## 基于DDS的QT聊天小程序

环境：rehl6.2 x64

ndds.5.1.0

Qt 4.8.6

## 制作IDL文件

当我们需要结合QT和DDS来构建一个聊天小程序的时候，第一步应该是创建IDL文件，IDL文件的作用是保存所有传输数据的类型，本聊天程序只需要传输文字信息，因此只需要定义一个string类型变量即可，文件名保存为minichat.idl，如图1-1所示。

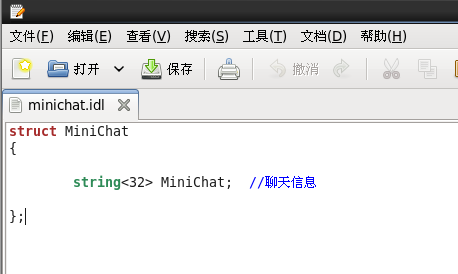


图1-1 IDL文件内容

## 利用IDL文件生成DDS示例代码

得到IDL文件之后，我们使用RTI工具可以生成C++语言的示例运行代码。步骤如下：（1）在minichat.idl文件所在文件夹中打开一个终端，输入：



**提供直接复制的代码：**

rtiddsgen -language C++ -example x64Linux2.6gcc4.1.1 -replace -ppDisable minichat.idl

执行后会生成如图2-1所示文件。

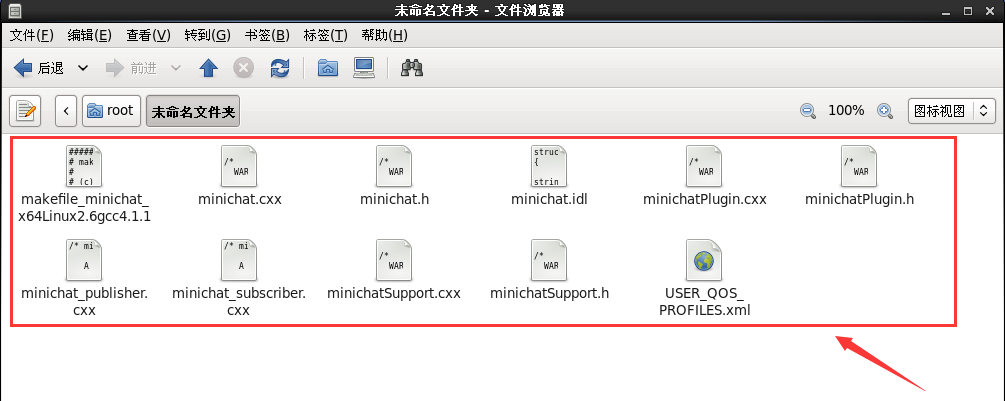


图2-1 示例代码文件

（2）为了检验生成示例代码是否正确，我们进行测试。继续在该终端中输入：

**提供直接复制的代码：**

make -f makefile\_minichat\_x64Linux2.6gcc4.1.1

将会得到一个objs文件夹，文件夹中有minichat\_publisher和minichat\_subscriber两个可执行文件，它们一个作为发送端，一个作为接收端，如图2-2所示：

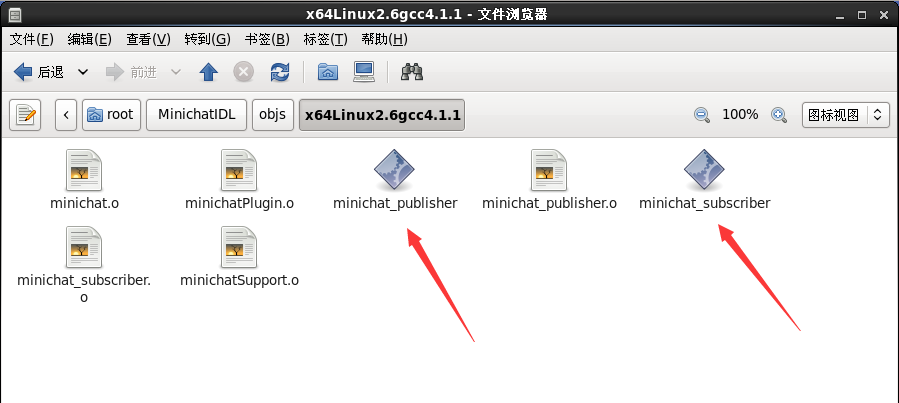


图2-2 objs文件夹

分别在两个终端中运行这两个可执行文件，如果得到图2-3，图2-4所示效果，则说明示例代码生成没有问题。

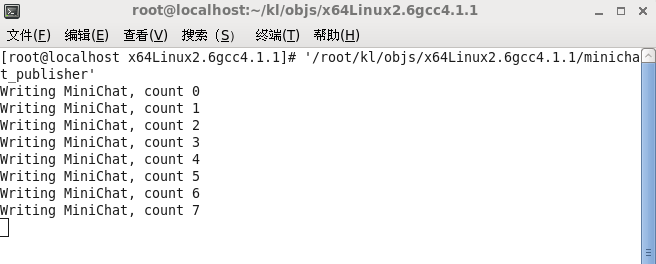


图2-3 minichat\_publisher 文件运行图

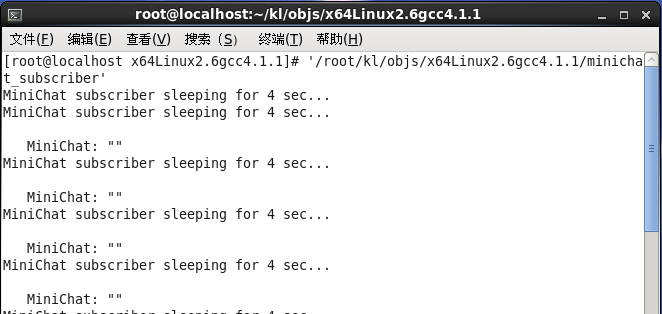


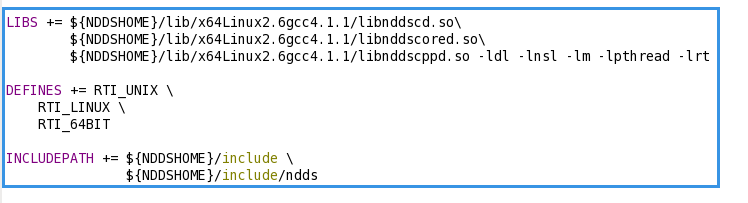
图2-4 minichat\_subscriber文件运行图（说明接收到了消息）

## 制作消息发送端程序

我们的聊天应用分为两个部分，一个是发送端程序，一个是接收端程序，先来讲发送端程序的制作。

#### 3.1创建QT工程

打开Qt Creator创建一个Qt GUI应用，工程名字为pub，并在pub.pro中添加如下代码：



**提供直接复制的代码：**

LIBS += ${NDDSHOME}/lib/x64Linux2.6gcc4.1.1/libnddscd.so\

${NDDSHOME}/lib/x64Linux2.6gcc4.1.1/libnddscored.so\

${NDDSHOME}/lib/x64Linux2.6gcc4.1.1/libnddscppd.so -ldl -lnsl -lm -lpthread -lrt

DEFINES += RTI\_UNIX \

RTI\_LINUX \

RTI\_64BIT

INCLUDEPATH += ${NDDSHOME}/include \

${NDDSHOME}/include/ndds

#### 3.2 编辑UI界面

作为发送端，本界面只需要三个部件组成，一个输入框，一个显示框，一个发送按钮。使用qt design拖拽两个text edit控件充当输入框和显示框，一个push button控件充当发送按钮。如图3-2-1所示。

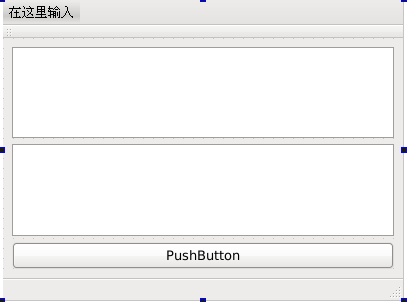


图3-2-1 发送端界面

接着我们需要修改控件对象名，输入窗口对象名为writeMessageEdit

显示窗口对象名为showMessageEdit，发送按钮对象名为sendButton。

#### 3.3 代码实现

3.3.1 引入DDS示例代码

将图3-3-1所示的DDS生成示例文件添加到pub工程中。

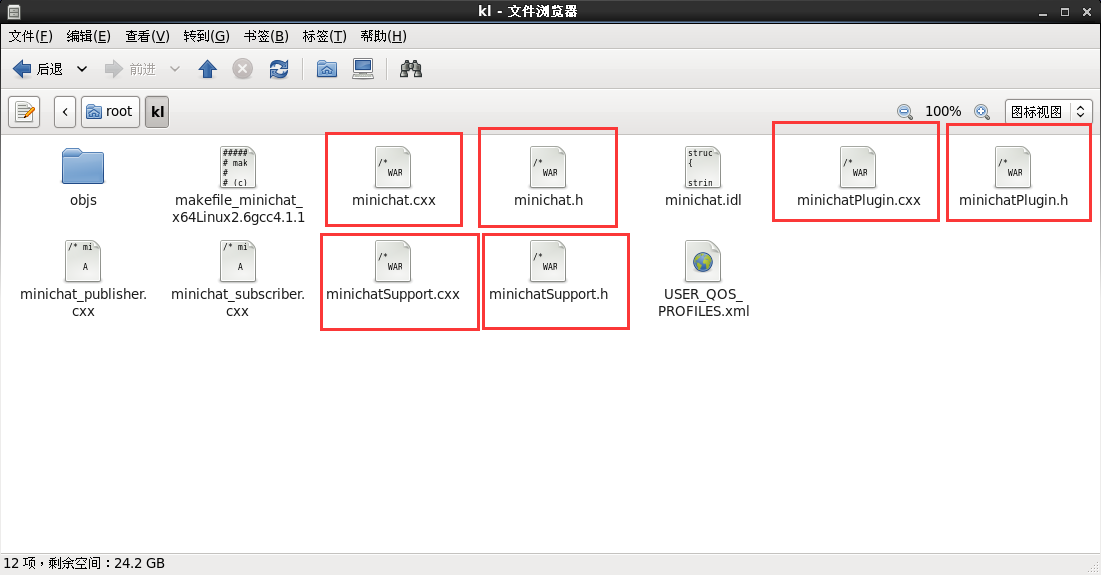


图3-3-1 需要用到的DDS示例代码文件

3.3.2 编写业务代码

（1）main.cpp



图3-3-2 main.cpp

**提供直接复制的代码：**

QTextCodec \*codec = QTextCodec::codecForName("System");

QTextCodec::setCodecForTr(codec);

QTextCodec::setCodecForCStrings(codec);

QTextCodec::setCodecForLocale(codec);

添加如图所示的代码，防止中文消息出现乱码。

（2）mainwindow.cpp

|  |  |
| --- | --- |
| 函数名 | 功能 |
| changeButtonStateSlot（） | 当存在输入信息时，发送按钮可用 |
| appendMessageSlot（） | 当点击发送按钮时，触发该槽函数，  (1)获取输入框的数据，  (2)触发信号:将该数据发送到至testthread类对象，修改变量content的值  (3）调用t->start（）函数，启动多线程 |

关键点：

1. 对象t是QThread类型，也就是多线程类型，t->start()是表示调用对象t的run函数，run函数会自动运行在新的线程之中，所以数据发送过程必须放到run函数中，如果不了解多线程的话，可以参照《C++ GUI Qt 4 编程》第14章。

（3）testthread.cpp

|  |  |
| --- | --- |
| 函数名 | 功能 |
| run() | 将变量message的值用write（）函数发送出去 |
| publishDDS() | 修改变量message的值为输入框的值 |
| setupQrtiDDS() | 设置域为0，设置participant，publisher，topic，writer的QoS（服务质量策略） |
| publisher\_shutdown（） | 回收域参与者 |

关键点：

这部分代码是核心，需要我们把生成示例代码文件minichat\_publisher.cxx中的代码修改一下，拿来使用即可，可以不懂具体代码意义，但是要学会使用这个套路。

1. 函数setupQrtiDDS()的功能可以看做DDS服务的一些准备工作，如果要使用DDS分发数据，就必须做好这些准备工作，否则报错。

首先设置域为0，然后下一部分代码直接把如下图所示在minichat\_publisher.cxx文件的代码复制过来即可。

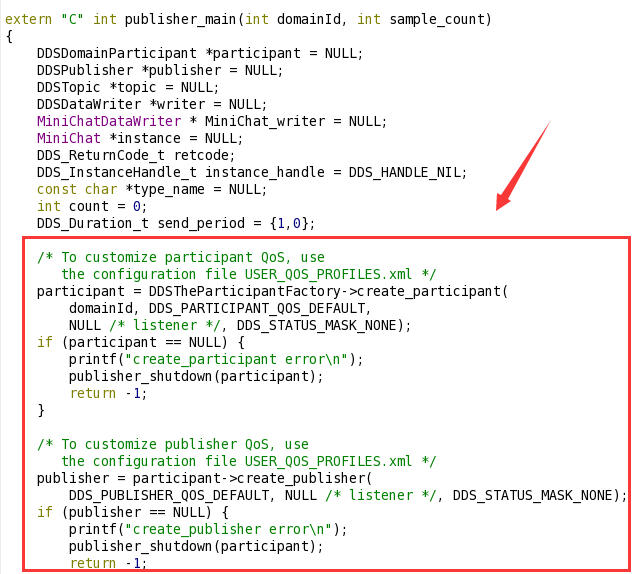


图3-3-3 DDS准备工作代码（1）

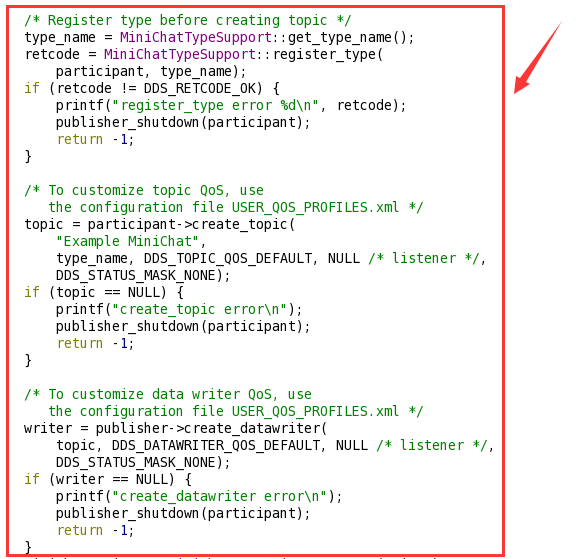


图3-3-4 DDS准备工作代码（2）

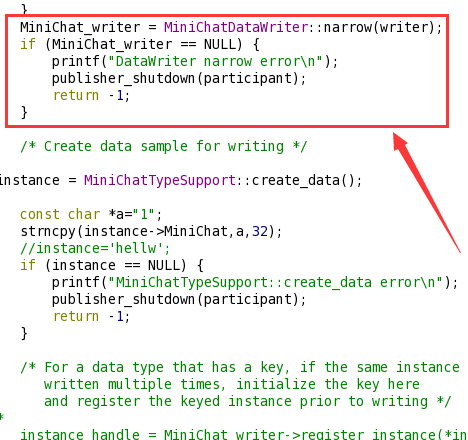


图3-3-5 DDS准备工作代码（3）

1. run（）函数起到的作用是发射数据，我们把minichat\_publisher.cxx文件中分发数据的代码复制过来，并且加以修改，主要修改是
2. 添加一句代码



**提供直接复制的代码：**

strncpy(instance->MiniChat,message.toUtf8().constData(),32);

它的作用是把message变量添加到instance，因为我们最终发出的数据都要存在instance中。

2．去掉if判断语句，因为run函数无返回值

3．修改for循环判断语句 判断条件为count<1，因为我们一次只需要发送一条数据

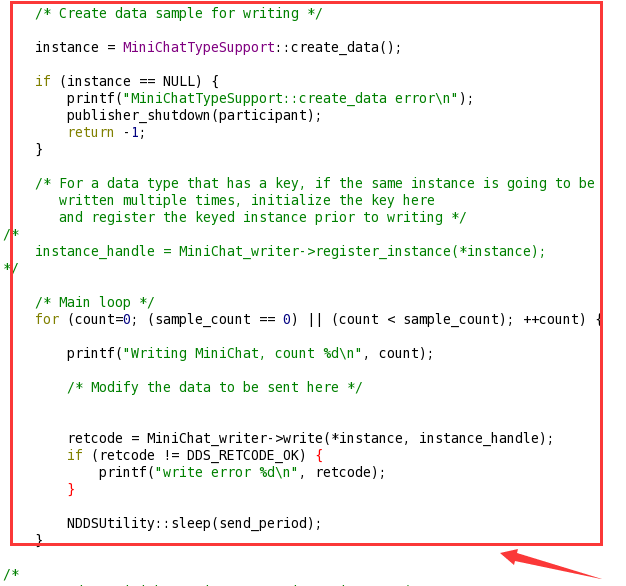


图3-3-6 DDS分发数据代码

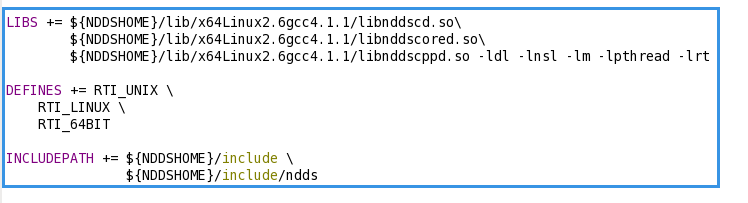
1. publisher\_shutdown（）函数最为简单，直接复制minichat\_publisher.cxx文件中的publisher\_shutdown（）代码即可，记得在头文件中声明该函数。

## 制作消息接收端程序

制作完消息发送端程序之后，我们开始制作消息接收端程序。

#### 4.1创建QT工程

打开Qt Creator创建一个Qt GUI应用，工程名字为sub，并在sub.pro中添加如下代码：

**提供直接复制的代码：**

LIBS += ${NDDSHOME}/lib/x64Linux2.6gcc4.1.1/libnddscd.so\

${NDDSHOME}/lib/x64Linux2.6gcc4.1.1/libnddscored.so\

${NDDSHOME}/lib/x64Linux2.6gcc4.1.1/libnddscppd.so -ldl -lnsl -lm -lpthread -lrt

DEFINES += RTI\_UNIX \

RTI\_LINUX \

RTI\_64BIT

INCLUDEPATH += ${NDDSHOME}/include \

${NDDSHOME}/include/ndds

#### 4.2 编辑UI界面

作为接收端，本界面只需要一个部件组成，即一个显示框，如图4-2-1所示。

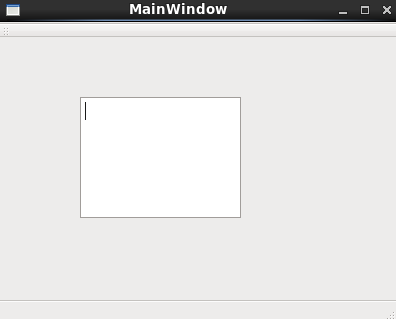


图4-2-1 发送端界面

接着我们需要修改控件对象名，显示窗口对象名为showMessage。

#### 4.3 代码实现

4.3.1 引入DDS示例代码

将图3-3-1所示的DDS生成示例文件添加到sub工程中。

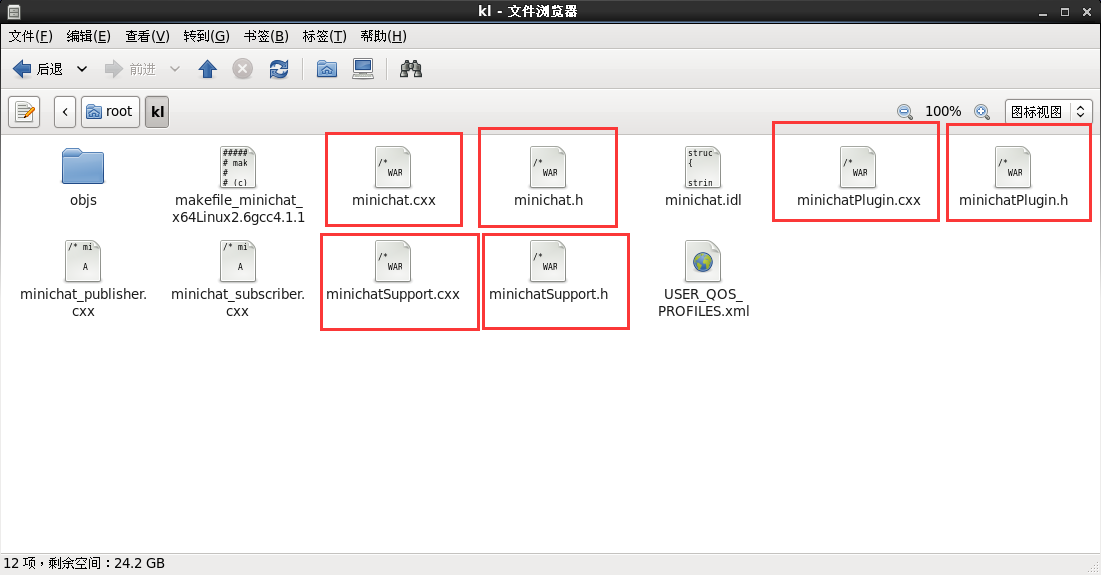


图4-3-1 需要用到的DDS示例代码文件

4.3.2 编写业务代码

（1）main.cpp



图4-3-2 main.cpp

**提供直接复制的代码：**

QTextCodec \*codec = QTextCodec::codecForName("System");

QTextCodec::setCodecForTr(codec);

QTextCodec::setCodecForCStrings(codec);

QTextCodec::setCodecForLocale(codec);

添加如图所示的代码，防止中文消息出现乱码。

添加如图所示的代码，防止中文消息出现乱码。

（2）mainwindow.cpp

|  |  |
| --- | --- |
| 函数名 | 功能 |
| setUp\_rtiDDS\_handler（） | 实例化rtiDDSlistener ，rtiDDS对象 |
| readListenerData（） | 将监听得到的数据发送到显示框 |

关键点：

1. 实例化rtiDDSlistener ，rtiDDS对象，主要是为了信号与槽的连接，有关于这两个对象的功能将在它们自己的部分详细介绍。

（3）qrtidds.cpp

|  |  |
| --- | --- |
| 函数名 | 功能 |
| setupQrtiDDS() | 设置域为0，设置participant，publisher，topic，writer的QoS（服务质量策略） |
| subscriber\_shutdown（） | 回收域参与者 |

关键点：

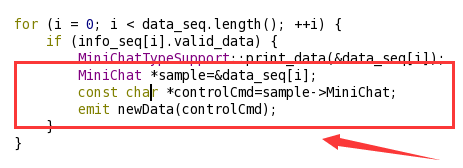
1. 如果有细心的同学就会发现，setupQrtiDDS()的代码构成很明显和发送端程序中setupQrtiDDS()的代码是一样的，是的，我们只需要把minichat\_subscriber.cxx文件中相同位置的代码复制过来，然后将create\_datareader（）中的监听器参数修改为我们自己定义的rtiDDSlistener 即可。
2. subscriber\_shutdown（）更简单了，minichat\_subscriber.cxx文件中相同函数名代码复制即可。

（4）qrtiddslistener.cpp

|  |  |
| --- | --- |
| 函数名 | 功能 |
| on\_data\_available () | 监听发送过来的数据，并发射信号通知显示窗口 |

关键点：

1. 这个类比较特殊，为了同时兼顾使用监听器功能以及能够使用信号与槽的机制，我们必须继承QObject，DDSDataReaderListener两个类
2. 监听器类的原理很简单，我们只需要继承DDSDataReaderListener类，并且重写on\_data\_available（）方法即可。
3. 所以我们将minichat\_subscriber.cxx文件中on\_data\_available（）方法的代码复制过来，然后稍作修改即可，就是在for循环中添加如下代码



它表示是每当有数据传送过来的时候，就发射信号给UI界面，让界面显示出来。

## 程序效果

同时运行发送端和接收端程序，输入端输入消息后，点击发送，接收端立刻显示该消息，如图5-1所示。

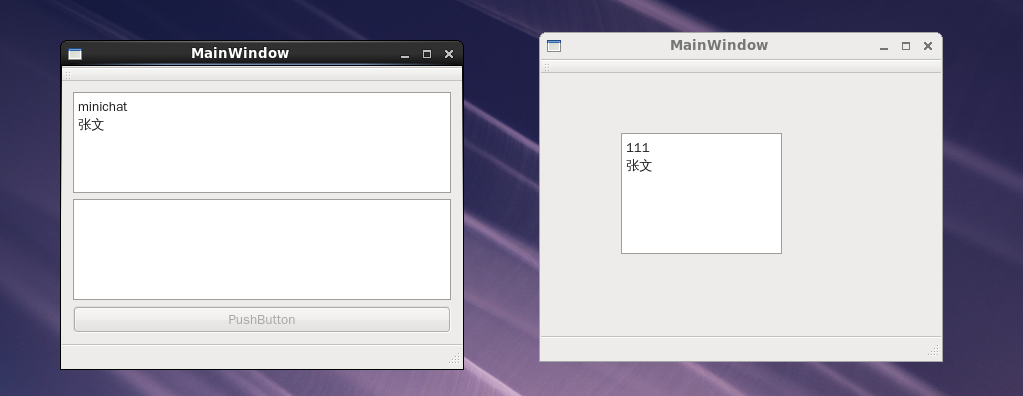


图5-1 程序效果图