Qt 串口通信专题教程

查看以前的教程：Qt 编写串口通信程序全程图文讲解

查看 Wincom 和 Lincom 介绍：Qt 跨平台串口通信软件 Wincom 与 Lincom

下载软件，文档和源码：资源下载

————————————2010 年 7 月 8 日更新————————————————————

网友 赵文杰 使用多线程完成的 linux 下的串口通信。

下载源码：下载

——

以下是正文：

前言

去年我使用 Qt 编写串口通信程序时，将自己的学习过程写成了教程（Qt 编写串口通信程序全程图文讲解），但是由于时间等原因，我只实现了 Windows 下的串口通信，并没有去做 Linux 下的。自从教程发布到网上后，就不断有人提出相关的问题，而其中问的最多的就是，怎样在 Linux 下实现串口通信。因为有计划安排，而且没有开发板，所以一直没能去研究，也就没能给出很好的解决办法。前些天，网友 hqwfreefly 用 Qt 写了一个叫 linucom 的 Linux 下串口调试程序，实现了 Linux 的串口通信。而且，正好现在我有几天假期，所以就和 hqwfreefly 合作，将 linucom 更新为 Lincom，并且推出了 Windows 下的 Wincom，然后完成了这篇 Qt 编写串口通信程序的专题教程，也算完成了我的一个心愿。

教程概述

该教程分三部分讲述，第一部分讲解 qextserialport 类的一些东东；第二部分讲解在 Windows 下使用 qextserialport 类实现串口通信的方法，这里将讲述两种不同的方法；第三部分讲解在 Linux 下利用 qextserialport 类实现串口通信的方法。

在这个教程中我们更注重知识的讲解，而不是界面的设计。关于界面和其他应用问题，你可以查看以前的串口通信教程或者查看一下 Wincom 软件的源码。

第一部分 Qextserialport 类介绍

在 Qt 中并没有特定的串口控制类，现在大部分人使用的是第三方写的 qextserialport 类，我们这里也使用了该类。

一、文件下载

文件下载地址：

http://sourceforge.net/projects/qextserialport/files/

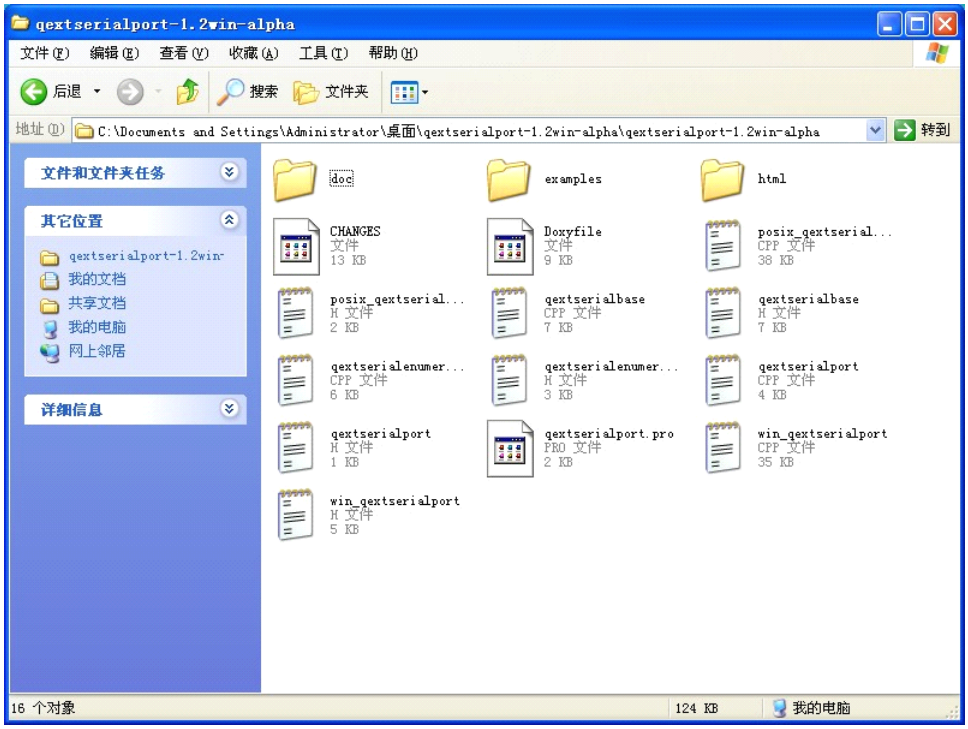
也可以下载我上传到网盘上的：

http://good.gd/494307.htm

二、文件内容介绍

1.下载到的文件为 qextserialport-1.2win-alpha ，解压并打开后其内容如下。

（点击图片可以查看清晰大图）



下面分别介绍：

（1）doc 文件夹中的文件内容是 QextSerialPort 类和 QextBaseType 的简单的说明，我们可以使用记事本程序将它们打开。

（2）examples 文件夹中是几个例子程序，可以看一下它的源码，不过想运行它们好像会出很多问题啊。

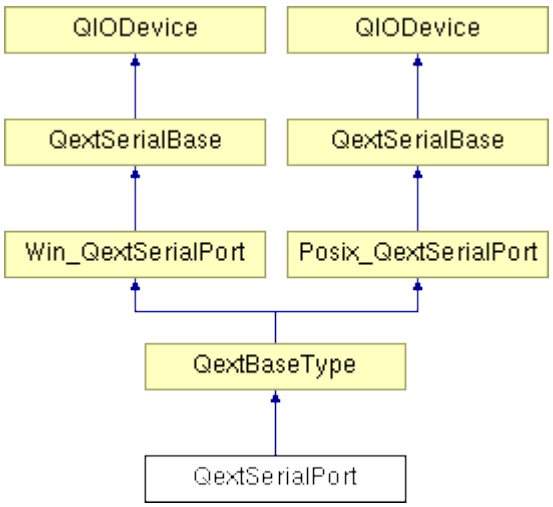
（3）html 文件夹中是 QextSerialPort 类的使用文档。

（4）然后就是剩下的几个文件了。其中 qextserialenumerator.cpp 及 qextserialenumerator.h 文件中定义的 QextSerialEnumerator 类是用来获取平台上可用的串口信息的。不过，这个类好像并不怎么好用，而且它不是我们关注的重点，所以下面就不再介绍它了。

（5）qextserialbase.cpp 和 qextserialbase.h 文件定义了一个 QextSerialBase 类，win_qextserialport.cpp 和 win_qextserialport.h 文件定义了一个 Win_QextSerialPort 类，posix_qextserialport.cpp 和 posix_qextserialport.h 文件定义了一个 Posix_QextSerialPort 类，qextserialport.cpp 和 qextserialport.h 文件定义了一个 QextSerialPort 类。这个 QextSerialPort 类就是我们上面所说的那个，它是所有这些类的子类，是最高的抽象，它屏蔽了平台特征，使得在任何平台上都可以使用它。

2.几个类的简单介绍。

下面是这几个类的关系图。



可以看到它们都继承自 QIODevice 类，所以该类的一些函数我们也可以直接来使用。图中还有一个 QextBaseType 类，其实它只是一个标识，没有具体的内容，它用来表示 Win_QextSerialPort 或 Posix_QextSerialPort 中的一个类，因为在 QextSerialPort 类中使用了条件编译，所以 QextSerialPort 类既可以继承自

Win_QextSerialPort 类，也可以继承自 Posix_QextSerialPort 类，所以使用了 QextBaseType 来表示。这一点我们可以在 qextserialport.h 文件中看到。再说 QextSerialPort 类，其实它只是为了方便程序的跨平台编译，使用它可以在不同的平台上，根据不同的条件编译继承不同的类。所以它只是一个抽象，提供了几个构造函数而已，并没有具体的内容。在 qextserialport.h 文件中的条件编译内容如下：

```
/*POSIX CODE*/

#ifdef _TTY_POSIX_

#include “posix_qextserialport.h”

#define QextBaseType Posix_QextSerialPort



/*MS WINDOWS CODE*/

#else

#include “win_qextserialport.h”

#define QextBaseType Win_QextSerialPort

#endif
```

所以，其实我们没有必要使用这个类，直接使用 Win_QextSerialPort 或 Posix_QextSerialPort 就可以了。当然如果你想使用这个类，实现同样的源程序可以直接在 Windows 和 Linux 下编译运行，那么一定要注意在 Linux 下这里需要添加 #define _TTY_POSIX_ 。而我们这里为了使得程序更明了，所以没有使用该类，下面也就不再介绍它了。

QextSerialBase 类继承自 QIODevice 类，它提供了操作串口所必需的一些变量和函数等，而 Win_QextSerialPort 和 Posix_QextSerialPort 均继承自 QextSerialBase 类，Win_QextSerialPort 类添加了 Windows 平台下操作串口的一些功能，Posix_QextSerialPort 类添加了 Linux 平台下操作串口的一些功能。所以说，在 Windows 下我们使用 Win_QextSerialPort 类，在 Linux 下我们使用 Posix_QextSerialPort 类。

3. 在 QextSerialBase 类中还涉及到了一个枚举变量 QueryMode。

它有两个值 Polling 和 EventDriven 。QueryMode 指的是读取串口的方式，下面我们称为查询模式，我们将 Polling 称为查询方式 Polling，将 EventDriven 称为事件驱动方式。

事件驱动方式 EventDriven 就是使用事件处理串口的读取，一旦有数据到来，就会发出 readyRead()信号，我们可以关联该信号来读取串口的数据。在事件驱动的方式下，串口的读写是异步的，调用读写函数会立即返回，它们不会冻结调用线程。

而查询方式 Polling 则不同，读写函数是同步执行的，信号不能工作在这种模式下，而且有些功能也无法实现。但是这种模式下的开销较小。我们需要自己建立定时器来读取串口的数据。

在 Windows 下支持以上两种模式，而在 Linux 下只支持 Polling 模式。

三、小结。

这里讲了这么多，最后要说的只是，我们在 Qt 中使用这个类编写串口程序，根据平台的不同只需要分别使用四个文件。

在 Windows 下是：

qextserialbase.cpp 和 qextserialbase.h 以及 win_qextserialport.cpp 和 win_qextserialport.h

在 Linux 下是：

qextserialbase.cpp 和 qextserialbase.h 以及 posix_qextserialport.cpp 和 posix_qextserialport.h

而在 Windows 下我们可以使用事件驱动 EventDriven 方式，也可以使用查询 Polling 方式，但是在 Linux 下我们只能使用查询 Polling 方式。

第二部分 在 Windows 下编写串口通信程序

我们的环境是 Windows xp，Qt4.6.3 及 Qt Creator2.0。

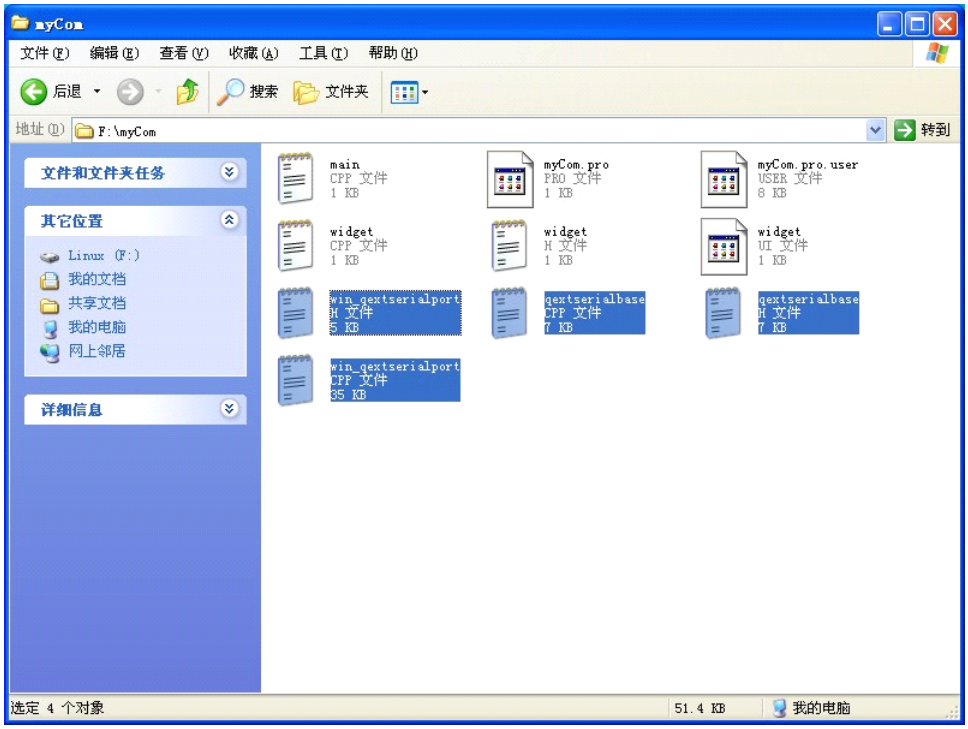
第一，下面我们首先使用事件驱动来实现串口通信。

1.新建工程。

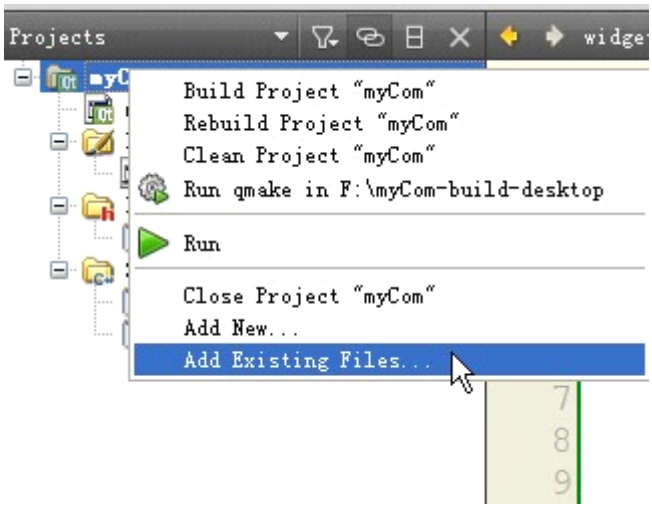
我们在 Qt Creator 中新建 Qt Gui 工程，命名为 myCom，Base Class 选择 QWidget。

2.添加文件。

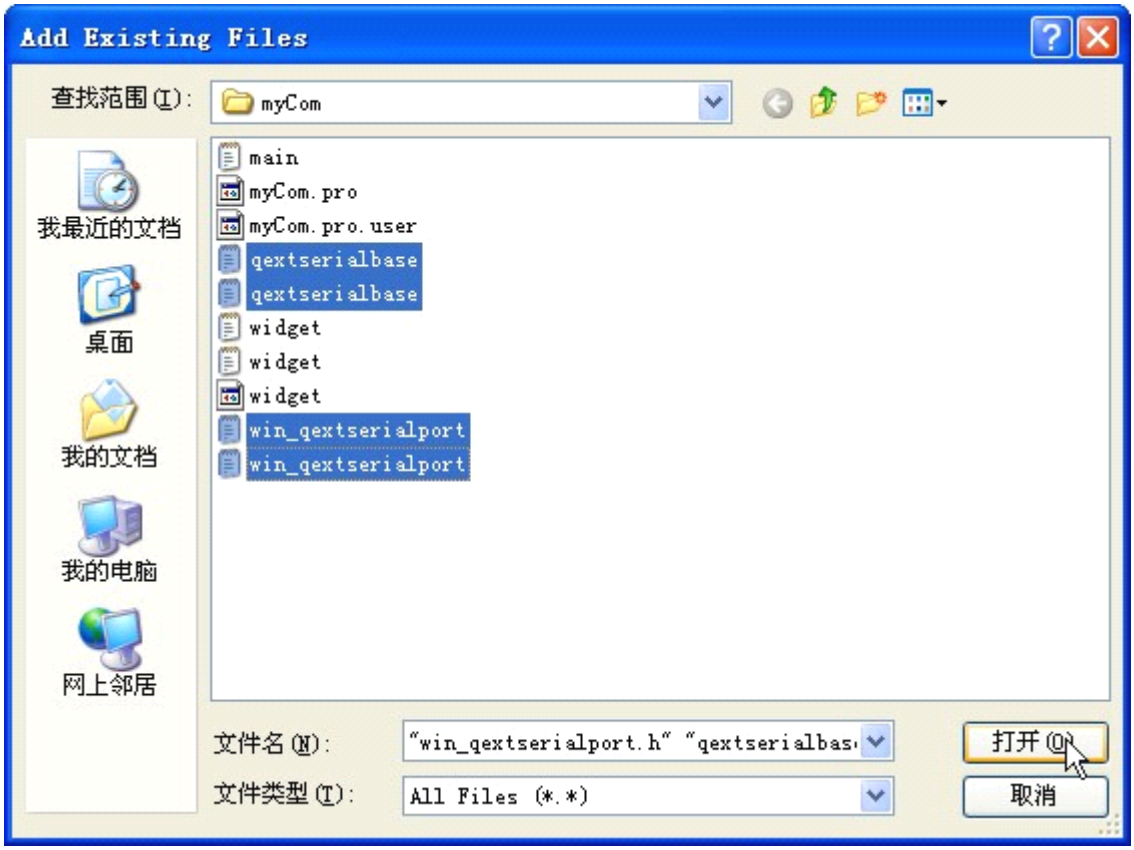
我们将那四个文件添加到工程文件夹中。如下图。



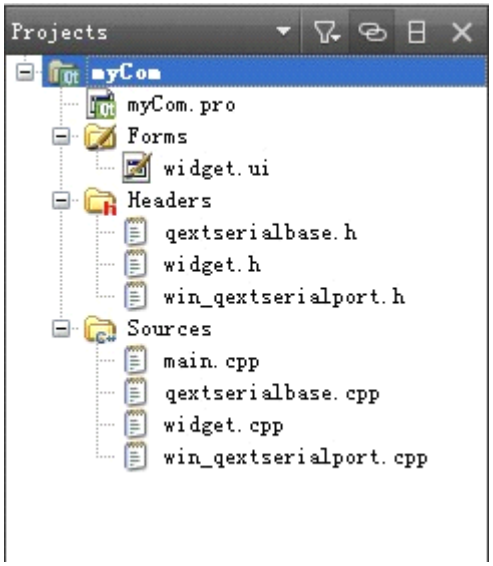
然后我们将这四个文件添加到工程中，在 Qt Creator 的工程列表中的工程文件夹上点击鼠标右键，在弹出的菜单中选择“Add Existing Files”菜单。如下图。



我们在弹出的对话框中选中四个文件，按下“打开”按钮即可，如下图。

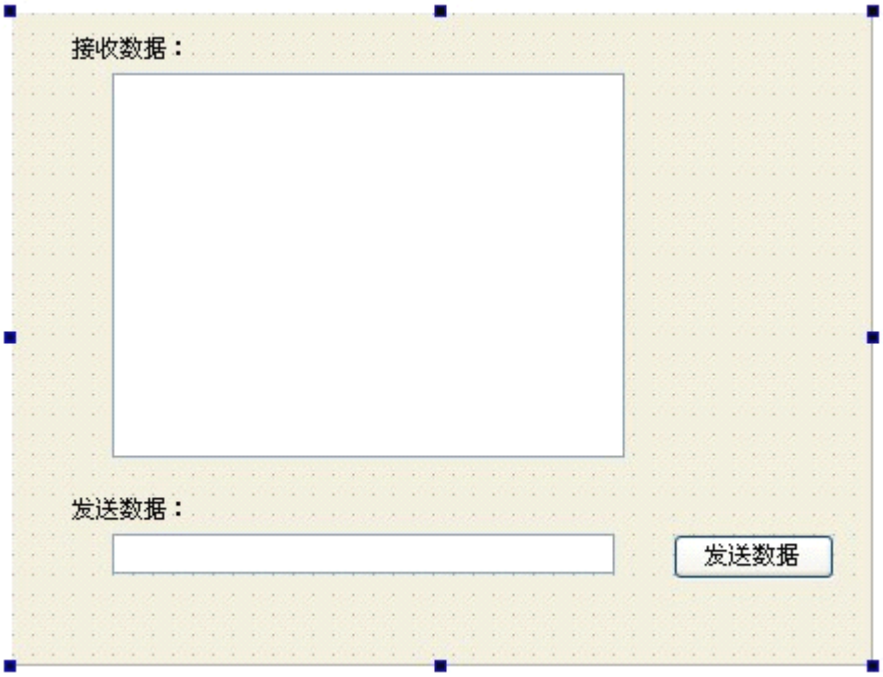


最终工程文件列表如下图。



3. 更改界面。

我们将界面设计如下。



其中的 Text Browser 部件用来显示接收到的数据，Line Edit 部件用来输入要发送的数据，Push Button 按钮用来发送数据。我们保持各部件的属性为默认值即可。

4. 我们在 widget.h 文件中进行对象及函数声明。

添加头文件包含：#include “win_qextserialport.h”

然后在 private 中声明对象：Win_QextSerialPort *myCom;

声明私有槽函数：

```
private slots:

    void on_pushButton_clicked(); //”发送数据”按钮槽函数

void readMyCom();  //读取串口
```

5.在 widget.cpp 文件中进行更改。

在构造函数中添加代码，完成后，构造函数内容如下：

```
Widget::Widget(QWidget *parent) :

    QWidget(parent),

    ui(new Ui::Widget)
```

```
{
    ui->setupUi(this);
    myCom = new
Win_QextSerialPort(“COM1″,QextSerialBase::EventDriven);
    //定义串口对象，指定串口名和查询模式，这里使用事件驱动 EventDriven
    myCom ->open(QIODevice::ReadWrite);
    //以读写方式打开串口
    myCom->setBaudRate(BAUD9600);
    //波特率设置，我们设置为 9600
    myCom->setDataBits(DATA_8);
    //数据位设置，我们设置为 8 位数据位
    myCom->setParity(PAR_NONE);
    //奇偶校验设置，我们设置为无校验
    myCom->setStopBits(STOP_1);
    //停止位设置，我们设置为 1 位停止位
    myCom->setFlowControl(FLOW_OFF);
    //数据流控制设置，我们设置为无数据流控制
    myCom->setTimeout(500);
    //延时设置，我们设置为延时 500ms,这个在 Windows 下好像不起作用
    connect(myCom,SIGNAL(readyRead()),this,SLOT(readMyCom()));
    //信号和槽函数关联，当串口缓冲区有数据时，进行读串口操作
}
```

实现槽函数：

```
void Widget::readMyCom() //读取串口数据并显示出来
```

```
{
    QByteArray temp = myCom->readAll();
    //读取串口缓冲区的所有数据给临时变量 temp
    ui->textBrowser->insertPlainText(temp);
    //将串口的数据显示在窗口的文本浏览器中
}
void Widget::on_pushButton_clicked()  //发送数据
{
    myCom->write(ui->lineEdit->text().toAscii());
    //以 ASCII 码形式将数据写入串口
}
```

6.此时，我们运行程序，效果如下。

可以看到，已经成功完成通信了。

（注：我们这里下位机使用的是单片机，它使用串口与计算机的 COM1 相连。单片机上运行的程序的功能是，接收到一个字符便向上位机发送一个字符串然后发送接收到的字符。）

两个重要问题的讲解：

一、关于数据接收。

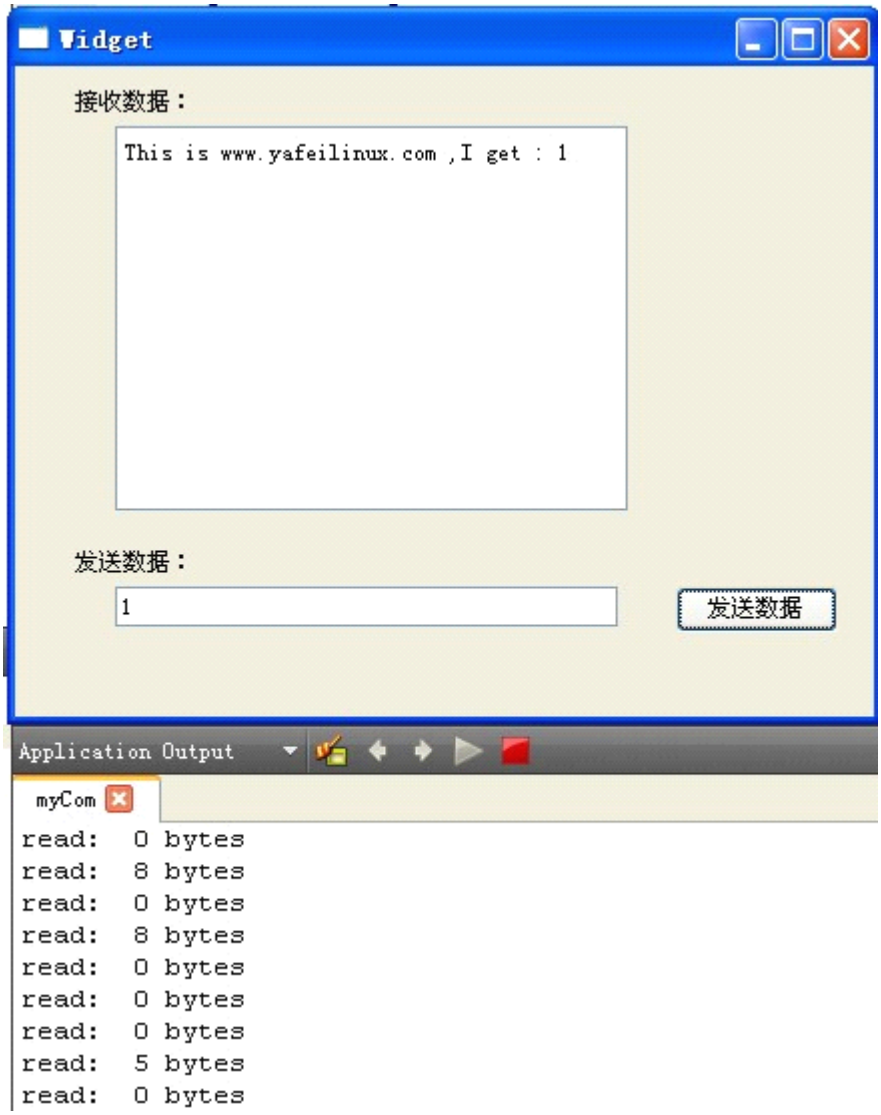
我们想在程序中对接收的数据进行控制，但是 readyRead()信号是一旦有数据到来就发射的，不过我们可以使用 bytesAvailable()函数来检查已经获得的字节数，从而对数据接收进行控制。

（1）我们在 widget.cpp 中添加头文件包含：#include <QDebug>

然后在读串口函数中添加一行代码，如下：

```
void Widget::readMyCom() //读取串口数据并显示出来
{
    qDebug() << “read: “<<myCom->bytesAvailable()<<” bytes”;
    //我们输出每次获得的字节数
    QByteArray temp = myCom->readAll();
    ui->textBrowser->insertPlainText(temp);
}
```

运行程序，效果如下：



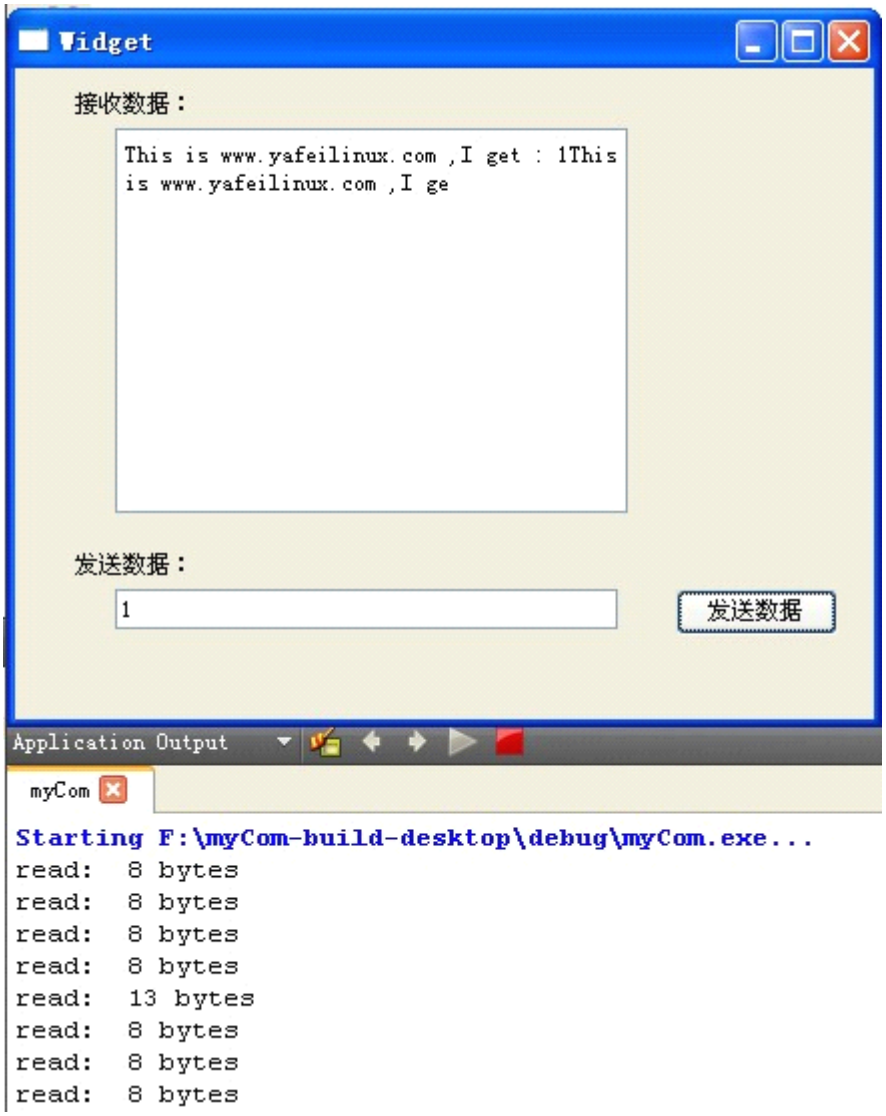
可以看到，我们获取的数据并不是一次获得的。

（2）利用上面的结论，我们可以让串口缓冲区拥有了一定的数据后再读取。

```
void Widget::readMyCom()
{
if(myCom->bytesAvailable() >=8 )
//如果可用数据大于或等于 8 字节再读取
    {
        qDebug() << “read: “<<myCom->bytesAvailable()<<” bytes”;
        QByteArray temp = myCom->readAll();
```

```
        ui->textBrowser->insertPlainText(temp);
    }
}
```

运行程序，效果如下：



我们发送了两次数据，可以看到，这样实现了每 8 个字节读取一次，而最后剩余的不够 8 个字节的数据将会和后面的数据一起读出。

然后我们将 8 改为 3，发送一次数据，效果如下：

改为 7，发送两次数据，效果如下：

改为 11，发送两次数据，效果如下：

改为 17，发送三次数据，效果如下：

重要结论：我们发送一次数据，应该获得 37 字节的数据，然后我们对比上面的结果，发现了什么？是的，其实串口每次读取 8 字节的数据放到缓冲区，只有数

据总数小于 8 字节时，才会读取小于 8 字节的数据。为了再次验证我们的结论，我们可以将上面程序中的“>=”改为“==”，那么只有 8 的倍数才能读取数据（当然这里 37 也可以），你可以测试一下。

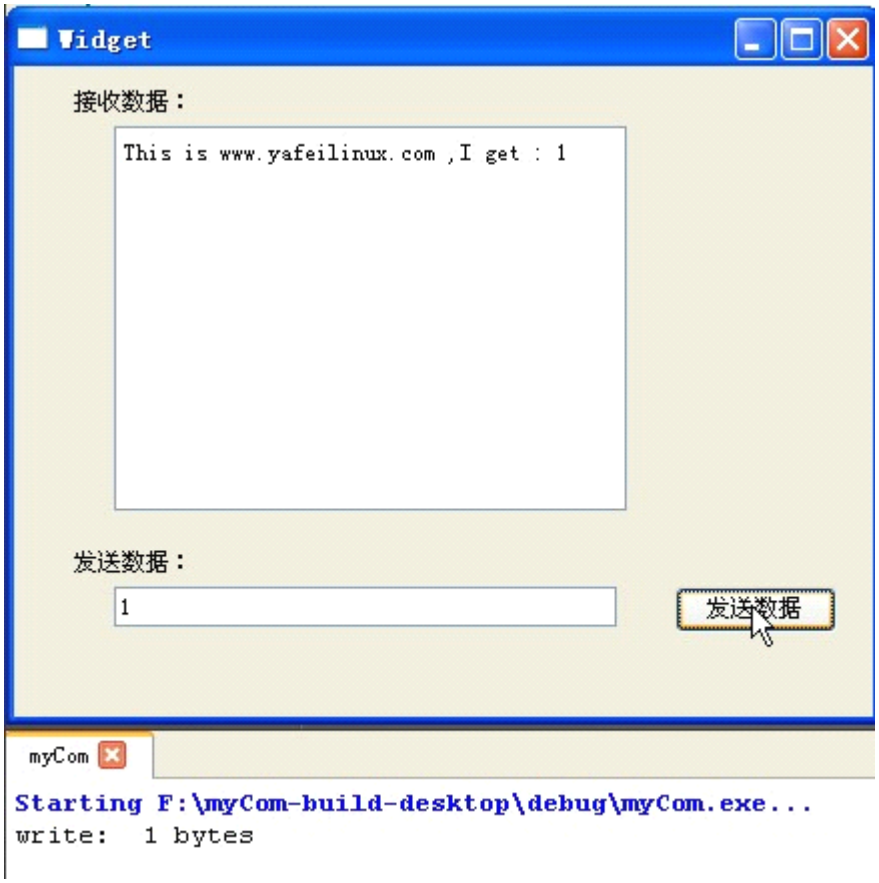
关于接收数据方面，可以根据你自己的需要再去进行研究和改进，这里只是抛砖引玉。

二、关于发送数据。

我们也可以使用函数获取要发送的数据的大小，这里有个 bytesToWrite()可以获取要发送的字节数。例如将发送数据更改如下：

```
void Widget::on_pushButton_clicked()  //发送数据

{

    myCom->write(ui->lineEdit->text().toAscii());

    qDebug() << “write: “<<myCom->bytesToWrite()<<” bytes”;

    //输出要发送的字节数

}
```

运行后效果如下：



当然，对于要发送的数据的大小我们不是很关心，而且它还有很多方法可以实现，这个还有个 bytesWritten()信号函数来获取已经发送的数据的大小，不过好像它不是很好用。这里将它们提出来，只是供大家参考而已。

第二，使用查询方式 Polling 来实现串口通信。

这里再次说明，Polling 方式是不能使用 readyRead()信号的，所以我们需要自己设置定时器，来不断地读取缓冲区的数据。

1.我们在 widget.h 中声明一个定时器对象。

添加头文件包含：#include <QTimer>

添加 private 变量：QTimer *readTimer;

2.我们在 widget.cpp 文件中的构造函数中更改。

（1）将串口定义更改为：

myCom = new Win_QextSerialPort(“COM1”,QextSerialBase::Polling);

//定义串口对象，指定串口名和查询模式，这里使用 Polling

（2）定义定时器，并将以前的关联更改为定时器的关联。

```
readTimer = new QTimer(this);

    readTimer->start(100);

    //设置延时为 100ms

    connect(readTimer,SIGNAL(timeout()),this,SLOT(readMyCom()));
```

//信号和槽函数关联，延时一段时间，进行读串口操作

3.此时运行程序，便可以正常收发数据了。

重点：关于延时问题。

上面的程序中可以进行数据的接收了，但是好像中间的延时有点长，要等一会儿才能收到数据，而且即便我们将定时器改为 10ms 也不行。问题在哪里呢？其实真正控制串口读写时间的不是我们的定时器，而是延时 timeout。我们在构造函数中设置了延时：

```
myCom->setTimeout(500);
```

//延时设置，我们设置为延时 500ms

我们前面说延时并不起作用，那是因为是在事件驱动的情况下，一旦有数据到来就会触发 readyRead()信号，所以延时不起作用。但是现在，真正控制串口读写数据间隔的就是这个函数。这里值得注意，我们现在所说的串口读写是指底层的串口读写，从上面的程序中我们也可以看到，我们每隔 100ms 去读串口，确切地说，应该是去读串口缓冲区。而 timeout 才是正真的读取串口数据，将读到的数据放入串口缓冲区。所以如果 timeout 时间很长，即便我们的定时器时间再短，也是读不到数据的。所以我们这里需要将 timeout 设置为较小的值，比如 10。
我们更改代码：

```
myCom->setTimeout(10);
```

这样再运行程序，我们就可以很快地获得数据了。

关于数据接收：事件驱动那里的结论依然有用，不过这里更多的是靠读取的时间间隔来控制。

关于发送数据：这时 bytesToWrite()函数就不再那么好用了。

第三部分 在 Linux 下编写串口通信程序

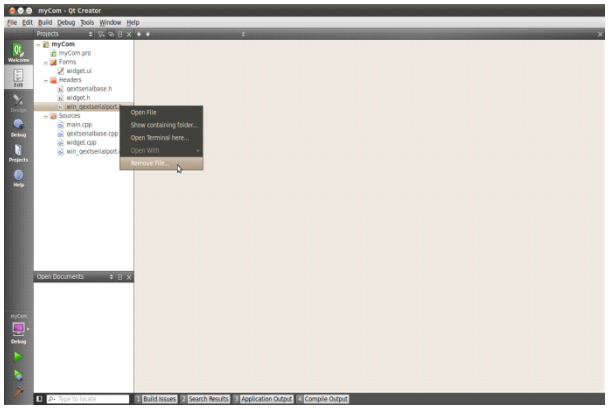
我这里的环境是 Ubuntu 10.04，Qt 4.6.3 和 Qt Creator2.0 。上面已经提到，在 Linux 下只能使用 Polling 的方式读取串口数据，所以我们将上面 Windows 下的应用 Polling 的程序在 Linux 下重新编译。我们使用 Qt Creator 打开该工程，然后进行下面的操作。

1. 文件替换。

将工程中的 win_qextserialport.cpp 和 win_qextserialport.h 文件替换成 posix_qextserialport.cpp 和 posix_qextserialport.h 文件。

（1）我们先删除工程中的 win_qextserialport.cpp 和 win_qextserialport.h 文件。

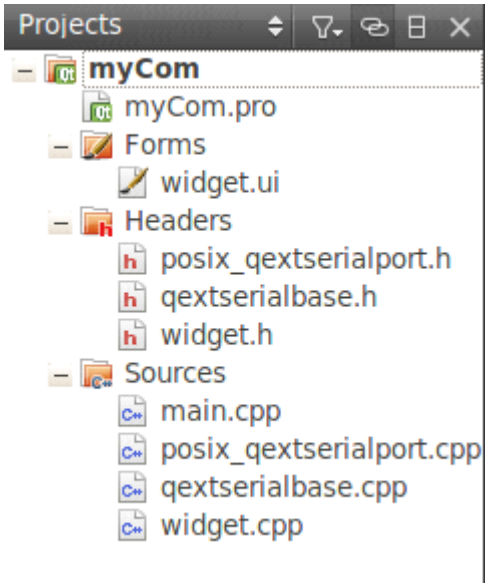
在工程列表中用鼠标右击 win_qextserialport.h，然后选择“Remove File”选项。如下图。



在弹出的对话框中我们选中“Delete file permanently”选项，确保删除了工程文件夹中的文件。如下图。

然后我们使用同样的方法删除 win_qextserialport.cpp 文件。

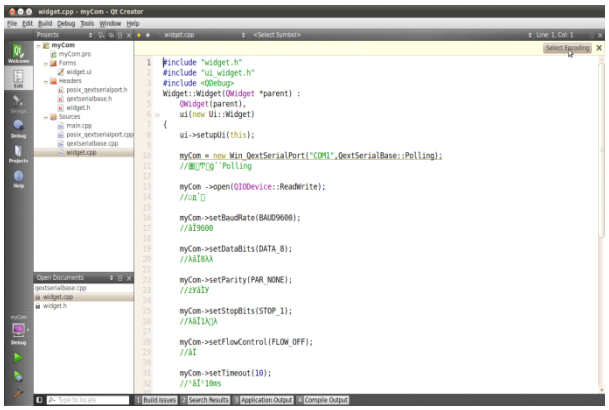
（2）我们按照 Windows 下添加文件的方法，向工程中添加 posix_qextserialport.cpp 和 posix_qextserialport.h 文件。最终工程文件列表如下。



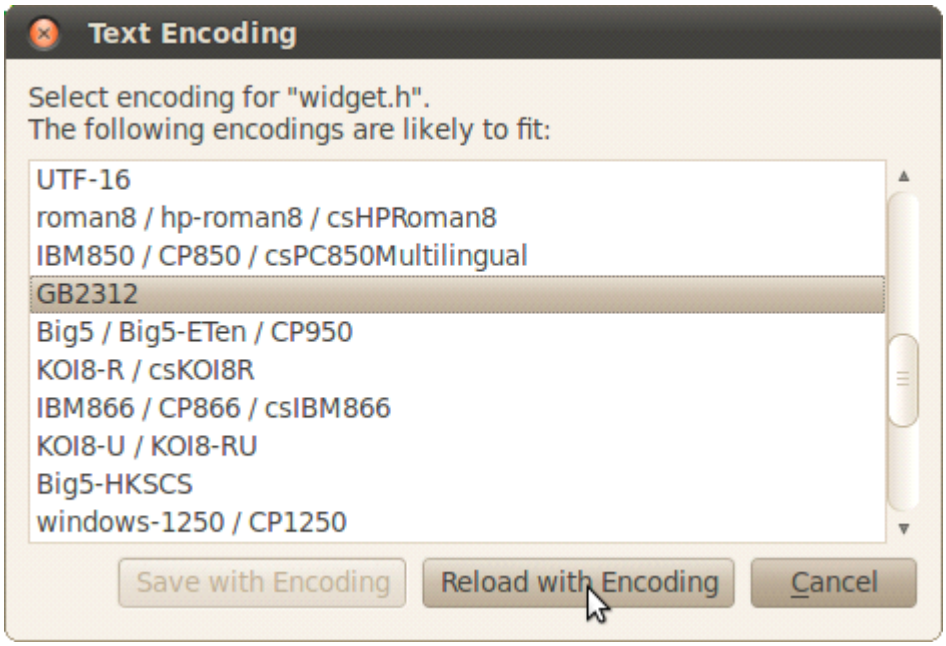
2. 设置编码。

（这是因为两个系统使用的默认编码不同造成的，如果你那里没有该问题，可以跳过这一步）

现在我们打开 widget.cpp 文件，发现中文出现乱码，而且无法编辑。在编辑器最上面有一个黄色提示条和一个“Select Encoding”按钮，我们点击该按钮。如下图。



在弹出的对话框中我们选择“GB2312”。按下“Reload with Encoding”按钮，中文就可以正常显示了。



3. 更改程序。

在 widget.h 文件中：

将以前的#include “win_qextserialport.h”更改为#include “posix_qextserialport.h”

将以前的 Win_QextSerialPort *myCom;更改为 Posix_QextSerialPort *myCom;

在 widget.cpp 文件中：

将以前的 myCom = new Win_QextSerialPort(“COM1”,QextSerialBase::Polling);

更改为：myCom = new Posix_QextSerialPort(“/dev/ttyS0”,QextSerialBase::Polling);

（这里一定要注意串口名称的写法。）

4. 下面我们运行程序。

这时可能会出现以下提示。

错误是说一个函数的调用出现了问题。我们点击该错误，定位到出错的位置，然后将那个函数中的第一个参数删除即可。如下图。

```
// setTimeout(Settings.Timeout_Sec, Settings.Timeout_Millisec);
 setTimeout(Settings.Timeout_Millisec);
```

5. 再次运行程序，这时已经可以正常运行了。

6. 小结

可以看到将 Windows 下的串口程序在 Linux 下重新编译是很简单的，我们只需要替换那两个文件，然后更改一下头文件包含，对象定义和串口名即可。

结尾

本教程比较详细的讲述了使用 Qt 在 Windows 下和 Linux 下编写串口通信程序的方法，但是对于串口通信的内容还有很多，我们现在还无法全部涵盖。希望广大网友可以提出宝贵建议，将 Wincom 软件进行功能扩展，或者将本教程继续更新下去。

如果你喜欢本教程的写作风格，而且您也是 Qt 爱好者，您可以访问我们的网站，这里有一系列教程和软件供您参考学习，当然也希望您能为我们的网站添砖加瓦，让我们一起为 Qt 及 Qt Creator 的普及贡献自己的力量。

关于我们

yafeilinux 不是个人，而是一个团队！

网站：www.yafeilinux.com 邮箱：yafeilinux@vip.163.com QQ 群：158054692

合作者 hqwfreefly

邮箱：hqwemail@163.com 个人主页：http://hi.baidu.com/hqwfreefly