飞鸽传书——Qt版的QQ

[飞鸽传书——Qt版的QQ 1](#_Toc62450000)

[1 PART03 --- 搭建聊天对话框界面 1](#_Toc62450001)

[1.1 内容更新 1](#_Toc62450002)

[1.2 任务目标 2](#_Toc62450003)

[1.3 任务实现 3](#_Toc62450004)

[1.3.1 修改登陆窗口 3](#_Toc62450005)

[1.3.1.1 删除注册账号、找回密码两个Qlabel 3](#_Toc62450006)

[1.3.1.2 修改相关字段如下 3](#_Toc62450007)

[1.3.2 添加注册对话框 3](#_Toc62450008)

[1.3.2.1 前置和预编译 3](#_Toc62450009)

[1.3.2.2 声明定义 4](#_Toc62450010)

[1.3.2.3 布局 5](#_Toc62450011)

[1.3.2.4 信号与槽 5](#_Toc62450012)

[1.3.3 修改main.cpp 7](#_Toc62450013)

[1.4 实现效果 8](#_Toc62450014)

# 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

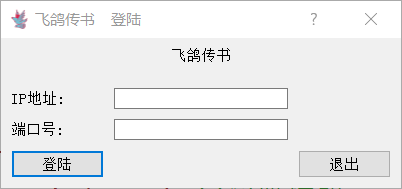
  + 对登陆窗口进行修改，使其变成连接服务器的窗口

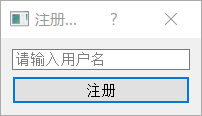
  + 增加一个对话框，用以注册用户名(用户名在服务器端保存)

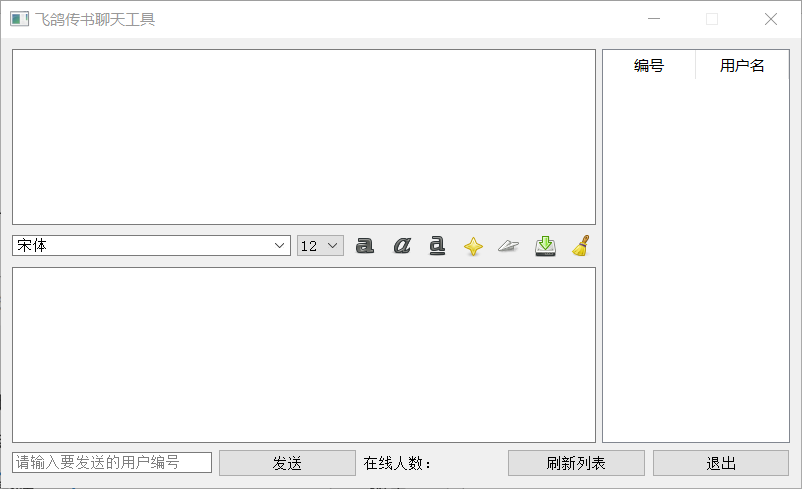
  + 最新的服务器（龙拐竹写的）

## 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();

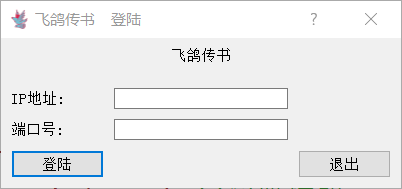
        }

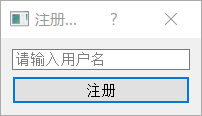
    }

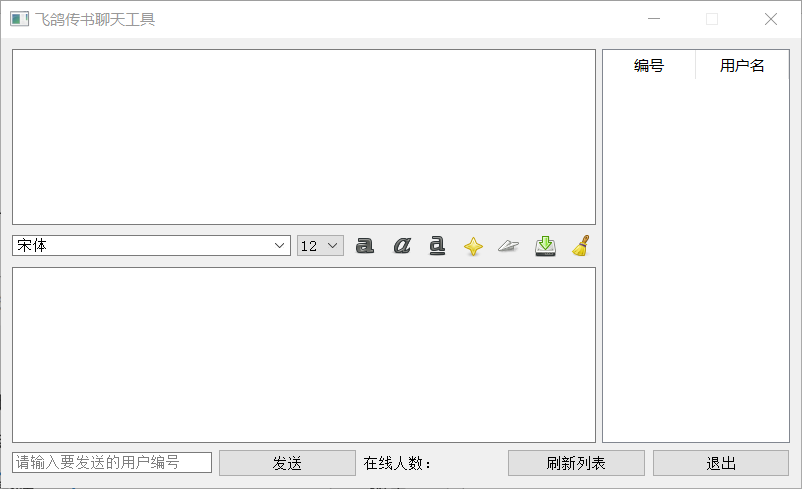
    return a.exec();

}

## 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);

}