显示
this->setCentralWidget(view);
```

## 1.3 绘制椭圆

### 1、实例参照

- 光盘/QtCode/QtSix/Qt03/Qt03.pro

### 2、实例实现

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QGraphicsScene>
#include <QGraphicsView>
```

```
//实例 QGraphicsScene
QGraphicsScene *scene = new QGraphicsScene;
//绘制椭圆
scene->addEllipse(50, 50, 100, 120);
//实例 QGraphicsView
QGraphicsView *view = new QGraphicsView(scene);
//显示
this->setCentralWidget(view);
```

## 1.4 显示静态图像

### 1、实例参照

- 光盘/QtCode/QtSix/Qt04/Qt04.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
protected:
    void paintEvent(QPaintEvent *);
```

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QPainter>
```

```
void MainWindow::paintEvent(QPaintEvent *)
{
    //实例 QPixmap
    QPixmap image(":/new/prefix1/dog");
```

```
//实例 QPainter 绘制类
QPainter painter(this);
//绘制图片
painter.drawPixmap(20, 20, 200, 257, image);
}
```

## 1.5 显示动态图像

### 1、实例参照

- 光盘/QtCode/QtSix/Qt05/Qt05.pro

### 2、实例实现

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QLabel>
#include <QTimer>
#include <QMovie>
```

```
//实例 QLabel
QLabel *label = new QLabel(this);
label->setGeometry(QRect(50, 50, 88, 51));
//实例 QMovie
QMovie *movie = new QMovie(":/new/prefix1/img");
//3 秒后图片消失
QTimer::singleShot( 3*1000, label, SLOT(close()));
label->setMovie(movie);
movie->start();
```

## 1.6 图片水平移动

### 1、实例参照

- 光盘/QtCode/QtSix/Qt06/Qt06.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QLabel>
#include <QImage>
#include <QPushButton>
```

```
private:
    QLabel *label;
    QImage *img;
    QPushButton *button;
private slots:
    void moveImg();
```

打开 MainWindow.cpp 文件编写如下代码。

```
//实例 QLabel
label = new QLabel(this);
label->setGeometry(QRect(50, 50, 75, 77));

//实例 QImage
img = new QImage;
//QImage 加载图片
img->load(":/new/prefix1/hd");
//label 显示图片
label->setPixmap(QPixmap::fromImage(*img));

//实例 button
button = new QPushButton(this);
button->setGeometry(QRect(50, 150, 80, 25));
button->setText("移动");
connect(button, SIGNAL(clicked()), this, SLOT(moveImg()));
```

```
//点击移动
int i = 50;
void MainWindow::moveImg()
{
    //移动 label 控件
    label->move(i, 50);
    i += 10;
}
```

## 1.7 图片翻转

### 1、实例参照

- 光盘/QtCode/QtSix/Qt07/Qt07.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QLabel>
#include <QImage>
#include <QPushButton>
```

```
private:
    QLabel *label;
    QImage *img;
    QPushButton *hBt;
    QPushButton *vBt;
    QPushButton *angleBt;
private slots:
    void hShow();
    void vShow();
    void angleShow();
```

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QMatrix>
```

```
//实例 QLabel
label = new QLabel(this);
label->setGeometry(QRect(160, 50, 75, 77));

//实例 QImage
img = new QImage;
//QImage 加载图片
img->load(":/new/prefix1/hd");
//label 显示图片
label->setPixmap(QPixmap::fromImage(*img));

//实例水平操作按钮
hBt = new QPushButton(this);
```

```
hBt->setGeometry(QRect(50, 200, 80, 25));
hBt->setText("水平翻转");
connect(hBt, SIGNAL(clicked()), this, SLOT(hShow()));

//实例垂直操作按钮
vBt = new QPushButton(this);
vBt->setGeometry(QRect(160, 200, 80, 25));
vBt->setText("垂直操作");
connect(vBt, SIGNAL(clicked()), this, SLOT(vShow()));

//实例角度操作按钮 88°
angleBt = new QPushButton(this);
angleBt->setGeometry(QRect(270, 200, 80, 25));
angleBt->setText("角度操作");
connect(angleBt, SIGNAL(clicked()), this, SLOT(angleShow()));
```

```
//水平操作
void MainWindow::hShow()
{
    //水平翻转
    *img = img->mirrored(true, false);
    //显示图片
    label->setPixmap(QPixmap::fromImage(*img));
}

//垂直操作
void MainWindow::vShow()
{
    //垂直翻转
    *img = img->mirrored(false, true);
    //显示图片
    label->setPixmap(QPixmap::fromImage(*img));
}

//角度操作
void MainWindow::angleShow()
{
    QMatrix matrix;
    //88 度角
    matrix.rotate(88);
    *img = img->transformed(matrix);
    label->setPixmap(QPixmap::fromImage(*img));
}
```

```
}
```

## 1.8 图片缩放

### 1、实例参照

- 光盘/QtCode/QtSix/Qt08/Qt08.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
#include <QLabel>
#include <QImage>
#include <QPushButton>
```

```
private:
    QLabel *label;
    QImage *img;
    QPushButton *bigBt;
    QPushButton *smallBt;
private slots:
    void bShow();
    void sShow();
```

打开 MainWindow.cpp 文件编写如下代码。

```
//实例 QLabel
label = new QLabel(this);
label->setGeometry(QRect(160, 50, 75, 77));

//实例 QImage
img = new QImage;
//QImage 加载图片
img->load(":/new/prefix1/hd");
//label 显示图片
label->setPixmap(QPixmap::fromImage(*img));

//实例放大按钮
bigBt = new QPushButton(this);
bigBt->setGeometry(QRect(50, 200, 80, 25));
```



```
bigBt->setText("放大");
connect(bigBt, SIGNAL(clicked()), this, SLOT(bShow()));

//实例缩小按钮
smallBt = new QPushButton(this);
smallBt->setGeometry(QRect(260, 200, 80, 25));
smallBt->setText("缩小");
connect(smallBt, SIGNAL(clicked()), this, SLOT(sShow()));
```

```
//放大操作
void MainWindow::bShow()
{
    *img = img->scaled(100, 100, Qt::IgnoreAspectRatio);
    label->setPixmap(QPixmap::fromImage(*img));
}

//缩小操作
void MainWindow::sShow()
{
    *img = img->scaled(60, 60, Qt::IgnoreAspectRatio);
    label->setPixmap(QPixmap::fromImage(*img));
}
```

## 1.9 图片中加文字

### 1、实例参照

- 光盘/QtCode/QtSix/Qt09/Qt09.pro

### 2、实例实现

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QImage>
#include <QPainter>
#include <QLabel>
```

```
//实例 QLabel
QLabel *label = new QLabel(this);
label->setGeometry(QRect(50, 50, 300, 25));
```

```
label->setText("图片已经生成，保存在项目文件中【text.jpg】。");

//实例 QImage
QImage image = QPixmap(":/new/prefix1/bg").toImage();
//实例 QPainter
QPainter painter(&image);
//设置画刷模式
painter.setCompositionMode(QPainter::CompositionMode_SourceIn);
//改变画笔和字体
QPen pen = painter.pen();
pen.setColor(Qt::red);
QFont font = painter.font();
font.setBold(true); //加粗
font.setPixelSize(50); //改变字体大小
painter.setPen(pen);
painter.setFont(font);
//将文字绘制在图片中心位置
painter.drawText(image.rect(), Qt::AlignCenter, "你好，朋友。");
//这个为图片的压缩度。0/100
int n = 100;
//保存图片
image.save("text.jpg", "JPG", n);
```

## 2.0 图像扭曲

### 1、实例参照

- 光盘/QtCode/QtSix/Qt10/Qt10.pro

### 2、实例实现

打开 MainWindow.h 文件编写如下代码。

```
protected:
    void paintEvent(QPaintEvent *);
```

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QPainter>
```

```
void MainWindow::paintEvent(QPaintEvent *)
{
    //实例 QPainter
    QPainter painter(this);
    //实例 QPixmap
    QPixmap pix;
    //加载图片
    pix.load(":/new/prefix1/bg");
    //原图显示
    painter.drawPixmap(0, 50, 183, 160, pix);
    //扭曲
    painter.shear(0.5, 0); //横向扭曲
    //绘制扭曲图
    painter.drawPixmap(190, 50, 183, 160, pix);
}
```

## 1.11 模糊效果

### 1、实例参照

- 光盘/QtCode/QtSix/Qt11/Qt11.pro

### 2、实例实现

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QLabel>
#include <QGraphicsBlurEffect>
```

```
//实例 QLabel
QLabel *label = new QLabel(this);
//QLabel 位置
label->setGeometry(QRect(50, 50, 254, 153));
//实例 QImage
QImage *img = new QImage;
//加载图片
img->load(":/new/prefix1/yl");
label->setPixmap(QPixmap::fromImage(*img));
//实例 QGraphicsBlurEffect
```

```
QGraphicsBlurEffect *effect = new QGraphicsBlurEffect(this);  
//模糊值，值越大越模糊  
effect->setBlurRadius(5);  
label->setGraphicsEffect(effect);
```

## 1.12 着色效果

### 1、实例参照

- 光盘/QtCode/QtSix/Qt12/Qt12.pro

### 2、实例实现

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QLabel>  
#include <QGraphicsColorizeEffect>
```

```
//实例 QLabel  
QLabel *label = new QLabel(this);  
//QLabel 位置  
label->setGeometry(QRect(50, 50, 254, 153));  
//实例 QImage  
QImage *img = new QImage;  
//加载图片  
img->load(":/new/prefix1/1.yu");  
label->setPixmap(QPixmap::fromImage(*img));  
//实例 QGraphicsColorizeEffect  
QGraphicsColorizeEffect *effect = new QGraphicsColorizeEffect(this);  
//设定颜色  
effect->setColor(QColor(0, 0, 192));  
label->setGraphicsEffect(effect);
```

## 1.13 阴影效果

### 1、实例参照

- 光盘/QtCode/QtSix/Qt13/Qt13.pro

### 2、实例实现

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QLabel>
#include <QGraphicsDropShadowEffect>
```

```
//实例 QLabel
QLabel *label = new QLabel(this);
//QLabel 位置
label->setGeometry(QRect(50, 50, 254, 153));
//实例 QImage
QImage *img = new QImage;
//加载图片
img->load(":/new/prefix1/you");
label->setPixmap(QPixmap::fromImage(*img));
//实例 QGraphicsDropShadowEffect
QGraphicsDropShadowEffect *effect = new QGraphicsDropShadowEffect(this);
//设定阴影
effect->setOffset(8, 8);
label->setGraphicsEffect(effect);
```

## 1.14 透明效果

### 1、实例参照

- 光盘/QtCode/QtSix/Qt14/Qt14.pro

### 2、实例实现

打开 MainWindow.cpp 文件编写如下代码。

```
#include <QLabel>
#include <QGraphicsOpacityEffect>
```

```
//实例 QLabel
QLabel *label = new QLabel(this);
//QLabel 位置
label->setGeometry(QRect(50, 50, 254, 153));
//实例 QImage
QImage *img = new QImage;
//加载图片
img->load(":/new/prefix1/you");
label->setPixmap(QPixmap::fromImage(*img));
```

```
//实例 QGraphicsOpacityEffect
QGraphicsOpacityEffect *effect = new QGraphicsOpacityEffect(this);
//设定透明值
effect->setOpacity(0.5);
label->setGraphicsEffect(effect);
```

## 第七章 多媒体应

---

# 用

---

## 1.1 音频、视频播放器

### 1、实例参照

- 光盘/QtCode/ QtSeven/Qt01/ Qt01.pro

### 2、实例实现

打开 Qt01.pro 工程文件编写如下代码。

```
CONFIG       += qaxcontainer
HEADERS      = playerwindow.h
```

打开 mainwindow.h 文件编写如下代码。

```
#include <QAxWidget>
#include <QToolButton>
#include <QSlider>
#include <QWidget>
```

```
//Q_ENUMS() 宏的参数是枚举类型。
Q_ENUMS(ReadyStateConstants)
```

```
//播放状态 枚举类型
enum PlayStateConstants { Stopped = 0, Paused = 1, Playing = 2 };
    enum ReadyStateConstants { Uninitialized = 0, Loading = 1,
                                Interactive = 3, Complete = 4 };

//播放进度
protected:
    void timerEvent(QTimerEvent *event);
```

```
private:
    QAxWidget *axWidget;//播放器
    QToolButton *openButton;//浏览按钮
    QToolButton *playPauseButton;//暂停播放按钮
    QToolButton *stopButton;//停止播放按钮
    QSlider *seekSlider;//进度条
    QString fileFilters;//文件格式
    int updateTimer;//播放进度
    QWidget *widget;//QWidget

private slots:
    void onPlayStateChange(int oldState, int newState);
    void onReadyStateChange(ReadyStateConstants readyState);
    void onPositionChange(double oldPos, double newPos);
    void sliderValueChanged(int newValue);
    void openFile();
```

打开 mainwindow.cpp 文件编写如下代码。



```
#include <QHBoxLayout>
#include <QVBoxLayout>
#include <QFileDialog>
```

```
//实例 QAxWidget (播放器插件)
axWidget = new QAxWidget;
//注册表键值, 调用插件
axWidget->setControl("{22D6F312-B0F6-11D0-94AB-0080C74C7E95}");
axWidget->setProperty("ShowControls", false);
axWidget->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Expanding);
connect(axWidget, SIGNAL(PlayStateChange(int, int)),
this, SLOT(onPlayStateChange(int, int)));
connect(axWidget, SIGNAL(ReadyStateChange(ReadyStateConstants)),
this, SLOT(onReadyStateChange(ReadyStateConstants)));
connect(axWidget, SIGNAL(PositionChange(double, double)),
this, SLOT(onPositionChange(double, double)));

//实例打开文件按钮
openButton = new QToolButton;
openButton->setText(tr("浏览.."));
connect(openButton, SIGNAL(clicked()), this, SLOT(openFile()));

//实例播放暂停按钮
playPauseButton = new QToolButton;
playPauseButton->setText(tr("播放"));
playPauseButton->setEnabled(false);
connect(playPauseButton, SIGNAL(clicked()), axWidget, SLOT(Play()));

//停止按钮
stopButton = new QToolButton;
stopButton->setText(tr("停止"));
stopButton->setEnabled(false);
connect(stopButton, SIGNAL(clicked()), axWidget, SLOT(Stop()));

//进度条
seekSlider = new QSlider(Qt::Horizontal);
seekSlider->setEnabled(false);
connect(seekSlider, SIGNAL(valueChanged(int)),
this, SLOT(sliderValueChanged(int)));
connect(seekSlider, SIGNAL(sliderPressed()),
axWidget, SLOT(Pause()));

//可播放格式, 此处简写下面几个
```

```
fileFilters = tr("Video files (*.mpg *.mpeg *.avi *.wmv *.mp4)\n"
"Audio files (*.mp3 *.wav)");
//初始化播放时间进度
updateTimer = 0;

//三个按钮横向布局
QHBoxLayout *buttonLayout = new QHBoxLayout;
buttonLayout->addWidget(openButton);
buttonLayout->addWidget(playPauseButton);
buttonLayout->addWidget(stopButton);

//纵向布局，上面播放器下面三个按钮
QVBoxLayout *mainLayout = new QVBoxLayout;
mainLayout->addWidget(axWidget);
mainLayout->addLayout(buttonLayout);
mainLayout->addWidget(seekSlider);

//显示布局控件
widget = new QWidget();
widget->setLayout(mainLayout);
this->setCentralWidget(widget);
```

```
//选择音视频文件
void MainWindow::openFile()
{
    //定义 QString 变量接收文件
    QString fileName = QFileDialog::getOpenFileName(this,
                                                    tr("Select File"), ".", fileFilters);

    //如果文件不为空，则播放
    if (!fileName.isEmpty())
        axWidget->setProperty("FileName",
                              QDir::toNativeSeparators(fileName));
}

//进度条随播放器移动
void MainWindow::timerEvent(QTimerEvent *event)
{
    if (event->timerId() == updateTimer) {
        double curPos = axWidget->property("CurrentPosition").toDouble();
        onPositionChange(-1, curPos);
    } else {
        QWidget::timerEvent(event);
    }
}
```

```
}

void MainWindow::onPlayStateChange(int, int newState)
{
    playPauseButton->disconnect(axWidget);

    switch (newState) {
    case Stopped:
        if (updateTimer) {
            killTimer(updateTimer);
            updateTimer = 0;
        }
    case Paused:
        connect(playPauseButton, SIGNAL(clicked()), axWidget, SLOT(Play()));
        stopButton->setEnabled(false);
        playPauseButton->setText(tr("播放"));
        break;
    case Playing:
        if (!updateTimer)
            updateTimer = startTimer(100);
        connect(playPauseButton, SIGNAL(clicked()),
                axWidget, SLOT(Pause()));
        stopButton->setEnabled(true);
        playPauseButton->setText(tr("暂停"));
    }
}

void MainWindow::onReadyStateChange(ReadyStateConstants ready)
{
    if (ready == Complete) {
        double duration = 60 * axWidget->property("Duration").toDouble();
        seekSlider->setMinimum(0);
        seekSlider->setMaximum(int(duration));
        seekSlider->setEnabled(true);
        playPauseButton->setEnabled(true);
    }
}

void MainWindow::onPositionChange(double, double newPos)
{
    seekSlider->blockSignals(true);
    seekSlider->setValue(int(newPos * 60));
    seekSlider->blockSignals(false);
}
```

```

}

void MainWindow::sliderValueChanged(int newValue)
{
    seekSlider->blockSignals(true);
    axWidget->setProperty("CurrentPosition", double(newValue) / 60);
    seekSlider->blockSignals(false);
}

```

## 1.2 播放 Flash 动画

### 1、实例参照

- 光盘/QtCode/ QtSeven/Qt02/ Qt02.pro

### 2、实例实现

打开 Qt02.pro 工程文件编写如下代码。

```
CONFIG+=qaxcontainer
```

打开 mainwindow.cpp 文件编写如下代码。

```

#include <QAxWidget>
#include <QWidget>
#include <QVBoxLayout>
#include <QDir>

```

```

//实例 QAxWidget
QAxWidget *flash = new QAxWidget(0,0);
//显示大小
flash->resize(460,370);    //设置该空间的初始大小
//设置注册表键值，调用 ActiveX 插件
flash->setControl(QString::fromUtf8("{d27cdb6e-ae6d-11cf-96b8-444553540000}"));
//设定控制器
//文件路径，注意如果发布可执行文件，播放文字路径需要拷贝一份
flash->dynamicCall("LoadMovie(long, string)",0,QDir::currentPath()
+ "/Qt02/file/map.swf");
//纵向布局
QVBoxLayout *layout = new QVBoxLayout;
//布局填充 QAxWidget
layout->addWidget(flash);

```

```
//实例 QWidget
QWidget *widget = new QWidget();
widget->setLayout(layout);
//界面显示
this->setCentralWidget(widget);
//界面标题为路径注意查看路径。
this->setWindowTitle(QDir::currentPath());
```

## 1.3 播放图片动画

### 1、实例参照

- 光盘/QtCode/ QtSeven/Qt03/ Qt03.pro

### 2、实例实现

打开 mainwindow.h 工程文件编写如下代码。

```
#include <QLabel>
#include <QPixmap>
#include <QTimer>
```

```
private:
    QPixmap *pm;
    QLabel *label;
    QTimer *timer;

private slots:
    void changeImg();
```

打开 mainwindow.cpp 工程文件编写如下代码。

```
//实例 QPixmap
pm = new QPixmap;
//加载第一张图片
pm->load(":/image/a0");
//实例 QLabel
label = new QLabel(this);
//位置
label->setGeometry(QRect(50, 50, 150, 150));
//Label 加载 QPixmap
label->setPixmap(*pm);
```

```
//实例 QTimer
timer = new QTimer;
//事件
connect(timer, SIGNAL(timeout()), this, SLOT(changeImg()));
//启动 QTimer
timer->start(100);
```

```
//初始值
int i = -1;

void MainWindow::changeImg()
{
    //i 不可大于 7，因为只准备了 8 张图片，i 从 0 开始。
    if(i < 7)
    {
        //每次 QTimer 执行 i+1;
        i++;
        //资源文件图片变量
        QString str = ":/image/a";
        //QDebug(str.toUtf8().append(QString::number(i, 10)));
        //加载图片路径
        pm->load(str.toUtf8().append(QString::number(i, 10)));
        //QLabel 显示图片
        label->setPixmap(*pm);
    }else
    {
        //循环一圈后，回到初始值，继续执行。
        i = -1;
    }
}
```

# 第八章 系统操作

---

---

## 1.1 获取屏幕分辨率

### 1、实例参照

- 光盘/QtCode/ QtEight /Qt01/ Qt01.pro

### 2、实例实现

打开 mainwindow.cpp 文件编写如下代码。

```
#include <QLabel>
#include <QDesktopWidget>
```

```
//实例 QLabel
QLabel *label = new QLabel(this);
//QLabel 位置
label->setGeometry(QRect(50, 50, 200, 25));
//实例 QDesktopWidget
QDesktopWidget *desktopWidget = QApplication::desktop();
//实例 QRect 接收屏幕信息
QRect screenRect = desktopWidget->screenGeometry();
//定义字符串
QString str = "屏幕分辨率为: ";
//屏幕宽度
int sWidth = screenRect.width();
//屏幕高度
int sHeight = screenRect.height();
//输出结果
label->setText(str+QString::number(sWidth, 10)+
"+QString::number(sHeight, 10));
```

X

## 1.2 获取本机名、IP 地址

### 1、实例参照

- 光盘/QtCode/ QtEight /Qt02/ Qt02.pro

### 2、实例实现

打开 mainwindow.cpp 文件编写如下代码。



```
#include <QDebug>
#include <QtNetwork/QHostInfo>
```

```
//获取计算机名称
QString localHostName = QHostInfo::localHostName();
QDebug() << "计算机名: " << localHostName;

//获取 IP 地址
QHostInfo info = QHostInfo::fromName(localHostName);
//遍历地址, 只获取 IPV4 地址
foreach(QHostAddress address, info.addresses())
{
    if(address.protocol() == QAbstractSocket::IPv4Protocol)
    {
        qDebug() << "ipV4 地址: " << address.toString();
    }
}
```

## 1.3 根据网址获取 IP 地址

### 1、实例参照

- 光盘/QtCode/ QtEight /Qt03/ Qt03.pro

### 2、实例实现

打开 mainwindow.h 文件编写如下代码。

```
#include <QtNetwork/QHostInfo>
#include <QLabel>
#include <QPushButton>
#include <QLineEdit>
```

```
private:
    QLabel *label;
    QPushButton *button;
    QLineEdit *edit;
    QLabel *result;
private slots:
    void sendUrl();
    void lookedUp(const QHostInfo &host);
```

打开 mainwindow.cpp 文件编写如下代码。

```
#include <QtNetwork/QHostInfo>
```

```
//实例 QLabel
label = new QLabel(this);
label->setGeometry(QRect(50, 50, 40, 25));
label->setText("网址: ");

//实例 QLineEdit
edit = new QLineEdit(this);
edit->setGeometry(QRect(100, 50, 150, 25));
edit->setText("www.baidu.com");

//实例 QPushButton
button = new QPushButton(this);
button->setGeometry(QRect(260, 50, 80, 25));
button->setText("查询");
connect(button, SIGNAL(clicked()), this, SLOT(sendUrl()));

//实例 QLabel
result = new QLabel(this);
result->setGeometry(QRect(50, 90, 150, 25));
```

```
void MainWindow::sendUrl()
{
    QHostInfo::lookupHost(edit->text(), this, SLOT(lookedUp(QHostInfo)));
}

void MainWindow::lookedUp(const QHostInfo &host)
{
    result->setText("IP 地址: "+host.addresses().first().toString());
}
```

## 1.4 判断键盘按下键值

### 1、实例参照

- 光盘/QtCode/ QtEight /Qt04/ Qt04.pro

## 2、实例实现

打开 mainwindow.h 文件编写如下代码。

```
#include <QLabel>
```

```
//键盘事件方法  
void keyPressEvent(QKeyEvent *event);
```

```
private:  
    QLabel *label;
```

打开 mainwindow.cpp 文件编写如下代码。

```
#include <QKeyEvent>
```

```
//实例 QLabel  
label = new QLabel(this);  
label->setGeometry(QRect(50, 50, 200, 25));  
label->setText("按 Q 键更改文字");
```

```
//键盘事件  
void MainWindow::keyPressEvent(QKeyEvent *event)  
{  
    //键值如果等于 Q  
    if(event->key() == Qt::Key_Q)  
    {  
        label->setText("你按了 Q 键");  
    }  
}
```

## 1.5 获取系统环境变量

### 1、实例参照

- 光盘/QtCode/ QtEight /Qt05/ Qt05.pro

### 2、实例实现

打开 mainwindow.cpp 文件编写如下代码。

```
#include <QListView>
#include <QStringListModel>
#include <QProcess>

//标题
this->setWindowTitle("获取 Path 环境变量");
//实例 QListView
QListView *listView = new QListView(this);
//QListView 位置
listView->setGeometry(QRect(10, 10, 380, 280));
//实例 QStringList 接收 path 系统变量
QStringList strList = QProcess::systemEnvironment();
//装载数据模型
QStringListModel *model = new QStringListModel(strList);
//绑定数据
listView->setModel(model);
```

## 1.6 执行系统命令

### 1、实例参照

- 光盘/QtCode/ QtEight /Qt06/ Qt06.pro

### 2、实例实现

打开 mainwindow.h 文件编写如下代码。

```
#include <QLineEdit>
#include <QPushButton>
#include <QPlainTextEdit>
#include <QProcess>
#include <QLabel>

private:
    QLineEdit *comm;    //dos 命令输入框
    QPlainTextEdit *outEdit;    //命令执行反馈框
    QPushButton *btClick;    //执行按钮
    QProcess *process;    //QProcess
    QString output;    //接收反馈信息
    QLabel *label;
```

```
private slots:
    void clickExecution(); //点击按钮事件
    void readOutput();    //QProcess 事件
```

打开 `mainwindow.cpp` 文件编写如下代码。

```
//实例命令输入对话框
comm = new QLineEdit(this);
comm->setText("ipconfig");
comm->setGeometry(QRect(20, 20, 260, 25));

//实例执行按钮
btClick = new QPushButton(this);
btClick->setText("执行");
btClick->setGeometry(QRect(290, 20, 80, 25));
connect(btClick, SIGNAL(clicked()), this, SLOT(clickExecution()));

//实例输出对话框
outEdit = new QTextEdit(this);
outEdit->setGeometry(QRect(20, 60, 350, 200));

//实例 QProcess
process = new QProcess;
connect(process, SIGNAL(readyRead()), this, SLOT(readOutput()));

//dos 命令查阅
label = new QLabel(this);
label->setGeometry(QRect(30, 265, 200, 25));
label->setText(tr("<a href='\"http://www.baidu.com/s?wd=dos 命令大全\"'>命令 DOS 查阅</a>"));
//开启超链接
label->setOpenExternalLinks(true);
```

```
//执行 DOS 命令
void MainWindow::clickExecution()
{
    //定义变量接收 dos 命令
    QString info = comm->text();
    //执行命令
    process->start(info);
    //绑定反馈值
    outEdit->setPlainText(output);
}
```

```
}  
  
//QProcess 事件  
void MainWindow::readOutput()  
{  
    //接收反馈信息  
    output +=process->readAll();  
    //将返回值绑定控件  
    outEdit->setPlainText(output);  
}
```

## 第九章 注册表

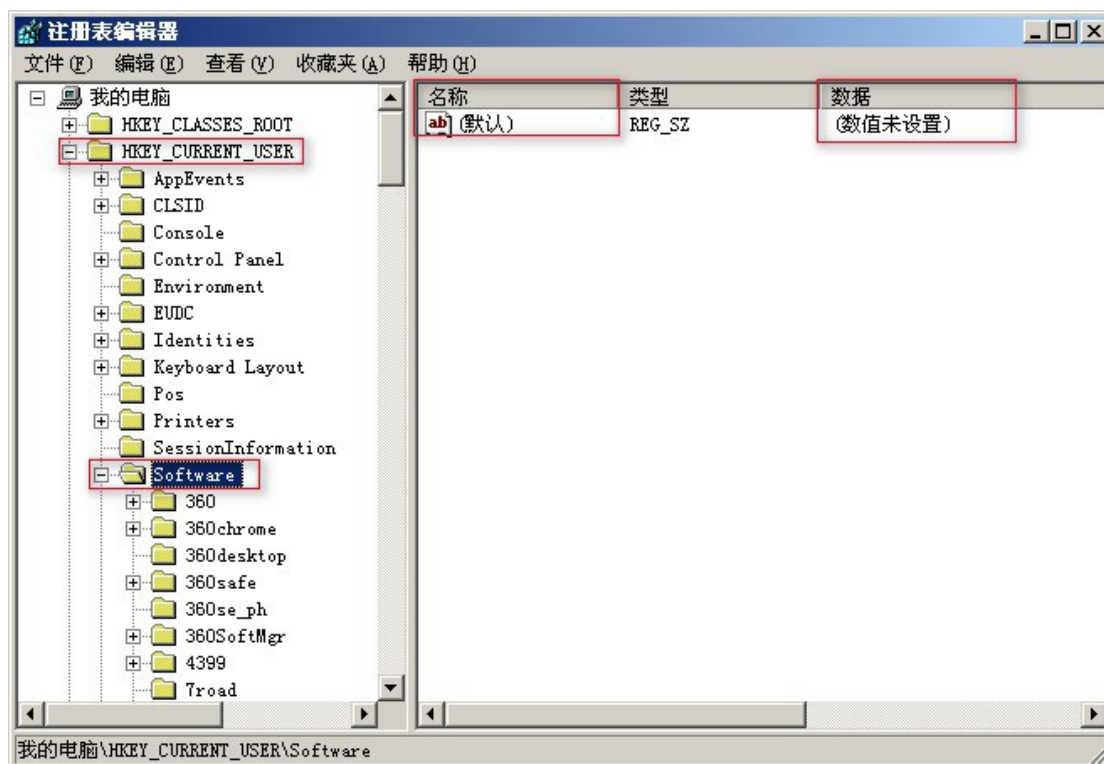
---

## 1.0 简要说明注册表

注册表实质上不是表，而是 windows 操作系统用于存储系统配置信息的核心文件。注册表中有大量数据，这些数据包括硬件设备的类型和配置参数、网络连接参数、软件版本安装、操作系统启动时的配置等软硬件信息。Windows 操作系统专门带有一个注册表编辑器，可以使用户查看和配置注册表。本章不使用 windows 自带注册表编辑器，而是 Qt 自带函数进行注册表操作。

注：因为注册表对系统的重要性，所以操作一定要谨慎，如果对注册表不熟悉，请谨慎操作。

注：打开系统注册表, 开始-运行- regedit-回车进入。如下图



这里我们简要操作 HKEY\_CURRENT\_USER--Software (这里主要存储系统软件信息)。

## 1.1 写入注册表

### 1、实例参照

- 光盘/QtCode/ QtNine /Qt01/ Qt01.pro

### 2、实例实现

打开 mainwindow.cpp 文件编写如下代码。

```
#include <QSettings>
```

```
//实例 QSettings  
//参数 1: 注册表位置  
//参数 2: 操作 windows 注册表 QSettings::NativeFormat  
//说明: QSettings::IniFormat 读写 ini 格式的配置文件, 前面用过。  
QSettings *reg = new QSettings("HKEY_CURRENT_USER\\Software\\Qt01",  
QSettings::NativeFormat);  
  
//设定值有修改, 没有创建。
```



```
reg->setValue("键名 001", "值 001");
reg->setValue("键名 002", true);
//用完删除 QSettings
delete reg;
```



注：程序之后，打开注册表查看，位置 `HKEY_CURRENT_USER\\Software\\Qt01`。

## 1.2 查找注册表

### 1、实例参照

- 光盘/QtCode/ QtNine /Qt02/ Qt02.pro

### 2、实例实现

打开 `mainwindow.cpp` 文件编写如下代码。

```
#include <QSettings>
#include <QLabel>
```

```
//输出键值
QLabel *label = new QLabel(this);
label->setGeometry(QRect(50, 50, 200, 25));

//实例 QSettings
//参数 1: 如果没有按照章节 Qt01 进行，则注册表中没有 Qt01。
QSettings *reg = new QSettings("HKEY_CURRENT_USER\\Software\\Qt01",
QSettings::NativeFormat);
//判断 value 是否为空，不为空则输出
if(reg->value("键名 001") != "")
{
label->setText("键名 001::"+reg->value("键名 001").toString());
}
//删除 QSettings
delete reg;
```

## 1.3 修改 IE 浏览器的默认主页

### 1、实例参照

- 光盘/QtCode/ QtNine /Qt03/ Qt03.pro

### 2、实例实现

说明：想要修改注册表中软件信息，首先需要找到该软件对应的键，例：IE 键路径如下：  
[HKEY\_CURRENT\_USER\Software\Microsoft\Internet Explorer\Main]子项，找到名称为  
“Start Page”的项。

打开 mainwindow.cpp 文件编写如下代码。

```
#include <QSettings>
```

```
//实例 QSettings
QSettings *reg = new
QSettings("HKEY_CURRENT_USER\\Software\\Microsoft\\Internet
Explorer\\Main",
QSettings::NativeFormat);
//判断 value 是否为空，不为空则输出
if(reg->value("Start Page") != "")
{
    //IE 默认主页修改为：百度首页
    reg->setValue("Start Page", "http://www.baidu.com");
}
//删除 QSettings
delete reg;
```

# 第十章 数据库基础

---

---

## 1.1 查询数据库驱动

### 1、实例参照

- 光盘/QtCode/ QtTen /Qt01/ Qt01.pro

## 2、实例实现

打开 Qt01.pro 工程文件编写如下代码。

```
QT += sql           //否则找不到
```

打开 mainwindow.cpp 文件编写如下代码。

```
#include <QtSql/QtSqlDatabase>
#include <QDebug>
#include <QStringList>
```

```
//实例 QStringList 接收驱动信息
QStringList drivers = QSqlDatabase::drivers();
//循环输出
foreach (QString driver, drivers) {
    qDebug() << "\r" << driver;
}
```

## 1.2 Qodbc 连接 Access 数据库

### 1、实例参照

- 光盘/QtCode/ QtTen /Qt02/ Qt02.pro

### 2、实例实现

注：操作 Access 数据库，只要系统装有 Office2003 标准安装，鼠标右键-新建-Microsoft Office Access 应用程序.mdb 即可，双击打开-新建 person 表。如下图：



保存之后将 db.mdb 数据库文件放到项目文件夹下。

打开 Qt01.pro 工程文件编写如下代码。

```
QT += sql           //否则找不到
```

打开 mainwindow.h 文件编写如下代码。

```
#include <QtSql/QtSqlDatabase>
```

```
QtSqlDatabase db;
//方法
bool openConnection();
```

打开 mainwindow.cpp 文件编写如下代码。

```
#include <QLabel>
#include <QtSql/QtSqlDatabase>
#include <QDir>
```

```
//实例 QLabel
QLabel *label = new QLabel(this);
label->setGeometry(QRect(50, 50, 200, 25));
//注意路径
this->setWindowTitle(QDir::currentPath()+"/Qt02/db.mdb");
//判断是否连接成功
```

```
if(openConnection())
{
    label->setText("连接成功");
}else
{
    label->setText("连接失败");
}
```

```
//数据库连接方法
bool MainWindow::openConnection()
{
    //实例 addDatabase
    db = QSqlDatabase::addDatabase("QODBC");
    //数据库驱动
    db.setDatabaseName("DRIVER={Microsoft Access Driver (*.mdb)};FIL={MS
Access};DBQ="+QDir::currentPath()+"/Qt02/db.mdb;UID='';PWD=''");
    //bool 型变量接收数据库是否连接成功
    bool isOk = db.open();
    if(isOk)
    {
        return true;
    }else
    {
        return false;
    }
}
```

## 1.3 插入数据

### 1、实例参照

- 光盘/QtCode/ QtTen /Qt03/ Qt03.pro

### 2、实例实现

打开 mainwindow.cpp 文件编写如下代码。

```
#include <QtSql/QSqlQuery>
```

```
//实例 QSqlQuery
QSqlQuery query;
```

```
//插入语句
query.prepare("insert into person(name) values (:name)");
//插入值
query.bindValue(":name", "张三");
//执行插入语句
if(query.exec())
{
    label->setText("数据插入成功");
} else
{
    label->setText("数据插入失败");
}
//关闭连接
db.close();
```

## 1.4 数据列表

### 1、实例参照

- 光盘/QtCode/ QtTen /Qt04/ Qt04.pro

### 2、实例实现

打开 mainwindow.cpp 文件编写如下代码。

```
#include <QTableView>
#include <QSqlTableModel>
```

```
//QTableView
QTableView *tableView = new QTableView(this);
tableView->setGeometry(QRect(50, 80, 310, 200));

//实例 QSqlTableModel
QSqlTableModel *model = new QSqlTableModel;
//为数据模型指定表
model->setTable("person");
//查询
model->select();
//填充模型
tableView->setModel(model);
//表格只读，默认单元格可修改
tableView->setEditTriggers(QAbstractItemView::NoEditTriggers);
```

## 1.5 操作 SQLite 数据库

### 1、实例参照

- 光盘/QtCode/ QtTen /Qt05/ Qt05.pro

### 2、实例实现

打开 Qt05.pro 工程文件编写如下代码。

```
QT += sql
```

打开 mainwindow.h 文件编写如下代码。

```
#include <QtSql/QtSqlDatabase>
```

```
//测试连接
bool connection();
//创建数据库
void createDB();
QtSqlDatabase db;
//绑定数据
void bindData();
```

打开 mainwindow.cpp 文件编写如下代码。

```
#include <QtSql/QtSqlDatabase>
#include <QLabel>
#include <QDir>
#include <QtSqlQuery>
#include <QTableView>
#include <QtSqlTableModel>
```

```
//实例 QLabel
QLabel *label = new QLabel(this);
label->setGeometry(QRect(50, 50, 200, 25));
//注意数据库路径
this->setWindowTitle(QDir::currentPath());

//判断数据库文件是否存在
bool dbFile = !QFile::exists(QDir::currentPath()+"/db.db");
```



```
if(connection())
{
//label->setText("连接成功");
//打开数据库，判断是否连接成功
if(dbFile)
{
    label->setText("不存在创建");
    //创建
    createDB();
    //绑定数据
    bindData();
} else
{
    //label->setText("存在");
    //绑定数据
    bindData();
}
//关闭数据库
db.close();
} else
{
    label->setText("连接失败");
}
```

```
//测试连接
bool MainWindow::connection()
{
    //实例 QSqlDatabase
    db = QSqlDatabase::addDatabase("QSQLITE");
    //数据库名称
    db.setDatabaseName("db.db");
    //判断是否可以打开
    if(!db.open())
    {
        return false;
    }
    return true;
}

//创建数据库
void MainWindow::createDB()
{
    //实例 QSqlQuery
```

```

    QSqlQuery query;
    //用 SQL 命令创建表，与 Access 一样
    //表名: person
    //id:自增，主键
    //name:存人名称
    //time: 时间自动添加
    query.exec("CREATE TABLE person (
        \"id INTEGER PRIMARY KEY AUTOINCREMENT, \"
        \"name VARCHAR(50) NOT NULL, \"
        \"time TIMESTAMP default CURRENT_TIMESTAMP)");

    //插入数据
    query.exec("INSERT INTO person (name) VALUES ('张三')");
    query.exec("INSERT INTO person (name) VALUES ('李四')");
    query.exec("INSERT INTO person (name) VALUES ('王五')");
}

//绑定数据
void MainWindow::bindData()
{
    //QTableView
    QTableView *tableView = new QTableView(this);
    tableView->setGeometry(QRect(50, 80, 310, 200));

    //实例 QSqlTableModel
    QSqlTableModel *model = new QSqlTableModel;
    //为数据模型指定表
    model->setTable("person");
    //查询
    model->select();
    //填充模型
    tableView->setModel(model);
    //表格只读，默认单元格可修改
    tableView->setEditTriggers(QAbstractItemView::NoEditTriggers);
}

```

## 1.6 SQLite 数据库视图管理器

本章节介绍一下作者在数据库应用这块常用的数据库视图管理器：

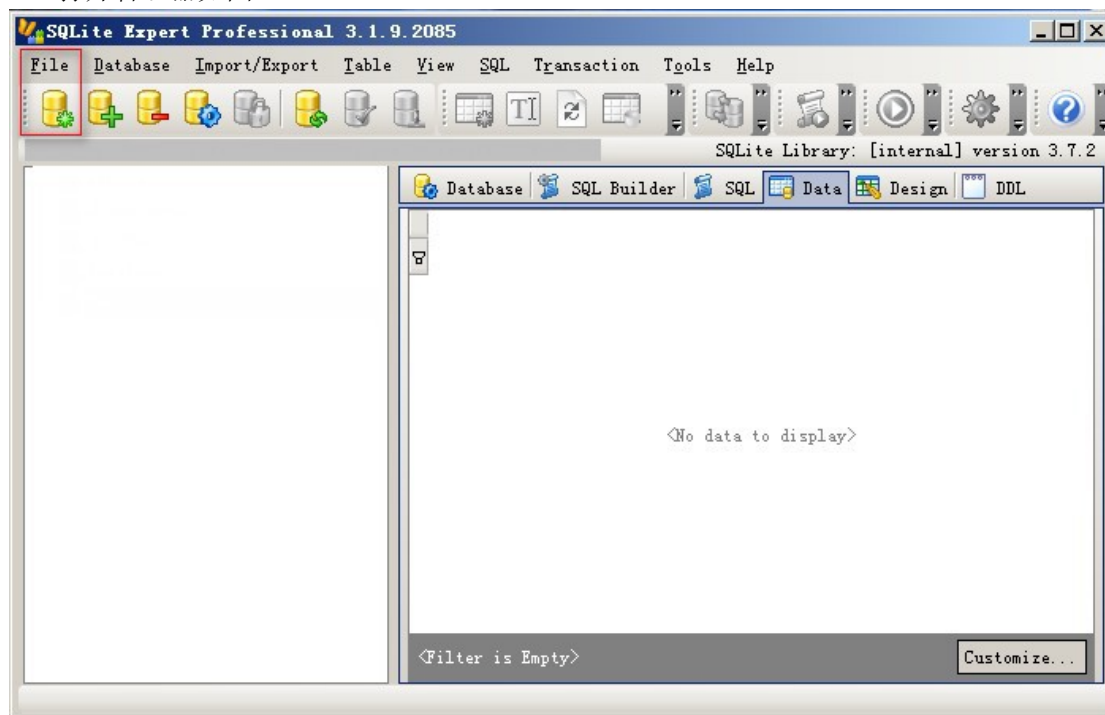
- 1、Oracle （PLSQL Developer 视图管理器）
- 2、SQLServer2000、2005、2008 以上都自带视图管理器
- 3、Access （Access 查询分析器）
- 4、Mysql（1、网页版视图管理，2、Navicat for Mysql）

## 5、SQLite (SQLite 视图管理器)

注：上述介绍数据库、视图管理器都可以在网上下载到，破解、注册就靠自己了。

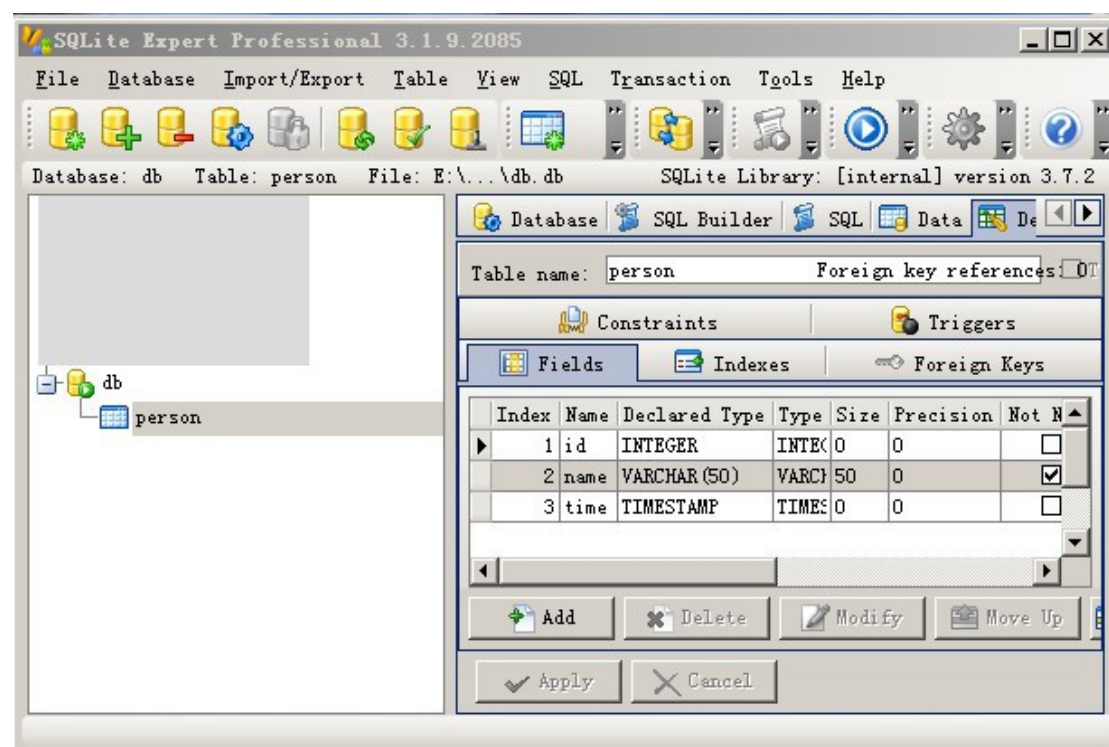
下面介绍一下 SQLite 视图管理器应用：

- 1、下载安装。
- 2、打开管理器如图。

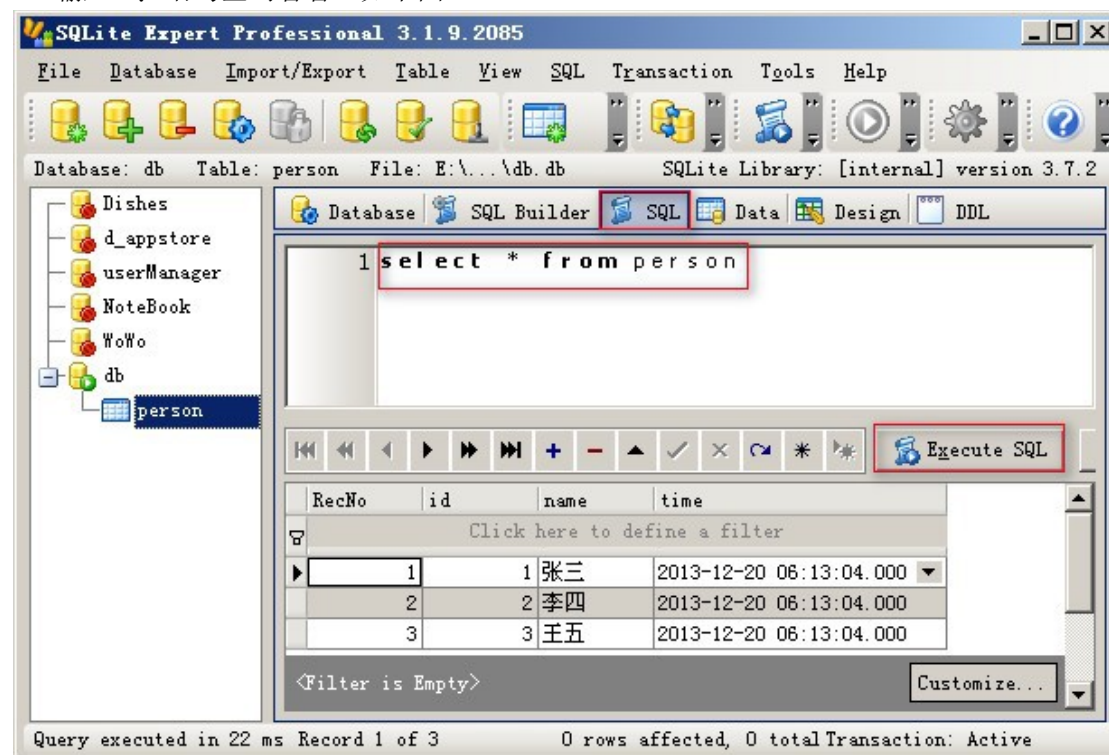


3、1.5 章节讲述的是通过程序创建数据库、表、字段，这里也可以通过管理器创建、添加数据，最后将数据库文件保存到对应的项目中也是可以的。

4、我们找到 1.5 章节中的数据库打开看看。



5、输入 SQL 语句查询看看，如下图。



# 第十一章 网络开发

---

## 1.1 点对点聊天服务端

### 1、实例参照

- 光盘/QtCode/ QtEleven /Qt01/ Qt01.pro

### 2、实例实现

打开 Qt01.pro 工程文件编写如下代码。

```
QT += network
```

打开 mainwindow.h 文件编写如下代码。

```
#include <QtNetwork>
#include <tcpserver.h>
```

```
private:
    QTcpServer *tcpServer;
```

```
int port;

private slots:
    void on_pushButton_clicked();
    void on_pushButton_2_clicked();
    void updateServer(QString, int);
```

打开 mainwindow.cpp 文件编写如下代码。

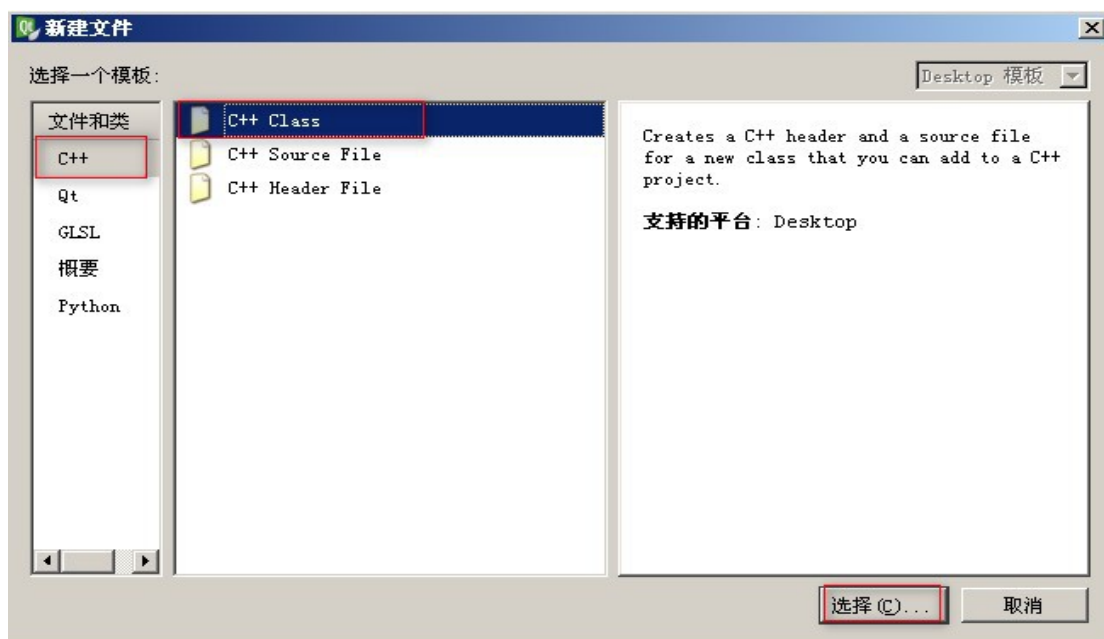
```
#include "tcpserver.h"
```

```
//关闭服务
void MainWindow::on_pushButton_2_clicked()
{
    QApplication->quit();
}

//开启服务
void MainWindow::on_pushButton_clicked()
{
    //协议端口号
    port = 8010;
    //实例 tcpServer
    tcpServer = new TcpServer(this, port);
    QObject::connect(tcpServer, SIGNAL(updateServer(QString, int)),
                     this, SLOT(updateServer(QString, int)));
    //开启服务后按钮不可点击
    ui->pushButton->setEnabled(false);
}

//监听
void MainWindow::updateServer(QString message, int length)
{
    ui->listWidget->addItem(message.left(length));
}
```

新建 tcpserver 类，项目右键-添加新文件如图：



打开 tcpserver.h 文件编写如下代码。

```
#include "tcpsocket.h"
#include <QtNetwork>
```

```
class TcpServer : public QTcpServer
{
    Q_OBJECT
public:
    explicit TcpServer(QObject *parent = 0, int port = 0);
    QList<TcpSocket*> tcpSocketList;
```



```
protected:
    void incomingConnection(int socketDescriptor);

signals:
    void updateServer(QString, int);
public slots:
    void updateClients(QString, int);
    void tcpDisconnected(int);
};
```

打开 tcpserver.cpp 文件编写如下代码。

```
#include <QtNetwork/QTcpSocket>
```

```
TcpServer::TcpServer(QObject *parent, int port) :
    QTcpServer(parent)
{
    listen(QHostAddress::Any, port);    //监听本机的 IP 地址和端口
}

void TcpServer::incomingConnection(int socketDescriptor)
{
    TcpSocket *tcpSocket = new TcpSocket(this);
    connect(tcpSocket, SIGNAL(updateClients(QString, int)),
            this, SLOT(updateClients(QString, int)));
    connect(tcpSocket, SIGNAL(disconnected(int)),
            this, SLOT(tcpDisconnected(int)));

    tcpSocket->setSocketDescriptor(socketDescriptor);
    tcpSocketList.append(tcpSocket);    //在 list 末尾插入数据
}

void TcpServer::updateClients(QString message, int length)
{
    emit updateServer(message, length);    //发射信号
    for(int i = 0; i < tcpSocketList.count(); i++)
    {
        QTcpSocket *temp = tcpSocketList.at(i);
        if(temp->write(message.toLatin1(), length) != length)
        {
            continue;
        }
    }
}
```

```

void TcpServer::tcpDisconnected(int descriptor)
{
    for(int i = 0; i < tcpSocketList.count(); i++)
    {
        QTcpSocket *temp = tcpSocketList.at(i);
        if(temp->socketDescriptor() == descriptor)
        {
            tcpSocketList.removeAt(i);
            return;
        }
    }
    return;
}

```

新建 tcpsocket 类，打开 tcpsocket.h 文件编写如下代码。

```

#include <QTcpSocket>

class TcpSocket : public QTcpSocket
{
    Q_OBJECT
public:
    TcpSocket(QObject *parent = 0);

signals:
    void updateClients(QString, int);
    void disconnected(int);
public slots:
    void dataReceived();
    void datadisconnected();
};

```

打开 tcpsocket.cpp 文件编写如下代码。

```

#include "tcpsocket.h"

TcpSocket::TcpSocket(QObject *parent) :
    QTcpSocket(parent)
{
    connect(this, SIGNAL(readyRead()),
            this, SLOT(dataReceived())); //准备接收
    connect(this, SIGNAL(disconnected()),

```

```

        this, SLOT(datadisconnected())); //断开连接信号
    }

void TcpSocket::dataReceived()
{
    while(this->bytesAvailable() > 0) //检查字节数
    {
        char buf[1024];
        int length = bytesAvailable();
        this->read(buf, length); //读取接收
        QString message = buf;
        emit updateClients(message, length); //发射信号
    }
}

void TcpSocket::datadisconnected()
{
    emit disconnected(this->socketDescriptor());
}

```

界面文件 mainwindow.ui 可以从 QtEleven/Qt01/Qt01/下拷贝，或者自己布局。

## 1.2 点对点聊天客户端

### 1、实例参照

- 光盘/QtCode/QtEleven/Qt02/Qt02.pro

### 2、实例实现

打开 Qt02.pro 工程文件编写如下代码。

```
QT += network
```

打开 mainwindow.h 文件编写如下代码。

```
#include <QtNetwork>
```

```

private:
    Ui::MainWindow *ui;

    QTcpSocket *tcpSocket;
    bool status;

```

```
    QString userName;

private slots:
    void on_pushButton_4_clicked();
    void on_pushButton_3_clicked();
    void on_pushButton_clicked();
    void on_pushButton_2_clicked();
    void tcpConnected();
    void tcpDisconnected();
    void dataReceived();
```

打开 mainwindow.cpp 文件编写如下代码。

```
//标题
this->setWindowTitle("客户端");

ui->pushButton->setDefault(true);
ui->pushButton->hide();
ui->pushButton_2->hide();
ui->pushButton_3->hide();
ui->lineEdit->hide();
ui->listWidget->hide();

ui->label->show();
ui->lineEdit_2->show();
ui->pushButton_4->show();
```

```
void MainWindow::on_pushButton_2_clicked()
{
    QString ipAddress = "127.0.0.1";
    int port = 8010;

    if(!status)
    {
        tcpSocket = new QTcpSocket(this);
        connect(tcpSocket, SIGNAL(connected()),
                this, SLOT(tcpConnected()));
        connect(tcpSocket, SIGNAL(disconnected()),
                this, SLOT(tcpDisconnected()));
        connect(tcpSocket, SIGNAL(readyRead()),
                this, SLOT(dataReceived()));
        tcpSocket->connectToHost(ipAddress, port);
    }
}
```

```
        status = true;
    }
    else
    {
        int length = 0;
        QString message = userName + tr(":Leave Chat Room");
        length = tcpSocket->write(message.toLatin1(), message.length());
        if(length != message.length())
        {
            return;
        }
        tcpSocket->disconnectFromHost();
        status = false;
    }
}

void MainWindow::tcpConnected()
{
    int length = 0;
    ui->pushButton_2->setText("退出连接");
    QString message = userName + tr(":Enter Chat Room");
    length = tcpSocket->write(message.toLatin1(), message.length());
    if(length != message.length())
    {
        return;
    }
}

void MainWindow::tcpDisconnected()
{
    ui->pushButton_2->setText("连接服务");
    ui->lineEdit_2->show();
    ui->label->show();
    ui->pushButton_4->show();

    ui->pushButton->hide();
    ui->pushButton_2->hide();
    ui->pushButton_3->hide();
    ui->lineEdit->hide();
    ui->listWidget->hide();
}

void MainWindow::dataReceived()
{
    while(tcpSocket->bytesAvailable() > 0)
```

```
{
    QByteArray datagram;
    datagram.resize(tcpSocket->bytesAvailable());

    tcpSocket->read(datagram.data(), datagram.length());
    QString message = datagram.data();
    ui->listWidget->addItem(message);
}
}

void MainWindow::on_pushButton_clicked()
{
    if(ui->lineEdit->text().isEmpty())
    {
        return;
    }
    QString message = userName + tr(":") + ui->lineEdit->text();
    tcpSocket->write(message.toLatin1(), message.length());
    ui->lineEdit->clear();
}

void MainWindow::on_pushButton_3_clicked()
{
    ui->listWidget->clear();
}

void MainWindow::on_pushButton_4_clicked()
{
    if(ui->lineEdit_2->text().isEmpty())
    {
        QMessageBox::warning(this, tr("warning!!"),
                               tr("please input you Name!"));
        return;
    }
    else
    {
        userName = ui->lineEdit_2->text();
        ui->lineEdit_2->hide();
        ui->label->hide();
        ui->pushButton_4->hide();

        ui->pushButton->show();
    }
}
```

```
    ui->pushButton_2->show();  
    ui->pushButton_3->show();  
    ui->lineEdit->show();  
    ui->listWidget->show();  
}  
}
```

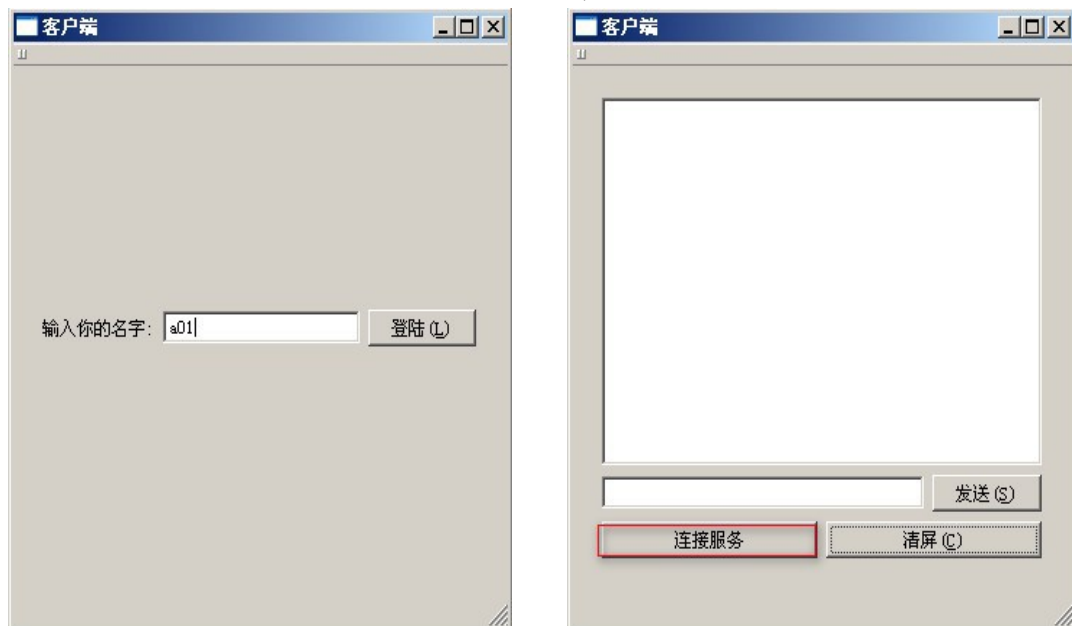
界面文件 mainwindow.ui 可以从 QtEleven/Qt02/ Qt02/下拷贝，或者自己布局。

服务端与客户端都做好之后，打开程序测试通信如下图。

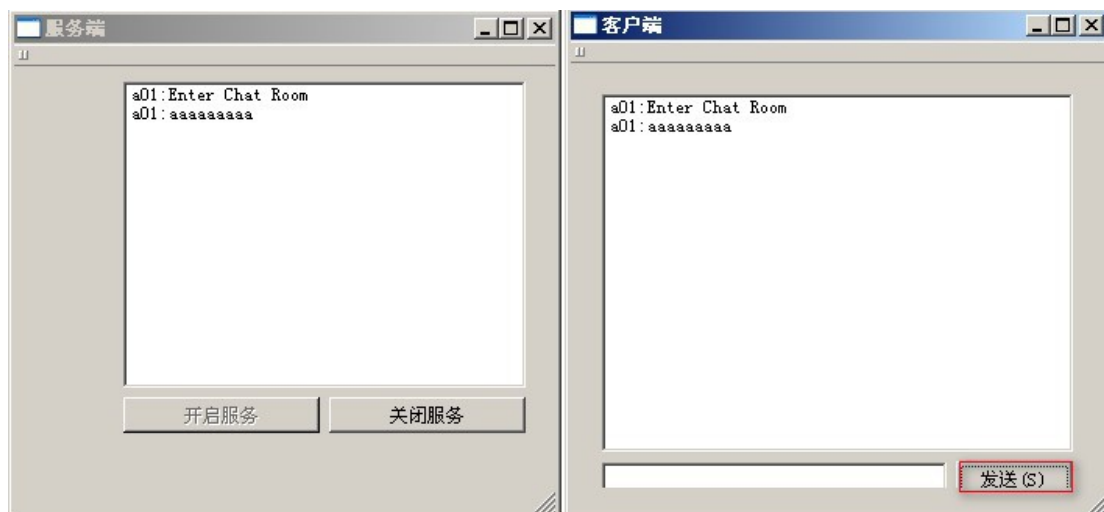
第一步：打开服务端程序-点击开启服务，开启服务后切忌不要关闭。



第二步：打开客户端程序-输入用户名作为标示, 点击登录。



第三步：客户端登录后，点击连接服务。



第四步：填写发送内容，点击发送，服务端就会收到客户端信息了。

## 1.3 局域网广播聊天

### 1、实例参照

- 光盘/QtCode/ QtEleven /Qt03/ Qt03.pro

### 2、实例实现

打开 Qt03.pro 工程文件编写如下代码。

```
QT += network
```

打开 widget.h 文件编写如下代码。

```
#include "tcpclient.h"
#include "tcpserver.h"
#include <QtNetwork>
#include <QTextCharFormat>
```

```
//枚举存储用户信息：ip、主机名、用户名...
enum MessageType{Message,NewParticipant,ParticipantLeft,FileName,Refuse};
```

```
QString getUsername();
QString getMessage();
```

```
protected:
```



```

void changeEvent(QEvent *e);
void sendMessage(MessageType type, QString serverAddress="");
void newParticipant(QString userName, QString localHostName, QString
ipAddress);
void participantLeft(QString userName, QString localHostName, QString time);
void closeEvent(QCloseEvent *);
void hasPendingFile(QString userName, QString serverAddress,
                    QString clientAddress, QString fileName);

    bool eventFilter(QObject *target, QEvent *event); //事件过滤器
private:
    Ui::Widget *ui;
    QUdpSocket *udpSocket;
    quint16 port;
    QString fileName;
    tcpserver *server;

    QString getIP();

    QColor color; //颜色
    bool saveFile(const QString& fileName); //保存聊天记录

private slots:
    void on_textUnderline_clicked(bool checked);
    void on_clear_clicked();
    void on_save_clicked();

    void on_textcolor_clicked();
    void on_textitalic_clicked(bool checked);
    void on_textbold_clicked(bool checked);
    void on_fontComboBox_currentFontChanged(QFont f);
    void on_fontsizecomboBox_currentIndexChanged(QString );
    void on_close_clicked();
    void on_sendfile_clicked();
    void on_send_clicked();
    void processPendingDatagrams();
    void sentFileName(QString);
    void currentFormatChanged(const QTextCharFormat &format);

```

打开 widget.cpp 文件编写如下代码。

```

#include <QMessageBox>
#include <QFileDialog>

```

```
#include <QKeyEvent>
#include <QScrollBar>
#include <QColorDialog>
```

```
this->resize(850, 550);
//输入框可获得焦点，比方说：回车发送消息
ui->textEdit->setFocusPolicy(Qt::StrongFocus);
//消息对话框关闭焦点事件
ui->textBrowser->setFocusPolicy(Qt::NoFocus);

ui->textEdit->setFocus();
ui->textEdit->installEventFilter(this);

udpSocket = new QUdpSocket(this);
port = 45454;
udpSocket->bind(port, QUdpSocket::ShareAddress
| QUdpSocket::ReuseAddressHint);
connect(udpSocket, SIGNAL(readyRead()), this, SLOT(processPendingDatagrams()));
sendMessage(NewParticipant);

server = new tcpserver(this);
connect(server, SIGNAL(sendFileName(QString)), this, SLOT(sentFileName(QString)));
connect(ui-
>textEdit, SIGNAL(currentCharFormatChanged(QTextCharFormat)), this, SLOT(currentFo
rmatChanged(const QTextCharFormat)));
```

```
void Widget::currentFormatChanged(const QTextCharFormat &format)
{
    //当编辑器的字体格式改变时，我们让部件状态也随之改变
    ui->fontComboBox->setCurrentFont(format.font());

    if(format.fontPointSize() < 9) //如果字体大小出错，因为我们最小的字体为9
        ui->fontsizecomboBox->setCurrentIndex(3); //即显示 12
    else ui->fontsizecomboBox->setCurrentIndex(
        ui->fontsizecomboBox-
>findText(QString::number(format.fontPointSize())));

    ui->textbold->setChecked(format.font().bold());
    ui->textitalic->setChecked(format.font().italic());
    ui->textUnderline->setChecked(format.font().underline());
    color = format.foreground().color();
}

void Widget::processPendingDatagrams() //接收数据 UDP
```

```
{
    while(udpSocket->hasPendingDatagrams())
    {
        QByteArray datagram;
        datagram.resize(udpSocket->pendingDatagramSize());
        udpSocket->readDatagram(datagram.data(), datagram.size());
        QDataStream in(&datagram, QIODevice::ReadOnly);
        int messageType;
        in >> messageType;
        QString userName, localHostName, ipAddress, message;
        QString time = QDateTime::currentDateTime().toString("yyyy-MM-dd
hh:mm:ss");
        switch(messageType)
        {
            case Message:
            {
                in >>userName >>localHostName >>ipAddress >>message;
                ui->textBrowser->setTextColor(Qt::blue);
                ui->textBrowser->setCurrentFont(QFont("Times New
Roman", 12));

                ui->textBrowser->append("[ " +userName+" ] "+ time);
                ui->textBrowser->append(message);
                break;
            }
            case NewParticipant:
            {
                in >>userName >>localHostName >>ipAddress;
                newParticipant(userName, localHostName, ipAddress);

                break;
            }
            case ParticipantLeft:
            {
                in >>userName >>localHostName;
                participantLeft(userName, localHostName, time);
                break;
            }
            case FileName:
            {
                in >>userName >>localHostName >> ipAddress;
                QString clientAddress, fileName;
                in >> clientAddress >> fileName;
                hasPendingFile(userName, ipAddress, clientAddress, fileName);
            }
        }
    }
}
```

```

        break;
    }
    case Refuse:
    {
        in >> userName >> localHostName;
        QString serverAddress;
        in >> serverAddress;
        QString ipAddress = getIP();

        if(ipAddress == serverAddress)
        {
            server->refused();
        }
        break;
    }
}
}

//处理新用户加入
void Widget::newParticipant(QString userName, QString localHostName, QString
ipAddress)
{
    bool bb = ui->tableWidget-
>findItems(localHostName, Qt::MatchExactly).isEmpty();
    if(bb)
    {
        QTableWidgetItem *user = new QTableWidgetItem(userName);
        QTableWidgetItem *host = new QTableWidgetItem(localHostName);
        QTableWidgetItem *ip = new QTableWidgetItem(ipAddress);
        ui->tableWidget->insertRow(0);
        ui->tableWidget->setItem(0,0,user);
        ui->tableWidget->setItem(0,1,host);
        ui->tableWidget->setItem(0,2,ip);
        ui->textBrowser->setTextColor(Qt::gray);
        ui->textBrowser->setCurrentFont(QFont("Times New Roman",10));
        ui->textBrowser->append(tr("%1 在线!").arg(userName));
        ui->onlineUser->setText(tr("在线人数: %1").arg(ui->tableWidget-
>rowCount()));
        sendMessage(NewParticipant);
    }
}

```

```
//处理用户离开
void Widget::participantLeft(QString userName, QString localHostName, QString
time)
{
    int rowNum = ui->tableWidget-
>findItems(localHostName, Qt::MatchExactly).first()->row();
    ui->tableWidget->removeRow(rowNum);
    ui->textBrowser->setTextColor(Qt::gray);
    ui->textBrowser->setCurrentFont(QFont("Times New Roman", 10));
    ui->textBrowser->append(tr("%1 于 %2 离开! ").arg(userName).arg(time));
    ui->onlineUser->setText(tr("在线人数: %1").arg(ui->tableWidget-
>rowCount()));
}

void Widget::changeEvent(QEvent *e)
{
    QWidget::changeEvent(e);
    switch (e->type()) {
    case QEvent::LanguageChange:
        ui->retranslateUi(this);
        break;
    default:
        break;
    }
}

QString Widget::getIP() //获取 ip 地址
{
    QList<QHostAddress> list = QNetworkInterface::allAddresses();
    foreach (QHostAddress address, list)
    {
        if(address.protocol() == QAbstractSocket::IPv4Protocol) //我们使用 IPv4
地址
            return address.toString();
    }
    return 0;
}

void Widget::sendMessage(MessageType type, QString serverAddress) //发送信息
{
    QByteArray data;
    QDataStream out(&data, QIODevice::WriteOnly);
    QString localHostName = QHostInfo::localHostName();
```

```
QString address = getIP();
out << type << getUsername() << localHostName;

switch(type)
{
    case ParticipantLeft:
    {
        break;
    }
    case NewParticipant:
    {
        out << address;
        break;
    }

    case Message :
    {
        if(ui->textEdit->toPlainText() == "")
        {
            QMessageBox::warning(0, tr("警告"), tr("发送内容不能为空"),
QMessageBox::Ok);
            return;
        }
        out << address << getMessage();
        ui->textBrowser->verticalScrollBar()->setValue(ui->textBrowser-
>verticalScrollBar()->maximum());
        break;
    }
    case FileName:
    {
        int row = ui->tableWidget->currentRow();
        QString clientAddress = ui->tableWidget->item(row, 2)->text();
        out << address << clientAddress << fileName;
        break;
    }
    case Refuse:
    {
        out << serverAddress;
        break;
    }
}
```

```
udpSocket->writeDatagram(data, data.length(), QHostAddress::Broadcast, port);
}

QString Widget::getUserName() //获取用户名
{
    QStringList envVariables;
    envVariables << "USERNAME.*" << "USER.*" << "USERDOMAIN.*"
                << "HOSTNAME.*" << "DOMAINNAME.*";
    QStringList environment = QProcess::systemEnvironment();
    foreach (QString string, envVariables)
    {
        int index = environment.indexOf(QRegExp(string));
        if (index != -1)
        {
            QStringList stringList = environment.at(index).split('=');
            if (stringList.size() == 2)
            {
                return stringList.at(1);
                break;
            }
        }
    }
    return false;
}

QString Widget::getMessage() //获得要发送的信息
{
    QString msg = ui->textEdit->toHtml();

    ui->textEdit->clear();
    ui->textEdit->setFocus();
    return msg;
}

void Widget::closeEvent(QCloseEvent *)
{
    sendMessage(ParticipantLeft);
}

void Widget::sentFileName(QString fileName)
{

```

```
this->fileName = fileName;
sendMessage(fileName);
}

void Widget::hasPendingFile(QString userName, QString serverAddress, //接收文件
                           QString clientAddress, QString fileName)
{
    QString ipAddress = getIP();
    if(ipAddress == clientAddress)
    {
        int btn = QMessageBox::information(this, tr("接受文件"),
                                           tr("来自%1(%2)的文件: %3, 是否接收? ")
                                           .arg(userName).arg(serverAddress).arg(fileName),
                                           QMessageBox::Yes, QMessageBox::No);
        if(btn == QMessageBox::Yes)
        {
            QString name = QFileDialog::getSaveFileName(0, tr("保存文件"), fileName);
            if(!name.isEmpty())
            {
                tcpclient *client = new tcpclient(this);
                client->setFileName(name);
                client->setHostAddress(QHostAddress(serverAddress));
                client->show();
            }
        }
        else{
            sendMessage(Refuse, serverAddress);
        }
    }
}

void Widget::on_send_clicked()//发送
{
    sendMessage(Message);
}

void Widget::on_sendfile_clicked()
{
    if(ui->tableWidget->selectedItems().isEmpty())
```



```
{
    QMessageBox::warning(0, tr("选择用户"), tr("请先从用户列表选择要传送的用户!"), QMessageBox::Ok);
    return;
}

server->show();
server->initServer();
}

void Widget::on_close_clicked() //关闭
{
    this->close();
}

bool Widget::eventFilter(QObject *target, QEvent *event)
{
    if(target == ui->textEdit)
    {
        if(event->type() == QEvent::KeyPress)
        {
            QKeyEvent *k = static_cast<QKeyEvent *>(event);
            if(k->key() == Qt::Key_Return)
            {
                on_send_clicked();
                return true;
            }
        }
    }
    return QWidget::eventFilter(target, event);
}

void Widget::on_fontComboBox_currentFontChanged(QFont f) //字体设置
{
    ui->textEdit->setCurrentFont(f);
    ui->textEdit->setFocus();
}

void Widget::on_fontsizecomboBox_currentIndexChanged(QString size)
{
    ui->textEdit->setFontPointSize(size.toDouble());
    ui->textEdit->setFocus();
}
```

```
void Widget::on_textbold_clicked(bool checked)
{
    if(checked)
        ui->textEdit->setFontWeight(QFont::Bold);
    else
        ui->textEdit->setFontWeight(QFont::Normal);
    ui->textEdit->setFocus();
}

void Widget::on_textitalic_clicked(bool checked)
{
    ui->textEdit->setFontItalic(checked);
    ui->textEdit->setFocus();
}

void Widget::on_textUnderline_clicked(bool checked)
{
    ui->textEdit->setFontUnderline(checked);
    ui->textEdit->setFocus();
}

void Widget::on_textcolor_clicked()
{
    color = QColorDialog::getColor(color, this);
    if(color.isValid())
    {
        ui->textEdit->setTextColor(color);
        ui->textEdit->setFocus();
    }
}

void Widget::on_save_clicked()//保存聊天记录
{
    if(ui->textBrowser->document()->isEmpty())
        QMessageBox::warning(0, tr("警告"), tr("聊天记录为空，无法保存！"),
        QMessageBox::Ok);
    else
    {
        //获得文件名
        QString fileName = QFileDialog::getSaveFileName(this, tr("保存聊天记录"), tr("聊天记录"), tr("文本(*.txt);;All File(*.*)"));
        if(!fileName.isEmpty())
            saveFile(fileName);
    }
}
```

```

    }
}

bool Widget::saveFile(const QString &fileName) //保存文件
{
    QFile file(fileName);
    if(!file.open(QFile::WriteOnly | QFile::Text))

    {
        QMessageBox::warning(this, tr("保存文件"),
            tr("无法保存文件 %1:\n %2").arg(fileName)
            .arg(file.errorString()));
        return false;
    }
    QTextStream out(&file);
    out << ui->textBrowser->toPlainText();

    return true;
}

void Widget::on_clear_clicked() //清空聊天记录
{
    ui->textBrowser->clear();
}

```

打开 tcpclient.h 文件编写如下代码。

```

#include <QTcpSocket>
#include <QHostAddress>
#include <QFile>
#include <QTime>

```

```

void setHostAddress(QHostAddress address);
void setFileName(QString fileName) {localFile = new QFile(fileName);}

```

```

protected:
    void changeEvent(QEvent *e);

private:
    Ui::tcpclient *ui;
    QTcpSocket *tcpClient;
    quint16 blockSize;

```

```

    QHostAddress hostAddress;
    quint16 tcpPort;

    quint64 TotalBytes;
    quint64 bytesReceived;
    quint64 bytesToReceive;
    quint64 fileNameSize;
    QString fileName;
    QFile *localFile;
    QByteArray inBlock;

    QTime time;

private slots:
    void on_tcpClientCancleBtn_clicked();
    void on_tcpClientCloseBtn_clicked();
    void newConnect();
    void readMessage();
    void displayError(QAbstractSocket::SocketError);

```

打开 tcpclient.cpp 文件编写如下代码。

```

#include <QFileDialog>
#include <QMessageBox>

```

```

this->setFixedSize(350, 180);
TotalBytes = 0;
bytesReceived = 0;
fileNameSize = 0;

tcpClient = new QTcpSocket(this);
tcpPort = 6666;
connect(tcpClient, SIGNAL(readyRead()), this, SLOT(readMessage()));
connect(tcpClient, SIGNAL(error(QAbstractSocket::SocketError)), this,
        SLOT(displayError(QAbstractSocket::SocketError)));

```

```

void tcpclient::changeEvent(QEvent *e)
{
    QDialog::changeEvent(e);
    switch (e->type()) {
    case QEvent::LanguageChange:
        ui->retranslateUi(this);

```

```
        break;
    default:
        break;
    }
}

void tcpclient::setHostAddress(QHostAddress address) //设置服务器地址并连接服务器
{
    hostAddress = address;
    newConnect();
}

void tcpclient::newConnect()
{
    blockSize = 0;
    tcpClient->abort();
    tcpClient->connectToHost(hostAddress, tcpPort);
    time.start();
}

void tcpclient::readMessage()
{
    QDataStream in(tcpClient);
    in.setVersion(QDataStream::Qt_4_6);

    float useTime = time.elapsed();
    if(bytesReceived <= sizeof(qint64)*2) {
        if((tcpClient->bytesAvailable() >= sizeof(qint64)*2) && (fileNameSize == 0)) {
            in>>TotalBytes>>fileNameSize;
            bytesReceived += sizeof(qint64)*2;
        }
        if((tcpClient->bytesAvailable() >= fileNameSize) && (fileNameSize != 0)) {
            in>>fileName;
            bytesReceived += fileNameSize;

            if(!localFile->open(QFile::WriteOnly)) {
                QMessageBox::warning(this, tr("应用程序"), tr("无法读取文件 %1:\n%2.").arg(fileName).arg(localFile->errorString()));
                return;
            }
        }
    }
}
```

```

        return;
    }
}

if(bytesReceived < TotalBytes){
    bytesReceived += tcpClient->bytesAvailable();
    inBlock = tcpClient->readAll();
    localFile->write(inBlock);
    inBlock.resize(0);
}

ui->progressBar->setMaximum(TotalBytes);
ui->progressBar->setValue(bytesReceived);
qDebug() << bytesReceived << "received" << TotalBytes;

double speed = bytesReceived / useTime;
ui->tcpClientStatusLabel->setText(tr("已接收 %1MB (%2MB/s) \n
共%3MB 已用时:%4秒\n 估计剩余时间: %5秒")
    .arg(bytesReceived /
(1024*1024))//已接收
    .arg(speed*1000/
(1024*1024), 0, 'f', 2)//速度
    .arg(TotalBytes / (1024 * 1024))//
总大小
    .arg(useTime/1000, 0, 'f', 0)//用时
    .arg(TotalBytes/speed/1000 -
useTime/1000, 0, 'f', 0));//剩余时间

if(bytesReceived == TotalBytes)
{
    tcpClient->close();
    ui->tcpClientStatusLabel->setText(tr("接收文件 %1 完
毕").arg(fileName));
    localFile->close(); //接收完文件后，一定要关闭，不然可能出问题
}
}

void tcpclient::displayError(QAbstractSocket::SocketError
socketError) //错误处理
{
    switch(socketError)
    {
        case QAbstractSocket::RemoteHostClosedError : break;
        default : qDebug() << tcpClient->errorString();
    }
}

```

```
}

void tcpclient::on_tcpClientCloseBtn_clicked() //关闭
{
    tcpClient->abort();
    this->close();
}

void tcpclient::on_tcpClientCancleBtn_clicked() //取消
{
    tcpClient->abort();
}
```

打开 tcpserver.h 文件编写如下代码。

```
#include <QTcpServer>
#include <QFile>
#include <QTime>
```

```
void refused();
void initServer();
```

```
protected:
    void changeEvent(QEvent *e);

private:
    Ui::tcpserver *ui;
    quint16 tcpPort;
    QTcpServer *tcpServer;
    QString fileName;
    QString theFileName;
    QFile *localFile;

    quint64 TotalBytes;
    quint64 bytesWritten;
    quint64 bytesToWrite;
    quint64 loadSize;
    QByteArray outBlock;

    QTcpSocket *clientConnection;

    QTime time;
```

```
private slots:
    void on_serverCloseBtn_clicked();
    void on_serverSendBtn_clicked();
    void on_serverOpenBtn_clicked();
    void sendMessage();

    void updateClientProgress(qint64 numBytes);
signals:
    void sendFileName(QString fileName);
```

打开 tcpserver.cpp 文件编写如下代码。

```
#include <QTcpSocket>
#include <QFileDialog>
#include <QMessageBox>
```

```
this->setFixedSize(350, 180);

tcpPort = 6666;
tcpServer = new QTcpServer(this);
connect(tcpServer, SIGNAL(newConnection()), this, SLOT(sendMessage()));

initServer();
```

```
void tcpserver::changeEvent(QEvent *e)
{
    QDialog::changeEvent(e);
    switch (e->type()) {
        case QEvent::LanguageChange:
            ui->retranslateUi(this);
            break;
        default:
            break;
    }
}

void tcpserver::sendMessage() //开始发送数据
{
    ui->serverSendBtn->setEnabled(false);
    clientConnection = tcpServer->nextPendingConnection();
```



```

connect(clientConnection, SIGNAL(bytesWritten(qint64)), SLOT(updateClientProgress(qint64)));

    ui->serverStatusLabel->setText(tr("开始传送文件 %1
! ").arg(fileName));

    localFile = new QFile(fileName);
    if(!localFile->open((QFile::ReadOnly))){//以只读方式打开
        QMessageBox::warning(this, tr("应用程序"), tr("无法读取文件
%1:\n%2").arg(fileName).arg(localFile->errorString()));
        return;
    }
    TotalBytes = localFile->size();
    QDataStream sendOut(&outBlock, QIODevice::WriteOnly);
    sendOut.setVersion(QDataStream::Qt_4_6);
    time.start(); //开始计时
    QString currentFile = fileName.right(fileName.size() -
fileName.lastIndexOf('/')-1);
    sendOut<<qint64(0)<<qint64(0)<<currentFile;
    TotalBytes += outBlock.size();
    sendOut.device()->seek(0);
    sendOut<<TotalBytes<<qint64((outBlock.size()-sizeof(qint64)*2));
    bytesToWrite = TotalBytes - clientConnection->write(outBlock);
    qDebug()<<currentFile<<TotalBytes;
    outBlock.resize(0);
}

void tcpserver::updateClientProgress(qint64 numBytes)//更新进度条
{
    bytesWritten += (int)numBytes;
    if(bytesToWrite > 0){
        outBlock = localFile->read(qMin(bytesToWrite, loadSize));
        bytesToWrite -= (int)clientConnection->write(outBlock);
        outBlock.resize(0);
    }
    else{
        localFile->close();
    }
    ui->progressBar->setMaximum(TotalBytes);
    ui->progressBar->setValue(bytesWritten);

    float useTime = time.elapsed();

```

```

        double speed = bytesWritten / useTime;
        ui->serverStatusLabel->setText(tr("已发送 %1MB (%2MB/s) \n 共%3MB  
已用时:%4 秒\n 估计剩余时间: %5 秒")
            .arg(bytesWritten / (1024*1024))//  
已发送  
            .arg(speed*1000/  
(1024*1024), 0, 'f', 2)//速度  
            .arg(TotalBytes / (1024 * 1024))//  
总大小  
            .arg(useTime/1000, 0, 'f', 0)//用时  
            .arg(TotalBytes/speed/1000 -  
useTime/1000, 0, 'f', 0));//剩余时间

        //num.printf("%.1f KB/s", (bytesWritten*1000) /  
(1024.0*time.elapsed()));
        if(bytesWritten == TotalBytes)
            ui->serverStatusLabel->setText(tr("传送文件 %1 成  
功").arg(theFileName));
    }

void tcpserver::on_serverOpenBtn_clicked() //打开
{
    fileName = QFileDialog::getOpenFileName(this);
    if(!fileName.isEmpty())
    {
        theFileName = fileName.right(fileName.size() -  
fileName.lastIndexOf('/')-1);
        ui->serverStatusLabel->setText(tr("要传送的文件为: %1  
").arg(theFileName));
        ui->serverSendBtn->setEnabled(true);
        ui->serverOpenBtn->setEnabled(false);
    }
}

void tcpserver::refused() //被对方拒绝
{
    tcpServer->close();
    ui->serverStatusLabel->setText(tr("对方拒绝接收!!!"));
}

void tcpserver::on_serverSendBtn_clicked() //发送
{

```

```
        if(!tcpServer->listen(QHostAddress::Any, tcpPort))//开始监听
        {
            qDebug() << tcpServer->errorString();
            close();
            return;
        }

        ui->serverStatusLabel->setText(tr("等待对方接收... ..."));
        emit sendFileName(theFileName);
    }

void tcpserver::on_serverCloseBtn_clicked()//退出
{
    if(tcpServer->isListening())
    {
        tcpServer->close();
        clientConnection->abort();
    }
    this->close();
}

void tcpserver::initServer()//初始化
{
    loadSize = 4*1024;
    TotalBytes = 0;
    bytesWritten = 0;
    bytesToWrite = 0;

    ui->serverStatusLabel->setText(tr("请选择要传送的文件"));
    ui->progressBar->reset();
    ui->serverOpenBtn->setEnabled(true);
    ui->serverSendBtn->setEnabled(false);

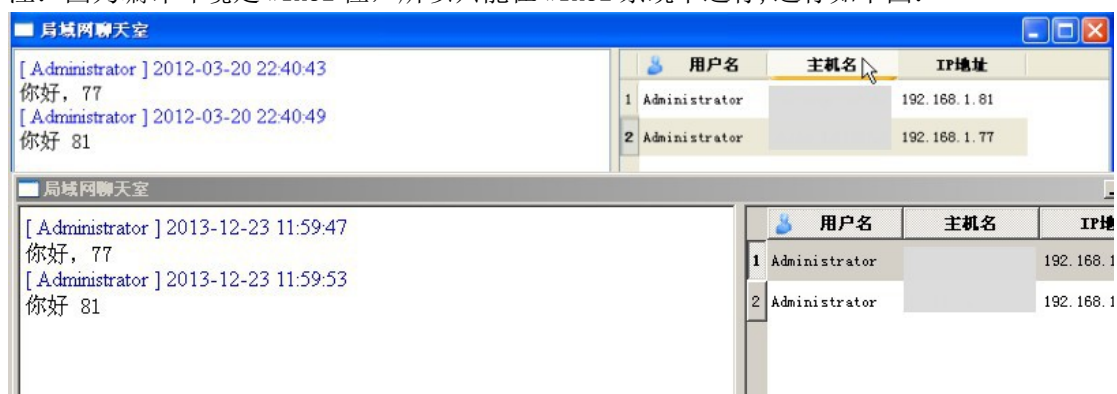
    tcpServer->close();
}
```

界面文件可以从 QtEleven/Qt03/ Qt03/下拷贝。

- 1、tcpclient.ui
- 2、tcpserver.ui
- 3、widget.ui

演示说明：打开 QtEleven/Qt03/ Qt03/局域网聊天.exe，这个是编译好的，可将其拷贝到局域网的其他机器，打开测试聊天。

注：因为编译环境是 Win32 位，所以只能在 Win32 系统下运行，运行如下图：



## 1.4 SMTP 协议发送邮件

### 1、实例参照

- 光盘/QtCode/ QtEleven /Qt04/ Qt04.pro

### 2、实例实现

打开 Qt04.pro 工程文件编写如下代码。

```
QT += network
```

打开 widget.h 文件编写如下代码。

```
#include <QLabel>
#include <QLineEdit>
#include <QPlainTextEdit>
#include <QPushButton>
#include <QGridLayout>
#include <QTcpSocket>
```

```
private:
    QLabel *shouLabel;
    QLabel *faLabel;
    QLabel *pwdLabel;
    QLabel *zhuLabel;
    QLabel *cLabel;
    QLineEdit *shouEdit;
    QLineEdit *faEdit;
```

```

    QLineEdit *pwdEdit;
    QLineEdit *zhuEdit;
    QPlainTextEdit *cEdit;
    QPushButton *faButton;
    QGridLayout *gLayout;
private slots:
    void faClick();
    void readWelcome(QTcpSocket & socket);
    void smtpCommunication(QTcpSocket & socket);
    void communication(QTcpSocket & socket, const char *msg);

```

打开 widget.cpp 文件编写如下代码。

```

#include <QMessageBox>
#include <QTcpSocket>

```

```

//实例 QGridLayout
gLayout = new QGridLayout();
shouLabel = new QLabel("收件人地址: ");
faLabel = new QLabel("发件人地址: ");
pwdLabel = new QLabel("QQ 邮箱密码: ");
zhuLabel = new QLabel("邮件主题: ");
cLabel = new QLabel("邮件内容: ");
//收件人
shouEdit = new QLineEdit();
//发件人
faEdit = new QLineEdit();
//密码
pwdEdit = new QLineEdit();
//输入状态*
pwdEdit->setEchoMode(QLineEdit::Password);
//主题
zhuEdit = new QLineEdit();
//内容
cEdit = new QPlainTextEdit();
//发送按钮
faButton = new QPushButton("发送邮件");
connect(faButton, SIGNAL(clicked()), this, SLOT(faClick()));

gLayout->addWidget(shouLabel, 0, 0, 1, 1);
gLayout->addWidget(shouEdit, 0, 1, 1, 1);
gLayout->addWidget(faLabel, 1, 0, 1, 1);

```

```
gLayout->addWidget(faEdit, 1, 1, 1, 1);
gLayout->addWidget(pwdLabel, 2, 0, 1, 1);
gLayout->addWidget(pwdEdit, 2, 1, 1, 1);
gLayout->addWidget(zhuLabel, 3, 0, 1, 1);
gLayout->addWidget(zhuEdit, 3, 1, 1, 1);
gLayout->addWidget(cLabel, 4, 0, 1, 1);
gLayout->addWidget(cEdit, 4, 1, 1, 1);
gLayout->addWidget(faButton, 5, 1, 1, 1);
```

```
this->setLayout(gLayout);
```

```
shouEdit->setText("xxx@126.com");
```

```
faEdit->setText("xxx@qq.com");
```

```
//点击发送邮件
```

```
void Widget::faClick()
```

```
{
    //简单判断一个不可为空
    if(shouEdit->text() == "")
    {
        QMessageBox::warning(this, "警告", tr("都不可以为空!!!"));
        return;
    }
    //实例QTcpSocket Mail 域名
    QTcpSocket socket;
    socket->connectToHost("smtp.qq.com", 25);
    if(socket->waitForConnected(2000))
    {
        qDebug() << "smtp 服务连接成功";
        readWelcome(socket);
        smtpCommunication(socket);
        socket->close();
    }else
    {
        qDebug() << "smtp 服务连接失败";
    }
}
```

```
void Widget::readWelcome(QTcpSocket & socket)
```

```
{
    char data[1024];
    if (socket->waitForReadyRead(-1) == true)
```

```

    {
        memset(data, '\0', sizeof(data));
        qDebug() << data << endl;;
    }
}

void Widget::smtpCommunication(QTcpSocket & socket)
{
    communication(socket, "helo qq.com\r\n");
    communication(socket, "auth login\r\n");
    //发件人邮箱
    communication(socket, QByteArray(faEdit->text().toUtf8()).toBase64()
+ "\r\n");
    //发件人邮箱密码
    communication(socket, QByteArray(pwdEdit->text().toUtf8()).toBase64()
+ "\r\n");
    //发件人邮箱
    communication(socket, "mail from: <" + faEdit->text().toUtf8() + ">\r\n");
    qDebug() << "mail from: <" + faEdit->text().toUtf8() + ">\r\n";
    //收件人邮箱
    communication(socket, "rcpt to: <" + shouEdit->text().toUtf8() + ">\r\n");
    //发送数据
    communication(socket, "data\r\n");
    //发送来源、主题、内容
    communication(socket, "From: " + faEdit->text().toUtf8() + "\r\nTo:
" + shouEdit->text().toUtf8() + "\r\n"
        "Subject: " + zhuEdit->text().toUtf8() + "\r\n\r\n"
        + cEdit->toPlainText().toUtf8() +
        "\r\n. \r\n");
    communication(socket, "quit\r\n");
    //qDebug() << "send email ok." << endl;
    QMessageBox::warning(this, "警告", tr("邮件发送成功!!!"));
}

void Widget::communication(QTcpSocket & socket, const char *msg)
{
    char data[1024];

    if (socket.write(msg, qstrlen(msg)) == -1)
        qDebug() << "@@@@@@@@@@@@ socket.write failed";
    socket.flush();

    if (socket.waitForReadyRead(-1) == true)

```

```

    {
        memset(data, '\0', sizeof(data));
        socket.readLine(data, 1024);
        qDebug() << data;
    }
}

```

注：使用 SMTP 协议发送邮件需要通过所在邮箱开启这项协议才可以使用，如上述例子通过 QQ 邮箱发给 126 邮箱，需要 QQ 邮箱开启 SMTP 协议，进入 QQ 邮箱-设置-账户-账户安全-POP3/SMTP 服务-点击开启即可。

## 1.5 调用系统 DLL 判断网络连接状态

### 1、实例参照

- 光盘/QtCode/ QtEleven /Qt05/ Qt05.pro

### 2、实例实现

注：Win32 网络连接 dll 文件名叫：wininet.dll，位置在 C:\WINDOWS\system32 目录下，将其拷贝到项目工程下。

打开 mainwindow.cpp 文件编写如下代码。

```

#include <QLibrary>
#include <QLabel>

//拨号
#define INTERNET_CONNECTION_MODEM 1
//局域网
#define INTERNET_CONNECTION_LAN 2
//代理上网
#define INTERNET_CONNECTION_PROXY 4
//代理被占用
#define INTERNET_CONNECTION_MODEM_BUSY 8
//定义函数指针
typedef bool(*ConnectFun)(int* lpdwFlags, int dwReserved);

```

```

//实例 QLabel 输出信息用
QLabel *label = new QLabel(this);
label->setGeometry(QRect(50, 50, 200, 25));

```



```
QLibrary myLib("wininet.dll");
//判断是否正确加载
if(myLib.load())
{
    bool bOnline=false;//是否在线
    int flags;
    //获取 dll 库中的函数 InternetGetConnectedState 函数地址
    ConnectFun myConnect=(ConnectFun)myLib.resolve("InternetGetConnectedState");
    //判断是否连网
    bOnline = myConnect(&flags, 0);
    if(bOnline)
    {
        if(flags & INTERNET_CONNECTION_MODEM)
        {
            label->setText("已连接-连接模式为：拨号上网");
        }
        else if(flags & INTERNET_CONNECTION_LAN)
        {
            label->setText("已连接-连接模式为：局域网");
        }
        else if(flags & INTERNET_CONNECTION_PROXY)
        {
            label->setText("已连接-连接模式为：代理上网");
        }
        else
        {
            label->setText("连接失败!!!");
        }
    }
    else
    {
        label->setText("没有网络连接");
    }
}
else
{
    label->setText("DLL 加载失败!!!");
}
```

# 第十二章 进程与线程

---

## 1.1 进程管理器

### 1、实例参照

- 光盘/QtCode/ QtTwelve/Qt01/ Qt01.pro

### 2、实例实现

打开 mainwindow.h 文件编写如下代码。

```
#include <QTableView>
#include <QStandardItemModel>

private:
    QTableView *tableView;
    QStandardItemModel *model;
    QString cPid;

private slots:
    //QTableView 行事件
    void sendContent(QModelIndex);
    //点击删除事件
    void deletePro();
```

打开 mainwindow.cpp 文件编写如下代码。

```
#include <QDebug>
#include "windows.h"    //1 顺序不可颠倒
#include "tlhelp32.h"    //2
#include <QPushButton>
#include <QMessageBox>
#include <QProcess>
```

```
//实例 QListView
tableView = new QTableView(this);
//QListView 位置
tableView->setGeometry(QRect(10, 10, 340, 430));
//实例 QPushButton
QPushButton *button = new QPushButton(this);
button->setGeometry(QRect(260, 450, 80, 25));
button->setText("结束进程");
connect(button, SIGNAL(clicked()), this, SLOT(deletePro()));
//实例数据模型
model = new QStandardItemModel();
model->setHorizontalHeaderItem(0, new QStandardItem("进程名"));
model->setHorizontalHeaderItem(1, new QStandardItem("PID"));

//获取系统快照句柄，可以获取进程、线程、模块、进程使用的堆的句柄
HANDLE hProcessSnap = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
if (hProcessSnap == FALSE )
{
    qDebug() << "调用失败";
    return;
}
PROCESSENTRY32 pe32;
pe32.dwSize = sizeof(PROCESSENTRY32);
BOOL bRet = Process32First(hProcessSnap, &pe32);
int i = 0;
while (bRet)
{
    i++;
    //前面 2 行不要
    if(i > 1)
    {
        //获取进程名
        QString pName = QString::fromWCharArray(pe32.szExeFile);
        //追加数据模型第一列
        model->setItem(i-2, 0, new QStandardItem(pName));
        //获取 PID 号
        QString pid = QString::number(pe32.th32ProcessID);
        //追加数据模型第二列
        model->setItem(i-2, 1, new QStandardItem(pid));
        bRet = Process32Next(hProcessSnap, &pe32);
    }
}
//清理 hProcessSnap
::CloseHandle(hProcessSnap);
```

```

//绑定数据
tableView->setModel(model);
//列宽
tableView->setColumnWidth(0, 190);
tableView->setColumnWidth(1, 130);
//选取整行
tableView->setSelectionBehavior(QAbstractItemView::SelectRows);
//不可修改
tableView->setEditTriggers(QAbstractItemView::NoEditTriggers);
//行单击事件
connect(tableView, SIGNAL(clicked(QModelIndex)), SLOT(sendContent(QModelIndex)));
//纵向头隐藏
tableView->verticalHeader()->setVisible(false);
//关闭滚动条
//tableView->setVerticalScrollBarPolicy(Qt::ScrollBarAlwaysOff);
//隔行变色
tableView->setAlternatingRowColors(true);

```

```

//QTableView 点击行事件
void MainWindow::sendContent(QModelIndex)
{
    //获取点击行索引
    int index = tableView->currentIndex().row();
    //将PID 值赋予 QString 类型保存
    cPid = model->data(model->index(index, 1)).toString();
}

//点击结束进程
void MainWindow::deletePro()
{
    if(cPid == "")
    {
        QMessageBox::warning(this, "警告", tr("请选择要结束的进程!!!"));
    }else
    {
        //Taskkill/pid 3184
        //结束进程命令
        QProcess process(0);
        QString dos = "Taskkill";
        QStringList list;
        list.append("/pid");
        list.append(cPid);
    }
}

```

```
//拼接参数，因为QProcess 不支持中文与空格
process.start(dos, list);
QMessageBox::warning(this, "警告", tr("成功结束!!!"));
}
}
```

## 1.2 线程 QThread 应用

### 1、实例参照

- 光盘/QtCode/ QtTwelve/Qt02/ Qt02.pro

### 2、实例实现

打开 mainwindow.h 文件编写如下代码。

```
#include <QPushButton>
#include <QThread>
```

```
private:
    QPushButton *starThread;
    QThread *thread;
private slots:
    //执行线程
    void startFun();
```

打开 mainwindow.cpp 文件编写如下代码。

```
#include <QDebug>
```

```
//执行线程 1
starThread = new QPushButton(this);
starThread->setText("执行线程 1");
starThread->setGeometry(QRect(30, 50, 80, 25));
connect(starThread, SIGNAL(clicked()), this, SLOT(startFun()));
```

```
void MainWindow::startFun()
{
    //实例线程
    thread = new QThread();
    //启动
    thread->start();
    int i=0;
    while(true)
    {
        //挂起1秒
        thread->sleep(1);
        i++;
        qDebug() << i;
        if(i > 2)
        {
            break;
        }
    }
    qDebug() << "结束";
}
```

### 1.3 线程 QRunnable 应用

#### 1、实例参照

- 光盘/QtCode/ QtTwelve/Qt03/ Qt03.pro

#### 2、实例实现

打开 mainwindow.h 文件编写如下代码。

```
#include <QPushButton>
#include <QProgressBar>
```

```
private:
    QPushButton *startButton;
    QProgressBar *progressBar;
    QProgressBar *progressBar2;

private slots:
    void startFun();
```

打开 mainwindow.cpp 文件编写如下代码。

```
#include <QProgressBar>
#include <QThreadPool>
#include <runnable.h>
```

```
//第一个 QProgressBar
progressBar = new QProgressBar(this);
progressBar->setGeometry(QRect(50, 50, 200, 16));
//初始值
progressBar->setValue(0);
//范围值
progressBar->setRange(0, 100000-1);

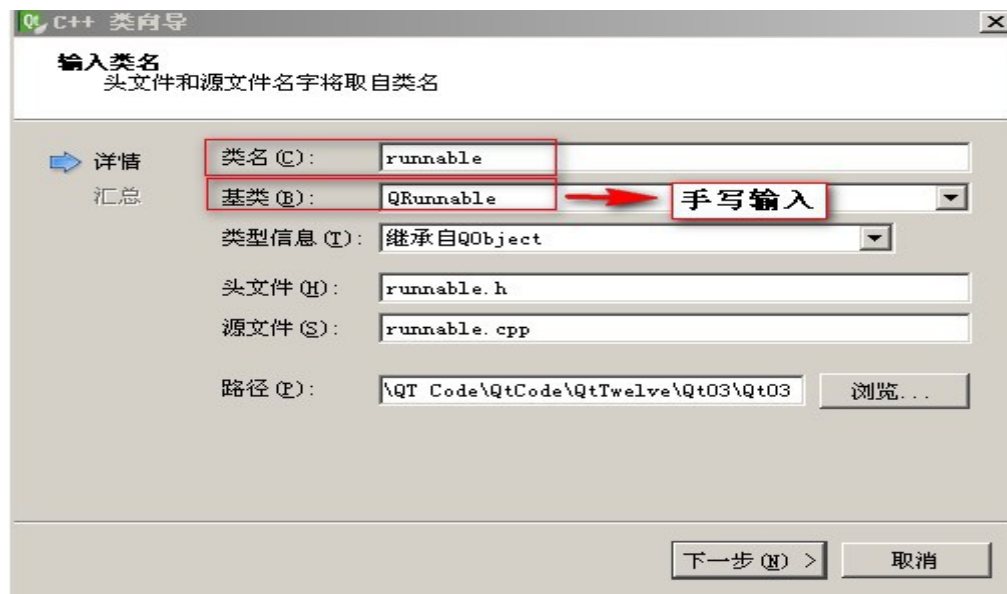
//第二个 QProgressBar
progressBar2 = new QProgressBar(this);
progressBar2->setGeometry(QRect(50, 100, 200, 16));
//初始值
progressBar2->setValue(0);
//范围值
progressBar2->setRange(0, 100000-1);

//实例按钮
startButton = new QPushButton(this);
startButton->setGeometry(QRect(260, 45, 80, 25));
startButton->setText("执行");
connect(startButton, SIGNAL(clicked()), this, SLOT(startFun()));
```

```
//开始执行
void MainWindow::startFun()
{
    //实例 runnable 类
    runnable *hInst = new runnable(progressBar);
    //实例第二个
    hInst = new runnable(progressBar2);
    //进程池
    QThreadPool::globalInstance()->start(hInst);
}
```

新建 runnable 类文件继承自 QRunnable，如下图。





打开 runnable.h 文件编写如下代码。

```
#include <QRunnable>
#include <QProgressBar>

class runnable : public QRunnable
{
public:
    runnable(QProgressBar *progressBar);
    virtual ~runnable();
    void run();
private:
    QProgressBar *m_ProgressBar;
};
```

打开 runnable.cpp 文件编写如下代码。

```
#include <QProgressBar>

runnable::runnable(QProgressBar *progressBar):
    QRunnable(), m_ProgressBar(progressBar)
{
    for(int i=0;i<100000;i++)
    {
        progressBar->setValue(i);
    }
}

runnable::~~runnable() {}
```

```
void runnable::run() {}
```

# 第十三章 数据安 全

---

---

## 1.1 QByteArray 加密数据

### 1、实例参照

- 光盘/QtCode/ QtThirteen/Qt01/ Qt01.pro

### 2、实例实现

打开 mainwindow.h 文件编写如下代码。

```
#include <QLabel>
#include <QPushButton>
#include <QLineEdit>
#include <QPlainTextEdit>
```

```
private:
    QLabel *label;
    QPushButton *btJ;
    QPushButton *btU;
    QLineEdit *edit;
    QPlainTextEdit *pEdit;
```

```
    QByteArray str;
private slots:
    void encFun();
    void jieFun();
```

打开 mainwindow.cpp 文件编写如下代码。

```
//标题
this->setWindowTitle("QByteArray 加密");

//布局
label = new QLabel(this);
label->setGeometry(QRect(30, 50, 100, 25));
label->setText("加密内容");

edit = new QLineEdit(this);
edit->setGeometry(QRect(90, 50, 290, 25));
edit->setText("你好");

pEdit = new QTextEdit(this);
pEdit->setGeometry(QRect(30, 80, 350, 150));

btJ = new QPushButton(this);
btJ->setGeometry(QRect(80, 240, 80, 25));
btJ->setText("加密");
connect(btJ, SIGNAL(clicked()), this, SLOT(encFun()));

btU = new QPushButton(this);
btU->setGeometry(QRect(220, 240, 80, 25));
btU->setText("解密");
connect(btU, SIGNAL(clicked()), this, SLOT(jieFun()));
```

## 1.2 AES 加密数据

### 1、实例参照

- 光盘/QtCode/ QtThirteen/Qt02/ Qt02.pro

### 2、实例实现

打开 mainwindow.h 文件编写如下代码。

```

#include <QLabel>
#include <QPushButton>
#include "taesclass.h"//引用 taesclass 头文件

private:
    QLabel *label;
    QPushButton *btJ;
    TAesClass *aes;
private slots:
    void encFun();

```

打开 mainwindow.cpp 文件编写如下代码。

```
#include <QDebug>
```

```

this->setWindowTitle("AES 加密");

label = new QLabel(this);
label->setGeometry(QRect(40, 40, 200, 25));
label->setText("调试中查看结果");

btJ = new QPushButton(this);
btJ->setGeometry(QRect(80, 80, 80, 25));
btJ->setText("加密");
connect(btJ, SIGNAL(clicked()), this, SLOT(encFun()));

```

```

//点击加密
void MainWindow::encFun()
{
    //实例 TAesClass 类
    aes = new TAesClass;
    //需要加密的字符串
    char mingwen[1024] = "你好";
    //转码
    DWORD size = strlen(mingwen);
    char miwen[1024];
    char jiemi[1024];
    //加入自己定义的密钥
    UCHAR key[1024] = "pwd";
    UCHAR *p = key;
    //进行初始化
    aes->InitializePrivateKey(16, p);
    //进行加密

```

```

    aes->OnAesEncrypt((LPVOID)mingwen, size, (LPVOID)miwen);
    //进行解密
    aes->OnAesUncrypt((LPVOID)miwen, (DWORD)sizeof(miwen), (LPVOID)jiemi);

    //调试输出信息
    qDebug() << "明文:" << mingwen;
    qDebug() << "密文:" << miwen;
    qDebug() << "解密:" << jiemi;
    //释放 TAesClass
    free(aes);
    //清0
    aes = 0;
}

```

注：这里需要将 taesclass.h 与 taesclass.cpp 文件拷贝到项目中，因为 QT 没有自带 AES 加密，taesclass 文件为网络资源，这里不做过分说明。

## 1.3 MD5 加密数据

### 1、实例参照

- 光盘/QtCode/ QtThirteen/Qt03/ Qt03.pro

### 2、实例实现

打开 mainwindow.cpp 文件编写如下代码。

```

//MD5 加密类
#include <QCryptographicHash>
#include <QLabel>

```

```

QLabel *label = new QLabel(this);
label->setGeometry(QRect(50, 50, 200, 25));

//被加密字符串
QString pwd = "nihao";
//加密后字符串
QString MD5;
//QByteArray
QByteArray ba, bb;
//实例 QCryptographicHash

```

```
QCryptographicHash md(QCryptographicHash::Md5);
ba.append(pwd);
md.addData(ba);
bb = md.result();
MD5.append(bb.toHex());
//MD5 号称是不可逆推，所以暂时没有解密
//MD5 用法：将加密字符就行比对。
//输出加密后字符串
label->setText(MD5);
```

## 1.4 生成随机数

### 1、实例参照

- 光盘/QtCode/ QtThirteen/Qt04/ Qt04.pro

### 2、实例实现

打开 mainwindow.cpp 文件编写如下代码。

```
#include <QTime>
#include <QLabel>
```

```
//根据时间产生随机数
QTime time;
time= QTime::currentTime();
qrand(time.msec()+time.second()*1000);
//随机数范围 1000
int xNum = qrand()%1000;

QLabel *label = new QLabel(this);
label->setGeometry(QRect(50, 50, 200, 25));
label->setText("随机数: "+QString::number(xNum));
```

# 第十四章 打包部署

---



## 1.1 FilePacker 打包

### 1、实例参照

- 光盘/QtCode/ QtFifteen/Qt01

### 2、实例实现

一： **hap-depends** 查询依赖动态库软件：

下载地址： <http://www.cr173.com/soft/6934.html>

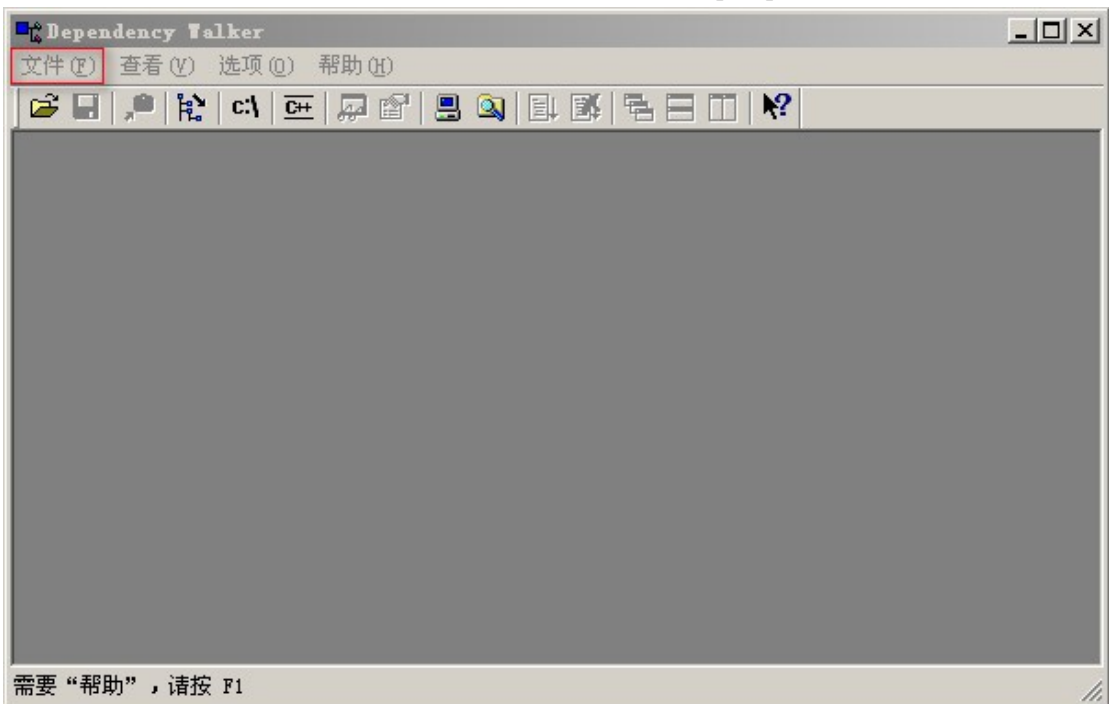
二： **filepack** 软件打包工具：

下载地址： <http://www.cr173.com/soft/19353.html>

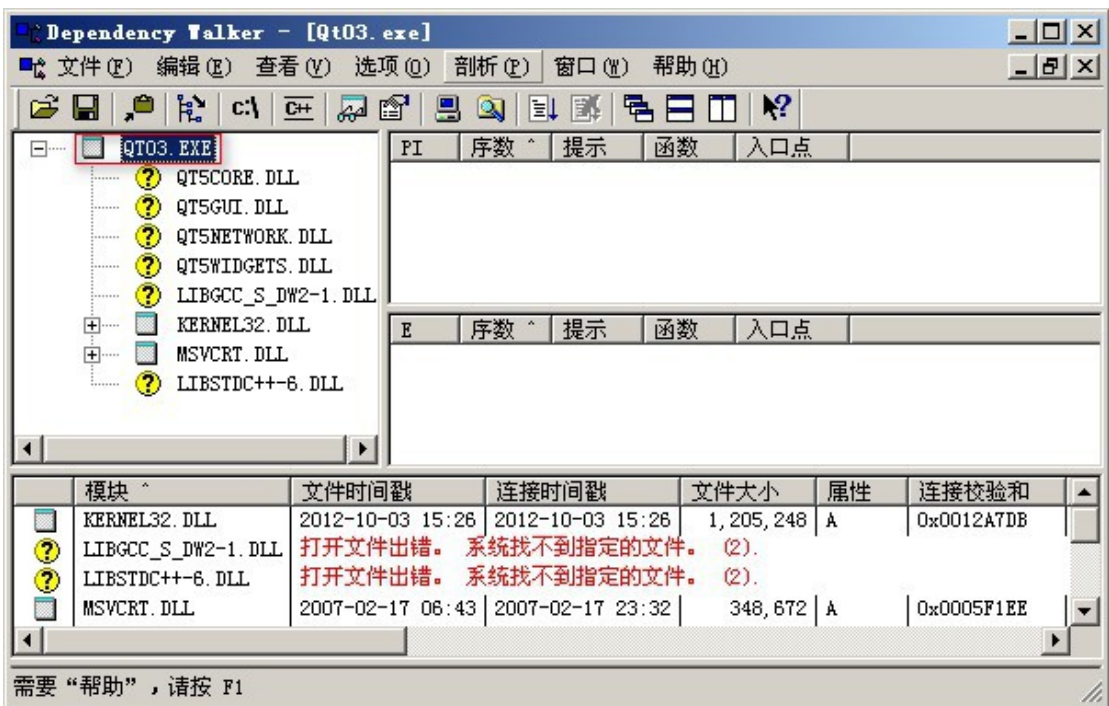
三：程序发布时，需要选择 release 发布程序。



发布后，将可执行文件\*.exe 程序拷贝到空文件夹下，将 hap-depends 软件打开如图：



点击文件-打开刚刚拷贝出来的.exe 程序， 如图：



打开 exe 程序后发现有很多 DLL 需要我们拷贝，这些缺少的 DLL 文件在 QtCreator 安装目录。我的 Creator 安装在 D 盘：D:\Qt\Qt5.1.1\5.1.1\mingw48\_32\bin，将上图中带问号图标的 DLL 全部拷贝到 exe 同级目录下，F5 刷新 hap-depends 软件，直到上图中没有带问号图标为止。

注：拷贝完 Dll 之后，还有一项很重要的事需要做，需要将 platforms 文件夹整个拷贝到 exe 同级目录下，platforms 文件夹位置：

D:\Qt\Qt5.1.1\5.1.1\mingw48\_32\plugins\platforms，如下图，完整的发布包。

platforms	文件夹	2013-12-23 11:55	
icudt51.dll	21,854 KB 应用程序扩展	2013-04-23 00:03	A
icuin51.dll	3,291 KB 应用程序扩展	2013-04-23 00:03	A
icuuc51.dll	1,933 KB 应用程序扩展	2013-04-23 00:03	A
libgcc_s_dw2-1.dll	533 KB 应用程序扩展	2013-04-18 02:18	A
libstdc++-6.dll	967 KB 应用程序扩展	2013-04-18 02:19	A
libwinpthread-1.dll	73 KB 应用程序扩展	2013-04-18 01:26	A
Qt03.exe	174 KB 应用程序	2013-12-23 11:16	A
Qt5Core.dll	4,290 KB 应用程序扩展	2013-11-20 10:55	A
Qt5Gui.dll	4,349 KB 应用程序扩展	2013-08-26 02:58	A
Qt5Network.dll	1,361 KB 应用程序扩展	2013-08-26 02:56	A
Qt5Widgets.dll	6,006 KB 应用程序扩展	2013-08-26 03:02	A

打包文件都准备好，开始打包，打开 FilePacker 软件，如图。



点击下一步



起个工程名，点击下一步



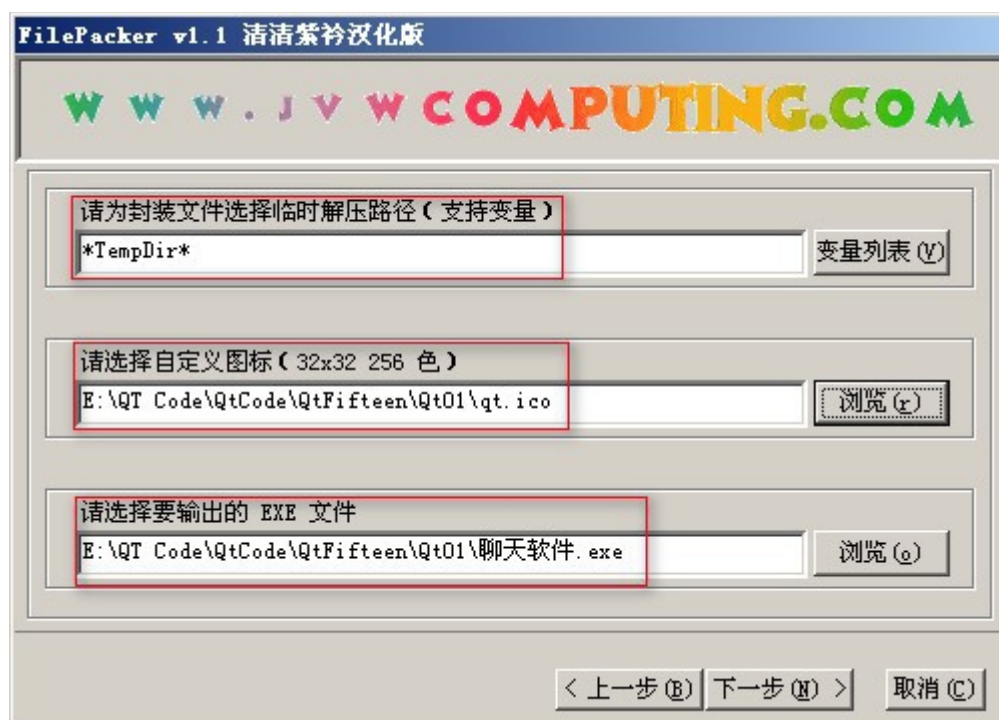
选择刚刚准备好的文件夹，点击下一步



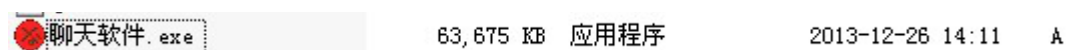
选择可执行文件 Qt03.exe，点击下一步



直接点击下一步



选择解压路径，配置程序图标，输出的 exe 文件名，点击下一步，发布成功如图



软件打包成功，一共只有 63MB 大小，很爽吧。

## 1.2 Inno Setup 打包

### 1、实例参照

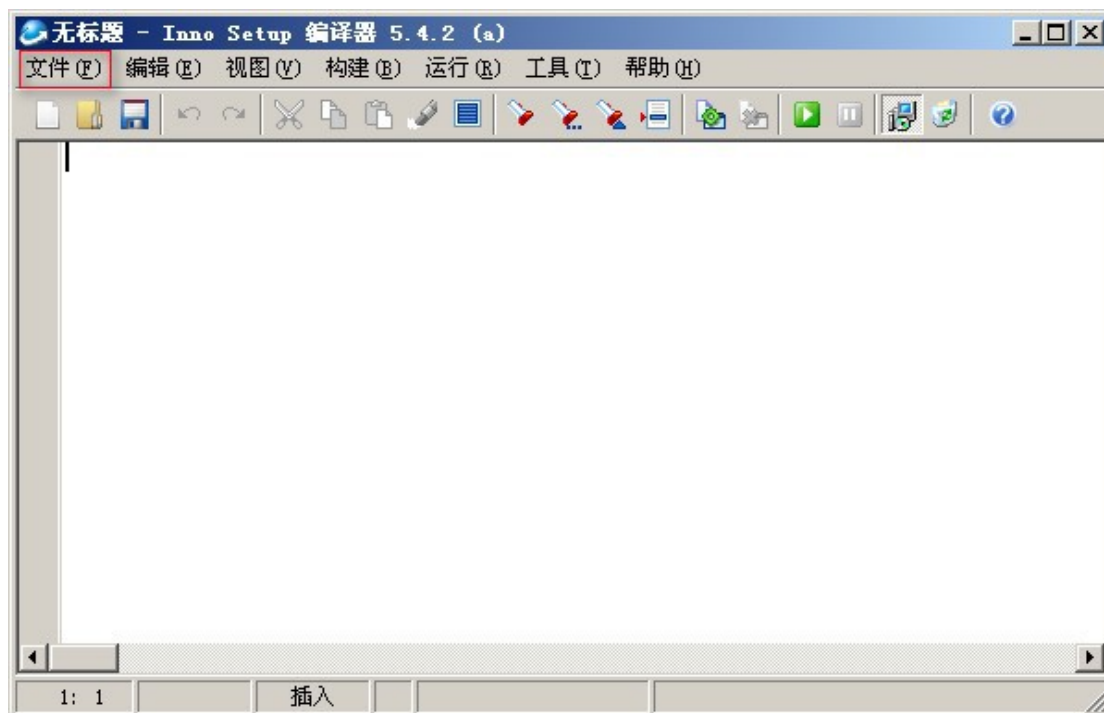
- 光盘/QtCode/ QtFifteen/Qt02

### 2、实例实现

首先还是将可执行程序与 Dll、platforms 文件夹准备好，之后下载 InnoSetup 软件，这里用到的版本为：InnoSetup 5.5.4 汉化版。

打开 InnoSetup 软件，程序主界面如下图





选择文件-新建





Inno Setup 脚本向导

**应用程序信息**  
请指定关于应用程序的基本信息。

**应用程序名称(M):**  
聊天

**应用程序版本(V):**  
1.5

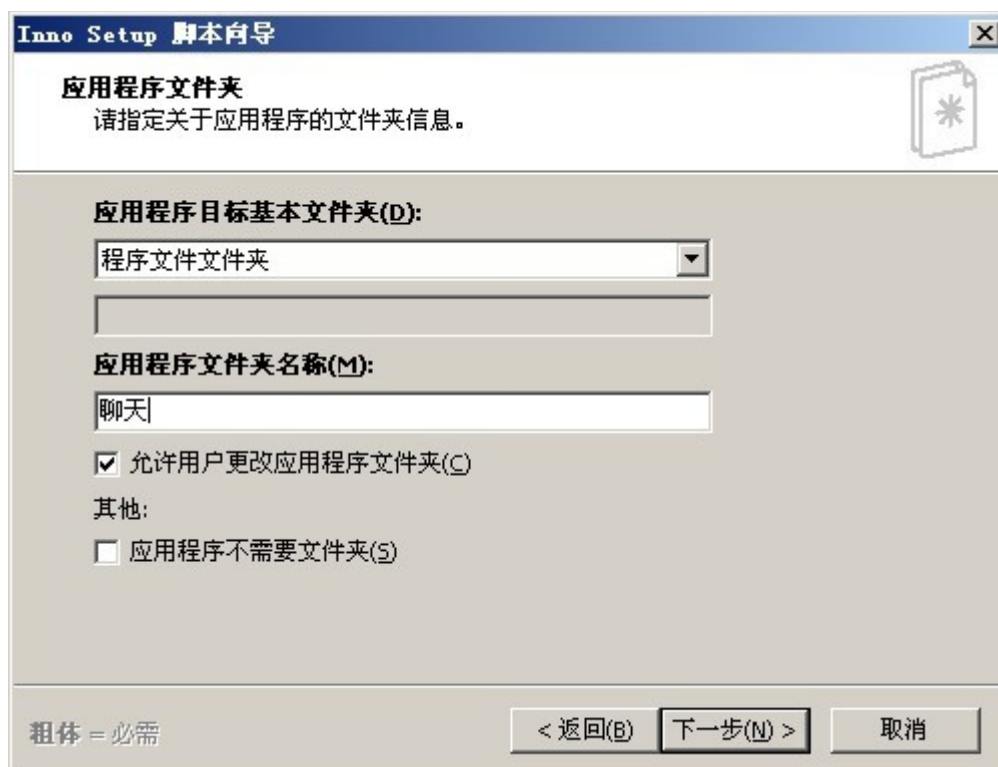
**应用程序发布者(P):**  
Longki

**应用程序网站(W):**  
http://www.baidu.com

粗体 = 必需

< 返回(B)    下一步(N) >    取消

添加应用程序信息，点击下一步



Inno Setup 脚本向导

**应用程序文件夹**  
请指定关于应用程序的文件夹信息。

**应用程序目标基本文件夹(D):**  
程序文件文件夹

**应用程序文件夹名称(M):**  
聊天

☒ 允许用户更改应用程序文件夹(C)

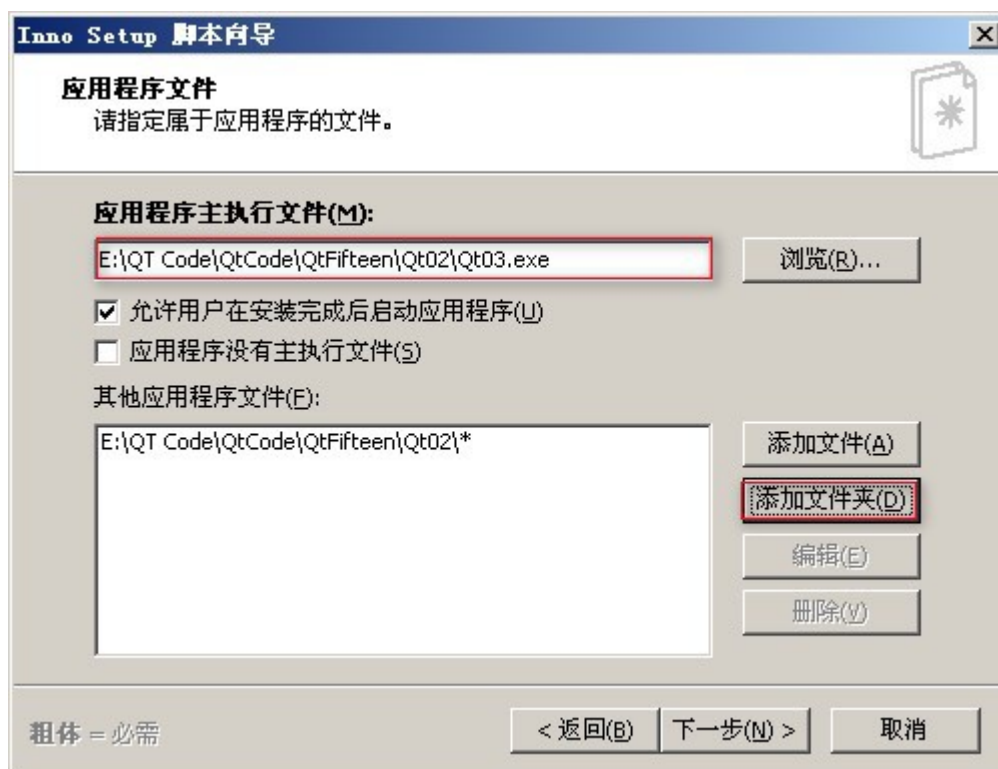
其他:  
☐ 应用程序不需要文件夹(S)

粗体 = 必需

< 返回(B)    下一步(N) >    取消

点击下一步

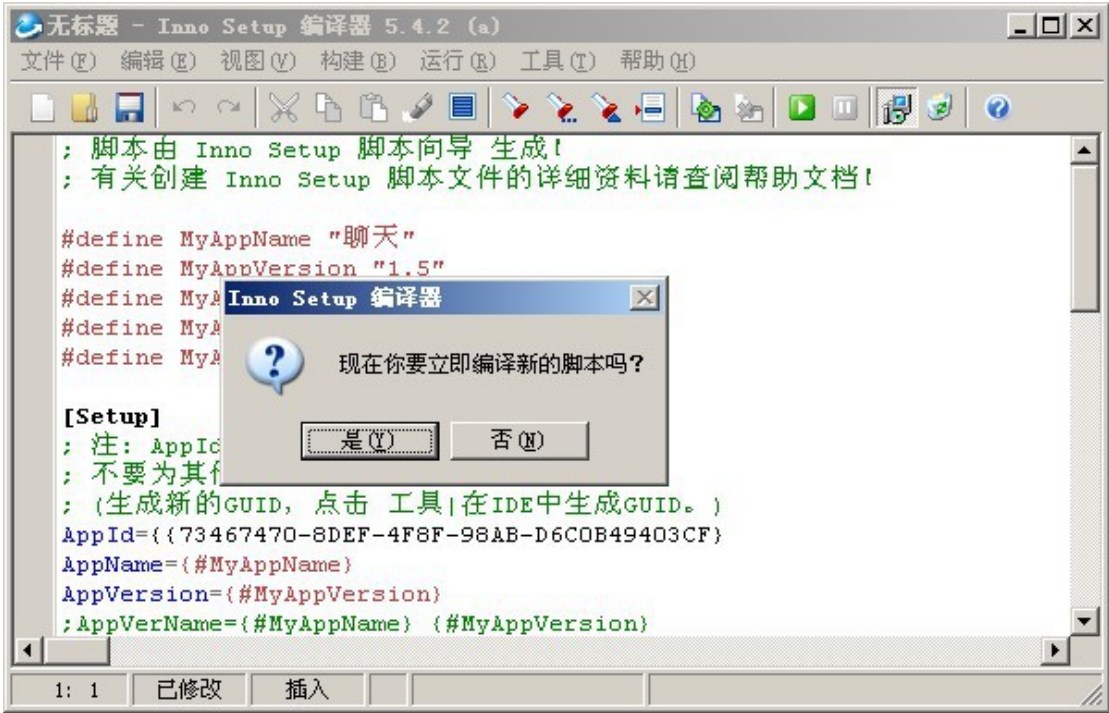






选择可执行文件，点击添加文件夹。



输出文件夹不要选择生成到 Qt02 项目中，因为该项目需要被打包，所以选择输出到桌面，之后一直点击下一步。



问是否立即执行编译，选择---是。

 聊天.exe	15,433 KB	应用程序	2013-12-27 10:34	A
 Qt02.iss	2 KB	Inno Setup 脚本	2013-12-27 10:34	A

编程成功后，关闭 inno setup 软件，聊天.exe 就是我们打包后的产物 15M 左右。