**计算机科学与技术学院课程设计成绩单**

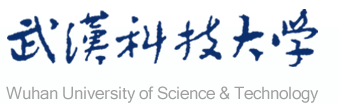
**课程名称：计算机网络课程设计**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **姓名** | **薛恩维** | | | **性别** | **男** | | **学号** | **201913137023** | | **班级** | **计卓1901** |
| **综合成绩** | |  | | | | **成绩等级** | | |  | | |
| **程序运行情况**  **（占总成绩20%）** | | | **□能正确运行 □基本能正确运行 □能运行但结果不完善**  **（20分） （15分） （10分）** | | | | | | | | |
| **程序功能的完善程度**  **（占总成绩10%）** | | | **□完善 □基本完善 □不完善**  **（10分） （8分） （5分）** | | | | | | | | |
| **程序结构的合理性**  **（占总成绩10%）** | | | **□合理 □基本合理 □不太合理**  **（10分） （8分） （5分）** | | | | | | | | |
| **对问题的答辩情况**  **（占总成绩40%）** | | | **□概念正确有创新 □能正确回答所有问题 □基本能正确回答**  **（40分） （35分） （30分）**  **□部分问题回答概念不清晰**  **（20分）** | | | | | | | | |
| **学生的工作态度与独立工作能力**  **（占总成绩10%）** | | | **□工作态度认真能独立完成任务 □工作态度认真但独立性较差**  **（10分） （8分）**  **□工作态度基本认真但缺乏独立性**  **（5分）** | | | | | | | | |
| **设计报告的规范性**  **（占总成绩10%）** | | | **□符合规范 □基本符合规范 □规范性较差**  **（10分） （8分） （5分）** | | | | | | | | |

A：90~100分 A-：85~89分 B+：82~84分 B：78~81分 B-：75~77分

C+：72~74分 C：68~71分 C-：64~67分 D：60~63分 F：<60分

武汉科技大学计算机科学与技术学院制表

** **

**计算机科学与技术学院**

**课 程 设 计 报 告**

**课程名称：计算机网络**

**专 业： 计算机科学与技术（卓越）**

**班 级：** 2019级 1班

**学 号：** 201913137023

**姓 名：** 薛恩维

### 题目3 基于TCP/UDP协议的数据传输程序

一.需求分析：

基于TCP/UDP协议编程的方式，编写程序模拟数据的传输过程。

程序功能：

1.用户的注册和登录：用户可以注册，然后登录到系统，没注册用户不能登录系统。

2.两个用户之间发送数据，数据类型包括但不限于文件信息，图片和文件等，接收方需要选择文件存储位置。

3.文件群发功能：发送方能选择系统中的多个用户，进行群发文件数据。

4.图片预览：发送方如果发送的图片数据，能预览图片。。

二.概要设计：

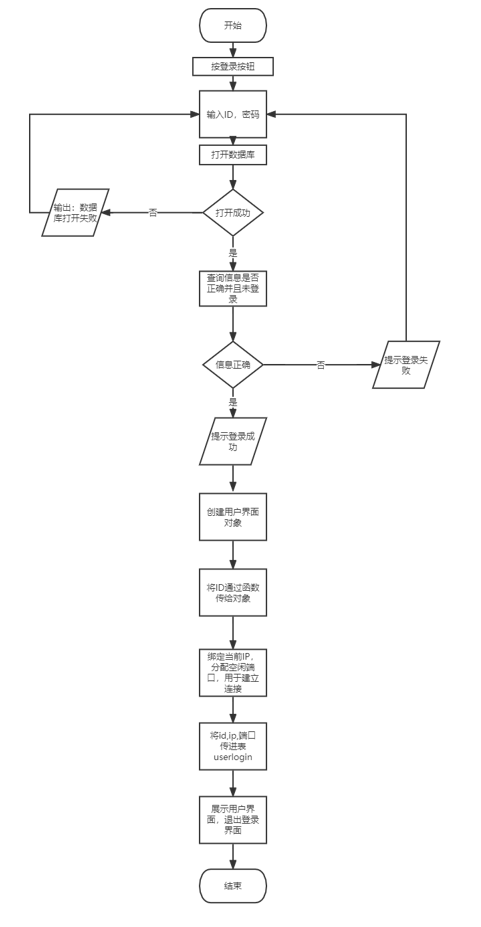
1.用户注册和登录部分算法:

用户需要注册和登录才能使用系统，没注册的用户不能登录，登录需要验证这个用户是否存在、是否已经登录并且输入正确的密码来证明自己是不是本人。注册需要判断这个用户是否存在。所以这个功能采用Sqlite数据库实现。建立用户信息数据库并创建用户数据表来存储相关信息。在登录或注册时进入数据库对相关信息进行查询，根据查询的结果来决定接下来执行什么步骤。

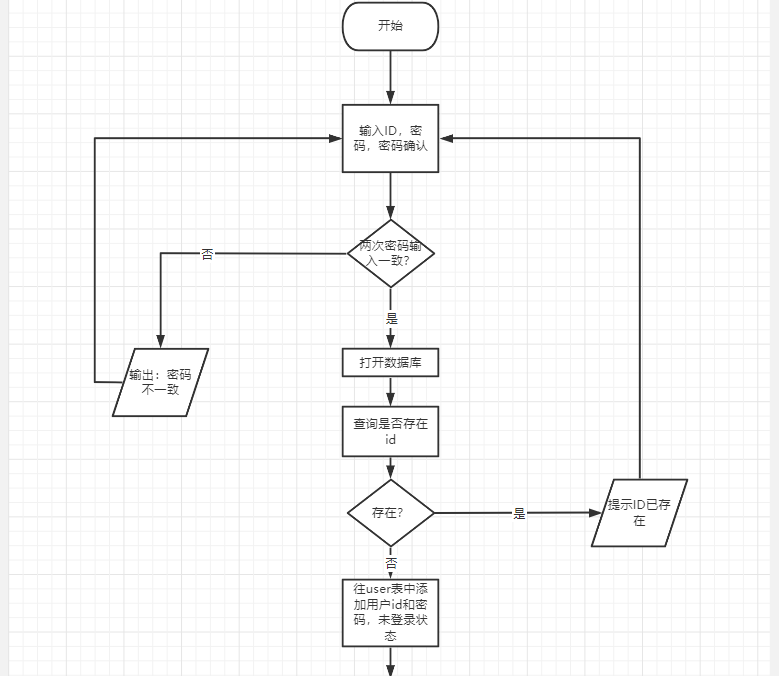
用户信息表user：用户名id（KEY）, 密码password和登录状态 is\_login。用于登录验证

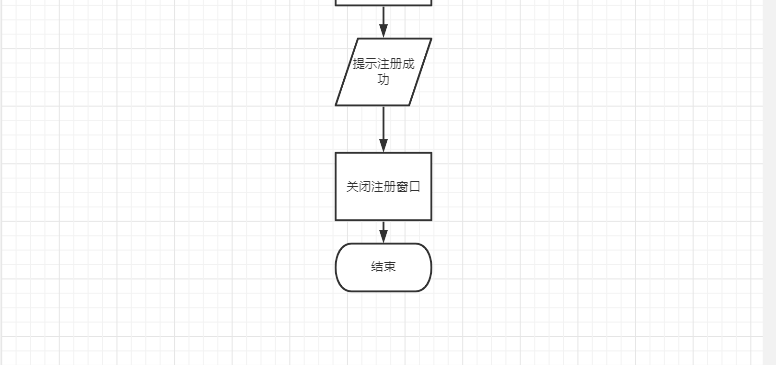
已登录用户网络信息表userlogin: 用户名id,（KEY） 网络地址ip,端口号port。用于与其他客户端建立连接

用户登录验证算法:



用户注册算法





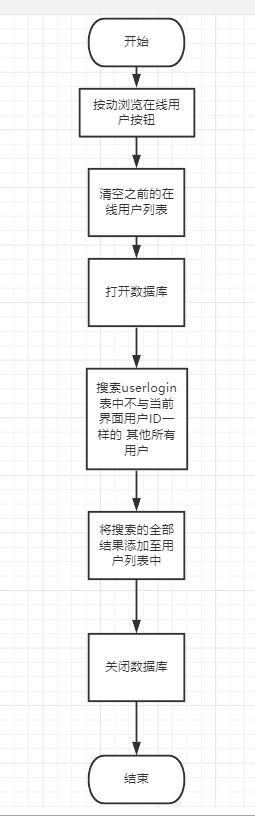
2.传输数据部分算法

通信协议的选择：响应快的UDP协议用来发文本数据，更稳定的TCP协议用于传输数据。

模式的选择：我选择了C/S模式来进行设计，这样用户之间可以直接点对点建立连接，不需要额外建立服务器来传输信息。每个用户端都相当于一个服务器端可以随时监听传输到的信息，发送信息时，发送方相当于一个客户端。

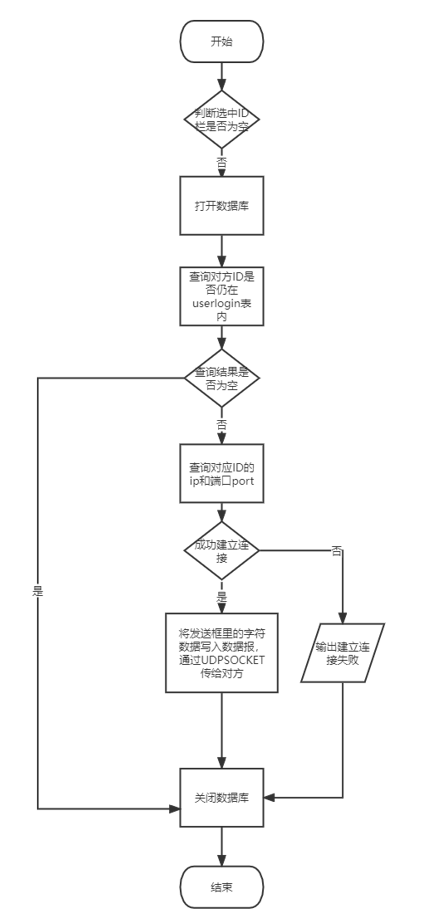
每个用户在登录之后都会自动绑定一个IP和端口，存入数据库中，之后会建立一个Udpsocket用于发送和接受文本数据和一个Tcpsocket用于接受传到这个IP下的端口的文件数据

手动更新在线用户列表算法

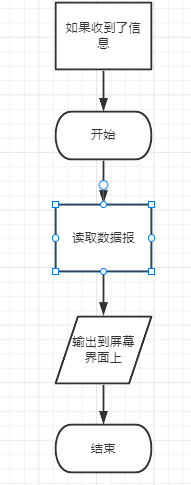


发送文本算法

发送端

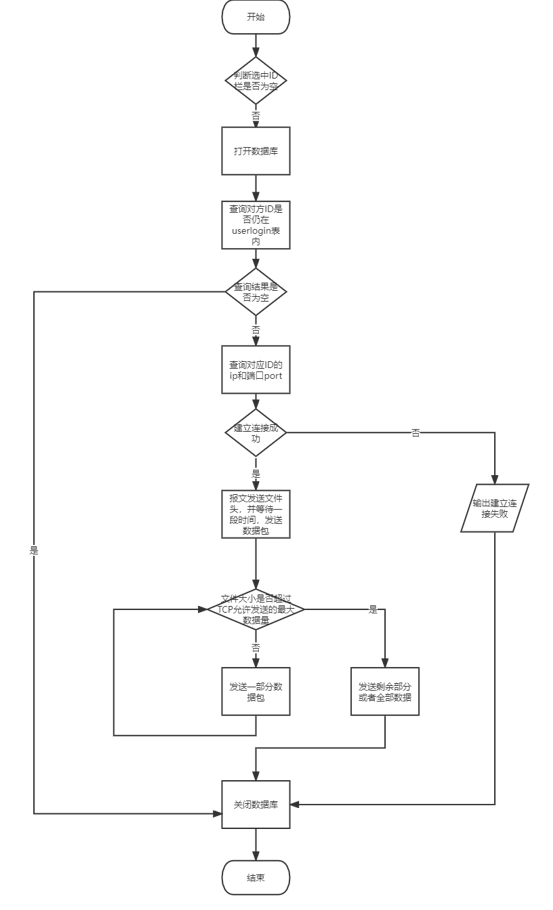


接收端

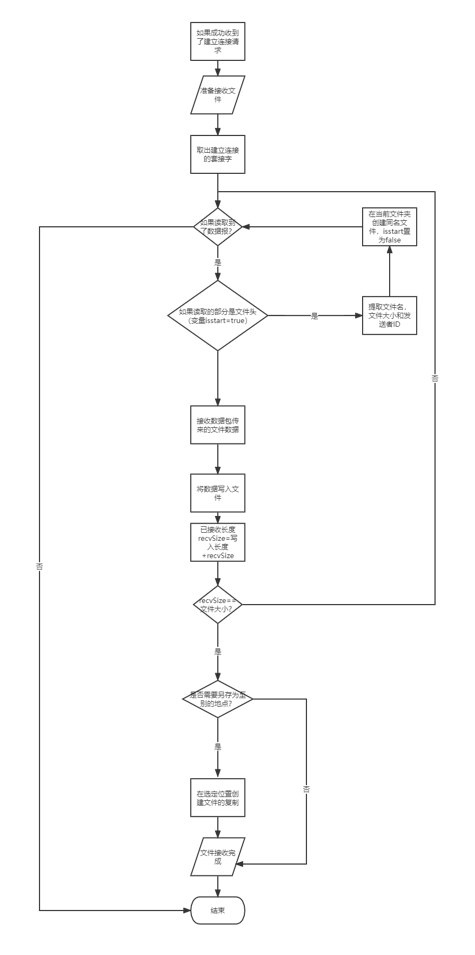


发送文件算法

发送端



接收端



群发文件算法

将要群发的ID填入IDlist中，采用for循环遍历IDlist，每遍历一次就建立一次连接，打开一次文件并发送，最后关闭文件。直到所有IDlist被遍历完为止。

图片预览

在发送文件算法中，文件传输完毕之后，使用Qpixmap类中的load函数对文件类型进行判定，如果是QT支持的图片文件类型，load返回true，则创建一个QGraphicscene容器，将文件加入容器中，通过QGrapihcview展示

存储结构数据说明

登录界面

public:

**MainWindow**(QWidget \*parent = nullptr);//构造函数

~***MainWindow***();//析构函数

private:

Ui::MainWindow \*ui;//ui窗口

QSqlDatabase user\_database;//数据库，默认连接至“database.db”文件，如果没有，则在构造函数中创建一个

private slots:

void **on\_exitButton\_clicked**();//退出

void **on\_loginButton\_clicked**();//登录按钮

void **on\_rigestButton\_clicked**();//注册界面打开

注册界面

public:

explicit **regist**(QWidget \*parent = nullptr);//构造函数

~***regist***();//析构

private slots:

void **on\_re\_button\_clicked**();//注册按钮

void **on\_ex\_button\_clicked**();//退出按钮

private:

Ui::regist \*ui;//ui界面

QSqlDatabase user\_database;//所连接的数据库

用户界面

public:

explicit **user**(QWidget \*parent = nullptr);//构造函数

~***user***();//析构函数

void **get\_id**(QString id);//获得登录界面ID

void **set\_id**();//将ID展示至用户界面

void **set\_port**();//将IP和端口放入数据库

void **dealMsg**();//接收文本数据

void **sendData**();//接收文件数据包

private slots:

void **on\_closeButton\_clicked**();//关闭窗口

void **on\_textSend\_clicked**();//发送文本

void **on\_viewbutton\_clicked**();//刷新在线用户列表

void **on\_fileButton\_clicked**();//选择文件

void **on\_sendButton\_clicked**();//对一个人发文件

void **on\_pushButton\_clicked**();//将ID添加至群发列表中

void **on\_pushButton\_2\_clicked**();//群发

void **on\_pushButton\_3\_clicked**();//移除被选中的ID

void **on\_pushButton\_4\_clicked**();//清空群发列表

private:

Ui::user \*ui;

QString user\_id;//登录用户ID

QSqlDatabase user\_database;

QUdpSocket \*udpSocket;// UDP套接字

QString ip;//用户IP

quint16 port;//用户端口

QTcpSocket \*tcpSocket;// TCP套接字

QTcpServer \*tcpServer;//服务器

QFile file; //文件对象，用于发送或保存

QString fname;//文件名

quint64 fileSize;//文件大小

quint64 sendSize;//已接收文件的大小

QTimer timer; //定时器

quint64 recvSize; //已经接收文件的大小

bool isstart;//发送的数据包是文件头还是正文

QStringList idlist;

图片预览界面

public:

explicit **picture**(QWidget \*parent = nullptr);//构造函数

void **showpicture**(QString fname);//预览图片

~***picture***();

private:

Ui::picture \*ui;

三.详细设计

登录

user\_database.open();

if (!user\_database.open())

{

qDebug() << "Error: Failed to connect database." << user\_database.lastError();

}

else

{

QSqlQuery sql\_query;

QString que=QString("select id from user where id='%1' and password='%2' and is\_login=%3;").arg(ui->ID\_INPUT->text()).arg(ui->PASSWORD\_INPUT->text()).arg(false);//创建SQL查询语句

sql\_query.exec(que);

sql\_query.next();

QString str=sql\_query.value(0).toString();

if(str==""){

QMessageBox::warning(this,QString QStringLiteral("通知"),QString QStringLiteral("登录失败，账号不存在，密码错误或者已登录"));

}

else{

// QMessageBox::warning(this,QString QStringLiteral("通知"),QString QStringLiteral("登录成功"));

QString now\_id=ui->ID\_INPUT->text();

QString que1=QString("update user set is\_login=%2 where id='%1';").arg(ui->ID\_INPUT->text()).arg(1);//更新登录状态

sql\_query.exec(que1);

user \*a=new user();//创建用户界面对象

a->get\_id(now\_id);//传给用户对象登录的ID

a->set\_id();

a->set\_port();//绑定端口和IP

a->show();

this->close();//关闭开始界面

}

}

user\_database.close();

注册

bool t=true;

QString str1 QStringLiteral("警告");

QString str2 QStringLiteral("密码不一致");

if(ui->pass\_word->text()!=ui->pass\_word\_sure->text()){

t=false;

QMessageBox::warning(this,str1,str2);//密码是否一致

}

else{

t=true;

}

user\_database.open();

if (!user\_database.open())

{

qDebug() << "Error: Failed to connect database." << user\_database.lastError();

}

else

{

QSqlQuery sql\_query;

QSqlQuery sql\_query1;

QString re1=QString("select id from user where id='%1';").arg(ui->new\_ID\_input->text());//查询ID是否存在

sql\_query1.exec(re1);

sql\_query1.first();

if(sql\_query1.value(0).toString()!=NULL){

t=false;

QMessageBox::warning(this,str1,QString QStringLiteral("ID已被注册"));

}

if(t==true)

{

if(ui->new\_ID\_input->text().length()<=20&&ui->pass\_word->text().length()<=20){

QString re=QString("insert into user values('%1','%2',%3);").arg(ui->new\_ID\_input->text()).arg(ui->pass\_word->text()).arg(false);//注册信息登入数据库

sql\_query.exec(re);

QMessageBox::warning(this,QString QStringLiteral("通知"),QString QStringLiteral("注册成功"));

user\_database.close();

this->close();

}

else{

QMessageBox::warning(this,QString QStringLiteral("通知"),QString QStringLiteral("注册失败，请检查密码和id是否合法"));

}

}

}

user\_database.close();

构造函数中绑定IP和端口号，建立对Udp连接和Tcp连接的监听

udpSocket = new QUdpSocket(this);

tcpServer = new QTcpServer(this);

QHostAddress address = QHostAddress(QHostAddress::LocalHost);

ip=address.toString();

udpSocket->bind(0,QUdpSocket::ShareAddress);//分配空闲端口，监听UDP连接

port=udpSocket->localPort();//将端口号通过函数传递给参数port，之后存于数据库中

isstart=true;

tcpServer->listen(QHostAddress::Any,port);//用于监听Tcp连接

UDP接收文本数据函数

connect(udpSocket, &QUdpSocket::readyRead, this, &user::dealMsg);//监听到UDP连接的反应

void user::**dealMsg**(){

//读取对方发送的内容

char buf[1024] = {0};

QHostAddress cliAddr; //对方地址

quint16 port; //对方端口

qint64 len = udpSocket->readDatagram(*buf*, sizeof(buf), *&cliAddr*, *&port*);

if(len > 0)

{

QString str = QString("%1")

.arg(buf);

//给编辑区设置内容

ui->Infotext->append(str);

}

}

TCP接收文件

connect(tcpServer, &QTcpServer::newConnection,//一旦建立起了连接

[=](){

ui->Infotext->append(QString QStringLiteral("准备接受文件"));

tcpSocket = tcpServer->*nextPendingConnection*();//取出用于建立连接的套接字

connect(tcpSocket, &QTcpSocket::readyRead,

[=](){

//取出接收的内容

if(tcpSocket->isValid()==false){

return ;

}

QByteArray buf = tcpSocket->readAll();

if(isstart==true)//文件头读取

{

//文件名

QString temp=QString(buf);

fname = temp.section("##", 0, 0);

//文件大小

fileSize = temp.section("##", 1, 1).toInt();

QString peername;

peername = temp.section("##", 2, 2);//发送人

recvSize = 0; //已经接收文件大小

isstart=false;

QString str = QString QStringLiteral("接收来自%3的文件: [%1: %2b]").arg(fname).arg(fileSize).arg(peername);

ui->Infotext->append(str);

file.setFileName(fname);

bool isOk = file.*open*(QIODevice::WriteOnly);//创建空白文件

if(false == isOk)

{

qDebug() << "WriteOnly error 49";

tcpSocket->*disconnectFromHost*(); //断开连接

tcpSocket->*close*(); //关闭套接字

return; //如果打开文件失败，中断函数

}

isstart=false;

}

else{

qint64 len = file.write(buf);//将数据写入文件

if(len >0){ //接收数据大于0

recvSize += len; //累计接收大小

}

if(recvSize == fileSize){//图片判定

QPixmap \*qp=new QPixmap();

if(qp->load(fname)){

picture \*p=new picture();

p->showpicture(fname);

p->show();//图片预览

}

file.*close*(); //关闭文件

QMessageBox messageBox(QMessageBox::NoIcon,QString QStringLiteral("通知"),QString QStringLiteral("文件已保存至当前文件夹，是否需要保存至别处"),QMessageBox::Yes|QMessageBox::No);//选择保存位置

int result=messageBox.*exec*();

if(result==QMessageBox::Yes)

{

QString SavePath = QFileDialog::getSaveFileName(this,"SAVE",fname);

if(!QFile::copy(fname,SavePath))

{

ui->Infotext->append(QString QStringLiteral("文件另存为失败"));

}

else{

QString str1 = QString QStringLiteral("另存为成功");

ui->Infotext->append(str1);

file.remove();

}

}

tcpSocket->*disconnectFromHost*();

tcpSocket->*close*();

QString str1 = QString QStringLiteral("文件接受完毕");

ui->Infotext->append(str1);

isstart=true;

}

}

});

});

发送文本

if(ui->ID\_box->currentText()==""){

QMessageBox::warning(this,QString QStringLiteral("通知"),QString QStringLiteral("对方ID不能为空"));

}

else{

user\_database.open();

QSqlQuery sql\_query1;

QString que1=QString("select id from userlogin where id='%1';").arg(ui->ID\_box->currentText());

sql\_query1.exec(que1);

sql\_query1.first();

user\_database.close();

if(sql\_query1.value(0).toString()==NULL){//确认对方是否离线

QMessageBox::warning(this,QString QStringLiteral("通知"),QString QStringLiteral("对方已离线，无法发送"));

return;

}

else{

user\_database.open();

QSqlQuery sql\_query1;

QString que1=QString("select \* from userlogin where id='%1';").arg(ui->ID\_box->currentText());

sql\_query1.exec(que1);

sql\_query1.first();

QString ip1=sql\_query1.value(1).toString();//锁定对方IP

qint16 port1=sql\_query1.value(2).toInt();//锁定对方端口

QString str = user\_id+QString QStringLiteral("(我)")+" to "+ui->ID\_box->currentText()+": "+ui->Sendtext->toPlainText();

QString str1 =user\_id+" : "+ui->Sendtext->toPlainText();

ui->Infotext->append(str);

qint64 len =udpSocket->writeDatagram(str1.toUtf8(),str1.toUtf8().size(), QHostAddress(ip1), port1);//发送数据报

if(len==-1){

QMessageBox::information(this,tr("error"),tr("writedatagram failed!"));

}

user\_database.close();

}

}

选取文件

QString filePath = QFileDialog::getOpenFileName(this, "open", "../");

if(false == filePath.isEmpty()){

fname.clear();

ui->lineEdit->text().clear();

fileSize = 0; //获取文件信息

QFileInfo info(filePath);

fname = info.fileName(); //获取文件名字

fileSize = info.size(); //获取文件大小

sendSize = 0; //发送文件的大小

//只读方式打开文件

//指定文件的名字

file.setFileName(filePath);

//打开文件

bool isOk = file.*open*(QIODevice::ReadOnly);

if(false == isOk)

{

qDebug() << "只读方式打开文件失败 106";

}

ui->lineEdit->setText(fname);

ui->sendButton->setEnabled(true);

ui->pushButton\_2->setEnabled(true);

}

else{

QMessageBox::warning(this,QString QStringLiteral("通知"),QString QStringLiteral("未选中文件"));

}

//读取文件数据，并将文件数据写入数据包

void user::**sendData**(){

quint64 len = 0;

do

{

//每次发送数据的大小

char buf[4\*1024] = {0};

len = 0;

//往文件中读数据

len = file.read(*buf*, sizeof(buf));

//发送数据，读多少，发多少

len =tcpSocket->write(buf, len);

//发送的数据需要累积

sendSize += len;

}while(len > 0);

}

发送文件

if(ui->ID\_box->currentText()==""){

QMessageBox::warning(this,QString QStringLiteral("通知"),QString QStringLiteral("对方ID不能为空"));

}

else{

user\_database.open();

QSqlQuery sql\_query1;

QString que1=QString("select id from userlogin where id='%1';").arg(ui->ID\_box->currentText());//是否离线？

sql\_query1.exec(que1);

sql\_query1.first();

user\_database.close();

if(sql\_query1.value(0).toString()==NULL){

QMessageBox::warning(this,QString QStringLiteral("通知"),QString QStringLiteral("对方已离线，无法建立连接"));

}

else{

user\_database.open();

QSqlQuery sql\_query1;

QString que1=QString("select \* from userlogin where id='%1';").arg(ui->ID\_box->currentText());

sql\_query1.exec(que1);

sql\_query1.first();

QString ip1=sql\_query1.value(1).toString();//获得IP

quint16 port1=sql\_query1.value(2).toUInt();//获得端口

tcpSocket = new QTcpSocket(this);

tcpSocket->*connectToHost*(QHostAddress(ip1), port1);//建立TCP连接

tcpSocket->*waitForConnected*(5000);

QAbstractSocket::SocketState s=tcpSocket->state();

if(s==QAbstractSocket::ConnectedState){//建立成功

ui->Infotext->append(QString QStringLiteral("已和对方建立连接，准备发送文件"));

ui->fileButton->setEnabled(false);

ui->sendButton->setEnabled(false);

//先发送文件头信息 文件名##文件大小##发送者名称

QString head = QString("%1##%2##%3").arg(fname).arg(fileSize).arg(user\_id);

QString str =QString QStringLiteral("%1 %2b").arg(fname).arg(fileSize);

ui->Infotext->append(str);

//发送头部信息

qint64 len = tcpSocket->write(head.toUtf8());

tcpSocket->*waitForBytesWritten*(20);

if(len > 0)//头部信息发送成功

{

sendData();

}

else

{

qDebug() << QString QStringLiteral("头部信息发送失败 142");

file.*close*();

ui->fileButton->setEnabled(true);

ui->sendButton->setEnabled(false);

}

ui->Infotext->append(QString QStringLiteral("发送完成"));

}

else{

ui->Infotext->append(QString QStringLiteral("连接失败，无法发送文件"));

}

user\_database.close();

ui->fileButton->setEnabled(true);

}

}

在线用户列表刷新

ui->ID\_box->clear();//清空之前的列表

user\_database.open();

QSqlQuery sql\_query1;

QString que1=QString("select id from userlogin where id!='%1';").arg(user\_id);//创建查询

sql\_query1.exec(que1);

while(sql\_query1.next()){

ui->ID\_box->addItem(sql\_query1.value(0).toString());//加入列表

}

user\_database.close();

添加ID至群发列表IDlist中

int t=1;

QString s=ui->ID\_box->currentText();

if(s==""){

QMessageBox::warning(this,QString QStringLiteral("通知"),QString QStringLiteral("添加ID不能为空"));

return;

}

for(int i=0;i<idlist.size();i++){

if(idlist[i]==s){

t=0;

break;

}

}

if(t==0){

QMessageBox::warning(this,QString QStringLiteral("通知"),QString QStringLiteral("对方已在群发名单中"));

return;

}

else{

idlist.append(s);

QMessageBox::warning(this,QString QStringLiteral("通知"),QString QStringLiteral("添加成功"));

QStandardItemModel \*ItemModel;

ItemModel = new QStandardItemModel();

int nCount = idlist.size();

for(int i = 0; i < nCount; i++)

{

QStandardItem \*item = new QStandardItem(idlist[i]);

ItemModel->appendRow(*item*);

}

ui->listView->*setModel*(*ItemModel*);

ui->listView->show();//将新的列表从LIST展示出来

}

从群发列表移除

int t=1;

int index=-1;

QString s=ui->ID\_box->currentText();

if(s==""){

QMessageBox::warning(this,QString QStringLiteral("通知"),QString QStringLiteral("移除ID不能为空"));

return;

}

for(int i=0;i<idlist.size();i++){

if(idlist[i]==s){

t=0;

index=i;

break;

}

}

if(t==1){

QMessageBox::warning(this,QString QStringLiteral("通知"),QString QStringLiteral("对方不在群发名单中"));

return;

}

else{

idlist.removeAt(index);

QMessageBox::warning(this,QString QStringLiteral("通知"),QString QStringLiteral("移除成功"));

QStandardItemModel \*ItemModel;

ItemModel = new QStandardItemModel();

int nCount = idlist.size();

for(int i = 0; i < nCount; i++)

{

QStandardItem \*item = new QStandardItem(idlist[i]);

ItemModel->appendRow(*item*);

}

ui->listView->*setModel*(*ItemModel*);

ui->listView->show();

}

群发

if(idlist.size()==0){

QMessageBox::warning(this,QString QStringLiteral("通知"),QString QStringLiteral("群发人员不能为空"));

ui->pushButton\_2->setEnabled(false);

return;

}

//开始群发

for(int i=0;i<idlist.size();i++){//代码基本结构同单发文件

QString s=idlist[i];

file.*open*(QIODevice::ReadOnly);//读文件

user\_database.open();

QSqlQuery sql\_query1;

QString que1=QString("select id from userlogin where id='%1';").arg(s);

sql\_query1.exec(que1);

sql\_query1.first();

user\_database.close();

if(sql\_query1.value(0).toString()==NULL){

QMessageBox::warning(this,QString QStringLiteral("通知"),QString QStringLiteral("对方已离线，无法建立连接"));

ui->pushButton\_2->setEnabled(false);

return;

}

user\_database.open();

QSqlQuery sql\_query2;

QString que2=QString("select \* from userlogin where id='%1';").arg(s);

sql\_query2.exec(que2);

sql\_query2.first();

QString ip1=sql\_query2.value(1).toString();

quint16 port1=sql\_query2.value(2).toUInt();

tcpSocket = new QTcpSocket(this);

tcpSocket->*connectToHost*(QHostAddress(ip1), port1);

tcpSocket->*waitForConnected*(5000);

QAbstractSocket::SocketState s1=tcpSocket->state();

if(s1==QAbstractSocket::ConnectedState){

ui->Infotext->append(QString QStringLiteral("已和对方建立连接，准备发送文件"));

ui->fileButton->setEnabled(false);

//先发送文件头信息 文件名##文件大小

QString head = QString("%1##%2##%3").arg(fname).arg(fileSize).arg(user\_id);

QString str =QString QStringLiteral("%1 %2b").arg(fname).arg(fileSize);

ui->Infotext->append(str);

//发送头部信息

qint64 len = tcpSocket->write(head.toUtf8());

tcpSocket->*waitForBytesWritten*(20);

if(len > 0)//头部信息发送成功

{

sendData();

}

else

{

qDebug() << QString QStringLiteral("头部信息发送失败 142");

file.*close*();

ui->fileButton->setEnabled(true);

ui->sendButton->setEnabled(false);

}

ui->Infotext->append(QString QStringLiteral("发送完成"));

}

else{

ui->Infotext->append(QString QStringLiteral("连接失败，无法发送文件"));

}

user\_database.close();

tcpSocket->*close*();

file.*close*();

}

ui->fileButton->setEnabled(true);

ui->pushButton\_2->setEnabled(false);

图片展示

//向容器中添加文件路径为fileName（QString类型）的文件

QGraphicsScene \*scene = new QGraphicsScene;

scene->addPixmap(QPixmap(fname));

//借助graphicsView（QGraphicsView类）控件显示容器的内容

ui->graphicsView->setScene(*scene*);

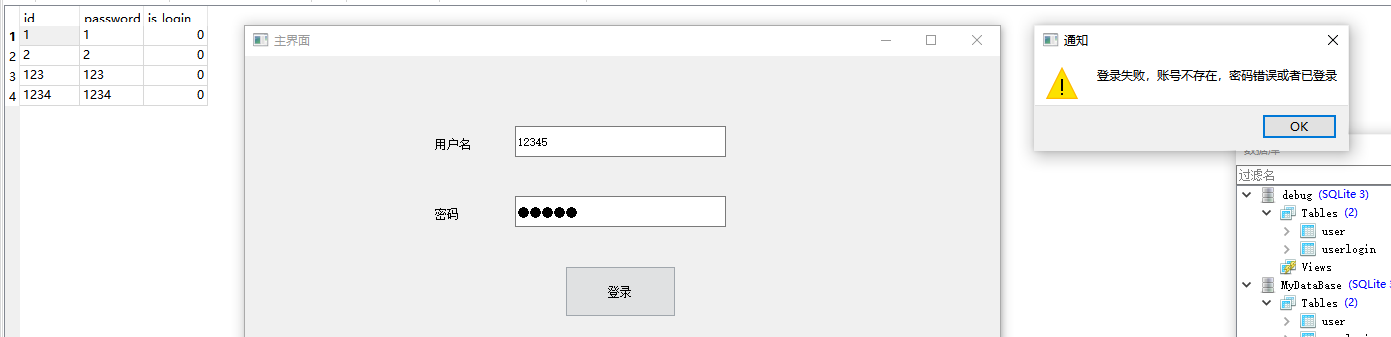
//开始显示

ui->graphicsView->show();

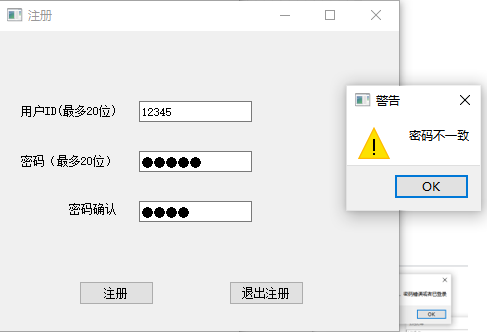
四.调试分析

功能实现截图

未注册用户无法登录，（图左边为数据库的user表）



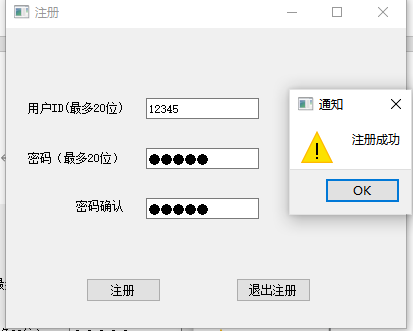
注册密码确认检测

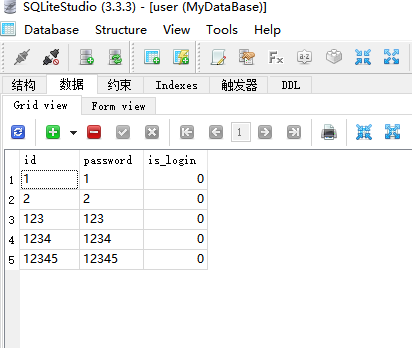


id重复检测

I

注册成功（密码同ID）





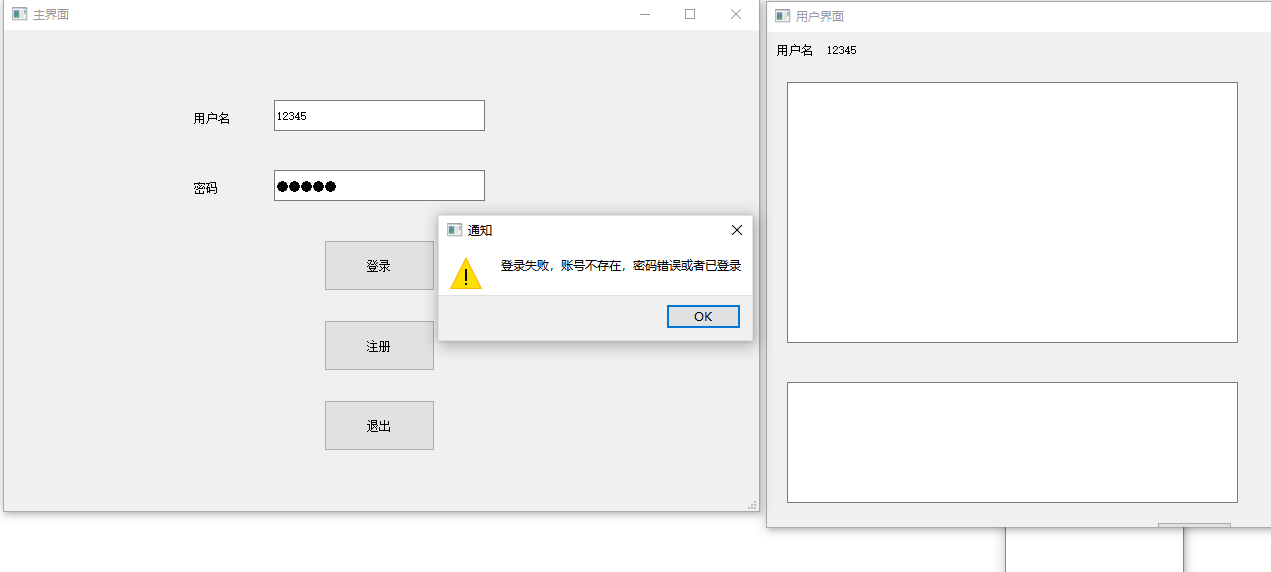
密码检测



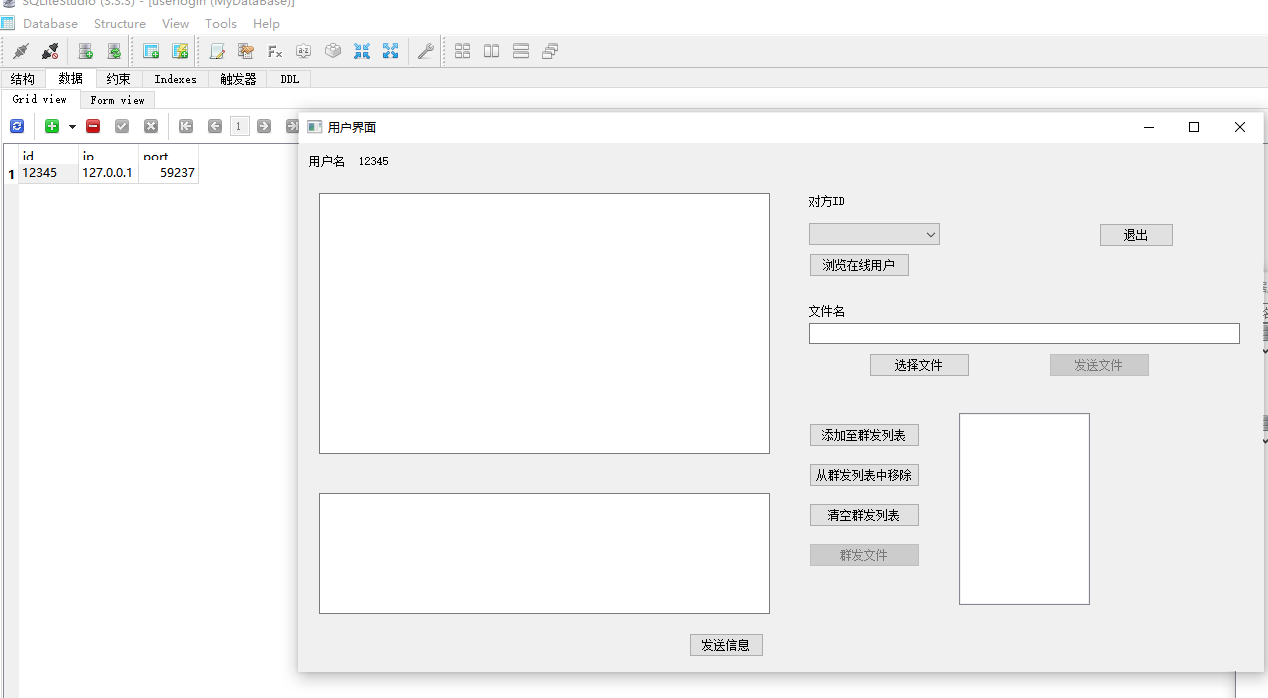
登录成功后切换到用户界面



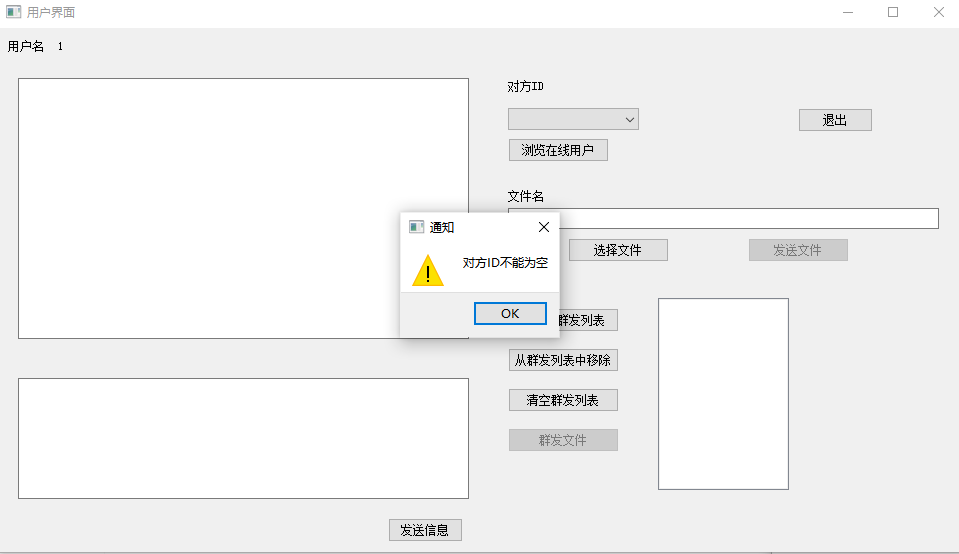
禁止重复登录



绑定IP和端口。



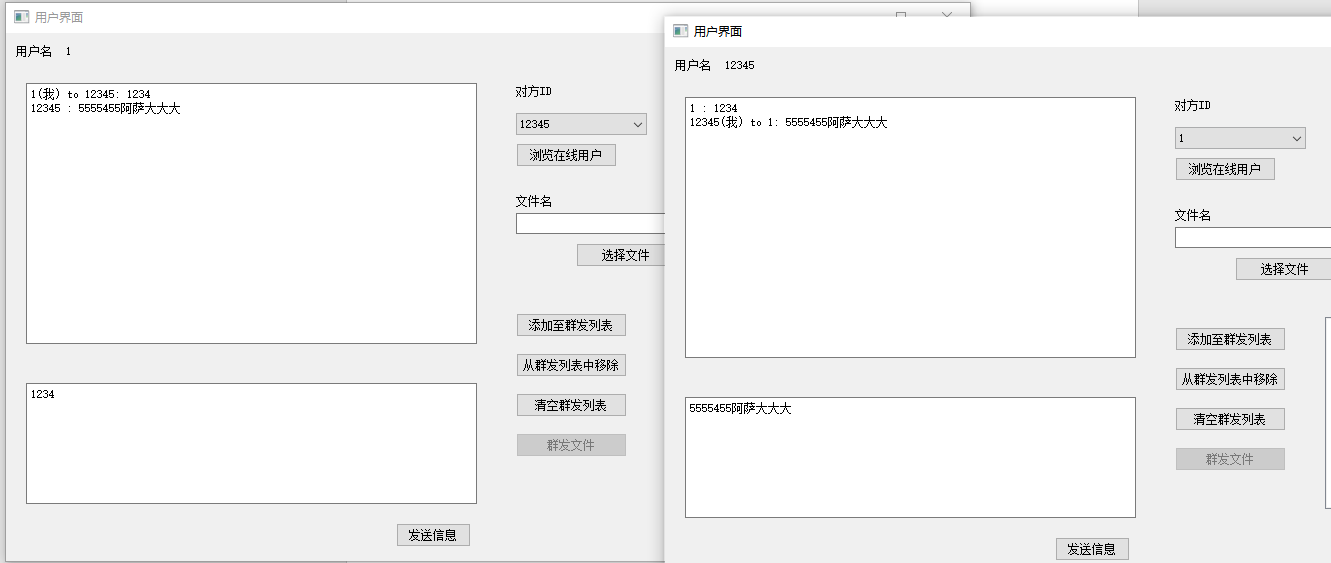
没有发送目标不能发送数据



选择发送目标

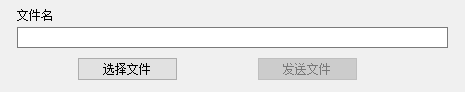


发送文本

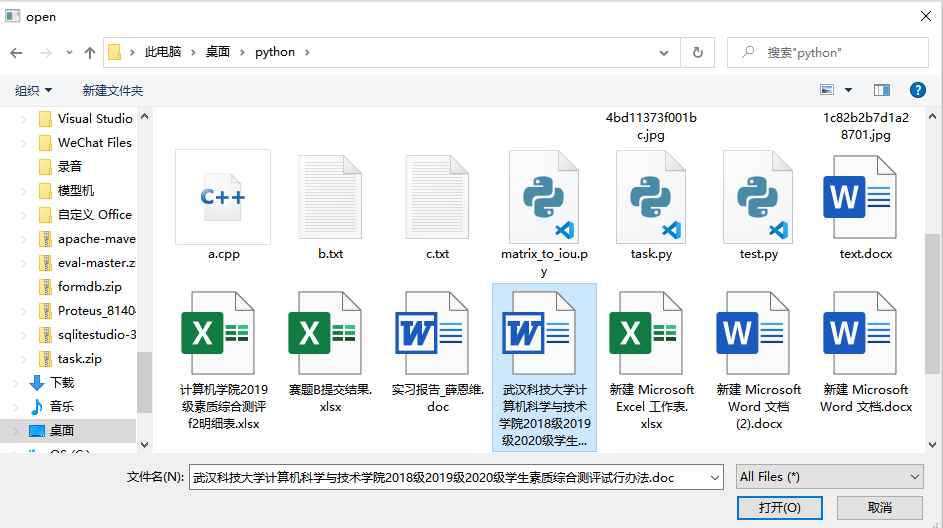


发送文件

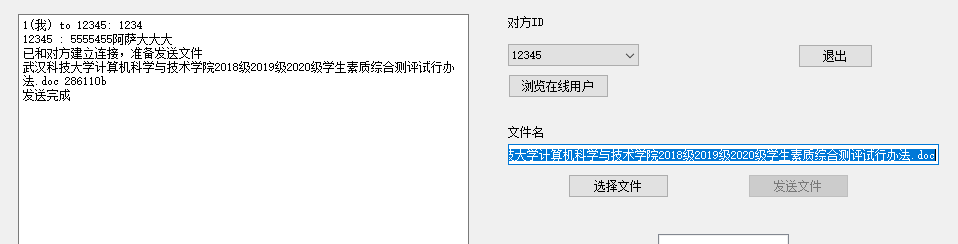
未选定文件前不能发送



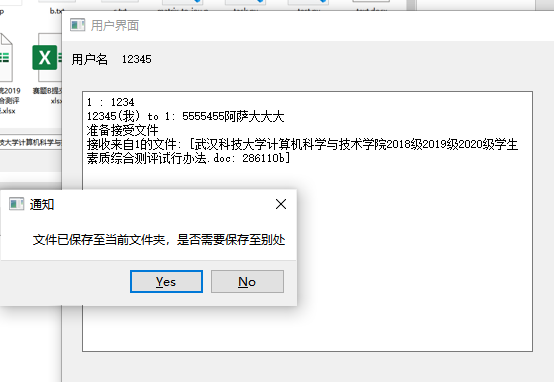
选择文件

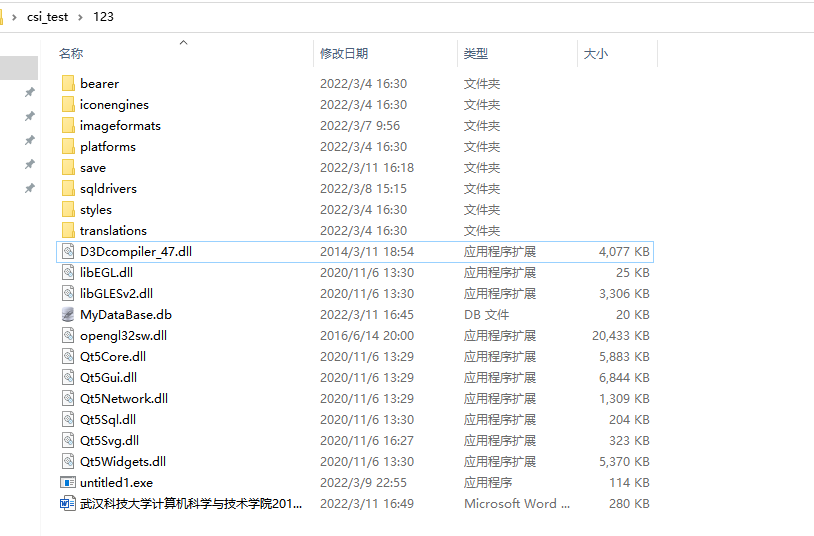


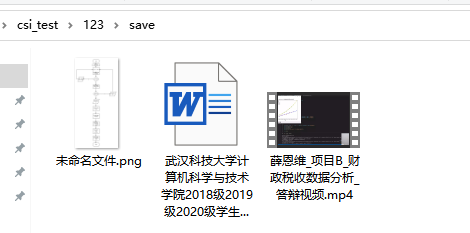
发送方



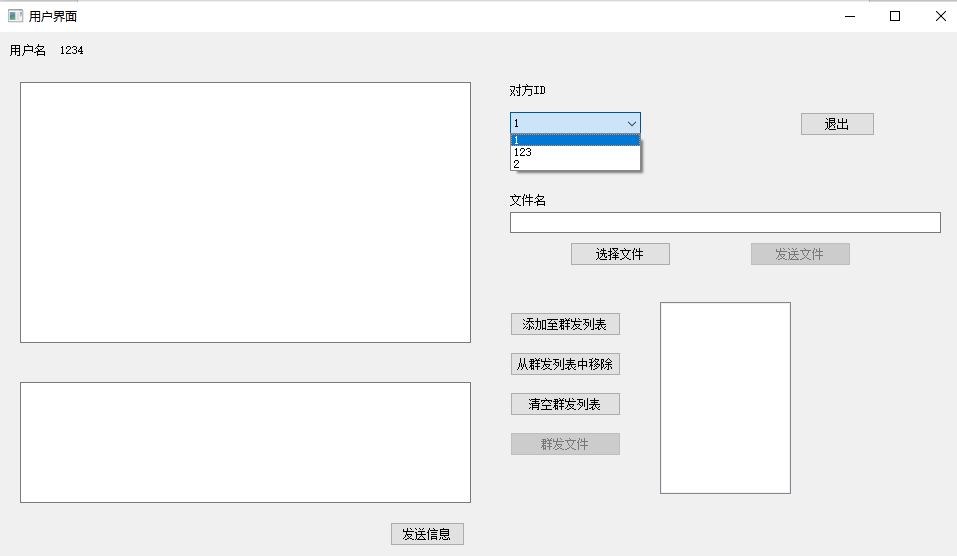
接收方



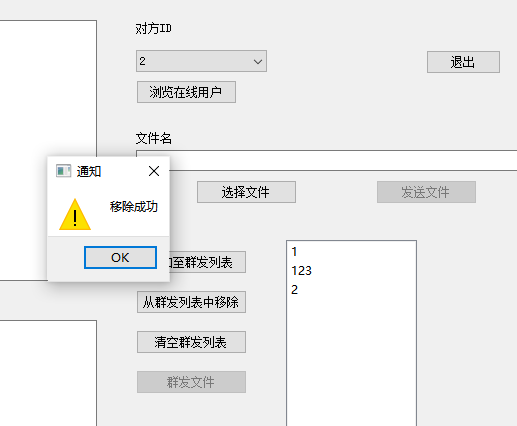


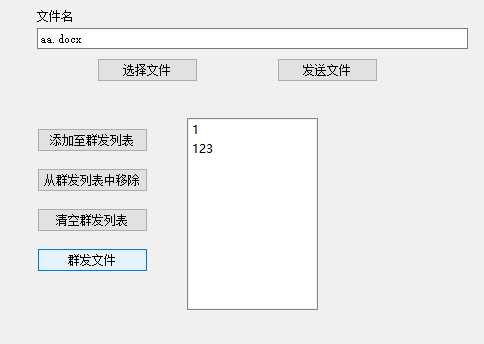


3.文件群发

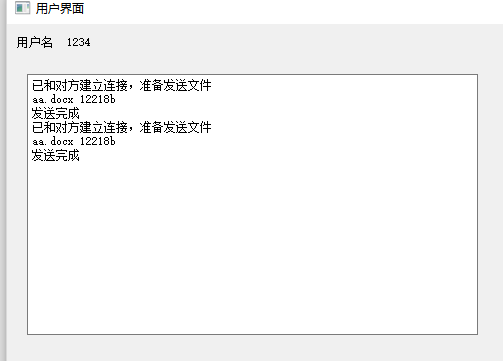


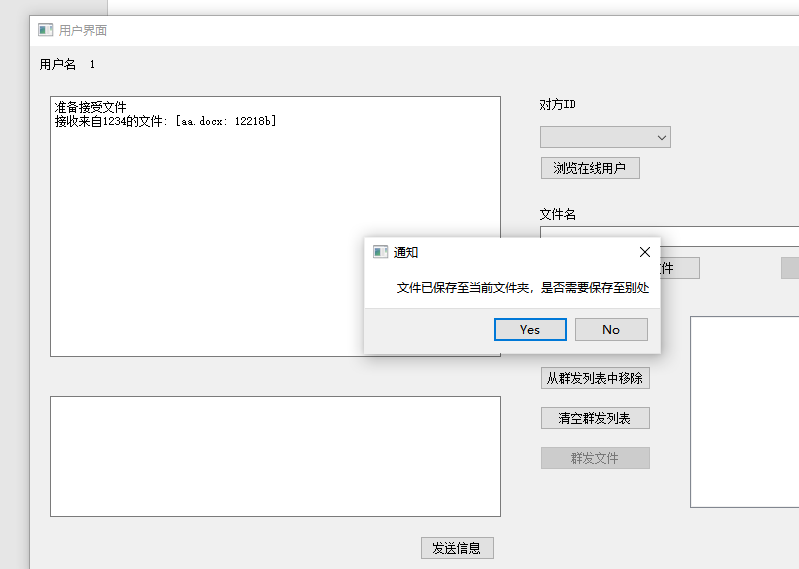
添加至群发名单

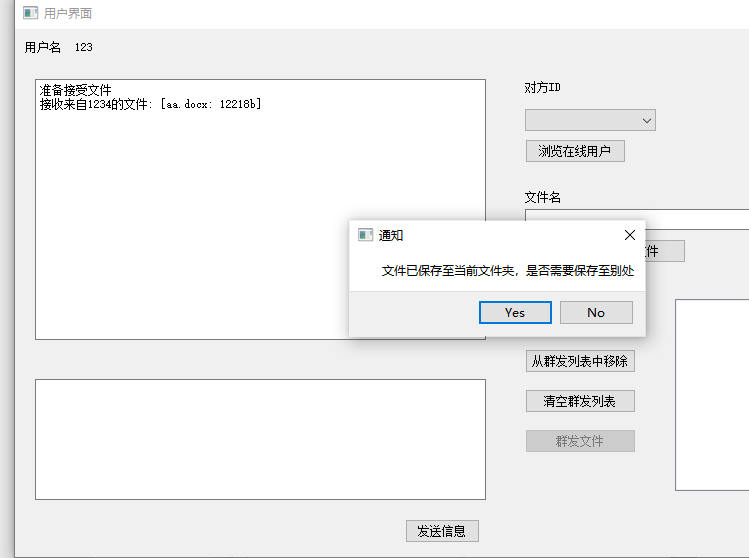




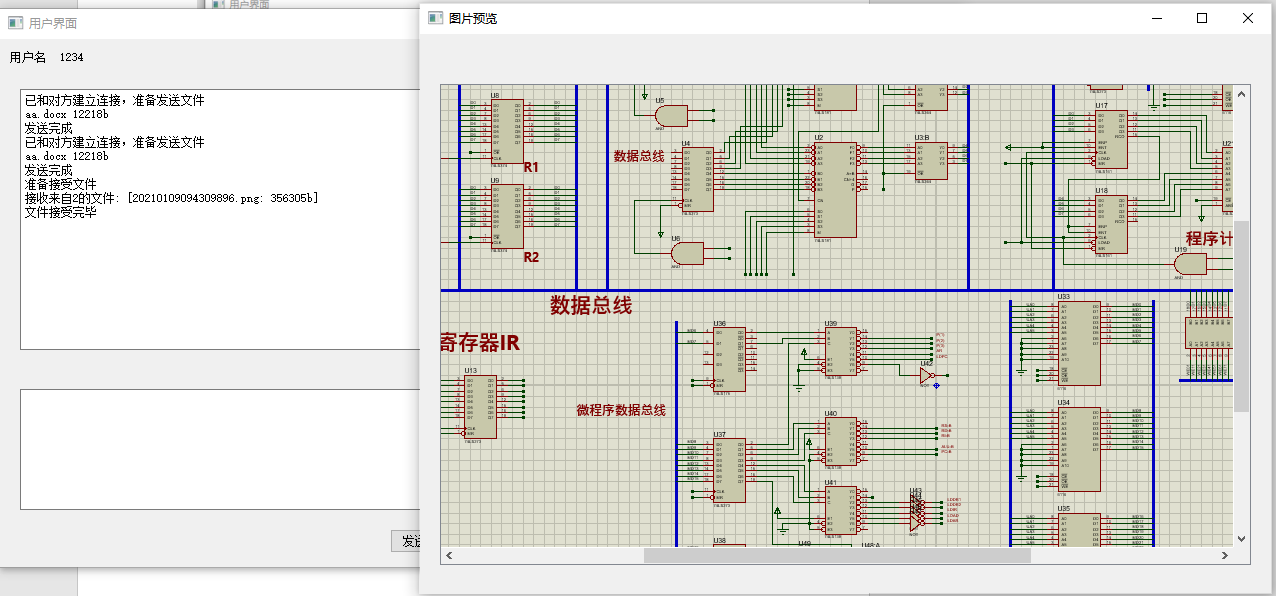
群发







4.图片预览



时间复杂度分析

本程序多次使用了数据库查询和修改，Sqlite数据库使用B-tree树数据结构存取数据，根据B-tree树的性质，查询算法的时间复杂度大概为O（logN）,具体到每个模块，如注册模块最少需要查询一次，时间复杂度为O（logN）,其他模块大多数为O（2logN）或O（3logN）,用时最长的是群发，大概需要O（（n-1）logN）。

设计和调试时存在的问题

1.TCP发包的大小和方式：发包不是一整个发过去，而是发一个个数据包，最大为64KB，发包的过程是连续不断的，一旦某个地方的逻辑出了问题就会丢包。解决方案就是一个一个排，理清逻辑

2.文件读写方式，文件读完之后就不会从头再读，当时在设计文件群发的时候没考虑到导致不能多发。在发送完之后必须得再次打开文件进行读取。

改进设想

程序的运行还是依赖设备的本地数据库，如果是跨设备部署就无法实行了。我设想搭建一个公共服务器端，用户端作为客户端，通过这个服务器来传递数据，用户信息数据库就部署在服务器端的设备上。