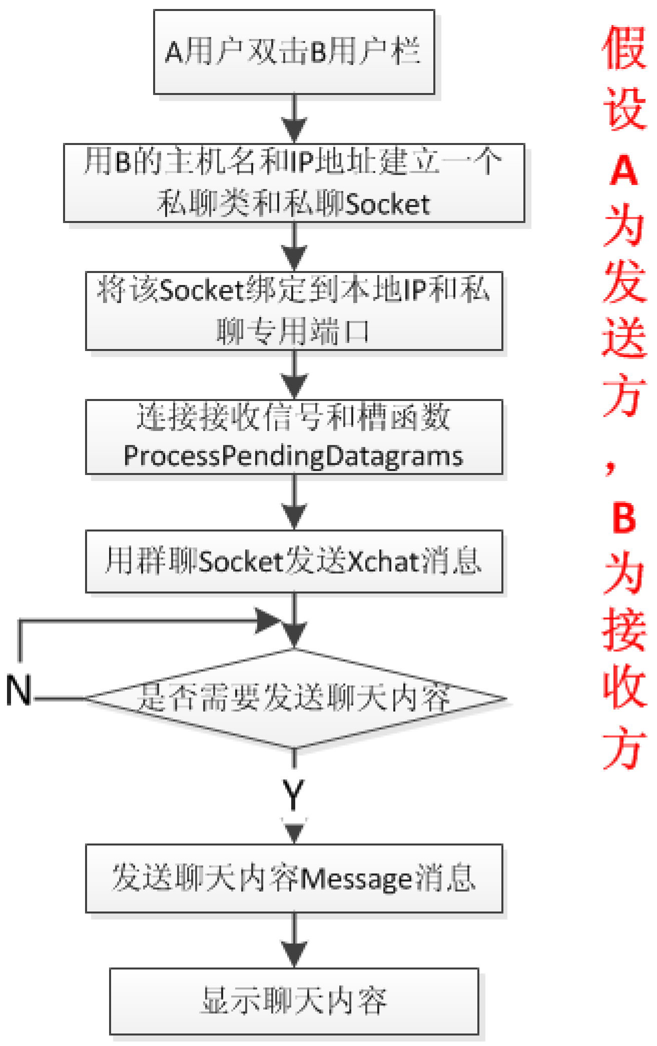
在上 2 次文章 Qt 学习之路_5(Qt TCP 的初步使用) Qt 学习之路_4(Qt UDP 的初步使用) 中已经初步介绍了群聊功能和文件传输功能，这一节中主要在这个基础上加入一个私聊功能。

参考文献依旧是：《Qt 及 Qt Quick 开发实战精解》一书中的第 5 个例子以及 http://www.yafeilinux.com/ 网站上的源码。另外这次的私聊功能也是参考网友 http://www.qtcn.org/bbs/read-htm-tid-32609.html 的，他的程序有些 bug，其中最严重的 bug 是当私聊第二次聊天的时候对方会接收不到信息。这次主要是将这个 bug 和其它一些小 bug 修补了，但是仍然有一个漏洞就是：当第二次私聊时，后面那个的发送方收到信息的时候有可能会多一个窗口弹出来。目前还找不到其原因。猜想是：在第一次聊天接收时关闭聊天窗口后，其内存没有释放。但是当窗口关闭时我们觉得其内存释放应该在 Qt 内部自己实现。

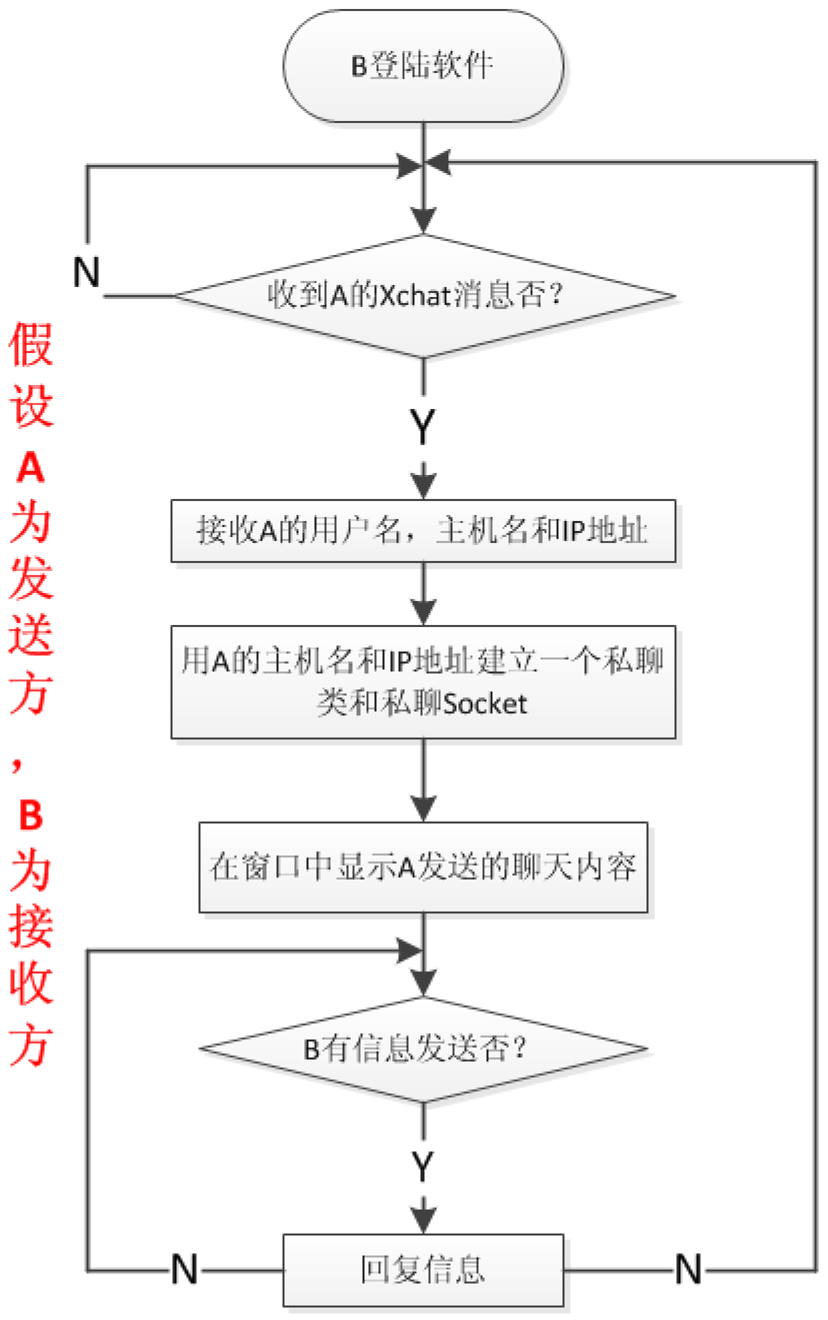
下面来讲一下私聊发送端和接收端具体实现过程。

发送端流程图如下：



A用户双击B用户栏
用B的主机名和IP地址建立一个私聊类和私聊Socket
将该Socket绑定到本地IP和私聊专用端口
连接接收信号和槽函数 ProcessPendingDatagrams
用群聊Socket发送Xchat消息
是否需要发送聊天内容
N
Y
发送聊天内容Message消息
显示聊天内容
假设A为发送方，B为接收方

接收端的流程图如下：


B登陆软件
收到A的Xchat消息否？
N
Y
接收A的用户名，主机名和IP地址
用A的主机名和IP地址建立一个私聊类和私聊Socket
在窗口中显示A发送的聊天内容
B有信息发送否？
Y
回复信息
N
N
假设A为发送方，B为接收方

下面来介绍下 2 者实现的具体过程：

A 方(主动开始首次发送的一方)：

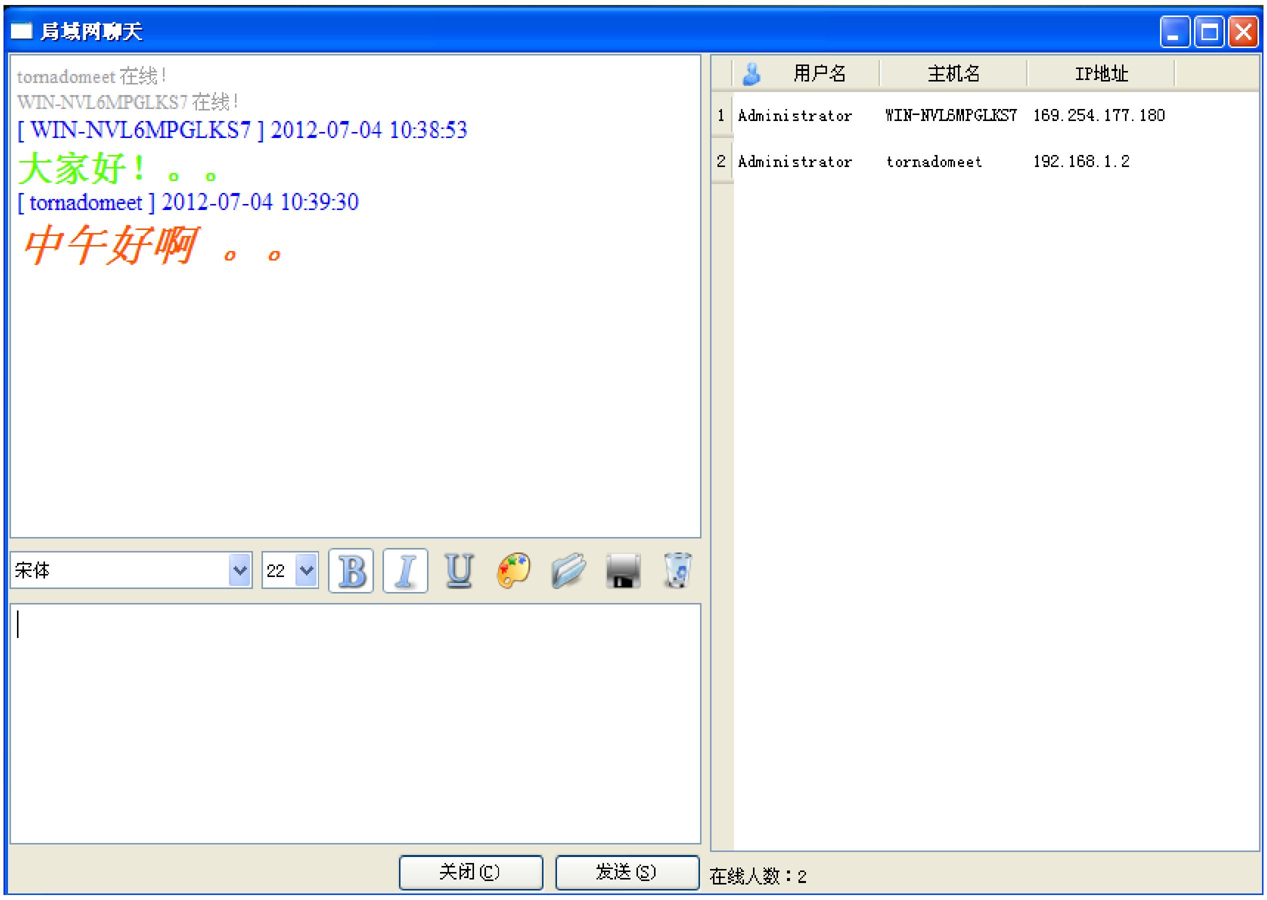
1. 在主窗口右侧双击自己想与之聊天的 B 方，此时 A 方实际上完成的工作有：用 B 方的主机名和 ip 地址新建了私聊的类 privatechat，在新建该类的过程中，已经设置了显示顶端为：与***聊天中，对方 IP：***，且绑定了本地 ip 和私聊的专用端口，同时设置了信号与槽的联系，即该端口如果有数据输入，则触发槽函数 processPendingDatagrams().该函数是 char.cpp 中的。
2. 当上面的新建私聊类完成后，用通讯对方 ip 地址和其群聊专用的端口（但用的是主 udp 群聊的 socket 进行的）将以下内容分别发送出去：消息类型(Xchat)，用户名，主机名，本地 ip 地址。完成后，在屏幕中显示私聊窗口。
3. 在私聊窗口中输入需要聊天的内容，单击发送键。该过程玩成的内容有：分别将消

息类型(Message)+用户名+本地名+本地 IP+消息内容本身通过私聊专用端口发送出去。在私聊窗口中显示主机名+聊天时间，换行后显示消息内容本身。

B 方(第一次信息是他人发送过来的):

1. 当 A 在 2 步骤中用群聊的方法发送其消息类型(Xchat)，其用户名，其主机名，其 ip 地址后，由于程序运行时已经初始化了 widget.cpp 中的构造函数，所以每个程序都绑定了本地地址+群聊专用的端口，一旦有数据传入，就触发 widget.cpp 中的槽函数 processPendingDatagrams().
2. 在 processPendingDatagrams()函数中，判断消息类型为 Xchat 后，接收缓存区内接收对方用户名，对方主机名和对方 ip 地址。并用接收到的主机名和 ip 地址新建一个私聊类。新建该私聊的过程与 A 中的步骤 1 一样。完后在程序中显示私聊窗口。
3. 当对方 A 按完发送按钮后，通过私聊专用端口绑定槽函数来触发 chart.cpp 中的 processPendingDatagrams()函数，该函数中先读取消息类型(Message)，然后依次读取用户名，主机名，ip 地址，消息内容本身，并将对方信息和消息内容显示在聊天窗口中。

实验结果如下：
群聊界面：

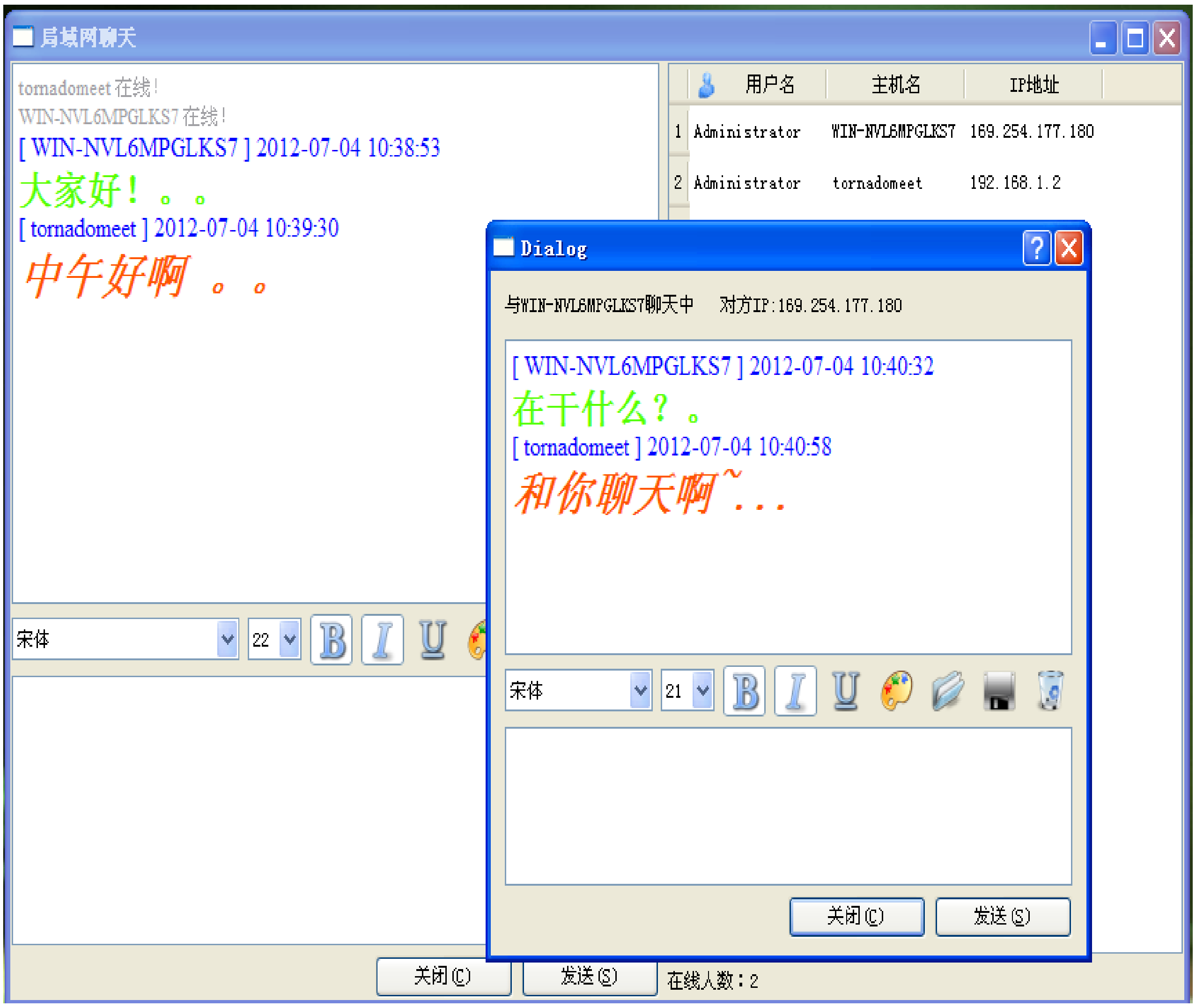


tornadomeet 在线！
WIN-NVL6MPGLKS7 在线！
[WIN-NVL6MPGLKS7] 2012-07-04 10:38:53
大家好！。。
[tornadomeet] 2012-07-04 10:39:30
中午好啊 。。

	用户名	主机名	IP地址
1	Administrator	WIN-NVL6MPGLKS7	169.254.177.180
2	Administrator	tornadomeet	192.168.1.2

在线人数：2

私聊界面：



局域网聊天

tornadomeet 在线！
WIN-NVL6MPGLKS7 在线！
[WIN-NVL6MPGLKS7] 2012-07-04 10:38:53
大家好！。。
[tornadomeet] 2012-07-04 10:39:30
中午好啊 。。

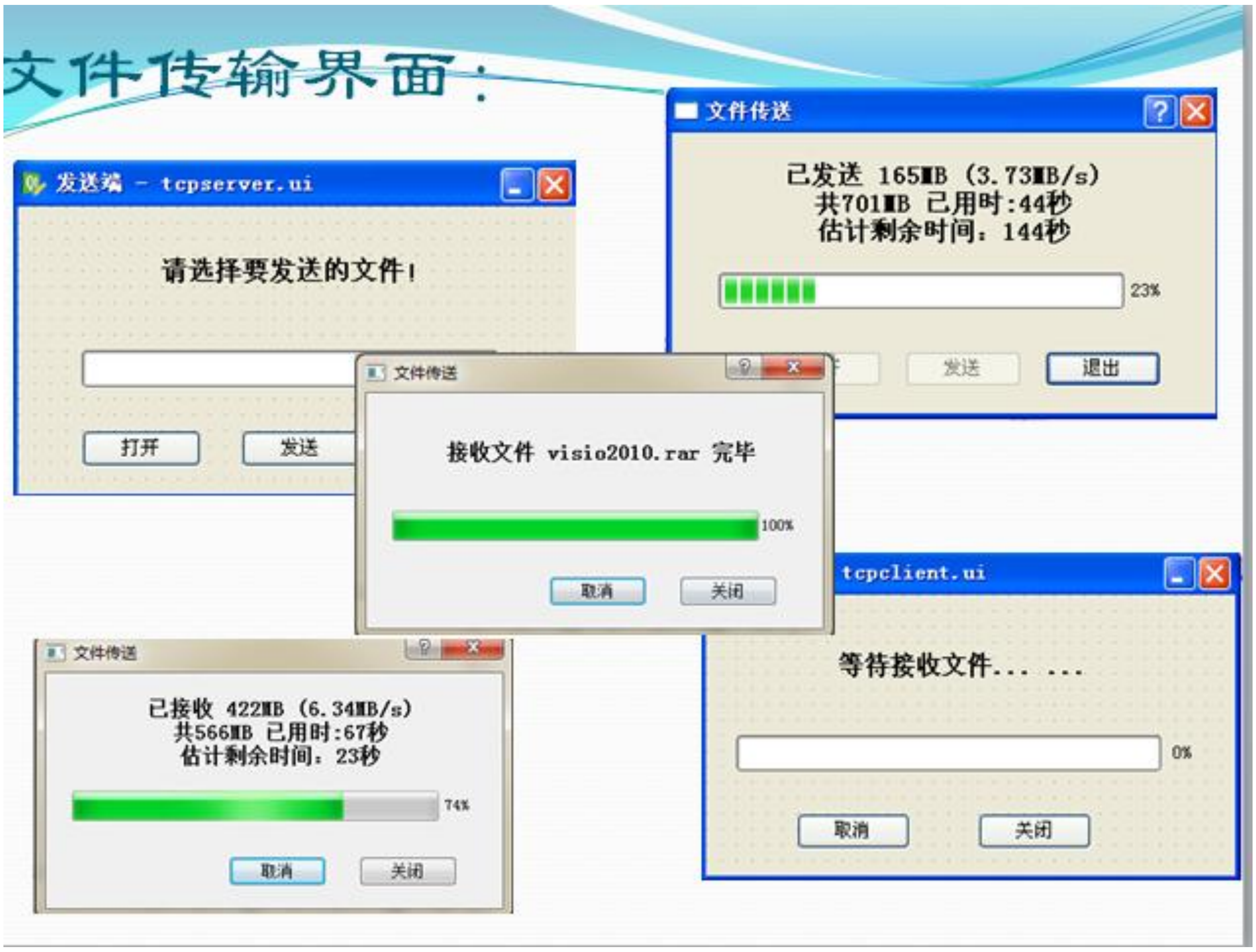
	用户名	主机名	IP地址
1	Administrator	WIN-NVL6MPGLKS7	169.254.177.180
2	Administrator	tornadomeet	192.168.1.2

Dialog

与WIN-NVL6MPGLKS7聊天中　对方IP:169.254.177.180

[WIN-NVL6MPGLKS7] 2012-07-04 10:40:32
在干什么？。
[tornadomeet] 2012-07-04 10:40:58
和你聊天啊~...

文件传输过程截图：



请选择要发送的文件！

已发送 165MB (3.73MB/s)
共701MB 已用时:44秒
估计剩余时间：144秒
23%

接收文件 visio2010.rar 完毕
100%

已接收 422MB (6.34MB/s)
共566MB 已用时:67秒
估计剩余时间：23秒
74%

等待接收文件... ...
0%

实验总结（下面几点只是暂时的理解）：

1. 使用类时，如果直接用构造函数定义该类的对象，则定义该类的函数接收时，该对象的生命也就结束了，所以如果要在其他函数中定义一个类的对象时并长久使用，可以使用 new 定义一个对象的初始指针。这样就在内存中永存了。
2. 如果某个窗口类需要显示时直接调用其指针->show()或者其对象-.show(),这个函数只是将内存中该类的对象显示出来而已（因为与界面有关），并不是重新建一个类对象。其表示该类的界面等可以显示，所以一旦 show 过即使改变了界面的内容，后面也无需一直调用 show 函数，界面会自动显示的。
3. 当关闭某个窗口时，只是将其隐藏，并没有释放其内存。

程序源码：

widget.h

```
#ifndef WIDGET_H
#define WIDGET_H

#include <QWidget>
#include <QtNetwork>
#include <QtGui>
```

```
#include "tcpclient.h"
#include "tcpserver.h"
#include "chat.h"
using namespace std::tr1;
namespace Ui {
    class Widget;
}

//enum MessageType
//{
//      Message,
//      NewParticipant,
//      ParticipantLeft,
//      FileName,
//      Refuse,
//      xchat
//};
//枚举变量标志信息的类型，分别为消息，新用户加入，和用户退出
class Widget : public QWidget
{
    Q_OBJECT

public:
    explicit Widget(QWidget *parent = 0);
    ~Widget();
    QString getUserName();
    QString getMessage();
    chat* privatechat;
    chat* privatechat1;

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

     bool eventFilter(QObject *target, QEvent *event);//事件过滤器
private:
    Ui::Widget *ui;
    QUdpSocket *udpSocket;
    qint32 port;
    qint32 bb;
    QString fileName;
    TcpServer *server;
    //chat *privatechat;

    QString getIP();

    QColor color;//颜色

    bool saveFile(const QString& fileName);//保存聊天记录
    void showxchat(QString name, QString ip);

private slots:
    void on_tableWidget_doubleClicked(QModelIndex index);
    void on_textUnderline_clicked(bool checked);
    void on_clear_clicked();
    void on_save_clicked();
    void on_textcolor_clicked();
    void on_textitalic_clicked(bool checked);
    void on_textbold_clicked(bool checked);
    void on_fontComboBox_currentFontChanged(QFont f);
    void on_fontsizecomboBox_currentIndexChanged(QString );
    void on_close_clicked();
    void on_sendfile_clicked();
    void on_send_clicked();
    void processPendingDatagrams();
    void sentFileName(QString);
    void currentFormatChanged(const QTextCharFormat &format);

signals:


};

#endif // WIDGET_H
```

widget.cpp:

```
#include "widget.h"
#include "ui_widget.h"
using namespace std::tr1;
Widget::Widget(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Widget)
{
    ui->setupUi(this);
    this->resize(850,550);
    ui->textEdit->setFocusPolicy(Qt::StrongFocus);
    ui->textBrowser->setFocusPolicy(Qt::NoFocus);

    ui->textEdit->setFocus();
    ui->textEdit->installEventFilter(this);//设置完后自动调用其 eventFilter 函数
    privatechat = NULL;
    privatechat1 = NULL;

    udpSocket = new QUdpSocket(this);
    port = 45454;
    bb = 0;
    udpSocket->bind(port,QUdpSocket::ShareAddress  | QUdpSocket::ReuseAddressHint);
    connect(udpSocket,SIGNAL(readyRead()),this,SLOT(processPendingDatagrams()));
    sendMessage(NewParticipant);

    server = new TcpServer(this);
    connect(server,SIGNAL(sendFileName(QString)),this,SLOT(sentFileName(QString)));

connect(ui->textEdit,SIGNAL(currentCharFormatChanged(QTextCharFormat)),this,SLOT(currentFo
rmatChanged(const QTextCharFormat)));

}

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

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

ui->fontsizecomboBox->setCurrentIndex(ui->fontsizecomboBox->findText(QString::number(forma
t.fontPointSize())));

    }

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

void Widget::processPendingDatagrams()      //接收数据 UDP
{
    while(udpSocket->hasPendingDatagrams())
    {
        QByteArray datagram;
        datagram.resize(udpSocket->pendingDatagramSize());
        udpSocket->readDatagram(datagram.data(),datagram.size());
        QDataStream in(&datagram,QIODevice::ReadOnly);
        int messageType;
        in >> messageType;
        QString userName,localHostName,ipAddress,message;
        QString time = QDateTime::currentDateTime().toString("yyyy-MM-dd hh:mm:ss");
        switch(messageType)
        {
            case Message:
                {
                    in >>userName >>localHostName >>ipAddress >>message;
                    ui->textBrowser->setTextColor(Qt::blue);
                    ui->textBrowser->setCurrentFont(QFont("Times  New Roman",12));
                    ui->textBrowser->append("[ " +localHostName+" ] "+ time);
                    ui->textBrowser->append(message);
                    break;
                }
            case NewParticipant:
                {
                    in >>userName >>localHostName >>ipAddress;
                    newParticipant(userName,localHostName,ipAddress);

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

                if(ipAddress == serverAddress)
                {
                    server->refused();
                }
                break;
            }
        case Xchat:
            {
                in >>userName >>localHostName >>ipAddress;
                showxchat(localHostName,ipAddress);//显示与主机名聊天中，不是用户名
                break;
            }
        }
    }
}

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

```
            ui->onlineUser->setText(tr("在线人数：%1").arg(ui->tableWidget->rowCount()));
            sendMessage(NewParticipant);
        }
}

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

Widget::~Widget()
{
        delete ui;
//        delete privatechat;
//        privatechat = NULL;
        //udpSocket
        //server
}

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

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

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


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

                case Message :
                        {
                                if(ui->textEdit->toPlainText() == "")
                                {
                                        QMessageBox::warning(0,tr(" 警 告 "),tr(" 发 送 内 容 不 能 为 空
"),QMessageBox::Ok);
                                        return;
                                }
                              out << address << getMessage();

ui->textBrowser->verticalScrollBar()->setValue(ui->textBrowser->verticalScrollBar()->maximum());
                              break;

                        }
                case FileName:
                        {
                                int row = ui->tableWidget->currentRow();
                                QString clientAddress = ui->tableWidget->item(row,2)->text();
                                out << address << clientAddress << fileName;
                break;
            }
        case Refuse:
            {
                out << serverAddress;
                break;
            }
    }
    udpSocket->writeDatagram(data,data.length(),QHostAddress::Broadcast, port);

}

QString Widget::getUserName()   //获取用户名
{
    QStringList envVariables;
    envVariables << "USERNAME.*" << "USER.*" << "USERDOMAIN.*"
                 << "HOSTNAME.*" << "DOMAINNAME.*";
    QStringList environment = QProcess::systemEnvironment();
    foreach (QString string, envVariables)
    {
        int index = environment.indexOf(QRegExp(string));
        if (index != -1)
        {

            QStringList stringList = environment.at(index).split('=');
            if (stringList.size() == 2)
            {
                return stringList.at(1);
                break;
            }
        }
    }
    return false;
}

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

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

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

void Widget::sentFileName(QString fileName)
{
    this->fileName = fileName;
    sendMessage(FileName);
}

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

            }

        }
        else{
            sendMessage(Refuse,serverAddress);
        }
    }
}

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

```
void Widget::on_sendfile_clicked()
{
        if(ui->tableWidget->selectedItems().isEmpty())
        {
                QMessageBox::warning(0,tr("选择用户"),tr("请先从用户列表选择要传送的用户！
"),QMessageBox::Ok);
                return;
        }
        server->show();
        server->initServer();
}

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

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

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

//字体大小设置
void Widget::on_fontsizecomboBox_currentIndexChanged(QString  size)
{
    ui->textEdit->setFontPointSize(size.toDouble());
    ui->textEdit->setFocus();
}

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

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

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

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

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

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

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

        return true;
}

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


void Widget::on_tableWidget_doubleClicked(QModelIndex  index)
{
        if(ui->tableWidget->item(index.row(),0)->text()  == getUserName() &&
                ui->tableWidget->item(index.row(),2)->text() == getIP())
        {
                QMessageBox::warning(0,tr(" 警 告 "),tr(" 你 不 可 以 跟 自 己 聊 天 ！ ！ ！
"),QMessageBox::Ok);
        }
        else
        {
        //        else
                if(!privatechat){
            //    chat *privatechatTemp;
                privatechat = new chat(ui->tableWidget->item(index.row(),1)->text(), //接收主机名
                                              ui->tableWidget->item(index.row(),2)->text())  ;//接收用户
```

```
IP
            }
//            if( privatechat->is_opened )delete privatechat;//如果其曾经显示过则删除掉
            QByteArray data;
            QDataStream out(&data,QIODevice::WriteOnly);
            QString localHostName = QHostInfo::localHostName();
            QString address = getIP();
            out << Xchat << getUserName() << localHostName << address;

udpSocket->writeDatagram(data,data.length(),QHostAddress::QHostAddress(ui->tableWidget->it
em(index.row(),2)->text()), port);

//
privatechat->xchat->writeDatagram(data,data.length(),QHostAddress::QHostAddress(ui->tableWi
dget->item(index.row(),2)->text()), 45456);
         //    if(!privatechat->is_opened)
                privatechat->show();
            privatechat->is_opened = true;
      //        (privatechat->a) = 0;
      }

}

void Widget::showxchat(QString name, QString ip)
{
//      if(!privatechat){
  // chat *privatechatTemp;
      if(!privatechat1)
      privatechat1 = new chat(name,ip);
//      privatechat = privatechatTemp;}
//      chat privatechat(name,ip);//如果不用 new 函数，则程序运行时只是闪烁显示一下就没
了，因为类的生命周期结束了
//      privatechat->is_opened = false;
  // privatechat->show();
   //privatechat.textBrowser.show();
   //privatechat->is_opened = true;
      bb++;
      //delete privatechat;

}
```

tcpclient.h:

```
#ifndef TCPCLIENT_H
#define TCPCLIENT_H

#include <QDialog>
#include <QTcpSocket>
#include <QHostAddress>
#include <QFile>
#include <QTime>
namespace Ui {
    class TcpClient;
}

class TcpClient : public QDialog
{
    Q_OBJECT

public:
    explicit TcpClient(QWidget *parent = 0);
    ~TcpClient();
    void setHostAddress(QHostAddress  address);
    void setFileName(QString  fileName){localFile  = new QFile(fileName);}

protected:
    void changeEvent(QEvent *e);

private:
    Ui::TcpClient *ui;
    QTcpSocket *tcpClient;
    quint16 blockSize;
    QHostAddress  hostAddress;
    qint16 tcpPort;

    qint64 TotalBytes;
    qint64 bytesReceived;
    qint64 bytesToReceive;
    qint64 fileNameSize;
    QString  fileName;
    QFile  *localFile;
    QByteArray inBlock;

    QTime time;

private slots:
    void on_tcpClientCancleBtn_clicked();
```

```
        void on_tcpClientCloseBtn_clicked();
        void newConnect();
        void readMessage();
        void displayError(QAbstractSocket::SocketError);
};

#endif // TCPCLIENT_H
```

tcpclient.cpp:

```
#include "tcpserver.h"
#include "ui_tcpserver.h"
#include <QTcpSocket>
#include <QFileDialog>
#include <QMessageBox>

TcpServer::TcpServer(QWidget *parent):QDialog(parent),
  ui(new Ui::TcpServer)
{
        ui->setupUi(this);
        this->setFixedSize(350,180);

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

        initServer();

}

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

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

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

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

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

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

}

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

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

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

}

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

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

void TcpServer::on_serverSendBtn_clicked()    //发送
{
        if(!tcpServer->listen(QHostAddress::Any,tcpPort))//开始监听
        {
```

```
        qDebug() << tcpServer->errorString();
        close();
        return;
    }

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

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

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

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

    tcpServer->close();

}
```

tcpserver.h:

```
#ifndef TCPSERVER_H
#define TCPSERVER_H

#include <QDialog>
#include <QTcpServer>
#include <QFile>
```

```
#include <QTime>

namespace Ui {
        class TcpServer;
}

class TcpServer : public QDialog
{
        Q_OBJECT

public:
        explicit TcpServer(QWidget *parent = 0);
        ~TcpServer();
        void refused();

        void initServer();



protected:
        void changeEvent(QEvent *e);

private:
        Ui::TcpServer *ui;
        qint16 tcpPort;
        QTcpServer *tcpServer;
        QString fileName;
        QString theFileName;
        QFile *localFile;

        qint64 TotalBytes;
        qint64 bytesWritten;
        qint64 bytesToWrite;
        qint64 loadSize;
        QByteArray outBlock;//缓存一次发送的数据

        QTcpSocket *clientConnection;

        QTime time;//计时器

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

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

};

#endif // TCPSERVER_H
```

tcpserver.cpp:

```
#include  "tcpserver.h"
#include  "ui_tcpserver.h"
#include  <QTcpSocket>
#include  <QFileDialog>
#include  <QMessageBox>

TcpServer::TcpServer(QWidget *parent):QDialog(parent),
  ui(new Ui::TcpServer)
{
        ui->setupUi(this);
        this->setFixedSize(350,180);

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

        initServer();

}

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

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

```
            break;
        default:
            break;
        }
}

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

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

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

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

}

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

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

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

}

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

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

void TcpServer::on_serverSendBtn_clicked()    //发送
{
        if(!tcpServer->listen(QHostAddress::Any,tcpPort))//开始监听
```

```
    {
        qDebug() << tcpServer->errorString();
        close();
        return;
    }

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

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

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

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

    tcpServer->close();

}
```

chat.h:

```
#ifndef CHAT_H
#define CHAT_H

#include <QDialog>
#include <QtNetwork>
```

```
#include <QtGui>
#include "tcpclient.h"
#include "tcpserver.h"

namespace Ui {
        class chat;
}

enum MessageType
{
        Message,
        NewParticipant,
        ParticipantLeft,
        FileName,
        Refuse,
        Xchat
};

class chat : public QDialog
{
        Q_OBJECT


public:
        ~chat();
//        chat();
        chat(QString pasvusername, QString pasvuserip);
        QString xpasvuserip;
        QString xpasvusername;
        QUdpSocket *xchat;
        qint32 xport;
        void sendMessage(MessageType type,QString serverAddress="");
        quint16 a;
//        static    qint32 is_opened = 0;
        bool is_opened;

public slots:



protected:
        void hasPendingFile(QString userName,QString serverAddress,    //接收文件
                                        QString clientAddress,QString fileName);
        void participantLeft(QString userName,QString localHostName,QString time);
        bool eventFilter(QObject *target, QEvent *event); //事件过滤器
private:
        Ui::chat *ui;
        TcpServer *server;
        QColor color;//颜色
        bool saveFile(const QString& fileName);//保存聊天记录
        QString getMessage();
        QString getIP();
        QString getUserName();
        QString message;
        QString fileName;

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

};

#endif // CHAT_H
```

chat.cpp:

```
#include  "chat.h"
#include  "ui_chat.h"

//chat::chat():ui(new Ui::chat)
//{
//        is_opened = false;
//}
```

```
chat::chat(QString pasvusername, QString pasvuserip) : ui(new Ui::chat)
{
        ui->setupUi(this);
        ui->textEdit->setFocusPolicy(Qt::StrongFocus);
        ui->textBrowser->setFocusPolicy(Qt::NoFocus);

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

        a = 0;
        is_opened = false;
//         this->is_opened = false;
        xpasvusername = pasvusername;
        xpasvuserip = pasvuserip;

        ui->label->setText(tr("与%1 聊天中      对方 IP:%2").arg(xpasvusername).arg(pasvuserip));

        //UDP 部分
        xchat = new QUdpSocket(this);
        xport = 45456;
  //     xchat->bind(xport, QUdpSocket::ShareAddress | QUdpSocket::ReuseAddressHint);
        xchat->bind( QHostAddress::QHostAddress(getIP()), xport );
        connect(xchat, SIGNAL(readyRead()), this, SLOT(processPendingDatagrams()));

        //TCP 部分
        server = new TcpServer(this);
        connect(server,SIGNAL(sendFileName(QString)),this,SLOT(sentFileName(QString)));


connect(ui->textEdit,SIGNAL(currentCharFormatChanged(QTextCharFormat)),this,SLOT(currentFo
rmatChanged(const QTextCharFormat)));
}

chat::~chat()
{
        is_opened = false;
        delete ui;
}

bool chat::eventFilter(QObject *target, QEvent *event)
{
        if(target == ui->textEdit)
        {
                if(event->type() == QEvent::KeyPress)//按下键盘某键
```

```
            {
                QKeyEvent *k = static_cast<QKeyEvent *>(event);
                if(k->key() == Qt::Key_Return)//回车键
                {
                    on_send_clicked();
                    return true;
                }
            }
        }
        return QWidget::eventFilter(target,event);
}

//处理用户离开
void chat::participantLeft(QString userName,QString localHostName,QString time)
{
        ui->textBrowser->setTextColor(Qt::gray);
        ui->textBrowser->setCurrentFont(QFont("Times  New Roman",10));
        ui->textBrowser->append(tr("%1  于  %2  离开！ ").arg(userName).arg(time));
}

QString chat::getUserName()    //获取用户名
{
        QStringList envVariables;
        envVariables << "USERNAME.*" << "USER.*" << "USERDOMAIN.*"
                                  << "HOSTNAME.*" << "DOMAINNAME.*";
        QStringList environment = QProcess::systemEnvironment();
        foreach (QString string, envVariables)
        {
            int index = environment.indexOf(QRegExp(string));
            if (index != -1)
            {

                QStringList stringList = environment.at(index).split('=');
                if (stringList.size() == 2)
                {
                    return stringList.at(1);
                    break;
                }
            }
        }
        return false;
}

QString chat::getIP()   //获取 ip 地址
```

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

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

            }

        }
        else{
            sendMessage(Refuse,serverAddress);
        }
    }
}

void chat::processPendingDatagrams()     //接收数据 UDP
{
    while(xchat->hasPendingDatagrams())
    {
        QByteArray datagram;
        datagram.resize(xchat->pendingDatagramSize());
        xchat->readDatagram(datagram.data(),datagram.size());
        QDataStream in(&datagram,QIODevice::ReadOnly);
        int messageType;
        in >> messageType;
        QString userName,localHostName,ipAddress,messagestr;
        QString time = QDateTime::currentDateTime().toString("yyyy-MM-dd hh:mm:ss");
        switch(messageType)
        {
            case Xchat:
            {
//                  ui.show();
                break;
            }
            case Message:
                {
                    //这 2 条语句都没有用。why？？、
                    /*this->hide();
                    this->close();*/
                    in >>userName >>localHostName >>ipAddress >>messagestr;
                    ui->textBrowser->setTextColor(Qt::blue);
                    ui->textBrowser->setCurrentFont(QFont("Times  New Roman",12));
                    ui->textBrowser->append("[ " +localHostName+" ] "+ time);//与主机名
聊天中
                    ui->textBrowser->append(messagestr);
            //          ui->textBrowser->show();
                    //this->textBrowser->setTextColor(Qt::blue);
                    //this->textBrowser->setCurrentFont(QFont("Times  New Roman",12));
                    //this->textBrowser->append("[ " +localHostName+" ] "+ time);//与主机
名聊天中
                    //this->textBrowser->append(messagestr);

            //          a ++;
            //          if( is_opened == false )//加了这句，接收端 B 不显示端口了
                    {
                        this->show();////解决 bug1.收到私聊消息后才显示
                //          ui->textBrowser->show();
                    //      this->show();
                //          ui->textBrowser->show();
                    //      ui.show();
                    //      if( this->show()  )
                //          this->hide();
                //          0 == a;
                        is_opened = true;
                    }
                    break;
                }
        case FileName:
            {
                in >>userName >>localHostName >> ipAddress;
                QString clientAddress,fileName;
                in >> clientAddress >> fileName;
                hasPendingFile(userName,ipAddress,clientAddress,fileName);
                break;
            }
        case Refuse:
            {
                in >> userName >> localHostName;
                QString serverAddress;
                in >> serverAddress;
                QString ipAddress = getIP();

                if(ipAddress  == serverAddress)
                {
                    server->refused();
                }
                break;
            }
        case ParticipantLeft:
            {
                in >>userName >>localHostName;
                participantLeft(userName,localHostName,time);
                QMessageBox::information(0,tr(" 本 次 对 话 关 闭 "),tr(" 对 方 结 束 了 对 话
"),QMessageBox::Ok);
                a = 1;
                ui->textBrowser->clear();
                //is_opened  = true;
            //      this->is_opened  = false;
                ui->~chat();
                close();
            //      delete ui;
            //      ui = 0;
                break;
            }
        }
    }
}
```

```
void chat::sentFileName(QString fileName)
{
	this->fileName = fileName;
	sendMessage(FileName);
}

QString chat::getMessage()  //获得要发送的信息
{
	QString msg = ui->textEdit->toHtml();
	qDebug()<<msg;
	ui->textEdit->clear();
	ui->textEdit->setFocus();
	return msg;
}

//通过私聊套接字发送到对方的私聊专用端口上
void chat::sendMessage(MessageType type , QString serverAddress)   //发送信息
{
	QByteArray data;
	QDataStream out(&data,QIODevice::WriteOnly);
	QString localHostName = QHostInfo::localHostName();
	QString address = getIP();
	out << type << getUserName() << localHostName;


	switch(type)
	{
	case ParticipantLeft:
		{
			break;
		}
	case Message :
		{
			if(ui->textEdit->toPlainText() == "")
			{
				QMessageBox::warning(0,tr(" 警 告 "),tr(" 发 送 内 容 不 能 为 空
"),QMessageBox::Ok);
				return;
			}
			message = getMessage();
			out << address << message;

ui->textBrowser->verticalScrollBar()->setValue(ui->textBrowser->verticalScrollBar()->maximum());
			break;
```

```
        }
    case FileName:
            {
                QString clientAddress = xpasvuserip;
                out << address << clientAddress << fileName;
                break;
            }
    case Refuse:
            {
                out << serverAddress;
                break;
            }
    }
    xchat->writeDatagram(data,data.length(),QHostAddress::QHostAddress(xpasvuserip),
45456);

}

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

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

ui->fontsizecomboBox->setCurrentIndex(ui->fontsizecomboBox->findText(QString::number(forma
t.fontPointSize())));

    }

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

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

```
}

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

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

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

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

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

bool chat::saveFile(const QString &fileName)//保存文件
```

```
{
        QFile file(fileName);
        if(!file.open(QFile::WriteOnly  | QFile::Text))

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

        return true;
}

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

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



void chat::on_close_clicked()
{
        sendMessage(ParticipantLeft);
        a = 1;
        ui->textBrowser->clear();
        //is_opened  = true;
//        this->is_opened  = false;
        close();
        ui->~chat();
        //this->close();
        /*delete ui;
        ui = 0;*/

}

void chat::on_send_clicked()
{
        sendMessage(Message);
        QString localHostName = QHostInfo::localHostName();
        QString time = QDateTime::currentDateTime().toString("yyyy-MM-dd hh:mm:ss");
        ui->textBrowser->setTextColor(Qt::blue);
        ui->textBrowser->setCurrentFont(QFont("Times New Roman",12));
        ui->textBrowser->append("[ " +localHostName+" ] "+ time);
        ui->textBrowser->append(message);
//        is_opened = true;
}

void chat::on_sendfile_clicked()
{
        server->show();
        server->initServer();
}
```

main:

```
#include <QtGui/QApplication>
#include "widget.h"

int main(int argc, char *argv[])
{
        QApplication a(argc, argv);
        Widget w;
        QTextCodec::setCodecForTr(QTextCodec::codecForLocale());
        w.show();
        return a.exec();
}
```