东软睿道

项目实训总结报告

项 目 名 称: QuickTalk

学 校 名 称:北京理工大学

项 目 讲 师: 崔浩

姓 名: 贾博凯

学 号: 1120201335

实 训 日 期: 2022年8月15日-8月25日

**1 引言**

这是一次难能可贵的基础项目实习经历，与以往的课程可以说形成了鲜明的对比，我写了大量的代码，也学会了很多代码设计思维，掌握了Linux基础知识，下面我来总结一下**。**

1.1项目概述

QuickTalk项目是一款实时通讯类的软件，可以让用户在局域网下实时传递消息，图片，文件等资源，实现mysql数据库支持，并且支持离线消息的存储，开通了群聊功能，丰富用户体验。

背景知识：Qt框架，mysql数据库，odbc操作。

参考资料：老师上课讲的demo,提供的素材，github。

意义：让我们有了更真实的项目体验，提高代码能力，转变代码思维。

1.2 需求分析

在现代高速的发展节奏中，人们已经很难在线下花大量的时间沟通，既浪费时间成本，也浪费经济成本，因此我们的QuickTalk项目提供的实时线上交流将更有利于大家提高效率。而且将逐步支持视频，语音通话的音视频传输功能，完善本项目的各模块，也能打造一个全新的社交平台。

1.3 运行环境

硬件：Window(10/11)/linux(centos 7/ubuntu 20.04)

软件：mysql(5.0以上)+qt(5.12.8)

**2 项目设计**

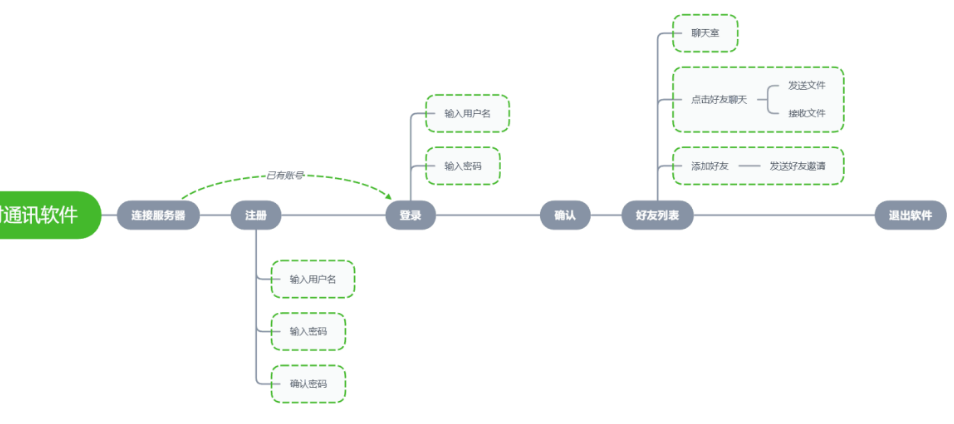
2.1 设计思路

利用tcp/udp 传输协议实现信息的传输，从客户端发送到服务端，经过服务端分析处理（访问数据库）后执行相应的操作，完成通信。各界面利用qt的UI美化，要符合基本的美感视觉。

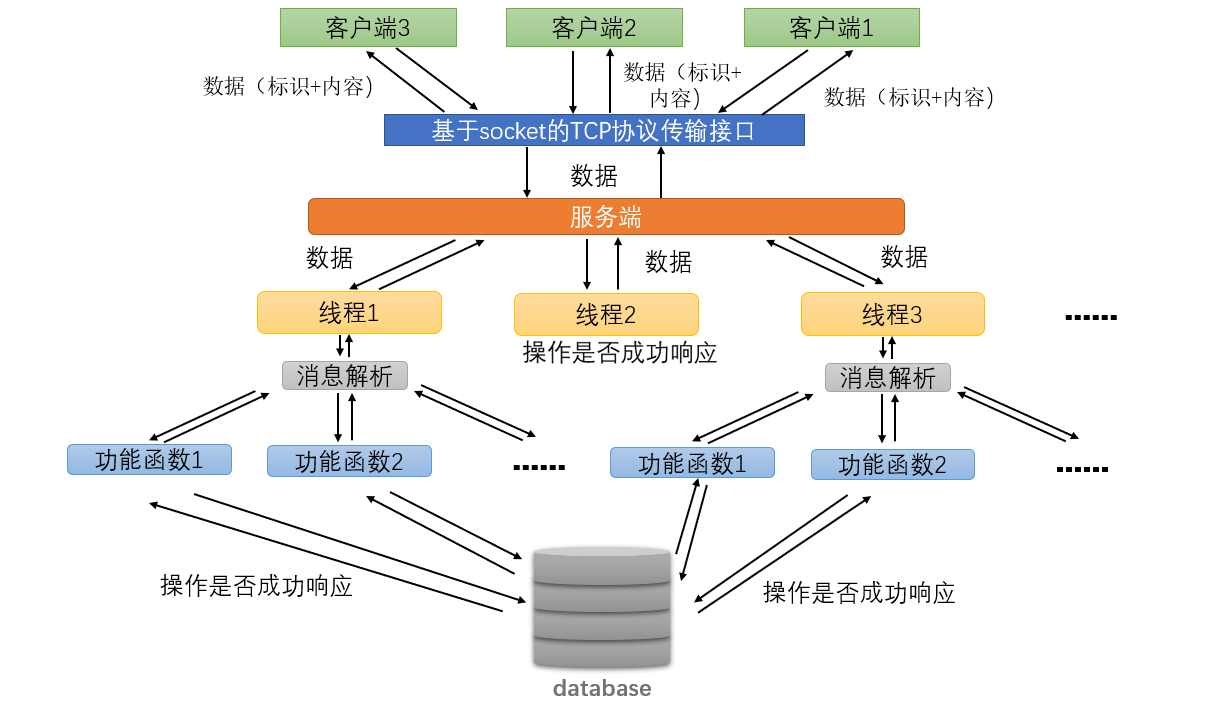
2.2 模块功能介绍

* 注册模块：基本信息填写，正则检验是否合理，传送数据到服务器，存储本地数据。
* 登录模块：利用socket传递信息检查数据信息是否匹配，在本地ini配置文件中检测用户是否存在，是否登录。
* 好友，群聊模块：服务器端建立对应的数据库存储消息。
* 聊天模块：利用前缀将信息传递到对应的客户端
* 文件传输模块：tcp传递信息后，利用udp实现点对点的文件传输。
* 更换背景，头像模块：更新头像，背景信息。
* 离线信息模块：获取离线信息。

2.3 模块结构图



2.4 程序流程图



2.5 功能设计分工

* 注册模块：

用户输入信息，写入本地ini配置文件，建立连接后利用socket传到服务端数据库存储起来；正则表达式检查输入信息的正确与否；信息正确以后，跳转到登录界面，并自动输入用户名和密码。

* 登录模块：

用户可以查询本地ini配置文件获取已经登陆过的账号密码信息，如果正确进入主界面，否则重新输入。检测该账号的state信息，如果已经登录，那么这时显示“用户已登录”。

* 好友，群聊模块：

在主界面点击添加好友后，会到服务器查询此用户信息，不存在返回“该用户不存在”，存在的话弹出提示框等对方确认。群聊自动加入，可以更新列表，一旦用户上线，要显示他的好友与群聊。

* 聊天模块：

将用户在自己的输入框的信息包装起来发给服务端，处理后发给对应的客户端，采用tcp稳定传输。文件传输也是同样的道理。

更换背景，头像模块：利用QPainter类封装实现背景和头像的更改。

* 离线信息模块：

利用访问用户state，传送数据到数据库，存储离线信息，当该用户的state更新之后，将信息传送到用户的聊天框。

**3 详细设计**

注册：服务端建立监听server，监听到客户端的连接请求时，建立与之对应的socket套接字建立通道。然后注册信息会发送给server，server处理消息后，查询数据库，将用户信息更新到数据库中。

登录：

本地查询ini配置文件，读取信息实现记录用户的功能，发送请求登录的消息到服务端，服务端解析到用户名，密码的信息，在拿到数据库中比对，如果正确，返回允许登录的信息，否则拒绝登录，同时查询用户state，如果已经在线，那么返回用户已经登陆的信息。

好友：

将要加的好友id传输到服务器，查询数据库，有信息就在对方那里返回收到好友请求的消息，询问是否同意，然后将结果在返回服务器，处理后发给原客户端。

聊天：

客户端发送聊天内容到服务器，经解析后，存储在数据库中，方便保存聊天记录，如果目标客户端state在线，再将该消息传到另一个客户端，否则暂缓传送，等待其状态改变，继续传送。

群聊：

客户端发送请求群聊信息，服务端查询数据库，可以直接创建，数据库会根据群聊名称创建一个群聊table,也可以加入，更新该群聊的成员信息。在其他人加入进来时，可以建立udp组播通信,实现更快的交流。

文件传输：

利用udp传输，查阅资料解决udp的传输丢包问题。

数据库:

利用mysql存储数据，未来可支持解决多线程并发，实现分布式存储功能。与qt程序交互也比较方便。主要实现用户信息存储，好友信息，消息列表，群聊信息的存储。

**4 实现方案**

注册：

用户输入信息后，首先询问数据库是否已有该用户id注册过，没有的话继续执行，然后检测邮箱，手机号格式是否正确，在确认密码是否一致，长度是否符合要求。

****

注册界面

代码：

connect(clientSocket,SIGNAL(dealMessage(QString)),this,SLOT(displayMsg(QString)) );

connect(clientSocket,SIGNAL(enroll\_success()),this,SLOT(success()));

connect(clientSocket,SIGNAL(enroll\_failure()),this,SLOT(failure()));

connect(clientSocket,SIGNAL(enroll\_name\_exists()),this,SLOT(name\_exists()));

//邮箱格式检测

QRegExp email\_exp("^\\w+([-+.]\\w+)\*@\\w+([-.]\\w+)\*\\.\\w+([-.]\\w+)\*$");//正则表达式

QRegExpValidator \*email\_val = new QRegExpValidator(email\_exp,ui->email);

ui->email->setValidator(email\_val);

//手机格式检测

QRegExp phone\_exp("^(13[0-9]|14[01456879]|15[0-35-9]|16[2567]|17[0-8]|18[0-9]|19[0-35-9])\\d{8}$");//正则表达式

QRegExpValidator \*phone\_val = new QRegExpValidator(phone\_exp,ui->phonenumber);

ui->phonenumber->setValidator(phone\_val);

//检测两次密码一致与否

else if(pwd!=check\_pwd){

ui->password->clear();

ui->checkpassword->clear();

ui->pwd\_check->setText("密码不一致!");

ui->pwd\_check->*setVisible*(true);

}

else if(pwd.length()<6){

ui->pwd\_check->setText("密码长度不足");

ui->pwd\_check->*setVisible*(true);

}

else {

QString info = "R#phone:" + phone +"/email:" + email + "/username:"+ name + "/password:" +pwd;

qDebug()<<info;

clientSocket->write(info.toUtf8());

}

登录：

注册之后，客户端会把用户信息存储在本地的ini文件下，方便实现记住密码的功能，检测账号密码正确后，还要检查用户的state，如果已经登录，显示用户已登陆，否则进入到主界面。

****

登录界面

void Login::**LoadConfig**(){

qDebug()<<"in loadconfig";

ReadInit("name",*namelist*);

ReadInit("pwd",*pwdlist*);

ReadInit("rem",*remlist*);

qDebug()<<namelist<<pwdlist<<remlist;

ui->name->addItems(namelist);

if(namelist.size()>0){

name = namelist.at(0);

}

if(pwdlist.size()>0){

pwd = pwdlist.at(0);

}

if(rem=="true"){

ui->checkBox->setChecked(true);

ui->name->setCurrentText(name);

ui->password->setText(pwd);

}

}

void Login::**SetLogin**(int index)//判断自动登录

{

if(pwdlist.size()>=index||index>0){

pwd = pwdlist.at(index-1);

rem = remlist.at(index-1);

if(rem== "true")

{

ui->checkBox->setChecked(true);//勾选记住用户名密码

ui->password->setText(pwd);

}else{

ui->checkBox->setChecked(false);

ui->password->setText("");

}

}

}

void Login::**WriteInit**(QString key, QStringList value){

QString path = "./login.ini";

//创建配置文件操作对象

QSettings \*config = new QSettings(path, QSettings::IniFormat);

QVariant variant;

variant.setValue(value);

//将信息写入配置文件

qDebug()<<value<<"!!!"<<endl;

config->beginGroup("config");

config->setValue(key, variant);

config->endGroup();

delete config;

}

好友：

搜索好友名字，如果存在，向对方发出添加好友的请求，然后对方会弹窗请求，接受才会添加成好友，然后更新自己的好友列表。

****

添加好友

void MainWindow::**on\_newfriend\_clicked**()

{

QString friend\_name = ui->search->text();

QString str = "F#My\_id:"+my\_name+"/Friend\_id:"+friend\_name;

clientSocket->sendMsg(str);

}

void MainWindow::**add\_friend**(QString info){

qDebug()<<info<<"in add friend";

if(info.contains("F#0")){

QMessageBox::information(this,"提示","已经是您的好友\n");

}

else if(info.contains("F#1")){

QMessageBox::information(this,"提示","添加好友成功\n");

clientSocket->sendMsg("G#my\_name:"+my\_name);

}

else if(info.contains("F#2")){

QMessageBox::information(this,"提示","加好友失败\n");

}

else if(info.contains("F#3")){

QMessageBox::information(this,"提示","您查找的用户不存在");

}else if(info.contains("F#4")){

qDebug()<<"in add friend:F#4"<<info;

QString us=info.mid(info.indexOf('/')+1);

QString str;

if(QMessageBox::Yes==QMessageBox::question(this,"加好友通知","用户"+us+"想加你为好友?")){

str = "FYMy\_id:"+my\_name+"/Friend\_id:"+us;

}else{

str = "FRMy\_id:"+my\_name+"/Friend\_id:"+us;

}

clientSocket->sendMsg(str);

}else if(info.contains("F#5")){

QMessageBox::information(this,"提示","加好友失败,对方拒绝了您的请求!!!");

}

fresh\_friend\_list(true,on\_friend);

fresh\_friend\_list(false,off\_friend);

}

void MainWindow::**friend\_list**(QString info){

on\_friend.clear();

off\_friend.clear();

grouplist.clear();

QStringList str = info.split("%");

qDebug()<<str;

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

if(str[i].contains("online")){

QString on = str[i].mid(13,-1);

on\_friend = on.split("@");

on\_friend.removeAll("");

}

if(str[i].contains("offline")){

QString off = str[i].mid(14,-1);

off\_friend = off.split("@");

off\_friend.removeAll("");

}

if(str[i].contains("G#1")){

QString on = str[i].mid(4,-1);

grouplist=on.split("@");

grouplist.removeAll("");

qDebug()<<"群聊信息列表";

qDebug()<<grouplist;

fresh\_group\_list(grouplist);

}

}

qDebug()<<"in friend\_list";

qDebug()<<on\_friend;

qDebug()<<off\_friend;

fresh\_friend\_list(true,on\_friend);

fresh\_friend\_list(false,off\_friend);

}

群聊：

和好友模块类似，但是在没有群聊的时候，会直接新建一个群聊，添加的时候可以直接加入，数据库直接建表，插入数据即可。

****

代码：

void MainWindow::**group\_list**(QString info)

{

QString on = info.mid(4,-1);

grouplist=on.split("@");

grouplist.removeAll("");

qDebug()<<"群聊信息列表";

qDebug()<<grouplist;

fresh\_group\_list(grouplist);

}

void MainWindow::**fresh\_group\_list**(QStringList grouplist)

{

qDebug()<<"in fresh group";

qDebug()<<grouplist ;

QList<QToolButton\*> btns = ui->group->findChildren<QToolButton\*>();//!!!!

foreach (QToolButton\* btn, btns){

delete btn;

}

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

btn=new QToolButton(this);

//加载图标

btn->setIcon(QPixmap(QString(":/image/chatgroup.png")));

//设置图片大小

btn->setIconSize(QPixmap(QString(":/image/chatgroup.png")).size()/8);

//设置网名

btn->setText(QString("%1").arg(grouplist[i]));

//设置为透明

btn->setAutoRaise(true);

//设置显示格式

btn->setToolButtonStyle(Qt::ToolButtonTextBesideIcon);

btn->setStyleSheet("font: 25 20pt;color: rgb(0, 0, 0);border-bottom : 5px groove #6699cc;");

btn->setGeometry(0, 0, 300, 100);

//on.push\_back(btn);

ui->grouplayout->addWidget(btn);

IsShow\_group.push\_back(false);

connect(btn,&QToolButton::clicked,[=](){

if(IsShow\_group[i])

{

QMessageBox::warning(this,"警告","该聊天框已被打开!");

return;

}

IsShow\_group[i]=true;

TalkWindow \*talkwindow=new TalkWindow(nullptr,my\_name,grouplist[i]);

talkwindow->setWindowIcon(btn->icon());

talkwindow->setWindowTitle(btn->text());

talkwindow->show();

//关闭时将对应的IsShow\_on变为false;

connect(talkwindow,&TalkWindow::closeWidget,this,[=](){

IsShow\_group[i]=false;

});

});

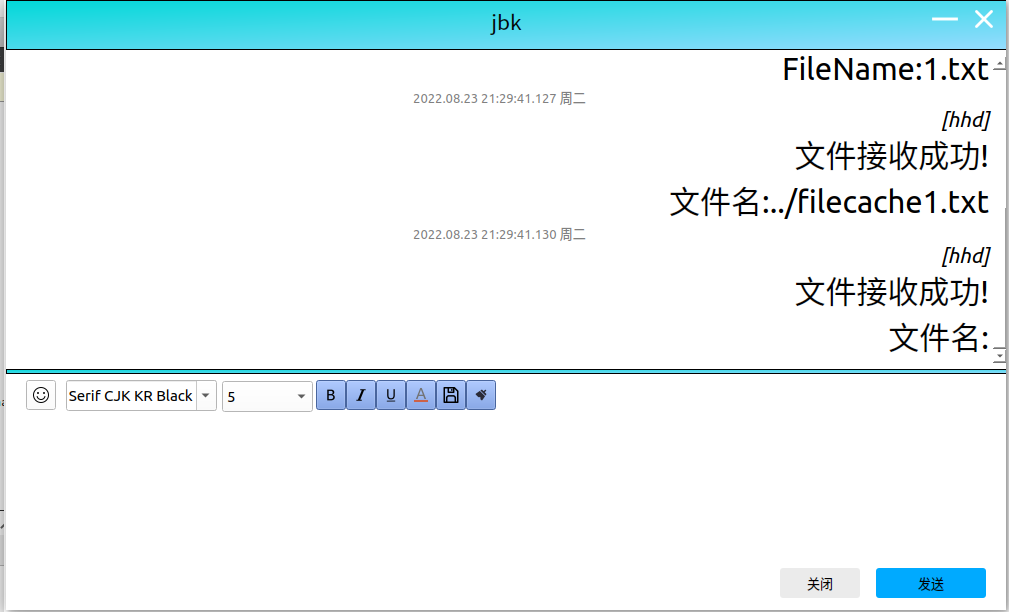
}

}

同时也会实现及时刷新的功能。

文件传输：

利用udp把文件打包发送给其他客户端，有丢包问题，尝试设置文件block大小和接受大小和速率来解决。

****

void UdpReciver::**readDatagrams**()

{

QFile final\_file;

qDebug()<<"刚进来";

QString File\_name;

qDebug()<<"in readDatagrams";

QHostAddress client\_address; //client ip addr

m\_data.clear();

int size=0;

while(m\_udpSocket->hasPendingDatagrams())

{

m\_data.resize(m\_udpSocket->pendingDatagramSize());

m\_udpSocket->readDatagram(m\_data.data(), m\_data.size(), &client\_address);

if(m\_data.contains("&&"))

{

int idx1 = m\_data.indexOf("&&");

int idx2 = m\_data.indexOf("##");

File\_name = m\_data.mid(idx1+2,idx2-idx1-2);

qDebug()<<"文件名："<<File\_name;

m\_data.clear();

//指定路径输出文件

final\_file.setFileName("../filecache"+File\_name);

qDebug()<<"真实文件名："<<final\_file.*fileName*();

continue;

}

QString strclient\_address = client\_address.toString();

size = m\_data.size();

qDebug() << "receive UDP's size:"<< m\_data.size();

}

if(final\_file.exists())

{

qDebug()<<"文件存在";

}

else

{

qDebug()<<"文件不存在";

}

QString str(m\_data);

if(!final\_file.*open*(QIODevice::ReadWrite))

{

qDebug()<<"打开失败";

}

else

{

qDebug()<<"打开成功";

}

int flag = final\_file.write(str.toUtf8(),m\_data.size());

if(flag == -1)

{

qDebug()<<"文件写入失败";

}

emit deliverInfo(final\_file.*fileName*());

final\_file.*close*();

//qDebug()<<"这是m\_data:"<<m\_data;

m\_data.clear();

}

void UdpSender::**send\_file**(QByteArray msg)

{

qDebug()<<"in send file";

m\_Socket->writeDatagram(msg, QHostAddress(m\_address), m\_port);

}

void UdpSender::**recMsg**(QByteArray msgInfo)

{

send\_file(msgInfo);

}

**5 系统测试**

首先测试服务端与数据库的连接，通过判断是否可以创建table，不能的话，检测mysql是否开启，驱动是否安装，用户密码是否匹配。

在测试服务端是否可以与客户端建立通信，如果可以，发送信息，不可以时，检查IP和port的设置，重新连接。

检查注册模块，去服务端监控信息是否发送成功，然后在看数据库是否有新增数据，再看客户端是否接受到反馈信息，如果成功，才能跳转到登录界面。

登陆模块也是先检查服务端的信息，正确后检查数据库，验证成功后看客户端是否受到对应消息，决定是否跳转主界面。

聊天模块检查双方发送消息与服务端的交互时候完整，在测试群聊，也是检查消息的完整性后，去数据库里查询是否有数据即可，离线消息可以查看用户的state存到离线消息表里，在该用户上线时发给他。

文件传输模块检测对应的客户端是否受到信息，文件信息是否完整，以及信息是否存储到数据库中。

**6 心得体会**

这门课程让我们有了一种在企业中进行项目开发的模拟经验，我们充分体会到了团队合作的效果，学会了很多新的知识。尤其是可以利用学过的c++的知识去实现一个可交互的应用，这直接让我们看到了计算机的魅力，让我们看到了自己学到的知识应用到实际是多么的震撼。

而我在这个过程中，充分学到了前后端交互，GitHub管理代码，项目逻辑构思，项目整体架构的知识，也大大提高了我的代码能力，让我能有兴趣多做一些这样的项目开发，充实自己的经历，也想继续带领我们的团队去克服更多的困难，完善我们的项目。同时，这个过程中也充斥着各种焦虑，甚至崩溃。有队友完不成任务，那就要帮他把他的模块完成，搭不好环境，要远程亲自帮他们搭，还要兼顾自己的任务进度，属实是手忙脚乱了。

不过我们最后终于是完成了这个还算可以的项目，达成了我们的目标！感谢老师，班主任，也感谢我的队友们，这个项目是我们所有人的心血，再接再厉，在创辉煌！