**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](http://www.yafeilinux.com/?p=387" \o "Permanent Link to 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](http://www.yafeilinux.com/?p=816" \o "Permanent Link to 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](http://www.yafeilinux.com/?page_id=9" \o "Permanent Link to 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](http://good.gd/613667.htm)

—————————————————————————————————————

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](http://www.yafeilinux.com/?p=387" \o "Permanent Link to 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/>

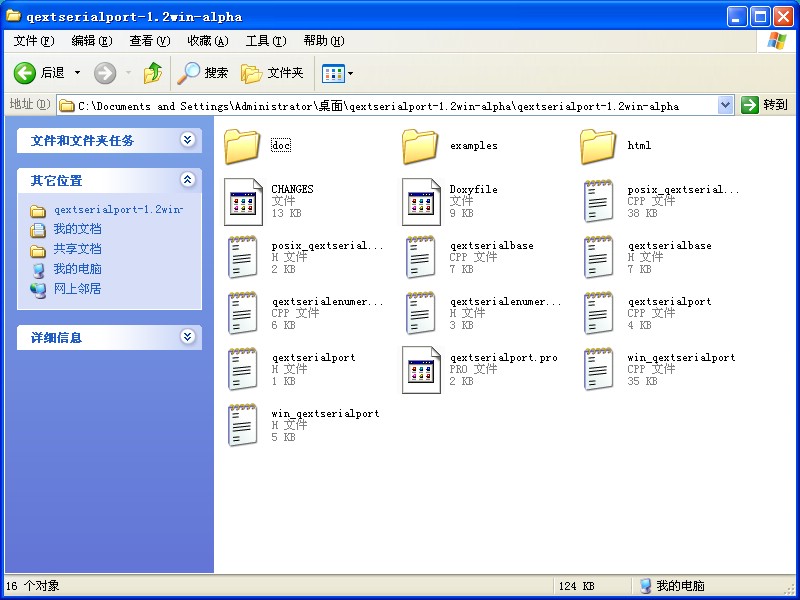
也可以下载我上传到网盘上的：

<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）**

[](http://j.imagehost.org/0610/1_96.jpg)

**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。

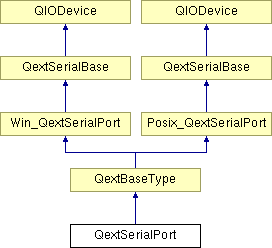
（3）html文件夹中是QextSerialPort类的使用文档。

（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。

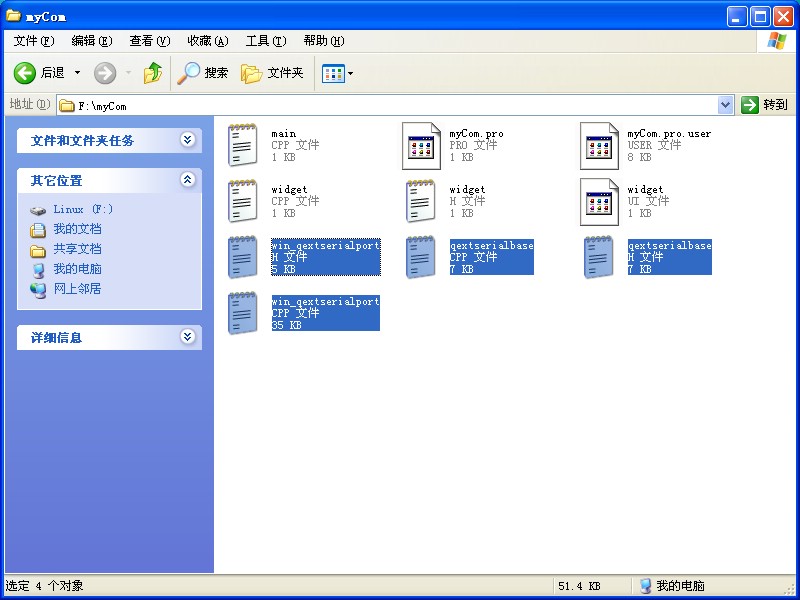
**第一，下面我们首先使用事件驱动来实现串口通信。**

**1.新建工程。**

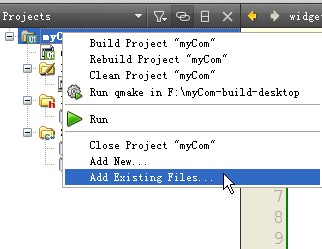
我们在Qt Creator中新建Qt Gui工程，命名为myCom，Base Class选择QWidget。

**2.添加文件。**

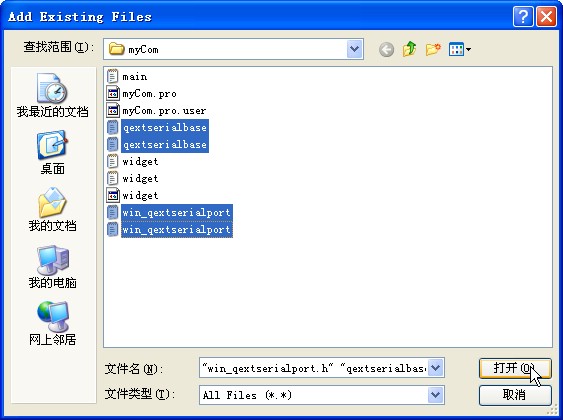
  我们将那四个文件添加到工程文件夹中。如下图。

[](http://i.imagehost.org/0420/win1.jpg)

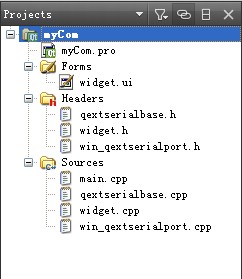
然后我们将这四个文件添加到工程中，在Qt Creator的工程列表中的工程文件夹上点击鼠标右键，在弹出的菜单中选择“Add Existing Files”菜单。如下图。



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。

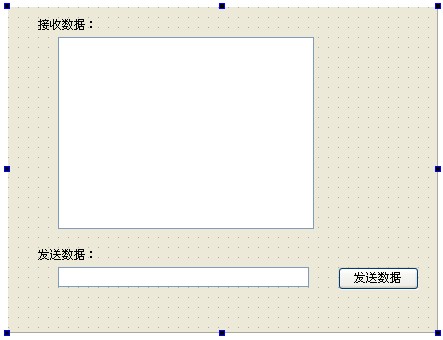
[](http://a.imagehost.org/0626/win3.jpg)

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”;

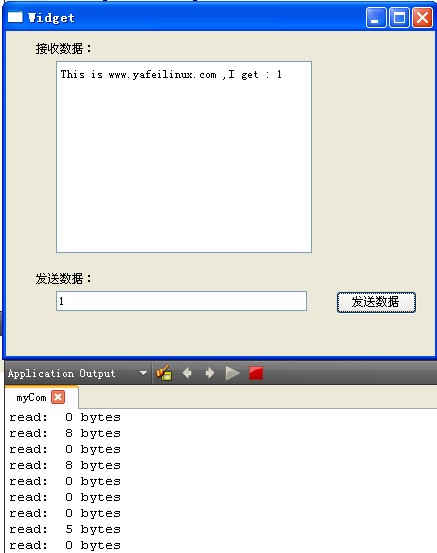
    //我们输出每次获得的字节数

    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”;

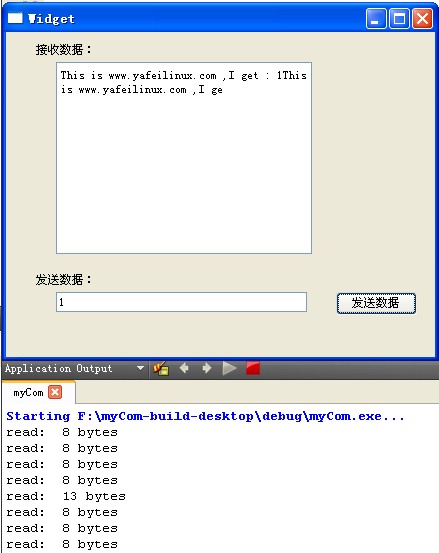
        QByteArray temp = myCom->readAll();

        ui->textBrowser->insertPlainText(temp);

   }

}

运行程序，效果如下：

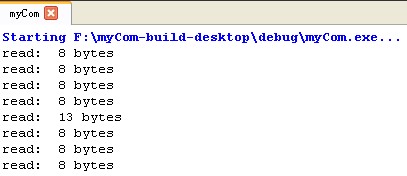


我们发送了两次数据，可以看到，这样实现了每8个字节读取一次，而最后剩余的不够8个字节的数据将会和后面的数据一起读出。

然后我们将8改为3，发送一次数据，效果如下：



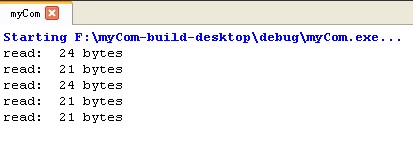
改为7，发送两次数据，效果如下：



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

{

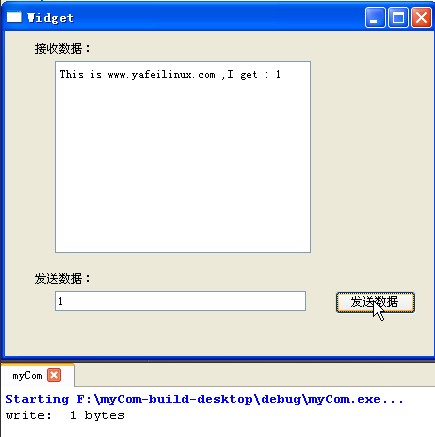
    myCom->write(ui->lineEdit->text().toAscii());

    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**

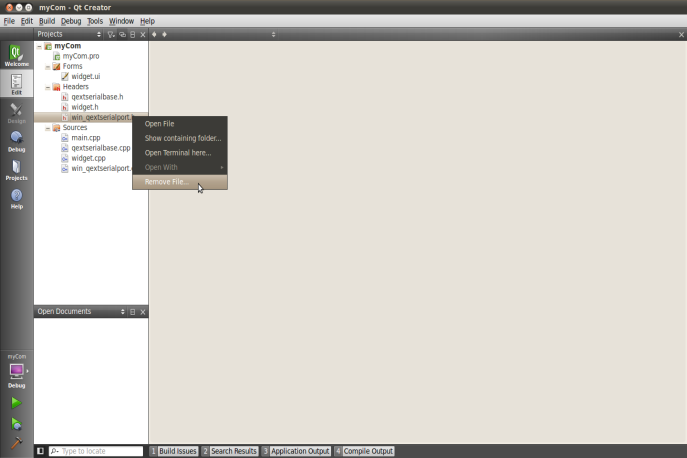
我这里的环境是Ubuntu 10.04，Qt 4.6.3和Qt Creator2.0 。上面已经提到，在Linux下只能使用Polling的方式读取串口数据，所以我们将上面Windows下的应用Polling的程序在Linux下重新编译。我们使用Qt Creator打开该工程，然后进行下面的操作。

**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。

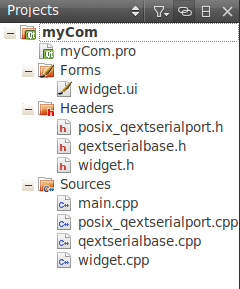
[](http://a.imagehost.org/0645/lin1.png)

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。

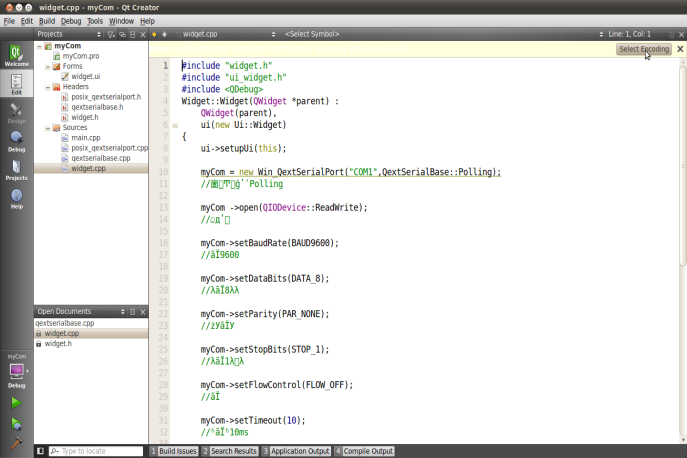
**（2）我们按照Windows下添加文件的方法，向工程中添加posix\_qextserialport.cpp和posix\_qextserialport.h文件。最终工程文件列表如下。**



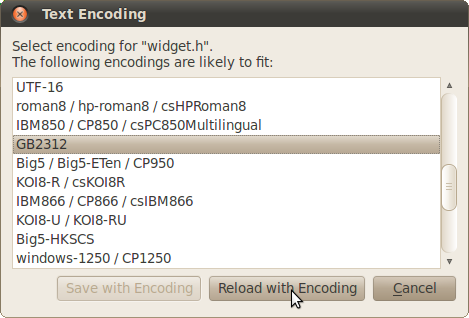
**2.设置编码。**

（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。

[](http://i.imagehost.org/0240/lin4.png)

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：

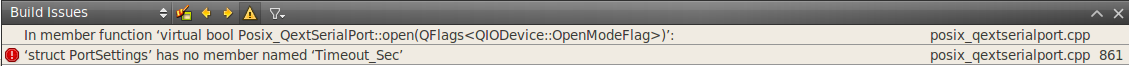
将以前的myCom = new Win\_QextSerialPort(“COM1″,QextSerialBase::Polling);

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。

[](http://a.imagehost.org/0854/lin6.png)

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。

Hosted by ImageHost.org

**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](http://www.yafeilinux.com/)   邮箱：[yafeilinux@vip.163.com](mailto:yafeilinux@vip.163.com)  QQ群：158054692

合作者 hqwfreefly

邮箱：[hqwemail@163.com](mailto:hqwemail@163.com)   个人主页：<http://hi.baidu.com/hqwfreefly>