飞鸽传书——Qt版的QQ

[飞鸽传书——Qt版的QQ 1](#_Toc62450193)

[1 前言 2](#_Toc62450194)

[2 PART01 ---- 简单的服务器和客户端的登陆界面 2](#_Toc62450195)

[2.1 Linux下（纯C代码） 2](#_Toc62450196)

[2.1.1 任务目标：实现最简单的服务器功能 2](#_Toc62450197)

[2.1.2 任务实现 2](#_Toc62450198)

[2.2 Windows下（Qt代码） 11](#_Toc62450199)

[2.2.1 任务目标：制作登录窗口 11](#_Toc62450200)

[2.2.2 任务实现： 11](#_Toc62450201)

[2.2.2.1 创建Login类 11](#_Toc62450202)

[2.2.2.2 类的前置声明与预编译指令 11](#_Toc62450203)

[2.2.2.3 声明并初始化需要的窗口插件 12](#_Toc62450204)

[2.2.2.4 使用栅格布局管理器布局 14](#_Toc62450205)

[2.2.2.5 设置窗口信息 15](#_Toc62450206)

[2.2.2.6 设置按钮相关的信号与槽 17](#_Toc62450207)

[2.2.2.7 编辑main.cpp，实现从登陆界面到主界面的跳转 18](#_Toc62450208)

[2.3 实现效果 19](#_Toc62450209)

[2.3.1 Server 19](#_Toc62450210)

[2.3.2 Client 20](#_Toc62450211)

[3 PART02 --- 搭建聊天对话框界面 20](#_Toc62450212)

[3.1 内容更新 20](#_Toc62450213)

[3.2 任务目标 20](#_Toc62450214)

[3.3 任务实现 21](#_Toc62450215)

[3.3.1 添加网络通讯模块(为Part3做准备) 21](#_Toc62450216)

[3.3.2 初始化窗口插件 21](#_Toc62450217)

[3.3.3 设置窗口属性 25](#_Toc62450218)

[3.3.3.1 添加图标资源 25](#_Toc62450219)

[3.3.3.2 字体大小组合框 25](#_Toc62450220)

[3.3.3.3 各种ToolButton属性修改 26](#_Toc62450221)

[3.3.3.4 设置表格空间QTableWidget的属性 28](#_Toc62450222)

[3.3.3.5 其他控件属性 30](#_Toc62450223)

[3.3.4 布局管理 30](#_Toc62450224)

[3.4 实现效果 33](#_Toc62450225)

[4 PART03 --- 搭建聊天对话框界面 33](#_Toc62450226)

[4.1 内容更新 33](#_Toc62450227)

[4.2 任务目标 34](#_Toc62450228)

[4.3 任务实现 35](#_Toc62450229)

[4.3.1 修改登陆窗口 35](#_Toc62450230)

[4.3.1.1 删除注册账号、找回密码两个Qlabel 35](#_Toc62450231)

[4.3.1.2 修改相关字段如下 35](#_Toc62450232)

[4.3.2 添加注册对话框 36](#_Toc62450233)

[4.3.2.1 前置和预编译 36](#_Toc62450234)

[4.3.2.2 声明定义 37](#_Toc62450235)

[4.3.2.3 布局 38](#_Toc62450236)

[4.3.2.4 信号与槽 38](#_Toc62450237)

[4.3.3 修改main.cpp 39](#_Toc62450238)

[4.4 实现效果 41](#_Toc62450239)

[4.5 服务器代码 41](#_Toc62450240)

# 前言

**平台：Windows、Linux**

**编译器：Qt Create 4.11.0**

# PART01 ---- 简单的服务器和客户端的登陆界面

+ 新增内容

  + Linux下的纯C客户端(仅实现了最简单的应用)

  + 登陆窗口（未绑定数据库）

## Linux下（纯C代码）

### 任务目标：实现最简单的服务器功能

### 任务实现

**文件：**QQ-version-of-Qt\PART01\Code\_in\_Ubuntu\server.c

#include <stdio.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <stdlib.h>

#include <string.h> *//memset*

#include <unistd.h>

#include <netdb.h>

#include <sys/types.h>

#define INET\_ADDRSTRLEN 16

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

{

*/\**

*\**

*\*argc ---- 指示程序启动时命令行参数的个数*

*\**

*\*argv ---- 则包含具体的参数字符串*

*\*/*

*//预备工作：检查参数是否正确*

    if (argc !=2)

    {

        printf("请输入正确的参数\n如：./server 5005\n\n");

        return -1;*//非成功结束*

    }

*//第一步：创建插座*

*/\**

*\*插座特点*

*1.系统不自动分配端口*

*2.初始化插座是主动的，需要手动调节成被动（监听）模式*

*\*功能：创建一个用于网络通信的I/O描述符（插座）*

*\*int socket(int family, int type, int protocol)*

*\*int socket(协议族    ,插座类型,协议类别)*

*\*/*

**int** serverfd;   *//服务器的插座*

    if((serverfd = socket(AF\_INET,SOCK\_STREAM,0)) == -1)

    {

        perror("socket");

        return -1;

    }               *//插座创建成功*

*//第二步 把服务端用于通信的地址和端口绑定到socket上*

*/\**

*\**

*\*int bind(int sockfd,const struct sockaddr \*myaddr,socklen\_t addrlen)*

*\*int bind(socket插座,指向特定于协议的地址结构指针,该地址结构的长度)*

*\*功能 将本地协议地址与服务器插座文件符(serverfd)绑定*

*\*sockaddr\_in ：在IPv4因特网域(AF\_INET)中,套接字地址结构用sockaddr\_in命名*

*\*/*

**struct** sockaddr\_in servaddr;    *//定义服务端地址信息的数据结构*

    memset(&servaddr,0,sizeof(servaddr));*//初始化servaddr*

    servaddr.sin\_family = AF\_INET;

*//协议族，在socket编程中只能是AF\_INET*

    servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

*//任意ip地址*

*//inet\_addr("192.168.190.134"); // 指定ip地址*

    servaddr.sin\_port = htons(atoi(argv[1]));*//指定通信端口*

    if(bind(serverfd,(**struct** sockaddr \*)&servaddr,sizeof(servaddr)) != 0)

    {

        perror("bind");

        close(serverfd);

        return -1;

    }                               *//开始绑定*

*//第3步：把socket设置为监听模式*

*/\**

*\*int listen(int sockfd, int backlog)*

*\*int listen(socket监听插座，连接队列的长度)*

*\*功能 将插座由主动修改为被动*

*\*    使os为该插座设置一个连接队列，用来记录所有连接到该插座的连接*

*\*/*

    if(listen(serverfd,10) != 0)

    {

        perror("listen");

        close(serverfd);

        return -1;

    }

*//第4步：接受客户端的连接*

*/\**

*\*int accept(int sockfd,struct sockaddr \*cliaddr,socklen\_t \*addrlen);*

*\*int accept(socket监听插座,用于存放客户端插座地址结构,插座地址结构体长度);*

*\*功能 从已连接队列中取出一个已经建立的连接，如果没有任何连接可用，则进入睡眠等待*

*\*accept函数返回的是一个已连接套接字，这个套接字代表当前这个连接*

*\*/*

**int** clientfd;       *//声明客户端的插座*

**int** socklen = sizeof(**struct** sockaddr\_in);  *//获得插座地址结构struct sockaddr\_in的大小*

**struct** sockaddr\_in clientaddr;      *//声明客户端地址数据结构*

*//接受连接*

    clientfd = accept(serverfd,(**struct** sockaddr \*)&clientaddr,(socklen\_t\*)&socklen);

    printf("客户端（%s）已连接。\n",inet\_ntoa(clientaddr.sin\_addr));

*//第5步：与客户端通信，接收客户端发过来的报文后，回复ok。*

*/\**

*\*ssize\_t recv(int sockfd, void \*buf,size\_t nbytes, int flags);*

*\*ssize\_t recv(客户端插座, 指向接收网络数据的缓冲区,*

*接收缓冲区的大小(以字节为单位), 套接字标志(常为0));*

*\*返回值：成功接收到的数据字节数*

*\*ssize\_t send(int sockfd, const void\* buf, size\_t nbytes, int flags);*

*\*ssize\_t send(服务端插座, 待发送数据缓存区的地址,*

*发送缓存区大小(以字节为单位), 套接字标志(常为0));*

*\*功能：用于发送数据 注意：不能用TCP协议发送0长度的数据包*

*\*返回值：成功发送的字节数*

*\*/*

**char** buffer[1024];      *//声明缓冲区*

    while (1)

    {

**int** iret;

        memset(buffer,0,sizeof(buffer));*//初始化缓冲区*

*//接收客户端的请求报文*

        if((iret = recv(clientfd,buffer,sizeof(buffer),0)) <= 0)

        {

            printf("iret=%d\n",iret);

            break;

        }

        printf("接受：%s\n",buffer);*//显示接受到的消息*

*//向客户端发送响应结果*

        if ( (iret=send(clientfd,buffer,strlen(buffer),0))<=0)

        {

            perror("send");

            break;

        }

        printf("发送：%s\n",buffer);*//显示发送出去的消息*

    }

*// 第6步：关闭socket，释放资源*

    close(serverfd);

    close(clientfd);

}

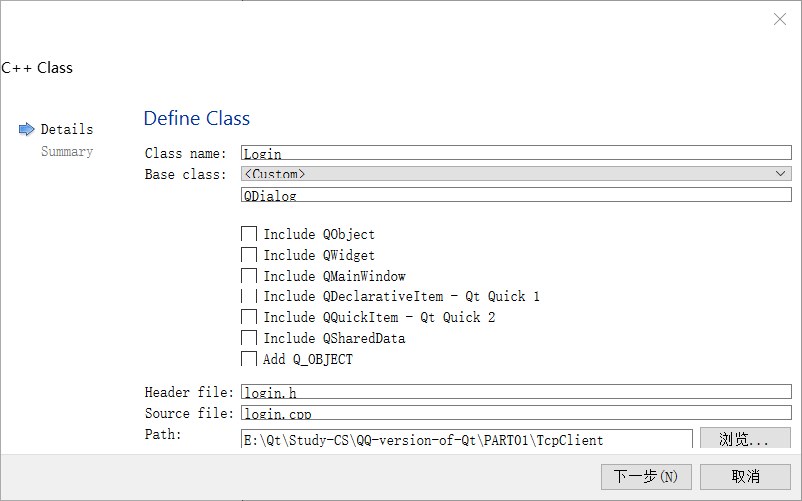
## Windows下（Qt代码）

### 任务目标：制作登录窗口

### 任务实现：

#### 创建Login类

右击项目名 🡪 add new 🡪 C++ 🡪 C++ Class 🡪如下图填写选择



#### 类的前置声明与预编译指令

文件：QQ-version-of-Qt\PART01\TcpClient\login.h

*//需要包含的类*

#include <QWidget>      *//窗口类*

#include <QDialog>      *//对话框类*

#include <QLabel>       *//标签类*

#include <QPushButton>  *//按钮类*

#include <QLineEdit>    *//输入行类*

#include <QMessageBox>  *//消息类*

#include <QGridLayout>  *//栅格布局管理器类*

*//类的前置声明*

**class** QLabel;

**class** QLineEdit;

**class** QPushButton;

#### 声明并初始化需要的窗口插件

文件：QQ-version-of-Qt\PART01\TcpClient\login.h

类的声明Class Login中：

**private:**

    QLabel \*usrLabel;*//文本：账号*

    QLabel \*pwdLabel;*//文本：密码*

    QLabel \*logoLabel;*//文本：logo*

    QLabel \*logup;*//文本：注册账号（用以日后连接数据库使用）*

    QLabel \*pswfind;*//文本：找回密码（用以日后连接数据库使用）*

    QLineEdit \*usrLineEdit;*//输入框：账号*

    QLineEdit \*pwdLineEdit;*//输入框：密码*

    QPushButton \*loginBtn;*//按钮：登陆*

    QPushButton \*exitBtn;*//按钮：退出*

文件：QQ-version-of-Qt\PART01\TcpClient\login.cpp

Login类中：

*//初始化窗口插件*

    logoLabel = new QLabel(this);

    usrLabel = new QLabel(this);

    pwdLabel = new QLabel(this);

    logup = new QLabel(this);

    pswfind = new QLabel(this);

    usrLineEdit = new QLineEdit(this);

    pwdLineEdit = new QLineEdit(this);

    loginBtn = new QPushButton(this);

    exitBtn = new QPushButton(this);

*//设置文本*

    logoLabel->setText("飞鸽传书");

    usrLabel->setText("用户名:");

    pwdLabel->setText("密码:");

    logup->setText("注册账户");

    pswfind->setText("找回密码");

    loginBtn->setText("登陆");

    exitBtn->setText("退出");

*//设置logo文本对齐方式（居中）*

    logoLabel->setAlignment(Qt::AlignHCenter);

#### 使用栅格布局管理器布局

文件：QQ-version-of-Qt\PART01\TcpClient\login.cpp

Login类中：

*//栅格布局管理器*

*//定义栅格布局管理器*

    QGridLayout \*loginLayout = new QGridLayout;

*//添加插件*

    loginLayout->addWidget(logoLabel,0,1);

    loginLayout->addWidget(usrLabel,2,0);

    loginLayout->addWidget(usrLineEdit,2,1);

    loginLayout->addWidget(logup,2,2);

    loginLayout->addWidget(pwdLabel,4,0);

    loginLayout->addWidget(pwdLineEdit,4,1);

    loginLayout->addWidget(pswfind,4,2);

    loginLayout->addWidget(loginBtn,6,0);

    loginLayout->addWidget(exitBtn,6,2);

*//设置管理器间距*

    loginLayout->setHorizontalSpacing(10);*//两列之间的间距*

    loginLayout->setVerticalSpacing(10);*//两行之间的间距*

    loginLayout->setContentsMargins(10,10,10,10);*//设置左侧、顶部、右侧和底部边距*

*//设置窗口大小*

    setMinimumSize(400,150);*//窗口最大大小*

    setMaximumSize(400,150);*//窗口最小大小*

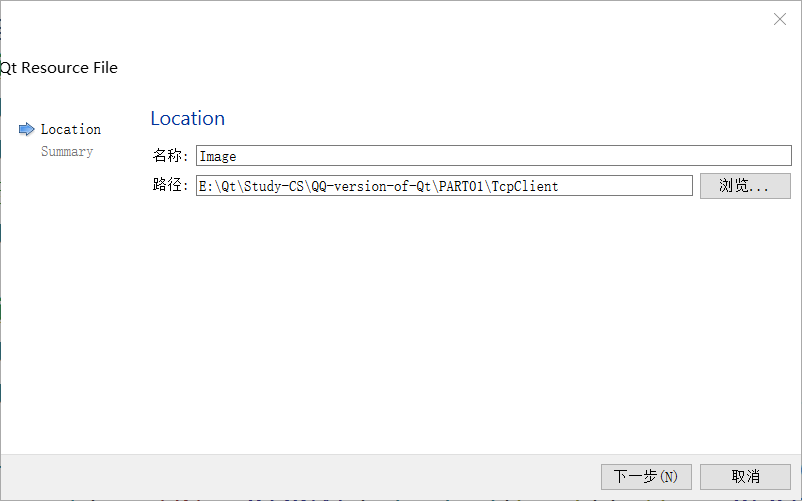
*//安装布局管理器*

    setLayout(loginLayout);

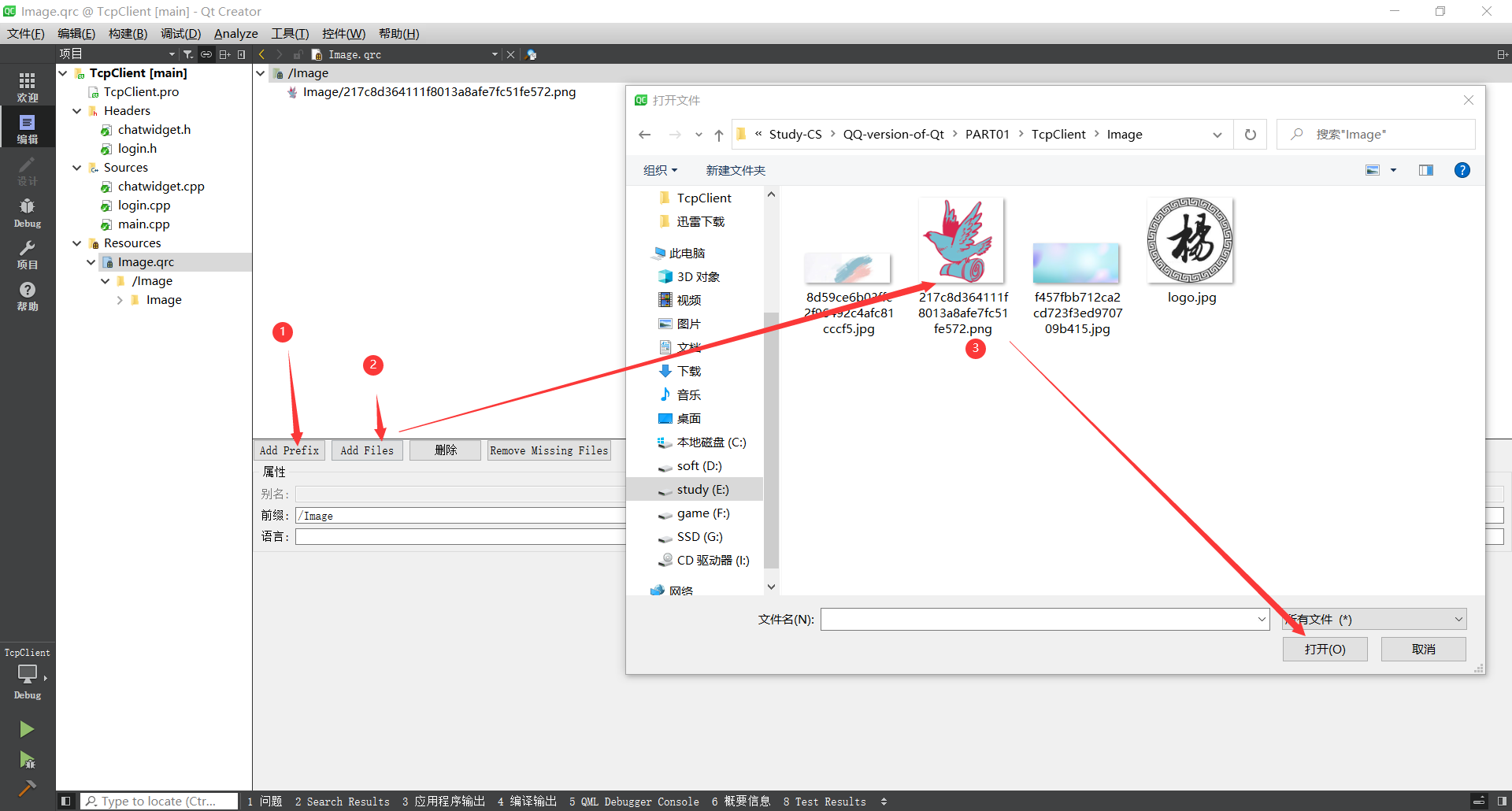
#### 设置窗口信息

文件：QQ-version-of-Qt\PART01\TcpClient\login.cpp

右击项目名 🡪 add new 🡪 Qt 🡪 Qt Resource File



右击资源文件Image.qrc 🡪Open in Editor



Login类中：

*//设置窗口名和窗口图标*

    setWindowTitle("飞鸽传书    登陆");

    setWindowIcon(QIcon(":/Image/Image/217c8d364111f8013a8afe7fc51fe572.png"));

#### 设置按钮相关的信号与槽

文件：QQ-version-of-Qt\PART01\TcpClient\login.h

类的声明Class Login中：

private slots:

**void** login();*//登陆进聊天室界面*

文件：QQ-version-of-Qt\PART01\TcpClient\login.cpp

Login类中：

*//设置信号与槽*

    connect(loginBtn,SIGNAL(clicked(**bool**)),this,SLOT(login()));

    connect(exitBtn,SIGNAL(clicked(**bool**)),this,SLOT(close()));

编辑login()函数

**void** Login::login()

{

    if(usrLineEdit->text().trimmed() == "yyz"

            && pwdLineEdit->text().trimmed() == "123456")

*//if(1) --- 为了便于测试，可以先设置if（1）*

    {

        accept();

    }

    else

    {

        QMessageBox::warning(this,"警告！","用户名或密码错误！",QMessageBox::Yes);

        usrLineEdit->clear();

        pwdLabel->clear();

        usrLineEdit->setFocus();

    }

}

#### 编辑main.cpp，实现从登陆界面到主界面的跳转

#include "chatwidget.h"

#include "login.h"

#include <QApplication>

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

{

    QApplication a(argc, argv);

    ChatWidget w;

    Login l;

    if(l.exec() == QDialog::Accepted)

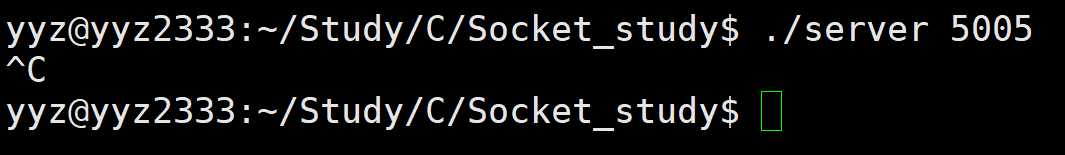
    w.show();

    return a.exec();

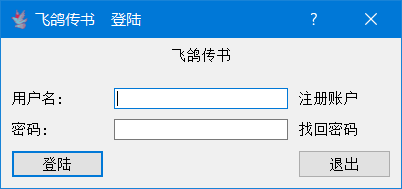
}

## 实现效果

### Server



### Client



# PART02 --- 搭建聊天对话框界面

## 内容更新

+ 已有内容

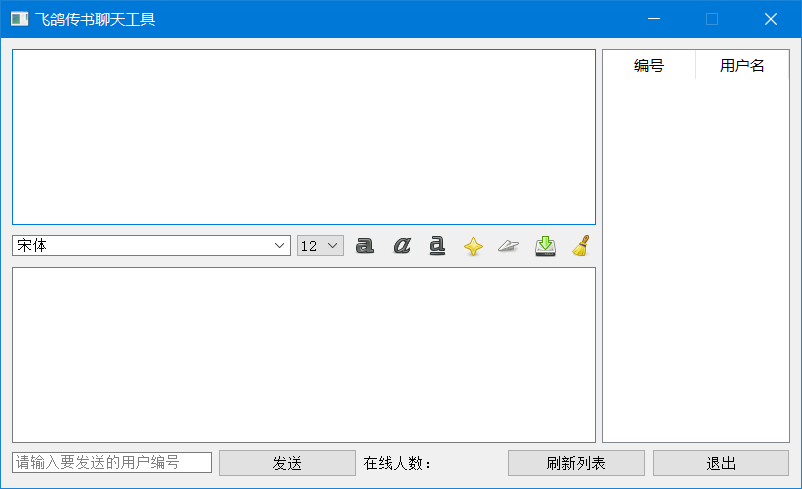
  + Linux下的纯C客户端(仅实现了最简单的应用)

  + 登陆窗口（未绑定数据库）

+ 新增内容

  + 客户端对话框界面

## 任务目标

实现聊天对话框界面如下：

## 任务实现

### 添加网络通讯模块(为Part3做准备)

文件：

QQ-version-of-Qt\PART02\TcpClient\TcpClient.pro

  Qt += network

### 初始化窗口插件

文件：

QQ-version-of-Qt\PART02\TcpClient\ chatwidget.h

预编译指令和类前置声明

#include <QWidget>

#include <QTextBrowser>

#include <QFontComboBox>

#include <QComboBox>

#include <QToolButton>

#include <QTextEdit>

#include <QTableWidget>

#include <QLineEdit>

#include <QPushButton>

#include <QLabel>

#include <QString>

#include <QHBoxLayout>

#include <QVBoxLayout>

#include <QGridLayout>

#include <QHeaderView>

**class** QTextBrowser;

**class** QFontComboBox;

**class** QComboBox;

**class** QToolButton;

**class** QTextEdit;

**class** QTableWidget;

**class** QLineEdit;

**class** QPushButton;

**class** QLabel;

class ChatWidget类声明中添加如下代码

**private:**

    QTextBrowser \*messageBroswer;*//信息文字浏览器*

    QFontComboBox \*fontComboBox;*//字体组合框*

    QComboBox \*sizeCombox;*//字体大小组合框*

    QToolButton \*boldTolBtn;*//加粗按钮*

    QToolButton \*italicToolBtn;*//倾斜按钮*

    QToolButton \*underlineToolBtn;*//下划线按钮*

    QToolButton \*colorToolBtn;*//更改字体颜色按钮*

    QToolButton \*sendToolBtn;*//传输文件按钮*

    QToolButton \*saveToolBtn;*//保存聊天记录按钮*

    QToolButton \*clearToolBtn;*//清空聊天记录按钮*

    QTextEdit \*messageTextEdit;*//消息输入框*

    QTableWidget \*userTableWidget;*//用户列表*

    QPushButton \*sendBtn;*//发送按钮*

    QLabel \*usrNumlabel;*//在线用户数*

QPushButton \*exitBtn;*//退出按钮*

QPushButton \*refreshBtn;*//刷新按钮*

    QLineEdit \*chooseusrNumber;*//选择用户编号*

文件：

QQ-version-of-Qt\PART02\TcpClient\ chatwidget.cpp

*//定义窗口插件*

    messageBroswer = new QTextBrowser(this);*//信息文字浏览器*

    fontComboBox = new QFontComboBox(this);*//字体组合框*

    sizeCombox = new QComboBox(this);*//字体大小组合框*

    boldTolBtn = new QToolButton(this);*//加粗按钮*

    italicToolBtn = new QToolButton(this);*//倾斜按钮*

    underlineToolBtn = new QToolButton(this);*//下划线按钮*

    colorToolBtn = new QToolButton(this);*//更改字体颜色按钮*

    sendToolBtn = new QToolButton(this);*//传输文件按钮*

    saveToolBtn = new QToolButton(this);*//保存聊天记录按钮*

    clearToolBtn = new QToolButton(this);*//清空聊天记录按钮*

    messageTextEdit = new QTextEdit(this);*//消息输入框*

    userTableWidget = new QTableWidget(this);*//用户列表*

    sendBtn = new QPushButton(this);*//发送按钮*

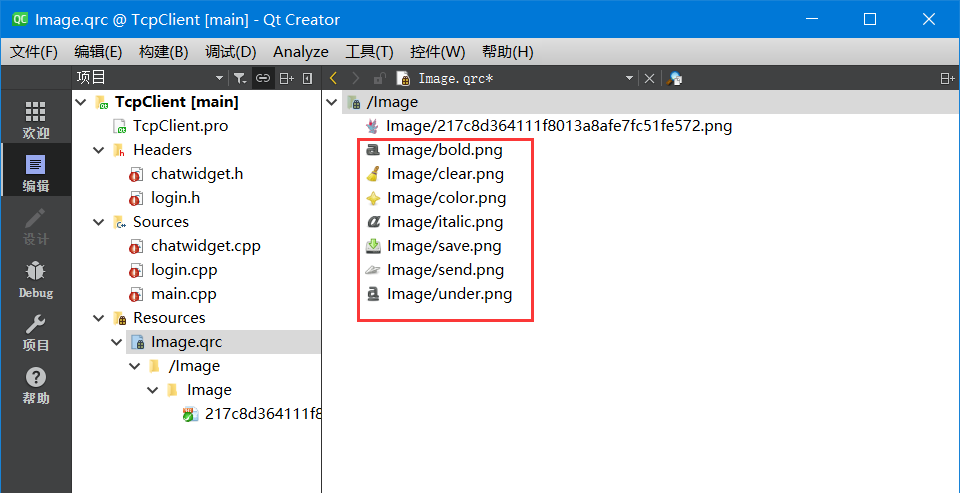
    usrNumlabel = new QLabel(this);*//在线用户数*

    exitBtn = new QPushButton(this);*//退出按钮*

    chooseusrNumber = new QLineEdit(this);*//选择用户编号*

### 设置窗口属性

#### 添加图标资源



#### 字体大小组合框

文件：

QQ-version-of-Qt\PART02\TcpClient\ chatwidget.cpp

*//添加下拉选项  9-22*

    for(**int** i =9;i <= 22;i++)

    {

        sizeCombox->addItem(QString::number(i));

    }

*//设置指定索引号显示*

    sizeCombox->setCurrentIndex(3);

#### 各种ToolButton属性修改

文件：

QQ-version-of-Qt\PART02\TcpClient\ chatwidget.cpp

*//按钮大小*

    boldTolBtn->setFixedSize(30,30);

    italicToolBtn->setFixedSize(30,30);

    underlineToolBtn->setFixedSize(30,30);

    colorToolBtn->setFixedSize(30,30);

    sendToolBtn->setFixedSize(30,30);

    saveToolBtn->setFixedSize(30,30);

    clearToolBtn->setFixedSize(30,30);

*//按钮图标大小*

    boldTolBtn->setIconSize(QSize(22,22));

    italicToolBtn->setIconSize(QSize(22,22));

    underlineToolBtn->setIconSize(QSize(22,22));

    colorToolBtn->setIconSize(QSize(22,22));

    sendToolBtn->setIconSize(QSize(22,22));

    saveToolBtn->setIconSize(QSize(22,22));

    clearToolBtn->setIconSize(QSize(22,22));

*//按钮设置自动浮起*

    boldTolBtn->setAutoRaise(true);

    italicToolBtn->setAutoRaise(true);

    underlineToolBtn->setAutoRaise(true);

    colorToolBtn->setAutoRaise(true);

    sendToolBtn->setAutoRaise(true);

    saveToolBtn->setAutoRaise(true);

    clearToolBtn->setAutoRaise(true);

*//按钮设置点击后按下*

    boldTolBtn->setChecked(true);

    italicToolBtn->setChecked(true);

    underlineToolBtn->setChecked(true);

*//设置按钮文本*

    boldTolBtn->setToolTip("加粗");

    italicToolBtn->setToolTip("倾斜");

    underlineToolBtn->setToolTip("下划线");

    colorToolBtn->setToolTip("更改字体颜色");

    sendToolBtn->setToolTip("传输文件");

    saveToolBtn->setToolTip("保存聊天记录");

    clearToolBtn->setToolTip("清空聊天记录");

*//设置按钮图标*

    boldTolBtn->setIcon(QIcon(":/Image/Image/bold.png"));

    italicToolBtn->setIcon(QIcon(":/Image/Image/italic.png"));

    underlineToolBtn->setIcon(QIcon(":/Image/Image/under.png"));

    colorToolBtn->setIcon(QIcon(":/Image/Image/color.png"));

    sendToolBtn->setIcon(QIcon(":/Image/Image/send.png"));

    saveToolBtn->setIcon(QIcon(":/Image/Image/save.png"));

    clearToolBtn->setIcon(QIcon(":/Image/Image/clear.png"));

#### 设置表格空间QTableWidget的属性

文件：

QQ-version-of-Qt\PART02\TcpClient\ chatwidget.cpp

*//设置表格空间QTableWidget的属性*

*//设置表格空间单选*

    userTableWidget->setSelectionMode(QAbstractItemView::SingleSelection);

*//设置一次选中一行*

    userTableWidget->setSelectionBehavior(QAbstractItemView::SelectRows);

*//设置不显示表格线*

    userTableWidget->setShowGrid(false);

*//设置列数+列宽*

    userTableWidget->setColumnCount(2);

*//setSectionResizeMode*

    userTableWidget->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);

    userTableWidget->verticalHeader()->setSectionResizeMode(QHeaderView::Stretch);

*//设置表头*

    QStringList strs;

    strs << QStringLiteral("编号") << QStringLiteral("用户名");

    userTableWidget->setHorizontalHeaderLabels(strs);

#### 其他控件属性

文件：

QQ-version-of-Qt\PART02\TcpClient\ chatwidget.cpp

*//设置各种普通按钮属性*

    sendBtn->setText(QString("发送"));

exitBtn->setText(QString("退出"));

refreshBtn->setText("刷新列表");

*//设置标签类属性*

    usrNumlabel->setText(QString("在线人数："));

### 布局管理

文件：

QQ-version-of-Qt\PART02\TcpClient\ chatwidget.cpp

*//////////////////////////////布局管理///////////////////////*

*//窗口一：各种ToolButton+字体组合框  水平布局*

    QHBoxLayout \*horizontalLayout\_1 = new QHBoxLayout;

    horizontalLayout\_1->addWidget(fontComboBox);

    horizontalLayout\_1->addWidget(sizeCombox);

    horizontalLayout\_1->addWidget(boldTolBtn);

    horizontalLayout\_1->addWidget(italicToolBtn);

    horizontalLayout\_1->addWidget(underlineToolBtn);

    horizontalLayout\_1->addWidget(colorToolBtn);

    horizontalLayout\_1->addWidget(sendToolBtn);

    horizontalLayout\_1->addWidget(saveToolBtn);

    horizontalLayout\_1->addWidget(clearToolBtn);

*//窗口二：信息文字浏览器+窗口一+消息输入框  垂直布局*

    QVBoxLayout \*horizontalLayout\_2 = new QVBoxLayout;

    horizontalLayout\_2->addWidget(messageBroswer);

    horizontalLayout\_2->addLayout(horizontalLayout\_1);

    horizontalLayout\_2->addWidget(messageTextEdit);

*//窗口三：窗口二+用户列表  水平布局*

    QHBoxLayout \*horizontalLayout\_3 = new QHBoxLayout;

    horizontalLayout\_3->addLayout(horizontalLayout\_2);

    horizontalLayout\_3->addWidget(userTableWidget);

*//窗口四：选择用户编号框+发送按钮+在线用户数+退出按钮+刷新按钮 水平布局*

    QHBoxLayout \*horizontalLayout\_4 = new QHBoxLayout;

    horizontalLayout\_4->addWidget(chooseusrNumber);

    horizontalLayout\_4->addWidget(sendBtn);

    horizontalLayout\_4->addWidget(usrNumlabel);

    horizontalLayout\_4->addWidget(refreshBtn);

    horizontalLayout\_4->addWidget(exitBtn);

*//窗口五：窗口三+窗口四  垂直布局*

    QVBoxLayout \*horizontalLayout\_5 = new QVBoxLayout;

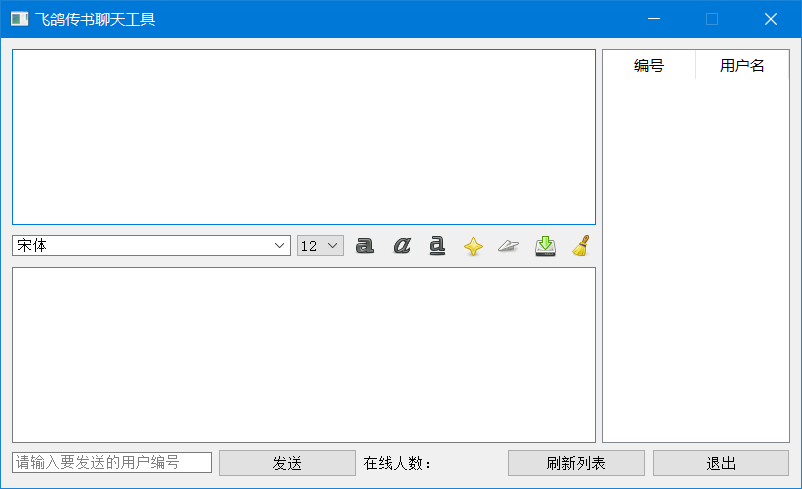
    horizontalLayout\_5->addLayout(horizontalLayout\_3);

    horizontalLayout\_5->addLayout(horizontalLayout\_4);

*//加入主窗口*

    setLayout(horizontalLayout\_5);

## 实现效果



# PART03 --- 搭建聊天对话框界面

## 内容更新

+ **\*\*前期第一次大修改\*\***

+ 已有内容

  + Linux下的纯C客户端(仅实现了最简单的应用)

  + 登陆窗口（未绑定数据库）

  + 客户端对话框界面

+ 新增内容

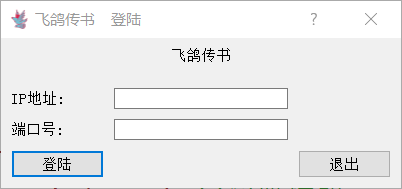
  + 对登陆窗口进行修改，使其变成连接服务器的窗口

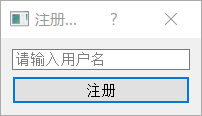
  + 增加一个对话框，用以注册用户名(用户名在服务器端保存)

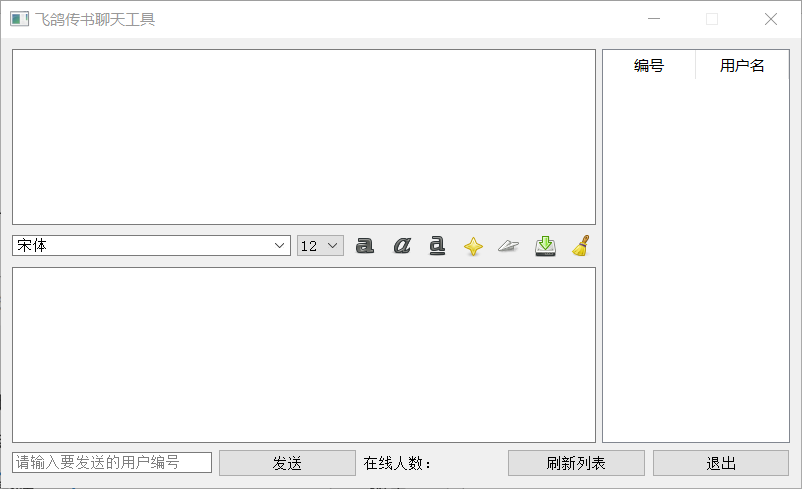
  + 最新的服务器（龙拐竹写的）

## 任务目标

实现聊天对话框界面如下：







## 任务实现

### 修改登陆窗口

#### 删除注册账号、找回密码两个Qlabel

#### 修改相关字段如下

文件：QQ-version-of-Qt\PART01\TcpClient\login.h

**private:**

    QLabel \*IPLabel;*//文本：IP地址*

    QLabel \*portLabel;*//文本：端口号*

    QLabel \*logoLabel;*//文本：logo*

    QLineEdit \*IPLineEdit;*//输入框：IP地址*

    QLineEdit \*portLineEdit;*//输入框：端口号*

    QPushButton \*loginBtn;*//按钮：登陆*

    QPushButton \*exitBtn;*//按钮：退出*

### 添加注册对话框

新建类：Registered

继承于：Qdialog

#### 前置和预编译

文件：QQ-version-of-Qt\PART01\TcpClient\ registered.h

#include <QDialog>

#include <QPushButton>

#include <QLineEdit>

#include <QString>

#include <QVBoxLayout>

#include <QMessageBox>

**class** QPushButton;

**class** QLineEdit;

**class** QVBoxLayout;

#### 声明定义

文件：QQ-version-of-Qt\PART01\TcpClient\ registered.h

class Registered类声明中

**private:**

    QLineEdit \*usrname;*//注册用户名输入框*

    QPushButton \*registered;*//注册按钮*

文件：QQ-version-of-Qt\PART01\TcpClient\ registered.cpp

class Registered类定义中

*//定义*

    setWindowTitle("注册用户");

    registered = new QPushButton(this);

    usrname = new QLineEdit(this);

*//文本*

    registered->setText(QString("注册"));

    usrname->setPlaceholderText(QString("请输入用户名"));

#### 布局

*//垂直布局*

    QVBoxLayout \*horizontalLayout\_1 = new QVBoxLayout;

    horizontalLayout\_1->addWidget(usrname);

    horizontalLayout\_1->addWidget(registered);

    setLayout(horizontalLayout\_1);

}

#### 信号与槽

文件：QQ-version-of-Qt\PART01\TcpClient\ registered.h

class Registered类声明中

private slots:

**void** registre();

文件：QQ-version-of-Qt\PART01\TcpClient\ registered.cpp

class Registered类定义中

*//信号与槽*

    connect(registered,SIGNAL(clicked(**bool**)),this,SLOT(login()));

void Registered::registre()中

**void** Registered::registre()

{

    if(1)

    {

        accept();

    }

    else

    {

        QMessageBox::warning(this,"警告！","连接错误",QMessageBox::Yes);

        exit(-1);

    }

}

### 修改main.cpp

#include "chatwidget.h"

#include "login.h"

#include "registered.h"

#include <QApplication>

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

{

    QApplication a(argc, argv);

    ChatWidget w;

    Login l;

    Registered r;

    if(l.exec() == QDialog::Accepted)

    {

        if(r.exec() == QDialog::Accepted)

        {

            w.show();

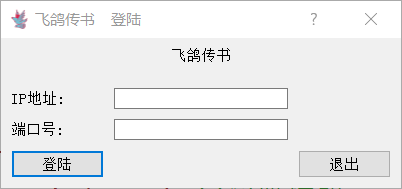
        }

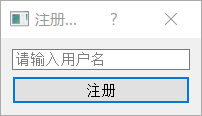
    }

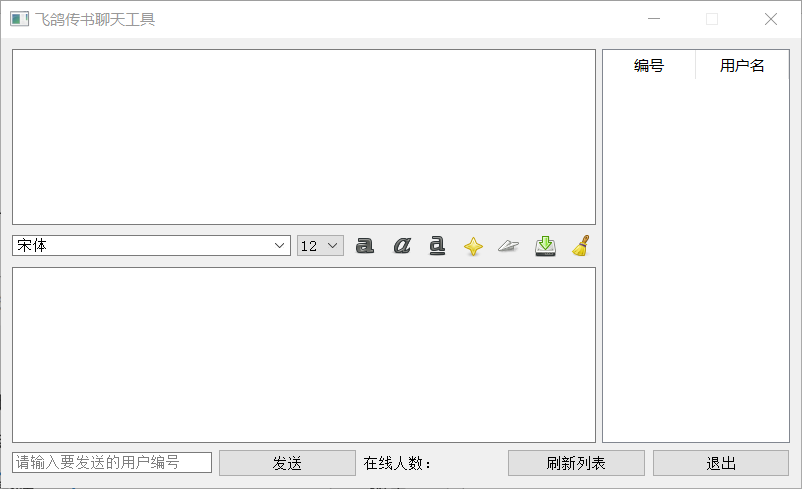
    return a.exec();

}

## 实现效果







## 服务器代码

#include<pthread.h>

#include<fcntl.h>

#include<netinet/in.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<sys/stat.h>

#include<sys/ipc.h>

#include<sys/shm.h>

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<arpa/inet.h>

#include<unistd.h>

#include<netdb.h>

#define LENGTH\_OF\_LISTEN\_QUEUE 20

#define BUFFER\_SIZE 1024

#define SHMBUFSZ 2048

#define FILE\_NAME\_MAX\_SIZE 512

#define ClientMAX 10

typedef **struct** clientinfo

{

**char** name[10];

**int** clientfd;

*//void \*sock\_fd;*

}clientinfo;

clientinfo infolist[ClientMAX];*//Clientlist*

**pthread\_mutex\_t** mutex;

**void** servant(**void** \* sock\_fd)

{

*//init*

**int** master\_fd = \*((**int** \*)sock\_fd);

**char** buffer[BUFFER\_SIZE];

    clientinfo Clientbuf[ClientMAX];

**int** masternum;

**int** temp;

    pthread\_mutex\_lock(&mutex);

    for (**int** i = 0; i < ClientMAX; i++)

        Clientbuf[i]=infolist[i];

    pthread\_mutex\_unlock(&mutex);

    while(1)

    {

        memset(buffer,9,sizeof(buffer));

        if (recv(master\_fd, buffer, BUFFER\_SIZE, 0) < 0)

        {

            perror("Server Recieve Data Failed:");

            break;

        }

*//idenfy message*

*//请求用户列表*

        if (buffer[0]=='2')

        {

            pthread\_mutex\_lock(&mutex);

            for (**int** i = 0; i < ClientMAX; i++)

                Clientbuf[i]=infolist[i];

            pthread\_mutex\_unlock(&mutex);

            for (**int** i = 0; i < ClientMAX; i++)

            {

                memset(buffer,9,sizeof(buffer));

                buffer[0]='2';

                buffer[1]=(**char**)i;

                strcat(buffer,Clientbuf[i].name);

                if (send(master\_fd,buffer,sizeof(buffer),0))

                    perror("User List send fail");

            }

        }

*//注册用户列表*

        if (buffer[0]=='3')

        {

            pthread\_mutex\_lock(&mutex);

            for (**int** i = 0; i < ClientMAX; i++)

            {

                if (infolist[i].clientfd=99)

                {

                    infolist[i].clientfd=master\_fd;

                    for (**int** j = 1; j < strlen(buffer); j++)

                    {

                        infolist[i].name[j-1]=buffer[j];

                    }

                    masternum=i;

                    break;

                }

            }

            pthread\_mutex\_unlock(&mutex);

        }

*//转发*

        if(buffer[0]=='0'||buffer[0]=='1')

        {

            temp=buffer[1];

            buffer[1]=master\_fd;

            if (send(Clientbuf[temp].clientfd,buffer,sizeof(buffer),0))

                perror("User List send fail");

        }

*//退出*

        if(buffer[0]=='4')

        {

            pthread\_mutex\_lock(&mutex);

                infolist[masternum].clientfd=99;

                memset(infolist[masternum].name,9,sizeof(infolist[masternum].name));

            pthread\_mutex\_unlock(&mutex);

            close(master\_fd);

            pthread\_exit(NULL);   *//terminate calling thread!*

        }

    }

}

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

{

**int** logfd;

**int** serverfd;

**int** clientfd;

**int** iret;

**int** shmid;

**char** \*shmadd;

**char** wrbuf**[]**=" connected.\n";

**char** buffer[1024];

**key\_t** key;

**pthread\_t** serverid;

**struct** sockaddr\_in servaddr;

**struct** sockaddr\_in clientaddr;

*//netinit*

**int** socklen = sizeof(**struct** sockaddr\_in);

    if((serverfd = socket(AF\_INET,SOCK\_STREAM,0)) == -1)

    {

        perror("socket");

        return -1;

    }

    memset(&servaddr,0,sizeof(servaddr));

*//TCP set*

    servaddr.sin\_family = AF\_INET;

    servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

    servaddr.sin\_port = htons(atoi(argv[1]));

*//TCP bind*

    if(bind(serverfd,(**struct** sockaddr \*)&servaddr,sizeof(servaddr)) != 0)

    {

        perror("bind");

        close(serverfd);

        return -1;

    }

*//TCP listen*

    if(listen(serverfd,10) != 0)

    {

        perror("listen");

        close(serverfd);

        return -1;

    }

*//localinit*

*//clientlist init*

    for (**int** i = 0; i < ClientMAX; i++)

    {

        infolist[i].clientfd=99;

        memset(&infolist[i].name,0,sizeof(infolist[i].name));

    }

*//open logfile*

*// if(logfd=open("./log.txt",O\_RDWR|O\_CREAT|O\_APPEND,0777))*

*// {*

*//     printf("Fail to open the logfile!\n");*

*//     return 1;*

*// }*

*//init pthread mutex*

    pthread\_mutex\_init(&mutex, NULL);

    while (1)

    {

*//get connection*

        socklen\_t clientaddrlength = sizeof(clientaddr);

        memset(buffer,0,sizeof(buffer));

        clientfd = accept(serverfd,(**struct** sockaddr \*)&clientaddr,(socklen\_t\*)&clientaddrlength);

        if(clientfd < 0)

        {

            perror("Server Accept Failed");

            break;

        }

        else

        {

*//write connection to the log*

            printf("Client（%s）has connected。Opening a new thread\n",inet\_ntoa(clientaddr.sin\_addr));

            strcpy(buffer,inet\_ntoa(clientaddr.sin\_addr));

            strcat(buffer,wrbuf);

            write(logfd,buffer,strlen(buffer));

        }

        if (pthread\_create(&serverid, NULL, (**void** \*)(&servant), (**void** \*)(&clientfd)) == -1)

        {

            fprintf(stderr, "pthread\_create error!\n");

            break;

        }

    }

    pthread\_mutex\_destroy(&mutex);

    close(serverfd);

    close(logfd);

}

# PART04 --- 连接窗口功能实现

## 内容更新

+ 已有内容

  + 最新的Linux下的C语言服务器（龙拐竹写的）

  + 连接窗口

  + 注册窗口

  + 客户端对话框界面

+ 新增内容

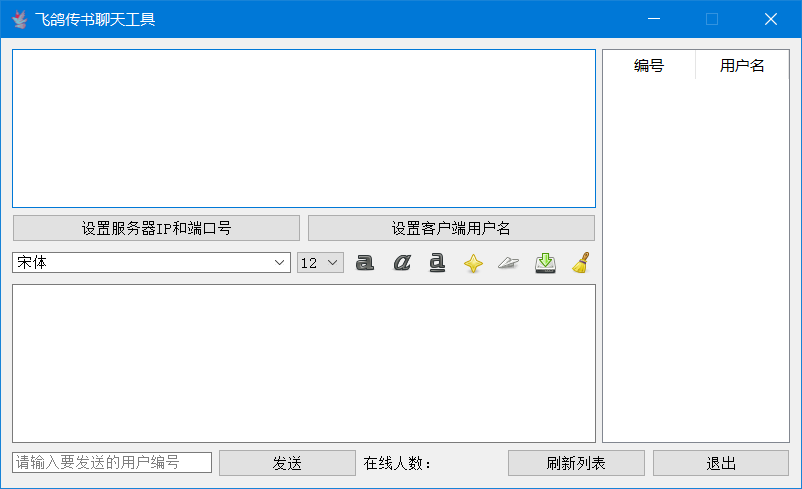
  + 项目大改版

    + 将聊天窗口作为主窗口，登陆、注册窗口设置为对话框

## 任务目标

对软件整体结构布局进行改版，要求以聊天窗口为主窗口，登陆、注册窗口以对话框的形式附属于主窗口，数据最终归聊天窗口。

界面效果如下：



## 任务实现

### 主窗口添加两个新按钮

文件：

QQ-version-of-Qt\PART02\TcpClient\ chatwidget.h

*//PART04改版添加*

    QPushButton \*loginBtn;*//登陆弹窗按钮*

    QPushButton \*regBtn;*//注册弹窗按钮*

文件：

QQ-version-of-Qt\PART02\TcpClient\ chatwidget.cpp

*//PART4大改版添加*

    loginBtn = new QPushButton(this);*//登陆*

    regBtn = new QPushButton(this);*//注册*

    loginBtn->setText(QString("设置服务器IP和端口号"));

    regBtn->setText(QString("设置客户端用户名"));

    QHBoxLayout \*horizontalLayout\_6 = new QHBoxLayout;

    horizontalLayout\_6->addWidget(loginBtn);

    horizontalLayout\_6->addWidget(regBtn);

    setWindowIcon(QIcon(":/Image/Image/217c8d364111f8013a8afe7fc51fe572.png"));

*//窗口二：信息文字浏览器+窗口一+消息输入框  垂直布局*

    QVBoxLayout \*horizontalLayout\_2 = new QVBoxLayout;

    horizontalLayout\_2->addWidget(messageBroswer);

*//PART04新增加内容*

    horizontalLayout\_2->addLayout(horizontalLayout\_6);

    horizontalLayout\_2->addLayout(horizontalLayout\_1);

    horizontalLayout\_2->addWidget(messageTextEdit);

### 设置信号与槽

文件：

QQ-version-of-Qt\PART02\TcpClient\ chatwidget.h

类的前置声明

**class** Login;

**class** Registered;

声明槽函数

*//PART04改版添加*

    QPushButton \*loginBtn;*//登陆弹窗按钮*

    QPushButton \*regBtn;*//注册弹窗按钮*

文件：

QQ-version-of-Qt\PART02\TcpClient\ chatwidget.cpp

连接信号与槽

*//PART04大改版添加*

    connect(loginBtn,SIGNAL(clicked(**bool**)),this,SLOT(setIP()));

    connect(regBtn,SIGNAL(clicked(**bool**)),this,SLOT(setusrname()));

定义槽函数

**void** ChatWidget::setIP()

{

    Login a;

    a.exec();

}

**void** ChatWidget::setusrname()

{

    Registered b;

    b.exec();

}

# PART04 --- 连接窗口功能实现

## 内容更新

+ 已有内容

  + 最新的Linux下的C语言服务器（龙拐竹写的）

  + 聊天框界面作为主界面

  + 登陆（设置IP）、注册（设置用户名）两个界面作为对话框依附于主界面

+ 新增内容

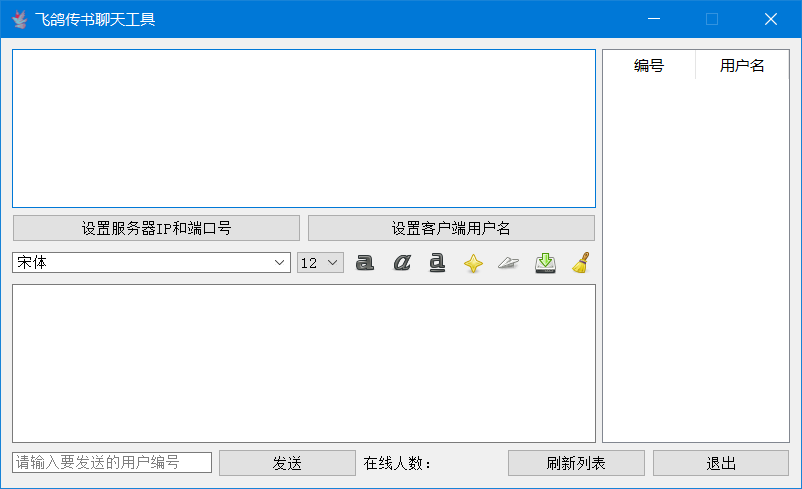
  + 最简单的客户端

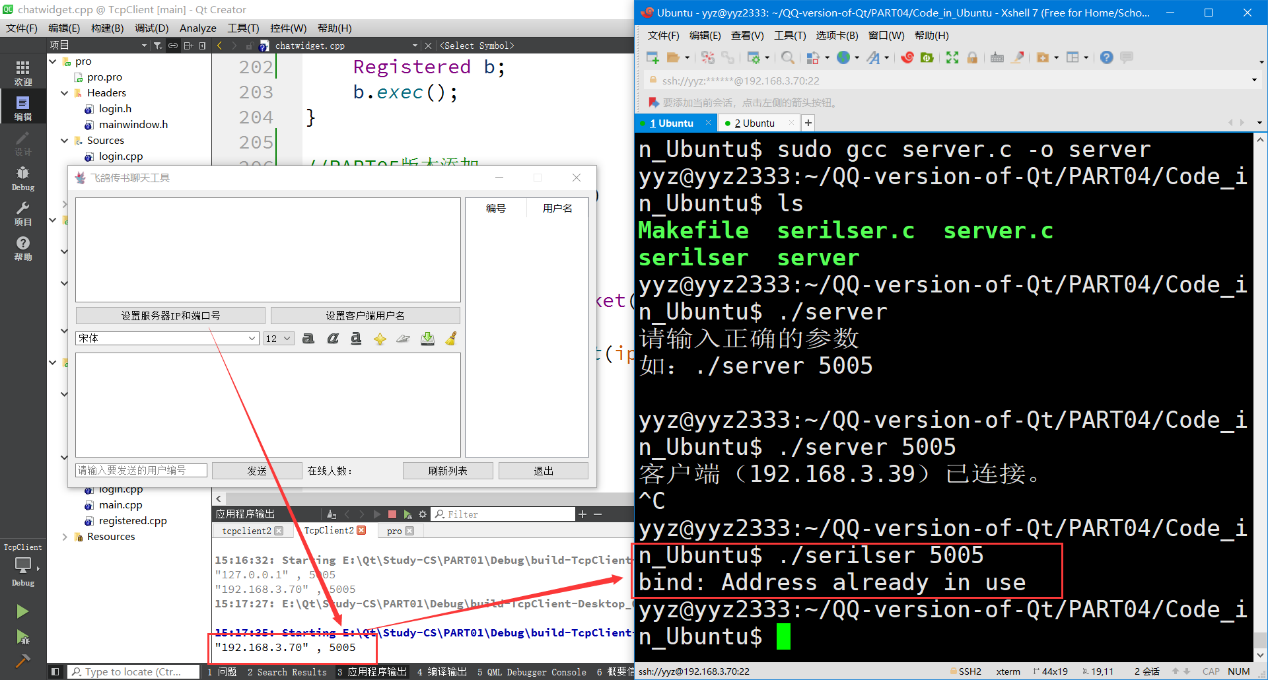
    + 能和服务端接通，获得反馈

## 任务目标

完成最简单的客户端功能，做到于服务端连通

界面效果如下：





## 任务实现

### 添加网络编程模块

文件：QQ-version-of-Qt\PART05\TcpClient\ TcpClient.pro

*##PART05添加*

*QT += network*

### 添加全局变量ip(ip地址)和port(系统端口)

#### 声明

注意：声明全局变量必须在类的外面声明，因为全局变量不属于这个类

文件：QQ-version-of-Qt\PART05\TcpClient\ login.h

*//全局变量*

**extern** QString ip;

**extern** **int** port;

#### 实例化

文件：QQ-version-of-Qt\PART05\TcpClient\ main.cpp

注意：全局变量声明后还需要进行实例化，同样是放在所有的类和函数外面，如果不实例化，系统在编译的时候会报错

QString ip;

**int** port;

### 设置socket

#### 添加预编译指令和类的前置声明

文件：QQ-version-of-Qt\PART05\TcpClient\ ChatClient.h

#include <QtNetwork>

#include <QAbstractSocket>

**class** QTcpSocket;

#### 声明socket

文件：QQ-version-of-Qt\PART05\TcpClient\ ChatClient.h

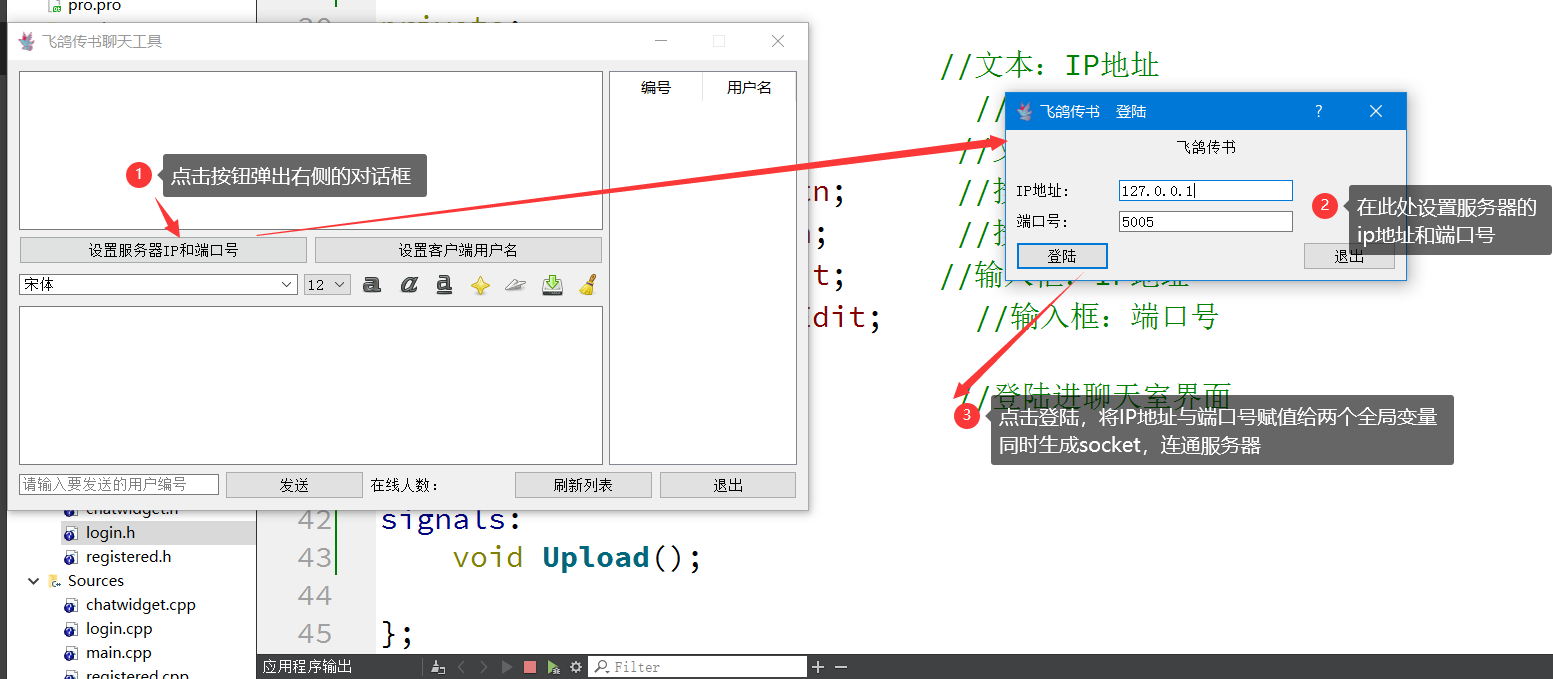
**public:**

*//PART05版本添加*

    QTcpSocket \*socket;

### 设置信号与槽

实现功能如下图：



#### 利用信号与槽机制实现主界面与对话框之间的切换

##### 生成一个信号

文件：

QQ-version-of-Qt\PART05\TcpClient\ login.h

signals:

**void** Upload();*//登录的信号*

##### 利用上面的信号进行信号与槽的切换

文件：

QQ-version-of-Qt\PART05\TcpClient\ login.cpp

*//设置信号与槽*

    connect(loginBtn,SIGNAL(clicked(**bool**)),this,SLOT(login()));

*//PART05版本添加  用于测试代码*

    IPLineEdit->setText(QString("192.168.3.70"));

    portLineEdit->setText(QString("5005"));

……………………

**void** Login::login()*//槽函数*

{

*//全局变量赋值*

    ip = this->IPLineEdit->text();

    port = this->portLineEdit->text().toInt();

    this->hide();*//隐藏对话框*

    emit Upload();*//发射信号*

    qDebug()<<ip<<","<<port;

}

文件：

QQ-version-of-Qt\PART05\TcpClient\ ChatClient.cpp

**void** ChatWidget::setIP()

{

*//PART05版本添加*

    Login a;

    connect(&a,SIGNAL(Upload()),this,SLOT(upload()));

    a.exec();

}

**void** ChatWidget::upload()

{

*//创建socket*

*//socket=IP地址+端口号*

    socket = new QTcpSocket(this);

*//连接*

    socket->connectToHost(ip,port);

}

## 实现成果

