TCP局域网聊天（2021.11.29）

不更新了
直接这个版本完结了，虽然还有许多内容值得完善

服务端:
cha.pro

```
QT += core gui

QT += network

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets



CONFIG += c++11



# You can make your code fail to compile if it uses
deprecated APIs.

# In order to do so, uncomment the following line.

#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000 #
disables all the APIs deprecated before Qt 6.0.0
```

```
SOURCES += \

 main.cpp \

 mainwindow.cpp



HEADERS += \

 mainwindow.h



FORMS += \

 mainwindow.ui



# Default rules for deployment.

qnx: target.path = /tmp/$${TARGET}/bin

else: unix:!android: target.path = /opt/$${TARGET}/bin

!isEmpty(target.path): INSTALLS += target
```

mainwindow.h

```

```

```
#ifndef MAINWINDOW_H

#define MAINWINDOW_H



#include <QMainWindow>

#include <QTcpServer>

#include <QTcpSocket>

#include <QVector>

#include <QStringListModel>



QT_BEGIN_NAMESPACE

namespace Ui { class MainWindow; }

QT_END_NAMESPACE



class MainWindow : public QMainWindow

{
```

```
 Q_OBJECT



public:

 QTcpServer *mserver; //创建QTcpServer对象

 QTcpSocket *msocket; //创建QTcpSocket对象

 //int num; //用户人数



 //用来显示谁在线

 QStringList who;

 QStringListModel *model;



 //学习链接：多客户端
https://blog.csdn.net/weixin_44916364/article/details/10058610

 QVector<QTcpSocket*> socketlist;
```

```
 MainWindow(QWidget *parent = nullptr);
 ~MainWindow();

private slots:
 void on_pushButton_clicked();
 void new_client(); //接收连接——获取与客户端通信的套接字
 void read_data(); //读取数据

 void send_data(QString which); //发送信息到客户端

 void on_pushButton_2_clicked();
 void on_pushButton_3_clicked();

private:
 Ui::MainWindow *ui;
```

```
};

#endif // MAINWINDOW_H
```

main.cpp

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
 QApplication a(argc, argv);
 MainWindow w;
 w.show();
 return a.exec();
}
```

mainwindow.cpp

```
#include "mainwindow.h"

#include "ui_mainwindow.h"



MainWindow::MainWindow(QWidget *parent):
QMainWindow(parent), ui(new Ui::MainWindow)
{

    ui->setupUi(this);

    //关联客户端连接信号(当有客户端连接的时候QTcpServer对象会发送
newConnection信号--关联槽函数)
    //1.创建QTcpSever对象
    mserver = new QTcpServer(this);
    msocket = NULL;

    //num = 0;

    //3.当服务器被客户端访问时，会发出newConnection()信号，因此为
该信号添加槽函数，并用一个QTcpSocket对象接受客户端访问

connect(mserver,&QTcpServer::newConnection,this,&MainWindow::n



}
```

```
MainWindow::~MainWindow()
{
    delete ui;
}



//监听——启动服务器
void MainWindow::on_pushButton_clicked()
{
    //2.侦听一个端口，使得客户端可以使用这个端口访问服务器
    //（listen(ip, port)）
    if( ui->textEdit->toPlainText() == "" ){            //
判断是否为空
        ui->textBrowser->append("服务器启动失败，请输入要监听的
端口号!");
    }else{
        mserver->listen(QHostAddress::Any,ui->textEdit-
>toPlainText().toUShort());
        ui->textBrowser->append("服务器启动成功，正在监听端
口"+ui->textEdit->toPlainText()+"!");
    }
}



//接受连接

void MainWindow::new_client()
{
```

```
{
    //获取与客户端通信的套接字
    msocket = mserver->nextPendingConnection();

    socketlist.append(msocket);          //将客户端的套接字加入
容器中

    //4.使用socket的write函数向客户端发送数据
    msocket->write("服务器连接成功! ");

    //获取客户端IP并显示
    QString ip = msocket->peerAddress().toString();
    ip.remove("::ffff:");
    quint16 port = msocket->peerPort();
    QString tmp = QString("客户端[%1:%2] 已上
线!").arg(ip).arg(port);
    ui->textBrowser->append(tmp);

    //关联读数据信号(客户端有数据到达服务器QTcpSocket对象会发送
readyRead信号--关联槽函数)
    //5.当socket接收缓冲区有新数据到来时，会发出readRead()信号，因
此为该信号添加槽函数以读取数据

connect(msocket,&QTcpSocket::readyRead,this,&MainWindow::read_

}



//读取数据

void MainWindow::read_data()
{
```

```
{
    //获取信号发送者
    QTcpSocket *msocket1 = dynamic_cast<QTcpSocket*>
(sender());

    //读取数据
    QString msg = msocket1->readAll();
    //获取对方IP和端口
    QString ip = msocket1->peerAddress().toString();
    quint16 port = msocket1->peerPort();
    ip.remove("::ffff:");
    //quint16 port = msocket1->peerPort();
    //QString tmp = QString("客户端[%1:%2]:
%3").arg(ip).arg(port).arg(msg);
    if( msg != "@断开连接@" ){              //如果传过来的不是
断开连接的信息
        QString tmp = msg;
        ui->textBrowser->append(tmp);

        //传送到每个客户端里面
        for(int i = 0 ; i < socketlist.size() ; i ++ ){
            socketlist[i]->write(tmp.toUtf8());
        }
    }else{              //如果传过来是断开连接的信息
        //num--;            //在线人数减少
        QString tmp = QString("客户端[%1:%2] 已下
线!").arg(ip).arg(port);
        ui->textBrowser->append(tmp);
        //QString tmp = "系统信息: "+ui->textEdit_2-
>toPlainText();

        //send_data(tmp);;
```

```
    }

}

//断开连接
void MainWindow::on_pushButton_2_clicked()
{
    //断开连接
    ui->textBrowser->append("服务器已经关闭!");
    mserver->disconnect();
    //num = 0;

}

//发送按钮
void MainWindow::on_pushButton_3_clicked()
{

    //向客户端发送信息
    //获取与客户端通信的套接字
    //QTcpSocket *msocket = mserver-
>nextPendingConnection();
    //使用socket的write函数向客户端发送数据
    QString tmp = "系统信息："+ui->textEdit_2->toPlainText();
    //注意这里要toUtf8，否则会报错
    //msocket->write(tmp.toUtf8());
    send_data(tmp);



}
```

```
 //发送信息到客户端
void MainWindow::send_data(QString which){
    //向每个客户端发送
    for(int i = 0 ; i < socketlist.size() ; i++ ){
        socketlist[i]->write(which.toUtf8());
    }

    //服务器本身日志添加
    ui->textBrowser->append(which);

}
```

mainwindow.ui

```
<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

 <class>MainWindow</class>

 <widget class="QMainWindow" name="MainWindow">

 <property name="windowModality">

 <enum>Qt::NonModal</enum>
```

```
 </property>

 <property name="geometry">

 <rect>

 <x>0</x>

 <y>0</y>

 <width>363</width>

 <height>456</height>

 </rect>

 </property>

 <property name="windowTitle">

 <string>聊天室 server</string>

 </property>

 <property name="layoutDirection">

 <enum>Qt::LeftToRight</enum>

 </property>

 <widget class="QWidget" name="centralwidget">
```

```
<widget class="QPushButton" name="pushButton">

<property name="geometry">

<rect>

<x>50</x>

<y>370</y>

<width>101</width>

<height>28</height>

</rect>

</property>

<property name="text">

<string>启动服务器</string>

</property>

</widget>

<widget class="QTextBrowser" name="textBrowser">

<property name="geometry">
```

```
<rect>

<x>10</x>

<y>20</y>

<width>321</width>

<height>241</height>

</rect>

</property>

<property name="placeholderText">

<string>日志</string>

</property>

</widget>

<widget class="QPushButton" name="pushButton_2">

<property name="geometry">

<rect>

<x>170</x>

<y>370</y>
```

```
<width>111</width>

<height>28</height>

</rect>

</property>

<property name="text">

<string>断开连接</string>

</property>

</widget>

<widget class="QTextEdit" name="textEdit_2">

<property name="geometry">

<rect>

<x>50</x>

<y>290</y>

<width>131</width>

<height>31</height>
```

```
 </rect>

 </property>

 <property name="html">

 <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML
4.0//EN&quot; &quot;http://www.w3.org/TR/REC-
html40/strict.dtd&quot;&gt;

&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot;
content=&quot;1&quot; /&gt;&lt;style
type=&quot;text/css&quot;&gt;

p, li { white-space: pre-wrap; }

&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-
family:'SimSun'; font-size:9pt; font-weight:400; font-
style:normal;&quot;&gt;

&lt;p style=&quot;-qt-paragraph-type:empty; margin-top:0px;
margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-
block-indent:0; text-indent:0px;&quot;&gt;&lt;br
/&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>

 </property>

 <property name="placeholderText">

 <string/>
```

```
</property>

</widget>

<widget class="QPushButton" name="pushButton_3">

<property name="geometry">

<rect>

<x>200</x>

<y>290</y>

<width>81</width>

<height>28</height>

</rect>

</property>

<property name="text">

<string>发送</string>

</property>

</widget>

<widget class="QLabel" name="label">
```

```
<property name="geometry">

<rect>

<x>60</x>

<y>330</y>

<width>101</width>

<height>21</height>

</rect>

</property>

<property name="text">

<string>端口: </string>

</property>

</widget>

<widget class="QTextEdit" name="textEdit">

<property name="geometry">

<rect>
```

```
 <x>120</x>

 <y>330</y>

 <width>151</width>

 <height>31</height>

 </rect>

 </property>

 <property name="html">

 <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML
4.0//EN&quot; &quot;http://www.w3.org/TR/REC-
html40/strict.dtd&quot;&gt;

&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot;
content=&quot;1&quot; /&gt;&lt;style
type=&quot;text/css&quot;&gt;

p, li { white-space: pre-wrap; }

&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-
family:'SimSun'; font-size:9pt; font-weight:400; font-
style:normal;&quot;&gt;

&lt;p style=&quot; margin-top:0px; margin-bottom:0px;
margin-left:0px; margin-right:0px; -qt-block-indent:0;
text-
```

```
indent:0px;&amp;quot;&amp;gt;9999&amp;lt;/p&amp;gt;&amp;lt;/body&amp;gt;&amp;lt;/html&amp;gt;
</string>

 </property>

 <property name="placeholderText">

 <string/>

 </property>

 </widget>

 </widget>

 <widget class="QMenuBar" name="menubar">

 <property name="geometry">

 <rect>

 <x>0</x>

 <y>0</y>

 <width>363</width>

 <height>26</height>

 </rect>
```

```
 </property>

 </widget>

 <widget class="QStatusBar" name="statusbar"/>

 </widget>

 <resources/>

 <connections/>

</ui>
```

客户端代码
chatt_kh.pro

```
QT += core gui

QT += network



greaterThan(QT_MAJOR_VERSION, 4): QT += widgets



CONFIG += c++11


```

```
# You can make your code fail to compile if it uses
deprecated APIs.

# In order to do so, uncomment the following line.

#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000 #
disables all the APIs deprecated before Qt 6.0.0



SOURCES += \

 main.cpp \

 mainwindow.cpp



HEADERS += \

 mainwindow.h



FORMS += \

 mainwindow.ui
```

```
# Default rules for deployment.

qnx: target.path = /tmp/$${TARGET}/bin

else: unix:!android: target.path = /opt/$${TARGET}/bin

!isEmpty(target.path): INSTALLS += target
```

mainwindow.h

```


#ifndef MAINWINDOW_H

#define MAINWINDOW_H



#include <QMainWindow>

#include <QMessageBox>

#include <QTcpServer>

#include <QTcpSocket>

#include <time.h>
```

```
QT_BEGIN_NAMESPACE

namespace Ui { class MainWindow; }

QT_END_NAMESPACE



class MainWindow : public QMainWindow

{

 Q_OBJECT



public:

 QString idname; //存储id名字

 QTcpSocket *msocket; //创建QTcpSocket对象

 MainWindow(QWidget *parent = nullptr);

 ~MainWindow();



private slots:

 void read_data();
```

```
 void on_pushButton_2_clicked();

 void on_pushButton_clicked();



 void on_pushButton_3_clicked();



private:

 Ui::MainWindow *ui;

};

#endif // MAINWINDOW_H
```

main.cpp

```


#include "mainwindow.h"



#include <QApplication>


```

```
int main(int argc, char *argv[])

{

 QApplication a(argc, argv);

 MainWindow w;

 w.show();

 return a.exec();

}
```

mainwindow.cpp

```
#include "mainwindow.h"

#include "ui_mainwindow.h"



MainWindow::MainWindow(QWidget *parent)

 : QMainWindow(parent)

 , ui(new Ui::MainWindow)
```

```
{

 ui->setupUi(this);

}



MainWindow::~MainWindow()

{

 delete ui;

}



//连接服务器

void MainWindow::on_pushButton_2_clicked()

{

 if( ui->textEdit_4->toPlainText() == "" ){

 QMessageBox::warning(this,tr("提示"),tr("请输入聊天用户
名"),tr("确定"));

 return;
```

```
 }else if( ui->textEdit->toPlainText() == "" ){

 QMessageBox::warning(this,tr("提示"),tr("请输入IP地
址"),tr("确定"));

 return;

 }else if( ui->textEdit_2->toPlainText() == "" ){

 QMessageBox::warning(this,tr("提示"),tr("请输入端口"),tr("确
定"));

 return;


 }

 //存储id名字

 idname = ui->textEdit_4->toPlainText();

 msocket = new QTcpSocket(this);

 //关联读数据信号(当QTcpSocket有数据可读会发送readyRead信号--关联
槽函数)


connect(msocket,&QTcpSocket::readyRead,this,&MainWindow::read_


 //调用对象方法链接服务器（connectToHost（服务器的IP，  端口））
```

```
 //msocket->connectToHost(ui->textEdit->toPlainText(),ui->textEdit_2->toPlainText().toUShort());

 msocket->connectToHost( QHostAddress::LocalHost,ui->textEdit_2->toPlainText().toUShort());




 //禁用按钮以及输入框(412,2),防止重复点击

 ui->pushButton_2->setDisabled(true);

 ui->textEdit->setDisabled(true);

 ui->textEdit_2->setDisabled(true);

 ui->textEdit_4->setDisabled(true);

}



void MainWindow::read_data()

{

 //读取数据

 QString msg = msocket->readAll();
```

```
 ui->textBrowser->append(msg);

}




//发送

void MainWindow::on_pushButton_clicked()

{


 if( msocket->state() == QAbstractSocket::ConnectedState ){
//判断连接状态：如果连接

 //向服务器发送数据

 QString data = QString("%1： %2").arg(idname).arg(ui-
>textEdit_3->toPlainText());

 msocket->write(data.toUtf8());

 }

 //客户端日志

 //QString tmp = QString("客户端： %1").arg(data);
```

```
 //ui->textBrowser->append(tmp);

}



//断开连接

void MainWindow::on_pushButton_3_clicked()

{

 if( msocket->state() == QAbstractSocket::ConnectedState ){
//判断连接状态：如果连接


 //断开前发送信息给客户端

 QString data = "@断开连接@";

 msocket->write(data.toUtf8());

 //主动和对方断开连接

 msocket->disconnectFromHost();

 msocket->close();



 }
```

```
 //解除禁用按钮以及输入框(412,2)
 ui->pushButton_2->setDisabled(false);
 ui->textEdit->setDisabled(false);
 ui->textEdit_2->setDisabled(false);
 ui->textEdit_4->setDisabled(false);

}
```

mainwindow.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
 <property name="geometry">
 <rect>
```

```
<x>0</x>

<y>0</y>

<width>410</width>

<height>549</height>

</rect>

</property>

<property name="windowTitle">


<string>聊天室 client </string>

</property>

<widget class="QWidget" name="centralwidget">

<widget class="QTextEdit" name="textEdit">

<property name="geometry">

<rect>

<x>190</x>

<y>380</y>
```

```
<width>151</width>

<height>31</height>

</rect>

</property>

<property name="contextMenuPolicy">

<enum>Qt::ActionsContextMenu</enum>

</property>


<property name="toolTip">

<string/>

</property>

<property name="toolTipDuration">

<number>0</number>

</property>

<property name="statusTip">

<string/>

</property>
```

```
 <property name="whatsThis">

 <string/>

 </property>

 <property name="accessibleName">

 <string/>

 </property>

 <property name="accessibleDescription">


 <string/>

 </property>

 <property name="html">

 <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML
4.0//EN&quot; &quot;http://www.w3.org/TR/REC-
html40/strict.dtd&quot;&gt;

&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot;
content=&quot;1&quot; /&gt;&lt;style
type=&quot;text/css&quot;&gt;

p, li { white-space: pre-wrap; }
```

```
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-
family:'SimSun'; font-size:9pt; font-weight:400; font-
style:normal;&quot;&gt;

&lt;p style=&quot; margin-top:0px; margin-bottom:0px;
margin-left:0px; margin-right:0px; -qt-block-indent:0;
text-
indent:0px;&quot;&gt;127.0.0.1&lt;/p&gt;&lt;/body&gt;&lt;/html
</string>

 </property>

 <property name="placeholderText">


 <string/>

 </property>

 </widget>

 <widget class="QTextEdit" name="textEdit_2">

 <property name="geometry">

 <rect>

 <x>190</x>

 <y>420</y>

 <width>151</width>
```

```
 <height>31</height>

 </rect>

 </property>

 <property name="html">

 <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML
4.0//EN&quot; &quot;http://www.w3.org/TR/REC-
html40/strict.dtd&quot;&gt;

&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot;

content=&quot;1&quot; /&gt;&lt;style
type=&quot;text/css&quot;&gt;

p, li { white-space: pre-wrap; }

&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-
family:'SimSun'; font-size:9pt; font-weight:400; font-
style:normal;&quot;&gt;

&lt;p style=&quot; margin-top:0px; margin-bottom:0px;
margin-left:0px; margin-right:0px; -qt-block-indent:0;
text-
indent:0px;&quot;&gt;9999&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;
</string>

 </property>
```

```
<property name="placeholderText">

<string/>

</property>

</widget>

<widget class="QPushButton" name="pushButton">

<property name="geometry">

<rect>


<x>252</x>

<y>290</y>

<width>111</width>

<height>28</height>

</rect>

</property>

<property name="text">

<string>发送</string>

</property>
```

```
</widget>

<widget class="QPushButton" name="pushButton_2">

<property name="geometry">

<rect>

<x>60</x>

<y>460</y>

<width>121</width>


<height>28</height>

</rect>

</property>

<property name="text">

<string>连接服务器</string>

</property>

</widget>

<widget class="QTextEdit" name="textEdit_3">
```

```
<property name="geometry">

<rect>

<x>50</x>

<y>290</y>

<width>191</width>

<height>31</height>

</rect>


</property>

<property name="placeholderText">

<string/>

</property>

</widget>

<widget class="QTextBrowser" name="textBrowser">

<property name="geometry">

<rect>

<x>50</x>
```

```
<y>20</y>

<width>321</width>

<height>261</height>

</rect>

</property>

<property name="placeholderText">

<string>信息框</string>


</property>

</widget>

<widget class="QPushButton" name="pushButton_3">

<property name="geometry">

<rect>

<x>220</x>

<y>460</y>

<width>121</width>
```

```
<height>28</height>

</rect>

</property>

<property name="text">

<string>断开连接</string>

</property>

</widget>


<widget class="QLabel" name="label">

<property name="geometry">

<rect>

<x>61</x>

<y>350</y>

<width>91</width>

<height>20</height>

</rect>

</property>
```

```
<property name="text">

<string>用户名：</string>

</property>

</widget>

<widget class="QLabel" name="label_2">

<property name="geometry">

<rect>


<x>61</x>

<y>390</y>

<width>91</width>

<height>20</height>

</rect>

</property>

<property name="text">

<string>服务器地址：</string>
```

```
</property>

</widget>

<widget class="QLabel" name="label_3">

<property name="geometry">

<rect>

<x>61</x>

<y>430</y>


<width>91</width>

<height>20</height>

</rect>

</property>

<property name="text">

<string>端口: </string>

</property>

</widget>

<widget class="QTextEdit" name="textEdit_4">
```

```
<property name="geometry">

<rect>

<x>190</x>

<y>340</y>

<width>151</width>

<height>31</height>

</rect>


</property>

</widget>

</widget>

<widget class="QMenuBar" name="menubar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>
```

```
  <width>410</width>

  <height>26</height>

  </rect>

  </property>

  </widget>

  <widget class="QStatusBar" name="statusbar"/>

  </widget>


  <resources/>

  <connections/>

 </ui>
```

（2021.11.1）

待补充的功能：

1.用户名

2.界面完善，参考（https://blog.csdn.net/zq9955/article/details/113542612?spm=1001.2101.3001.6650.1&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-1.no_search_link&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-1.no_search_link）

3.添加发送时间

4.私聊
5.显示在线人数以及人（服务器以及客户端）

其他杂七杂八：
文字类型
表情（颜文字）
文件传输
登陆注册验证

待完善：
1.服务端改成系统信息“”
2.连接服务器和断开连接的按钮的禁用
3.显示当前的连接状态

多客户端聊天（2021.10.31）

增加了：
多客户之间的传递
已完成：
1.客户端与服务器端信息的基本传递
1.服务器与客户端交互信息的完善
2.服务器可以向客户端发送信息

学习链接：多客户端
https://blog.csdn.net/weixin_44916364/article/details/100586104

关键:
1.套接字容器`QVector<QTcpSocket*> socketlist;`
2. 接受连接的时候，将套接字加入容器中

```
msocket = mserver->nextPendingConnection();
socketlist.append(msocket);          //将客户端的套接字加入容器中
```

3.接收数据的时候，或者发送数据的时候,利用套接字容器将信息发送到每个客户端

```
//传送到每个客户端里面
for(int i = 0 ; i < socketlist.size() ; i ++ ){
    socketlist[i]->write(tmp.toUtf8());
}
```

服务器端
mainwindow.h

```
#ifndef MAINWINDOW_H

#define MAINWINDOW_H



#include <QMainWindow>

#include <QTcpServer>

#include <QTcpSocket>

#include <QVector>



QT_BEGIN_NAMESPACE
```

```
namespace Ui { class MainWindow; }

QT_END_NAMESPACE



class MainWindow : public QMainWindow

{

 Q_OBJECT



public:

 QTcpServer *mserver; //创建QTcpServer对象

 QTcpSocket *msocket; //创建QTcpSocket对象



 //学习链接：多客户端
https://blog.csdn.net/weixin_44916364/article/details/10058610

 QVector<QTcpSocket*> socketlist;
```

```
 MainWindow(QWidget *parent = nullptr);
 ~MainWindow();


private slots:
 void on_pushButton_clicked();
 void new_client(); //接收连接——获取与客户端通信的套接字
 void read_data(); //读取数据


 void on_pushButton_2_clicked();
 void on_pushButton_3_clicked();


private:
 Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

main.cpp

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
 QApplication a(argc, argv);
 MainWindow w;
 w.show();
 return a.exec();
}
```

mainwindow.cpp

```
#include "mainwindow.h"
```

```
#include "ui_mainwindow.h"



MainWindow::MainWindow(QWidget *parent):
QMainWindow(parent), ui(new Ui::MainWindow)
{

    ui->setupUi(this);

    //关联客户端连接信号(当有客户端连接的时候QTcpServer对象会发送
newConnection信号--关联槽函数)
    //1.创建QTcpSever对象
    mserver = new QTcpServer(this);
    msocket = NULL;

    //3.当服务器被客户端访问时，会发出newConnection()信号，因此为
该信号添加槽函数，并用一个QTcpSocket对象接受客户端访问

connect(mserver,&QTcpServer::newConnection,this,&MainWindow::r



}



MainWindow::~MainWindow()
{
    delete ui;
}
```

```
//监听——启动服务器
void MainWindow::on_pushButton_clicked()
{
    //2.侦听一个端口，使得客户端可以使用这个端口访问服务器
    //（listen(ip, port)）
    if( ui->textEdit->toPlainText() == "" ){              //
判断是否为空
        ui->textBrowser->append("服务器启动失败，请输入要监听的
端口号!");
    }else{
        mserver->listen(QHostAddress::Any,ui->textEdit-
>toPlainText().toUShort());
        ui->textBrowser->append("服务器启动成功，正在监听端
口"+ui->textEdit->toPlainText()+"!");
    }
}



//接受连接
void MainWindow::new_client()
{
    //获取与客户端通信的套接字
    msocket = mserver->nextPendingConnection();

    socketlist.append(msocket);          //将客户端的套接字加入
容器中

    //1.使用msocket->write()函数向客户端发送数据
```

```
    //4.使用socket的write函数向客户端发送数据
    msocket->write("服务器连接成功! ");


    //获取客户端IP并显示
    QString ip = msocket->peerAddress().toString();
    ip.remove("::ffff:");
    quint16 port = msocket->peerPort();
    QString tmp = QString("客户端[%1:%2] 成功连
接!").arg(ip).arg(port);
    ui->textBrowser->append(tmp);



    //关联读数据信号(客户端有数据到达服务器QTcpSocket对象会发送
readyRead信号--关联槽函数)
    //5.当socket接收缓冲区有新数据到来时，会发出readRead()信号，因
此为该信号添加槽函数以读取数据

connect(msocket,&QTcpSocket::readyRead,this,&MainWindow::read_

}



//读取数据
void MainWindow::read_data()
{
    //获取信号发送者
    QTcpSocket *msocket1 = dynamic_cast<QTcpSocket*>
(sender());

    //读取数据
```

```
    //读取数据
    QString msg = msocket1->readAll();
    //获取对方IP和端口

    QString ip = msocket1->peerAddress().toString();
    ip.remove("::ffff:");
    quint16 port = msocket1->peerPort();
    QString tmp = QString("客户端[%1:%2]:
%3").arg(ip).arg(port).arg(msg);
    ui->textBrowser->append(tmp);

    //传送到每个客户端里面
    for(int i = 0 ; i < socketlist.size() ; i ++ ){
        socketlist[i]->write(tmp.toUtf8());
    }

}

//断开连接
void MainWindow::on_pushButton_2_clicked()
{
    //断开连接
    mserver->disconnect();

}

//发送按钮
void MainWindow::on_pushButton_3_clicked()
{

    //向客户端发送信息
    //获取与客户端通信的套接字
```

```
    //QTcpSocket *msocket = mserver-
>nextPendingConnection();
    //使用socket的write函数向客户端发送数据

    QString tmp = "[服务器]: "+ui->textEdit_2-
>toPlainText();
    //注意这里要toUtf8，否则会报错
    //msocket->write(tmp.toUtf8());

    //向每个客户端发送
    for(int i = 0 ; i < socketlist.size() ; i++ ){
        socketlist[i]->write(tmp.toUtf8());
    }

    //服务器本身日志添加
    ui->textBrowser->append(tmp);

}
```

mainwindow.ui

```
<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

 <class>MainWindow</class>

 <widget class="QMainWindow" name="MainWindow">
```

```
<property name="windowModality">

<enum>Qt::NonModal</enum>

</property>

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>520</width>

<height>412</height>

</rect>

</property>

<property name="windowTitle">

<string>聊天室 server</string>

</property>

<property name="layoutDirection">
```

```
<enum>Qt::LeftToRight</enum>

</property>

<widget class="QWidget" name="centralwidget">

<widget class="QTextEdit" name="textEdit">

<property name="geometry">

<rect>

<x>10</x>

<y>10</y>

<width>81</width>

<height>31</height>

</rect>

</property>

<property name="contextMenuPolicy">

<enum>Qt::ActionsContextMenu</enum>

</property>

<property name="toolTip">
```

```
<string notr="true"/>

</property>

<property name="placeholderText">

<string>端口</string>

</property>

</widget>

<widget class="QPushButton" name="pushButton">

<property name="geometry">

<rect>

<x>100</x>

<y>10</y>

<width>93</width>

<height>28</height>

</rect>

</property>
```

```
<property name="text">

<string>启动服务器</string>

</property>

</widget>

<widget class="QTextBrowser" name="textBrowser">

<property name="geometry">

<rect>

<x>10</x>

<y>50</y>

<width>466</width>

<height>144</height>

</rect>

</property>

<property name="placeholderText">

<string>日志</string>

</property>
```

```
</widget>

<widget class="QPushButton" name="pushButton_2">

<property name="geometry">

<rect>

<x>370</x>

<y>10</y>

<width>93</width>

<height>28</height>

</rect>

</property>

<property name="text">

<string>断开连接</string>

</property>

</widget>

<widget class="QTextEdit" name="textEdit_2">
```

```
 <property name="geometry">

 <rect>

 <x>10</x>

 <y>210</y>

 <width>471</width>

 <height>101</height>

 </rect>

 </property>

 <property name="html">

 <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML
4.0//EN&quot; &quot;http://www.w3.org/TR/REC-
html40/strict.dtd&quot;&gt;

&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot;
content=&quot;1&quot; /&gt;&lt;style
type=&quot;text/css&quot;&gt;

p, li { white-space: pre-wrap; }

&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-
family:'SimSun'; font-size:9pt; font-weight:400; font-
style:normal;&quot;&gt;
```

```
&lt;p style=&quot;-qt-paragraph-type:empty; margin-top:0px;
margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-
block-indent:0; text-indent:0px;&quot;&gt;&lt;br
/&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>

 </property>

 <property name="placeholderText">

 <string>发送数据</string>

 </property>

 </widget>

 <widget class="QPushButton" name="pushButton_3">

 <property name="geometry">

 <rect>

 <x>390</x>

 <y>320</y>

 <width>93</width>

 <height>28</height>

 </rect>
```

```
</property>

<property name="text">

<string>发送</string>

</property>

</widget>

</widget>

<widget class="QMenuBar" name="menubar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>520</width>

<height>26</height>

</rect>

</property>
```

```
 </widget>

 <widget class="QStatusBar" name="statusbar"/>

 </widget>

 <resources/>

 <connections/>

</ui>
```

客户端
mainwindow.h

```


#ifndef MAINWINDOW_H

#define MAINWINDOW_H



#include <QMainWindow>

#include <QTcpServer>

#include <QTcpSocket>


```

```
QT_BEGIN_NAMESPACE

namespace Ui { class MainWindow; }

QT_END_NAMESPACE



class MainWindow : public QMainWindow

{

 Q_OBJECT



public:

 QTcpSocket *msocket; //创建QTcpSocket对象

 MainWindow(QWidget *parent = nullptr);

 ~MainWindow();



private slots:

 void read_data();
```

```
 void on_pushButton_2_clicked();

 void on_pushButton_clicked();



 void on_pushButton_3_clicked();



private:

 Ui::MainWindow *ui;

};

#endif // MAINWINDOW_H
```

main.cpp

```


#include "mainwindow.h"



#include <QApplication>



```

```
int main(int argc, char *argv[])

{

 QApplication a(argc, argv);

 MainWindow w;

 w.show();

 return a.exec();

}
```

mainwindow.cpp

```
#include "mainwindow.h"

#include "ui_mainwindow.h"



MainWindow::MainWindow(QWidget *parent): QMainWindow(parent),
ui(new Ui::MainWindow)
{

    ui->setupUi(this);

    //关联客户端连接信号(当有客户端连接的时候QTcpServer对象会发送
newConnection信号--关联槽函数)
```

```
    //1.创建QTcpSever对象
    mserver = new QTcpServer(this);
    msocket = NULL;

    //3.当服务器被客户端访问时，会发出newConnection()信号，因此为该信号添加槽函数，并用一个QTcpSocket对象接受客户端访问

connect(mserver,&QTcpServer::newConnection,this,&MainWindow::new_client);

}



MainWindow::~MainWindow()
{
    delete ui;
}



//监听——启动服务器
void MainWindow::on_pushButton_clicked()
{
    //2.侦听一个端口，使得客户端可以使用这个端口访问服务器
    //（listen(ip, port)）
    if( ui->textEdit->toPlainText() == "" ){                //判断是否为空
        ui->textBrowser->append("服务器启动失败，请输入要监听的端口号!");
    }else{
```

```
        mserver->listen(QHostAddress::Any,ui->textEdit->toPlainText().toUShort());
        ui->textBrowser->append("服务器启动成功，正在监听端口"+ui->textEdit->toPlainText()+"!");
    }
}



//接受连接
void MainWindow::new_client()
{
    //获取与客户端通信的套接字
    msocket = mserver->nextPendingConnection();

    socketlist.append(msocket);          //将客户端的套接字加入容器中

    //4.使用socket的write函数向客户端发送数据
    msocket->write("服务器连接成功! ");

    //获取客户端IP并显示
    QString ip = msocket->peerAddress().toString();
    ip.remove("::ffff:");
    quint16 port = msocket->peerPort();
    QString tmp = QString("客户端[%1:%2] 成功连接!").arg(ip).arg(port);
    ui->textBrowser->append(tmp);
```

```
    //关联读数据信号(客户端有数据到达服务器QTcpSocket对象会发送
readyRead信号--关联槽函数)
    //5.当socket接收缓冲区有新数据到来时，会发出readRead()信号，因此
为该信号添加槽函数以读取数据

connect(msocket,&QTcpSocket::readyRead,this,&MainWindow::read
_data);
}



//读取数据
void MainWindow::read_data()
{
    //获取信号发送者
    QTcpSocket *msocket1 = dynamic_cast<QTcpSocket*>
(sender());

    //读取数据
    QString msg = msocket1->readAll();
    //获取对方IP和端口
    QString ip = msocket1->peerAddress().toString();
    ip.remove("::ffff:");
    quint16 port = msocket1->peerPort();
    QString tmp = QString("客户端[%1:%2]:
%3").arg(ip).arg(port).arg(msg);
    ui->textBrowser->append(tmp);

    //传送到每个客户端里面
    for(int i = 0 ; i < socketlist.size() ; i ++ ){
        socketlist[i]->write(tmp.toUtf8());
```

```
    }

}

//断开连接
void MainWindow::on_pushButton_2_clicked()
{
    //断开连接
    mserver->disconnect();

}

//发送按钮
void MainWindow::on_pushButton_3_clicked()
{

    //向客户端发送信息
    //获取与客户端通信的套接字
    //QTcpSocket *msocket = mserver->nextPendingConnection();
    //使用socket的write函数向客户端发送数据
    QString tmp = "[服务器]: "+ui->textEdit_2->toPlainText();
    //注意这里要toUtf8，否则会报错
    //msocket->write(tmp.toUtf8());

    //向每个客户端发送
    for(int i = 0 ; i < socketlist.size() ; i++ ){
        socketlist[i]->write(tmp.toUtf8());
    }

    //服务器本身日志添加
    ui->textBrowser->append(tmp);
```

```
}
```

mainwindow.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
 <property name="windowModality">
 <enum>Qt::NonModal</enum>
 </property>
 <property name="geometry">
 <rect>
 <x>0</x>
 <y>0</y>
 <width>520</width>
```

```
<height>412</height>

</rect>

</property>

<property name="windowTitle">

<string>聊天室 server</string>

</property>

<property name="layoutDirection">

<enum>Qt::LeftToRight</enum>

</property>

<widget class="QWidget" name="centralwidget">

<widget class="QTextEdit" name="textEdit">

<property name="geometry">

<rect>

<x>10</x>

<y>10</y>

<width>81</width>
```

```
<height>31</height>

</rect>

</property>

<property name="contextMenuPolicy">

<enum>Qt::ActionsContextMenu</enum>

</property>

<property name="toolTip">

<string notr="true"/>

</property>

<property name="placeholderText">

<string>端口</string>

</property>

</widget>

<widget class="QPushButton" name="pushButton">

<property name="geometry">
```

```
<rect>

<x>100</x>

<y>10</y>

<width>93</width>

<height>28</height>

</rect>

</property>

<property name="text">

<string>启动服务器</string>

</property>

</widget>

<widget class="QTextBrowser" name="textBrowser">

<property name="geometry">

<rect>

<x>10</x>

<y>50</y>
```

```
<width>466</width>

<height>144</height>

</rect>

</property>

<property name="placeholderText">

<string>日志</string>

</property>

</widget>

<widget class="QPushButton" name="pushButton_2">

<property name="geometry">

<rect>

<x>370</x>

<y>10</y>

<width>93</width>

<height>28</height>
```

```
</rect>

</property>

<property name="text">

<string>断开连接</string>

</property>

</widget>

<widget class="QTextEdit" name="textEdit_2">

<property name="geometry">

<rect>

<x>10</x>

<y>210</y>

<width>471</width>

<height>101</height>

</rect>

</property>

<property name="html">
```

```
 <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML
4.0//EN&quot; &quot;http://www.w3.org/TR/REC-
html40/strict.dtd&quot;&gt;

&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot;
content=&quot;1&quot; /&gt;&lt;style
type=&quot;text/css&quot;&gt;

p, li { white-space: pre-wrap; }

&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-
family:'SimSun'; font-size:9pt; font-weight:400; font-
style:normal;&quot;&gt;

&lt;p style=&quot;-qt-paragraph-type:empty; margin-top:0px;
margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-
block-indent:0; text-indent:0px;&quot;&gt;&lt;br
/&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>

 </property>

 <property name="placeholderText">

 <string>发送数据</string>

 </property>

 </widget>

 <widget class="QPushButton" name="pushButton_3">
```

```
<property name="geometry">

<rect>

<x>390</x>

<y>320</y>

<width>93</width>

<height>28</height>

</rect>

</property>

<property name="text">

<string>发送</string>

</property>

</widget>

</widget>

<widget class="QMenuBar" name="menubar">

<property name="geometry">
```

```
  <rect>

  <x>0</x>

  <y>0</y>

  <width>520</width>

  <height>26</height>

  </rect>

  </property>

  </widget>

  <widget class="QStatusBar" name="statusbar"/>

  </widget>

  <resources/>

  <connections/>

 </ui>
```

服务器与客户端信息的传递2.0(2021.10.29)

增加了:
1.服务器与客户端交互信息的完善
2.服务器可以向客户端发送信息

服务器
cha.pro

```
QT += core gui

QT += network

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets



CONFIG += c++11



# You can make your code fail to compile if it uses
deprecated APIs.

# In order to do so, uncomment the following line.

#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000 #
disables all the APIs deprecated before Qt 6.0.0



SOURCES += \

 main.cpp \

 mainwindow.cpp
```

```
HEADERS += \

 mainwindow.h



FORMS += \

 mainwindow.ui



# Default rules for deployment.

qnx: target.path = /tmp/$${TARGET}/bin

else: unix:!android: target.path = /opt/$${TARGET}/bin

!isEmpty(target.path): INSTALLS += target
```

mainwindow.h

```
#ifndef MAINWINDOW_H

#define MAINWINDOW_H
```

```
#include <QMainWindow>
#include <QTcpServer>
#include <QTcpSocket>


QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE


class MainWindow : public QMainWindow
{
 Q_OBJECT


public:
 QTcpServer *mserver; //创建QTcpServer对象
 QTcpSocket *msocket; //创建QTcpSocket对象
```

```
 MainWindow(QWidget *parent = nullptr);
 ~MainWindow();


private slots:
 void on_pushButton_clicked();
 void new_client(); //接收连接——获取与客户端通信的套接字
 void read_data(); //读取数据


 void on_pushButton_2_clicked();
 void on_pushButton_3_clicked();


private:
 Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

main.cpp

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
 QApplication a(argc, argv);
 MainWindow w;
 w.show();
 return a.exec();
}
```

mainwindow.cpp

```
#include "mainwindow.h"
```

```cpp
#include "ui_mainwindow.h"



MainWindow::MainWindow(QWidget *parent):
QMainWindow(parent), ui(new Ui::MainWindow)
{

    ui->setupUi(this);

    //关联客户端连接信号(当有客户端连接的时候QTcpServer对象会发送newConnection信号--关联槽函数)
    //1.创建QTcpSever对象
    mserver = new QTcpServer(this);
    msocket = NULL;

    //3.当服务器被客户端访问时，会发出newConnection()信号，因此为该信号添加槽函数，并用一个QTcpSocket对象接受客户端访问

connect(mserver,&QTcpServer::newConnection,this,&MainWindow::n

}



MainWindow::~MainWindow()
{
    delete ui;
}
```

```
//监听——启动服务器

void MainWindow::on_pushButton_clicked()
{
    //2.侦听一个端口，使得客户端可以使用这个端口访问服务器
    //（listen(ip, port)）
    if( ui->textEdit->toPlainText() == "" ){             //
判断是否为空
        ui->textBrowser->append("服务器启动失败，请输入要监听的
端口号!");
    }else{
        mserver->listen(QHostAddress::Any,ui->textEdit-
>toPlainText().toUShort());
        ui->textBrowser->append("服务器启动成功，正在监听端
口"+ui->textEdit->toPlainText()+"!");
    }
}



//接受连接
void MainWindow::new_client()
{
    //获取与客户端通信的套接字
    msocket = mserver->nextPendingConnection();

    //4.使用socket的write函数向客户端发送数据
    msocket->write("服务器连接成功! ");

    //获取客户端IP并显示
```

```
    QString ip = msocket->peerAddress().toString();
    ip.remove("::ffff:");
    quint16 port = msocket->peerPort();

    QString tmp = QString("客户端[%1:%2] 成功连
接!").arg(ip).arg(port);
    ui->textBrowser->append(tmp);



    //关联读数据信号(客户端有数据到达服务器QTcpSocket对象会发送
readyRead信号--关联槽函数)
    //5.当socket接收缓冲区有新数据到来时，会发出readRead()信号，因
此为该信号添加槽函数以读取数据

connect(msocket,&QTcpSocket::readyRead,this,&MainWindow::read_

}



//读取数据
void MainWindow::read_data()
{
    //获取信号发送者
    QTcpSocket *msocket1 = dynamic_cast<QTcpSocket*>
(sender());
    //读取数据
    QString msg = msocket1->readAll();
    //获取对方IP和端口
    QString ip = msocket1->peerAddress().toString();
    ip.remove("::ffff:");
```

```
    quint16 port = msocket1->peerPort();
    QString tmp = QString("客户端[%1:%2]:
%3").arg(ip).arg(port).arg(msg);

    ui->textBrowser->append(tmp);
}

//断开连接
void MainWindow::on_pushButton_2_clicked()
{
    //断开连接
    mserver->disconnect();

}

//发送按钮
void MainWindow::on_pushButton_3_clicked()
{

    //向客户端发送信息
    //获取与客户端通信的套接字
    //QTcpSocket *msocket = mserver-
>nextPendingConnection();
    //使用socket的write函数向客户端发送数据
    QString tmp = "[服务器]: "+ui->textEdit_2-
>toPlainText();
    //注意这里要toUtf8，否则会报错
    msocket->write(tmp.toUtf8());

    //服务器本身日志添加
    ui->textBrowser->append(tmp);

}
```

```
}
```

mainwindow.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
 <property name="windowModality">
 <enum>Qt::NonModal</enum>
 </property>
 <property name="geometry">
 <rect>
 <x>0</x>
 <y>0</y>
 <width>520</width>
```

```
<height>412</height>

</rect>

</property>

<property name="windowTitle">

<string>聊天室 server</string>

</property>

<property name="layoutDirection">

<enum>Qt::LeftToRight</enum>

</property>

<widget class="QWidget" name="centralwidget">

<widget class="QTextEdit" name="textEdit">

<property name="geometry">

<rect>

<x>10</x>

<y>10</y>

<width>81</width>
```

```
<height>31</height>

</rect>

</property>

<property name="contextMenuPolicy">

<enum>Qt::ActionsContextMenu</enum>

</property>

<property name="toolTip">

<string notr="true"/>

</property>

<property name="placeholderText">

<string>端口</string>

</property>

</widget>

<widget class="QPushButton" name="pushButton">

<property name="geometry">
```

```
<rect>

<x>100</x>

<y>10</y>

<width>93</width>

<height>28</height>

</rect>

</property>

<property name="text">

<string>启动服务器</string>

</property>

</widget>

<widget class="QTextBrowser" name="textBrowser">

<property name="geometry">

<rect>

<x>10</x>

<y>50</y>
```

```
<width>466</width>

<height>144</height>

</rect>

</property>

<property name="placeholderText">

<string>日志</string>

</property>

</widget>

<widget class="QPushButton" name="pushButton_2">

<property name="geometry">

<rect>

<x>370</x>

<y>10</y>

<width>93</width>

<height>28</height>
```

```
</rect>

</property>

<property name="text">

<string>断开连接</string>

</property>

</widget>

<widget class="QTextEdit" name="textEdit_2">

<property name="geometry">

<rect>

<x>10</x>

<y>210</y>

<width>471</width>

<height>101</height>

</rect>

</property>

<property name="html">
```

```
 <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML
4.0//EN&quot; &quot;http://www.w3.org/TR/REC-
html40/strict.dtd&quot;&gt;

&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot;
content=&quot;1&quot; /&gt;&lt;style
type=&quot;text/css&quot;&gt;

p, li { white-space: pre-wrap; }

&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-
family:'SimSun'; font-size:9pt; font-weight:400; font-
style:normal;&quot;&gt;

&lt;p style=&quot;-qt-paragraph-type:empty; margin-top:0px;
margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-
block-indent:0; text-indent:0px;&quot;&gt;&lt;br
/&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>

 </property>

 <property name="placeholderText">

 <string>发送数据</string>

 </property>

 </widget>

 <widget class="QPushButton" name="pushButton_3">
```

```
<property name="geometry">

<rect>

<x>390</x>

<y>320</y>

<width>93</width>

<height>28</height>

</rect>

</property>

<property name="text">

<string>发送</string>

</property>

</widget>

</widget>

<widget class="QMenuBar" name="menubar">

<property name="geometry">
```

```
  <rect>

  <x>0</x>

  <y>0</y>

  <width>520</width>

  <height>26</height>

  </rect>

  </property>

  </widget>

  <widget class="QStatusBar" name="statusbar"/>

  </widget>

  <resources/>

  <connections/>

 </ui>
```

客户端代码
chatt_kh.pro

```
QT += core gui

QT += network

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++11

# You can make your code fail to compile if it uses deprecated APIs.

# In order to do so, uncomment the following line.

#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000 # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \

 main.cpp \
```

```
 mainwindow.cpp



HEADERS += \

 mainwindow.h



FORMS += \

 mainwindow.ui



# Default rules for deployment.

qnx: target.path = /tmp/$${TARGET}/bin

else: unix:!android: target.path = /opt/$${TARGET}/bin

!isEmpty(target.path): INSTALLS += target
```

mainwindow.h

```


#ifndef MAINWINDOW_H
```

```
#define MAINWINDOW_H



#include <QMainWindow>

#include <QTcpServer>

#include <QTcpSocket>



QT_BEGIN_NAMESPACE

namespace Ui { class MainWindow; }

QT_END_NAMESPACE



class MainWindow : public QMainWindow

{

 Q_OBJECT



public:

 QTcpSocket *msocket; //创建QTcpSocket对象
```

```
 MainWindow(QWidget *parent = nullptr);
 ~MainWindow();


private slots:
 void read_data();
 void on_pushButton_2_clicked();
 void on_pushButton_clicked();


 void on_pushButton_3_clicked();


private:
 Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

main.cpp

```cpp
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
 QApplication a(argc, argv);
 MainWindow w;
 w.show();
 return a.exec();
}
```

mainwindow.cpp

```cpp
#include "mainwindow.h"
#include "ui_mainwindow.h"
```

```
MainWindow::MainWindow(QWidget *parent):
QMainWindow(parent), ui(new Ui::MainWindow)
{

    ui->setupUi(this);

    //关联客户端连接信号(当有客户端连接的时候QTcpServer对象会发送
newConnection信号--关联槽函数)
    //1.创建QTcpSever对象
    mserver = new QTcpServer(this);
    msocket = NULL;

    //3.当服务器被客户端访问时，会发出newConnection()信号，因此为
该信号添加槽函数，并用一个QTcpSocket对象接受客户端访问

connect(mserver,&QTcpServer::newConnection,this,&MainWindow::r

}



MainWindow::~MainWindow()
{
    delete ui;
}



//监听——启动服务器
```

```
//监听一启动服务器
void MainWindow::on_pushButton_clicked()
{
    //2.侦听一个端口，使得客户端可以使用这个端口访问服务器
    //（listen(ip, port)）

    if( ui->textEdit->toPlainText() == "" ){              //
判断是否为空
        ui->textBrowser->append("服务器启动失败，请输入要监听的
端口号!");
    }else{
        mserver->listen(QHostAddress::Any,ui->textEdit-
>toPlainText().toUShort());
        ui->textBrowser->append("服务器启动成功，正在监听端
口"+ui->textEdit->toPlainText()+"!");
    }
}



//接受连接
void MainWindow::new_client()
{
    //获取与客户端通信的套接字
    msocket = mserver->nextPendingConnection();

    //4.使用socket的write函数向客户端发送数据
    msocket->write("服务器连接成功! ");

    //获取客户端IP并显示
    QString ip = msocket->peerAddress().toString();
    ip.remove("::ffff:");
    quint16 port = msocket->peerPort();
```

```
    QString tmp = QString("客户端[%1:%2] 成功连
接!").arg(ip).arg(port);
    ui->textBrowser->append(tmp);



    //关联读数据信号(客户端有数据到达服务器QTcpSocket对象会发送
readyRead信号--关联槽函数)
    //5.当socket接收缓冲区有新数据到来时，会发出readRead()信号，因
此为该信号添加槽函数以读取数据

connect(msocket,&QTcpSocket::readyRead,this,&MainWindow::read_

}



//读取数据
void MainWindow::read_data()
{
    //获取信号发送者
    QTcpSocket *msocket1 = dynamic_cast<QTcpSocket*>
(sender());
    //读取数据
    QString msg = msocket1->readAll();
    //获取对方IP和端口
    QString ip = msocket1->peerAddress().toString();
    ip.remove("::ffff:");
    quint16 port = msocket1->peerPort();
    QString tmp = QString("客户端[%1:%2]:
%3").arg(ip).arg(port).arg(msg);
```

```
    ui->textBrowser->append(tmp);
}

//断开连接

void MainWindow::on_pushButton_2_clicked()
{
    //断开连接
    mserver->disconnect();

}

//发送按钮
void MainWindow::on_pushButton_3_clicked()
{

    //向客户端发送信息
    //获取与客户端通信的套接字
    //QTcpSocket *msocket = mserver-
>nextPendingConnection();
    //使用socket的write函数向客户端发送数据
    QString tmp = "[服务器]: "+ui->textEdit_2-
>toPlainText();
    //注意这里要toUtf8，否则会报错
    msocket->write(tmp.toUtf8());

    //服务器本身日志添加
    ui->textBrowser->append(tmp);

}
```

mainwindow.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
 <property name="windowModality">
 <enum>Qt::NonModal</enum>
 </property>
 <property name="geometry">
 <rect>
 <x>0</x>
 <y>0</y>
 <width>520</width>
 <height>412</height>
 </rect>
```

```
</property>

<property name="windowTitle">

<string>聊天室 server</string>

</property>

<property name="layoutDirection">

<enum>Qt::LeftToRight</enum>

</property>

<widget class="QWidget" name="centralwidget">

<widget class="QTextEdit" name="textEdit">

<property name="geometry">

<rect>

<x>10</x>

<y>10</y>

<width>81</width>

<height>31</height>
```

```
</rect>

</property>

<property name="contextMenuPolicy">

<enum>Qt::ActionsContextMenu</enum>

</property>

<property name="toolTip">

<string notr="true"/>

</property>

<property name="placeholderText">

<string>端口</string>

</property>

</widget>

<widget class="QPushButton" name="pushButton">

<property name="geometry">

<rect>

<x>100</x>
```

```
<y>10</y>

<width>93</width>

<height>28</height>

</rect>

</property>

<property name="text">

<string>启动服务器</string>

</property>

</widget>

<widget class="QTextBrowser" name="textBrowser">

<property name="geometry">

<rect>

<x>10</x>

<y>50</y>

<width>466</width>
```

```
<height>144</height>

</rect>

</property>

<property name="placeholderText">

<string>日志</string>

</property>

</widget>

<widget class="QPushButton" name="pushButton_2">

<property name="geometry">

<rect>

<x>370</x>

<y>10</y>

<width>93</width>

<height>28</height>

</rect>

</property>
```

```
 <property name="text">

 <string>断开连接</string>

 </property>

 </widget>

 <widget class="QTextEdit" name="textEdit_2">

 <property name="geometry">

 <rect>

 <x>10</x>

 <y>210</y>

 <width>471</width>

 <height>101</height>

 </rect>

 </property>

 <property name="html">

 <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML
4.0//EN&quot; &quot;http://www.w3.org/TR/REC-
```

```
html40/strict.dtd&amp;quot;&amp;gt;

&amp;lt;html&amp;gt;&amp;lt;head&amp;gt;&amp;lt;meta name=&amp;quot;qrichtext&amp;quot;
content=&amp;quot;1&amp;quot; /&amp;gt;&amp;lt;style
type=&amp;quot;text/css&amp;quot;&amp;gt;

p, li { white-space: pre-wrap; }

&amp;lt;/style&amp;gt;&amp;lt;/head&amp;gt;&amp;lt;body style=&amp;quot; font-
family:'SimSun'; font-size:9pt; font-weight:400; font-
style:normal;&amp;quot;&amp;gt;

&amp;lt;p style=&amp;quot;-qt-paragraph-type:empty; margin-top:0px;
margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-
block-indent:0; text-indent:0px;&amp;quot;&amp;gt;&amp;lt;br
/&amp;gt;&amp;lt;/p&amp;gt;&amp;lt;/body&amp;gt;&amp;lt;/html&amp;gt;</string>

 </property>

 <property name="placeholderText">

 <string>发送数据</string>

 </property>

 </widget>

 <widget class="QPushButton" name="pushButton_3">

 <property name="geometry">
```

```
<rect>

<x>390</x>

<y>320</y>

<width>93</width>

<height>28</height>

</rect>

</property>

<property name="text">

<string>发送</string>

</property>

</widget>

</widget>

<widget class="QMenuBar" name="menubar">

<property name="geometry">

<rect>

<x>0</x>
```

```
 <y>0</y>

 <width>520</width>

 <height>26</height>

 </rect>

 </property>

 </widget>

 <widget class="QStatusBar" name="statusbar"/>

 </widget>

 <resources/>

 <connections/>

</ui>
```

QT客户端与服务端通信(2021.10.29s)

```
//这个应该放在连接成功那里
 QTcpSocket *msocket = mserver->nextPendingConnection();

//接下来这些就是发送代码
 //使用socket的write函数向客户端发送数据
```

```
QString tmp = "[服务器]: "+ui->textEdit_2->toPlainText();
//注意这里要toUtf8，否则会报错
msocket->write(tmp.toUtf8());
```

QT之TCP服务器客户端连接0.1（2021.10.24）

（1）TCP知识点（链接：https://blog.csdn.net/l477918269/article/details/95613127）
在服务端方面：
TCP要建立监听（listen）和接收链接(accept)。
在客户端方面：
要接收链接（connect）

具体步骤：
TCP服务端
1.创建套接字（socket）
2.绑定端口(bind)
3.开始监听：
例子：饭店拉客
当我们节假日外出吃饭的时候，经常受到各个店家热情的拉客，当我们被成功拉进馆子之后，在外拉客的店员就不会再管我们了，而是交由店内的店员为我们提供服务。在这个例子中，“拉客”的店员就相当于我们的“监听”端口
listen函数
int listen(int socket, int backlo
参数：sock套接字。
backlog是一个数字，表示所传链表的大小。具体含义是服务端预留了一小部分资源，这部分资源由一个链表队列位置，这个位置是为了给在排队的进程预留的。作用是为了提高效率，但存在一定的成本。

4.接收请求
接收请求就是上面例子中的，在店内为你提供服务的店员。
accept函数
```
int accept(int socket, struct sockaddr* address, socklen_t*
address_len);
```
参数： socket套接字。 address接收到的结构体， address_len结构体的大小。
5.提供服务（自定义）

客户端：
1.创建套接字
2.创建连接
connect函数
```
int connect(int socket, const struct sockaddr* addr, struct
sockaddr* addr_len)
```
参数： socket套接字。 address接收到的结构体， address_len结构体的大小。

（2）学习参考链接：
https://blog.csdn.net/qq_42449351/article/details/100517623
https://blog.csdn.net/u014695839/article/details/70041771
（3）
问题：
服务器没有监听到
解决方案：
从 https://blog.csdn.net/omg_orange/article/details/73826694 得到灵感，将 `QTcpServer mserver;` 改成指针 `QTcpServer *mserver;`,并在生成窗口的那里new一个对象。
问题：
客户端没有连接成功
解决方案：
https://blog.csdn.net/WindSunLike/article/details/106248368

当我修改绑定的地址语句为 `QHostAddress::LocalHost` 之后，可以监听成功。

（4）代码
服务器端代码
笔记：
1、服务器除了使用到了QTcpSocket类，还需要用到QTcpSever类。即便如此，也只是比客户端复杂一点点，用到了6个步骤：

（1）创建QTcpSever对象

```
server = new QTcpServer();
```

（2）侦听一个端口，使得客户端可以使用这个端口访问服务器

```
server->listen(QHostAddress::Any, port);
```

（3）当服务器被客户端访问时，会发出newConnection()信号，因此为该信号添加槽函数，并用一个QTcpSocket对象接受客户端访问

```
connect(server,&QTcpServer::newConnection,this,&MainWindow::se


    void MainWindow::server_New_Connect()
    {
        //获取客户端连接
        socket = server->nextPendingConnection();
    }
```

（4）使用socket的write函数向客户端发送数据

```
socket->write(data);
```

（5）当socket接收缓冲区有新数据到来时，会发出readRead()信号，因此为该信号添加槽函数以读取数据

```
    QObject::connect(socket, &QTcpSocket::readyRead, this,
&MainWindow::socket_Read_Data);

    void MainWindow::socket_Read_Data()
    {
        QByteArray buffer;
        //读取缓冲区数据
        buffer = socket->readAll();
    }
```

（6）取消侦听

```
server->close();
```

cha.pro

```
QT += core gui

QT += network                   #注意这里必须加一个才能进行TCP连接
```

```
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets



CONFIG += c++11



# You can make your code fail to compile if it uses
deprecated APIs.

# In order to do so, uncomment the following line.

#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000 #
disables all the APIs deprecated before Qt 6.0.0



SOURCES += \

 main.cpp \

 mainwindow.cpp



HEADERS += \

 mainwindow.h
```

```
FORMS += \

 mainwindow.ui



# Default rules for deployment.

qnx: target.path = /tmp/$${TARGET}/bin

else: unix:!android: target.path = /opt/$${TARGET}/bin

!isEmpty(target.path): INSTALLS += target
```

mainwindow.h

```


#ifndef MAINWINDOW_H

#define MAINWINDOW_H



#include <QMainWindow>

#include <QTcpServer>

#include <QTcpSocket>
```

```
QT_BEGIN_NAMESPACE

namespace Ui { class MainWindow; }

QT_END_NAMESPACE



class MainWindow : public QMainWindow

{

 Q_OBJECT



public:

 QTcpServer *mserver; //创建QTcpServer对象

 MainWindow(QWidget *parent = nullptr);

 ~MainWindow();



private slots:
```

```
 void on_pushButton_clicked();

 void new_client(); //接收链接——获取与客户端通信的套接字

 void read_data(); //读取数据



private:

 Ui::MainWindow *ui;

};

#endif // MAINWINDOW_H
```

main.cpp

```


#include "mainwindow.h"



#include <QApplication>



int main(int argc, char *argv[])
```

```
{

 QApplication a(argc, argv);

 MainWindow w;

 w.show();

 return a.exec();

}
```

mainwindow.cpp

```
#include "mainwindow.h"

#include "ui_mainwindow.h"



MainWindow::MainWindow(QWidget *parent)

 : QMainWindow(parent)

 , ui(new Ui::MainWindow)
```

```
{

 ui->setupUi(this);

 //关联客户端连接信号(当有客户端连接的时候QTcpServer对象会发送newConnection信号--关联槽函数)
 //1.创建QTcpSever对象
 mserver = new QTcpServer(this);

 //3.当服务器被客户端访问时，会发出newConnection()信号，因此为该信号添加槽函数，并用一个QTcpSocket对象接受客户端访问

connect(mserver,&QTcpServer::newConnection,this,&MainWindow::r



}



MainWindow::~MainWindow()

{

 delete ui;

}



//监听——启动服务器
void MainWindow::on_pushButton_clicked()
```

```
{

 //2.侦听一个端口，使得客户端可以使用这个端口访问服务器
 //（listen(ip, port)）

 mserver->listen(QHostAddress::Any,ui->textEdit-
>toPlainText().toUShort());

}



//接受链接

void MainWindow::new_client()

{

 //获取与客户端通信的套接字

 QTcpSocket *msocket = mserver->nextPendingConnection();

 //4.使用socket的write函数向客户端发送数据

 msocket->write("connect");

 //关联读数据信号(客户端有数据到达服务器QTcpSocket对象会发送
readyRead信号--关联槽函数)
 //5.当socket接收缓冲区有新数据到来时，会发出readRead()信号，因此
为该信号添加槽函数以读取数据
```

```
connect(msocket,&QTcpSocket::readyRead,this,&MainWindow::read_

}



//读取数据
void MainWindow::read_data()
{
 //获取信号发送者
 QTcpSocket *msocket = dynamic_cast<QTcpSocket*>(sender());
 //读取数据
 QString msg = msocket->readAll();
 //获取对方IP
 QString ip = msocket->peerAddress().toString();
 ip.remove("::ffff:");
 ui->textBrowser->append(ip+":"+msg);
}
```

客户端代码:
笔记:

1. **使用QT的网络套接字需要.pro文件中加入一句:**

```
QT  += network
```

2. 1、客户端的代码比服务器稍简单，总的来说，使用QT中的QTcpSocket类与服务器进行通信只需要以下5步:

(1) 创建QTcpSocket套接字对象

```
socket = new QTcpSocket();
```

(2) 使用这个对象连接服务器

```
socket->connectToHost(IP, port);
```

(3) 使用write函数向服务器发送数据

```
socket->write(data);
```

(4) 当socket接收缓冲区有新数据到来时，会发出readRead()信号，因此为该信号添加槽函数以读取数据

```
    QObject::connect(socket, &QTcpSocket::readyRead, this,
&MainWindow::socket_Read_Data);
```

```
void MainWindow::socket_Read_Data()
{
    QByteArray buffer;
    //读取缓冲区数据
    buffer = socket->readAll();
}
```

（5）断开与服务器的连接（关于close()和disconnectFromHost()的区别，可以按F1看帮助）

```
socket->disconnectFromHost();
```

chatt_kh.pro

```
QT += core gui

QT += network                   #注意这句加上


greaterThan(QT_MAJOR_VERSION, 4): QT += widgets


CONFIG += c++11
```

```
# You can make your code fail to compile if it uses
deprecated APIs.

# In order to do so, uncomment the following line.

#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000 #
disables all the APIs deprecated before Qt 6.0.0



SOURCES += \

 main.cpp \

 mainwindow.cpp



HEADERS += \

 mainwindow.h



FORMS += \

 mainwindow.ui



# Default rules for deployment.
```

```
qnx: target.path = /tmp/$${TARGET}/bin

else: unix:!android: target.path = /opt/$${TARGET}/bin

!isEmpty(target.path): INSTALLS += target
```

mainwindow.h

```
#ifndef MAINWINDOW_H

#define MAINWINDOW_H



#include <QMainWindow>

#include <QTcpServer>

#include <QTcpSocket>



QT_BEGIN_NAMESPACE

namespace Ui { class MainWindow; }
```

```
QT_END_NAMESPACE



class MainWindow : public QMainWindow

{

 Q_OBJECT



public:

 QTcpSocket *msocket; //创建QTcpSocket对象

 MainWindow(QWidget *parent = nullptr);

 ~MainWindow();



private slots:

 void read_data();

 void on_pushButton_2_clicked();

 void on_pushButton_clicked();
```

```
private:

 Ui::MainWindow *ui;

};

#endif // MAINWINDOW_H
```

main.cpp

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])

{

 QApplication a(argc, argv);

 MainWindow w;

 w.show();
```

```
 return a.exec();

}
```

mainwindow.cpp

```


#include "mainwindow.h"

#include "ui_mainwindow.h"



MainWindow::MainWindow(QWidget *parent)

 : QMainWindow(parent)

 , ui(new Ui::MainWindow)

{


 ui->setupUi(this);
 //1.创建套接字对象
 msocket = new QTcpSocket;

 //关联读数据信号(当QTcpSocket有数据可读会发送readyRead信号--关联
槽函数)
 //4.当socket接收缓冲区有新数据到来时，会发出readRead()信号，因此
```

```
为该信号添加槽函数以读取数据

connect(msocket,&QTcpSocket::readyRead,this,&MainWindow::read_




}



MainWindow::~MainWindow()

{

 delete ui;

}



//连接服务器

void MainWindow::on_pushButton_2_clicked()


{

 //调用对象方法链接服务器（connectToHost（服务器的IP，  端口））

 //msocket->connectToHost(ui->textEdit->toPlainText(),ui-
>textEdit_2->toPlainText().toUShort());
```

```
 //2.使用这个对象连接服务器，注意这里改成了
QHostAddress::LocalHost
 msocket->connectToHost( QHostAddress::LocalHost,ui-
>textEdit_2->toPlainText().toUShort());

}




void MainWindow::read_data()

{

 //读取数据

 QString msg = msocket->readAll();

 ui->textBrowser->append(msg);

}




//发送

void MainWindow::on_pushButton_clicked()
```

```
{

 //发送数据

 QString data = ui->textEdit_3->toPlainText();
 //3.使用write函数向服务器发送数据
 msocket->write(data.toUtf8());

}
```

mainwindow.ui

```
<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

 <class>MainWindow</class>

 <widget class="QMainWindow" name="MainWindow">

 <property name="geometry">

 <rect>

 <x>0</x>

 <y>0</y>
```

```
<width>715</width>

<height>600</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<widget class="QWidget" name="centralwidget">

<widget class="QTextEdit" name="textEdit">

<property name="geometry">

<rect>

<x>40</x>

<y>20</y>

<width>241</width>

<height>31</height>

</rect>
```

```
</property>

</widget>

<widget class="QTextEdit" name="textEdit_2">

<property name="geometry">

<rect>

<x>310</x>

<y>20</y>

<width>161</width>

<height>31</height>

</rect>

</property>

</widget>

<widget class="QPushButton" name="pushButton">

<property name="geometry">

<rect>
```

```
<x>520</x>

<y>490</y>

<width>93</width>

<height>28</height>

</rect>

</property>

<property name="text">

<string>发送</string>

</property>

</widget>

<widget class="QPushButton" name="pushButton_2">

<property name="geometry">

<rect>

<x>500</x>

<y>20</y>

<width>93</width>
```

```
<height>28</height>

</rect>

</property>

<property name="text">

<string>连接服务器</string>

</property>

</widget>

<widget class="QTextEdit" name="textEdit_3">

<property name="geometry">

<rect>

<x>90</x>

<y>290</y>

<width>531</width>

<height>181</height>

</rect>
```

```
 </property>

 </widget>

 <widget class="QTextBrowser" name="textBrowser">

 <property name="geometry">

 <rect>

 <x>90</x>

 <y>90</y>

 <width>531</width>

 <height>191</height>

 </rect>

 </property>

 </widget>

 </widget>

 <widget class="QMenuBar" name="menubar">

 <property name="geometry">

 <rect>
```

```
  <x>0</x>
  <y>0</y>
  <width>715</width>
  <height>26</height>
  </rect>
  </property>
  </widget>
  <widget class="QStatusBar" name="statusbar"/>
  </widget>
  <resources/>
  <connections/>
</ui>
```