一、系统总体介绍

单词消除游戏由两类参与者组成：闯关者（即游戏玩家），出题者（为游戏增加游戏中使用单词）。游戏规则为，游戏每一轮，程序会根据该关卡难度，显示一个单词，一定时间后单词消失。闯关者需要在相应地方输入刚刚显示并消失的单词，如果闯关者输入正确（即闯关者输入的单词与刚刚显示的单词完全一致，包含大小写）则为通过。一关可以由一轮或者多轮组成。

二、开发工具

1.VSCode

2.Qt Creator 10.0.0 (Community)

3.Navicat MySQL 8.0

4.Qt Version 5.15.2

5.CMake 3.25.1

三、设计内容

闯关者属性含有：闯关者姓名、闯关者密码、已闯关关卡数、闯关者经验值、闯关者等级。出题者属性含有：出题者姓名、出题者密码、出题者出题数目、等级。

具体功能包括：

◼实现闯关者，出题者本地的注册、登录。

◼程序支持多人注册，同一时间只有一人可以登录。

◼实现游戏规则：出题者增加游戏中使用单词。游戏每一关，程序会根据该关卡难度，显示一个单词，一定时间后单词消失。闯关者需要在相应地方输入刚刚显示并消失的单词，如果闯关者输入正确则为通过。

◼任何角色均可查询所有闯关者、出题者，按照属性查找相应闯关者、出题者。

◼可以根据闯关者闯过关卡数、经验、等级等对闯关者排名，根据出题者出题数目、等级对出题者排名。

◼每一关的难度要有所增加，体现为如下两个条件：

1、进行轮数增多（即单词数目增加，如：前三关仅仅通过一个单词就过关，后续需要通过两个、三个甚至更多才过关）；

2、单词显示时间缩短（随着关卡的增加显示时间越来越短）。

◼闯关者每闯过一关，增加一定经验值。经验值会根据闯过的该关卡的关卡号、该关的闯关耗费时间共同决定。当经验值累计到一定程度闯关者等级增加。闯关失败需要重新闯该关。

◼游戏自带词库，而且已经注册的出题者可以为系统出题，即增加词库的新词，已经存在的单词不能再次添加。

◼每成功出题一次，更新该出题者的出题数目。出题者等级根据出题人成功出题数目来升级。

◼游戏为服务器多人游戏平台，使用客户端/服务器的方式，同一时间可以多人登录系统。

◼将所有闯关者、出题者信息保存在服务器。使用socket进行通信。客户端可以启动多个同时与服务器交互，服务器具有并发处理能力。

四、算法设计思路

本游戏绝大部分运算均在本地进行，服务端仅进行数据的增删改查。

1.类的设计

（1）客户端

客户端由多个界面和有关参与者的类组成。

有关参与者的类有三个，分别是Member，Player和Tester。其中Member包含两个类的基本属性：用户名、密码、经验和等级。

Player类包含通关数，Tester类包含出题数。

其余类分别实现登录、注册、查询、添加单词，单人多人游戏功能。

（2）服务端

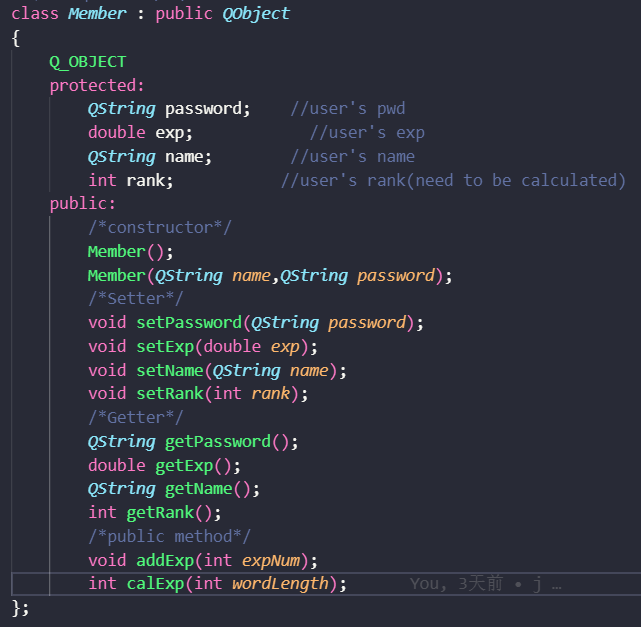
服务端主要是实现socket通信、多线程处理socket和数据库交互。

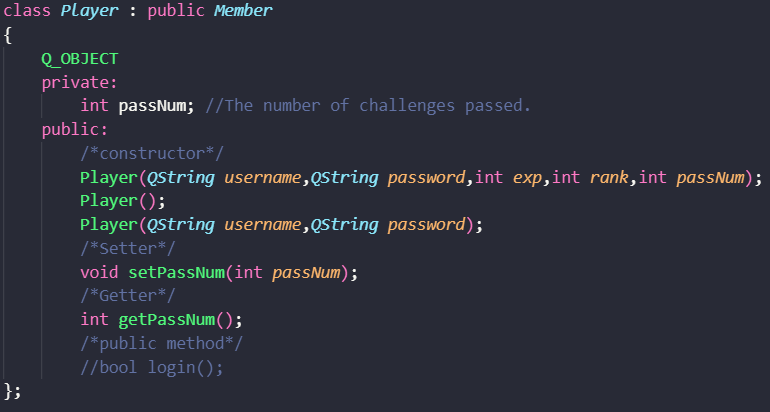
2.客户端界面介绍及功能实现

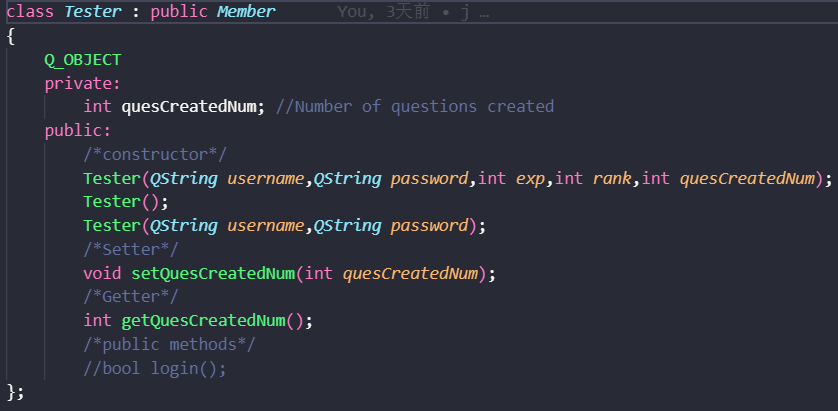
(1)有关参与者的类

由于玩家和出题者均存在用户名，用户密码，经验和等级这四个共同属性。因此设计Member类，Player类和Tester类继承自Member类和它们的Get，Set方法。

Player类独有属性通关数，Tester类独有属性出题数，分别存在Get，Set方法。







(2)注册登录界面(实现于widget.h widget.cpp中)

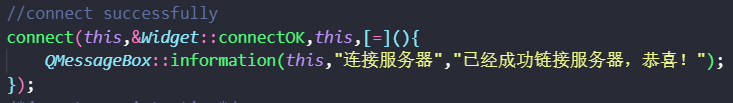
Ⅰ.连接服务器

ip指定为本机ip：127.0.0.1

端口为：8989

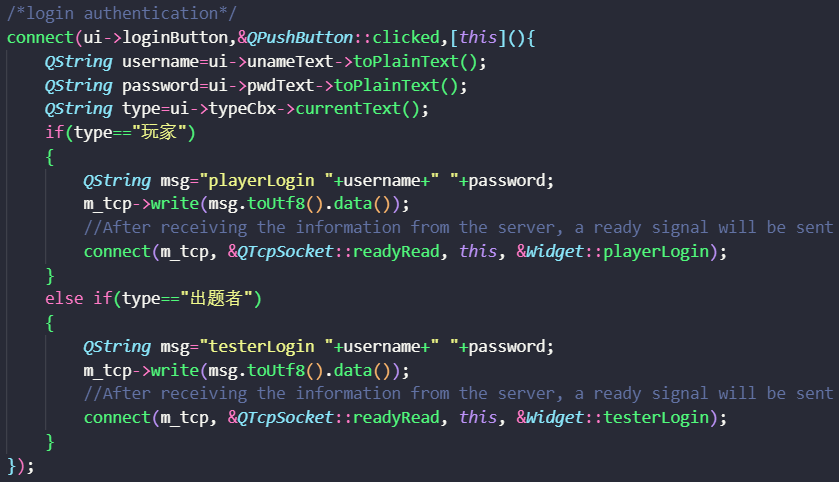
成功链接服务器后弹窗提醒。



(3)登录功能实现

点击登录按钮，会根据类型发送请求，注册信号，根据服务器端回复的数据来决定操作。服务器端与客户端约定好传输方式：

0代表失败，1代表成功，后面是该用户的所有数据。



为防止内存泄漏，本程序所有页面跳转均采用关闭当前所有界面，而后传递tcp参数和用户数据的方式。

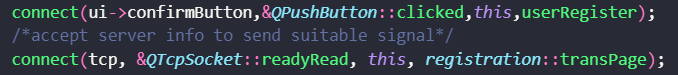


管理员登录同理：



Ⅱ.注册功能实现(实现于registration.cpp registration.h)

点击按钮，跳转到注册界面。采用QT独有信号槽机制，当点击确认注册按钮时，向服务器端发送请求。根据服务器端回复的数据来决定操作。服务器端与客户端约定好传输方式：0代表失败，1代表成功，然后跳转到登陆后界面。







(3)玩家操作界面(实现于playerPage.cpp playerPage.h中)



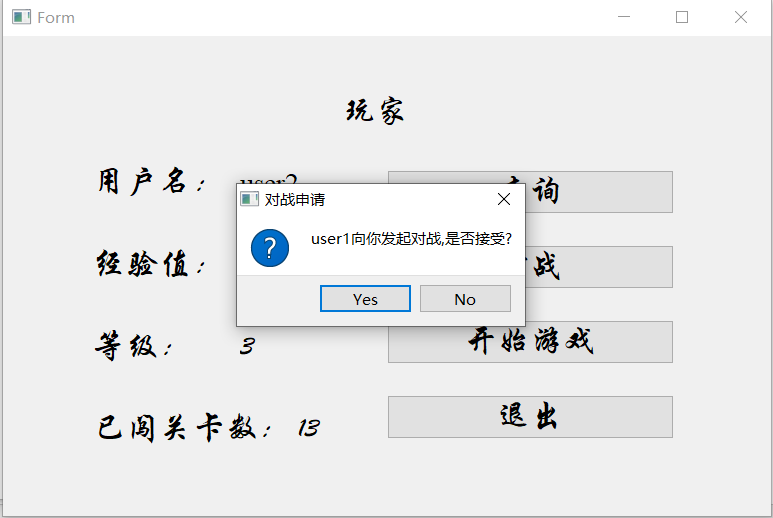
在玩家操作界面可以选择，查询，对战，单人游戏，退出功能和玩家信息查看，通过跳转界面实现。值得注意的是退出按钮，当退出该界面时，会跳转到登录界面，但是创建登陆界面时会新建一个socket链接。因此需要在退出程序时关闭已经创建的链接。先向服务器发送一个信号，代表该用户已经退出，以来更新数据表。



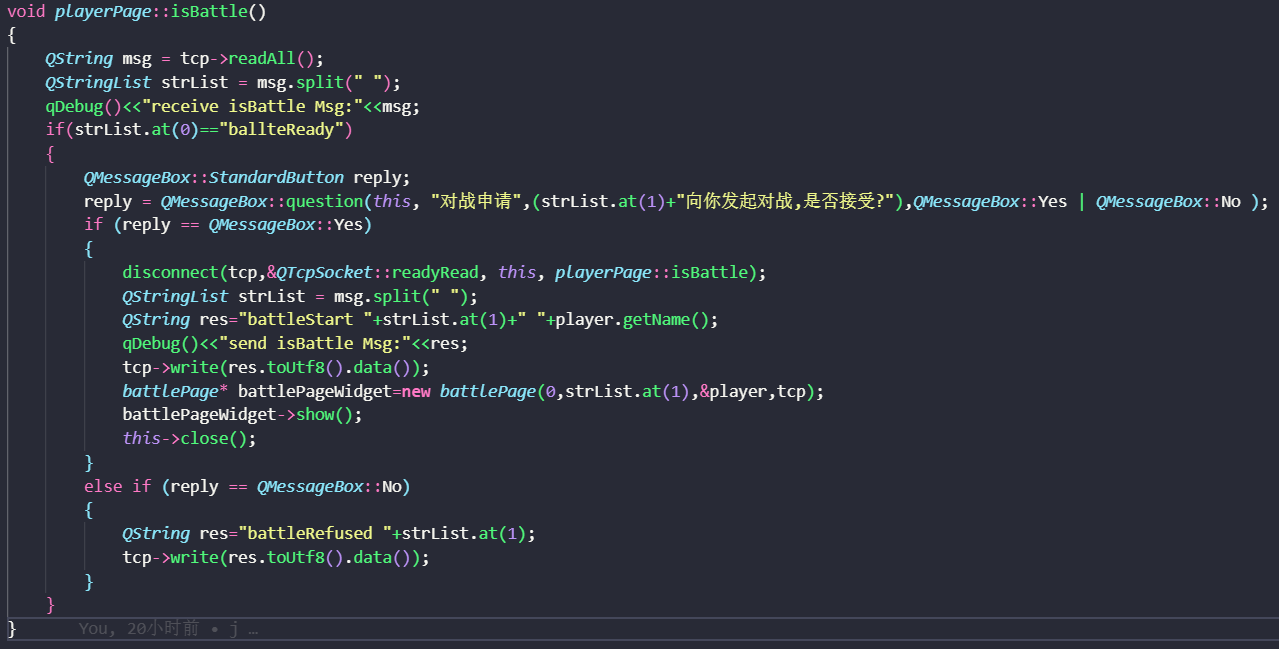
Ⅰ. 处理对战功能

先将信号与槽相连接，但退出该界面后，需要将所有链接断开，防止不同的信号抢夺同一条服务器发的信息。

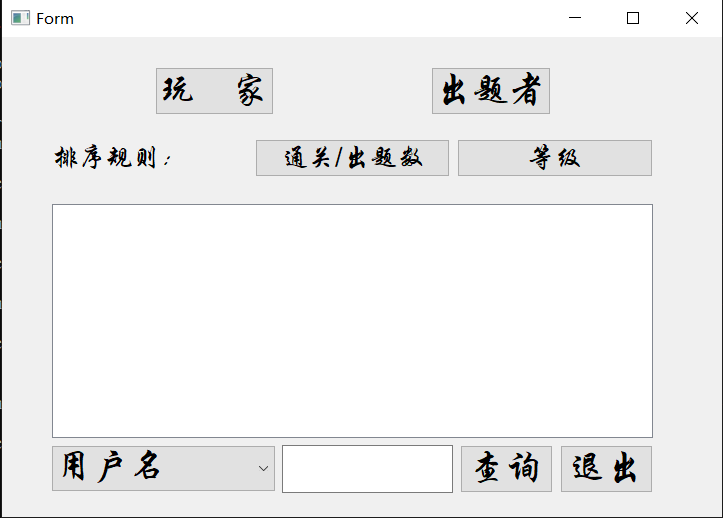




当收到服务器邀请对战，取下发起挑战人的信息，弹窗提醒。如果同意进行对战，则向服务器发送同意对战信息，并跳转到对战界面。如果拒绝对战，则向服务器发送拒绝对战信息。

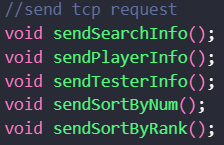
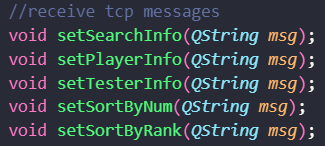
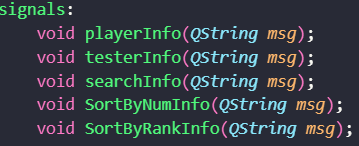


Ⅱ.查询界面(实现于searchpage.h searchpage.cpp)



可以查询所有玩家和出题者的信息，并根据排序规则对所有玩家或出题者进行重新排序。实现了根据用户名、经验、等级进行单\多个用户查询，详见具体代码。

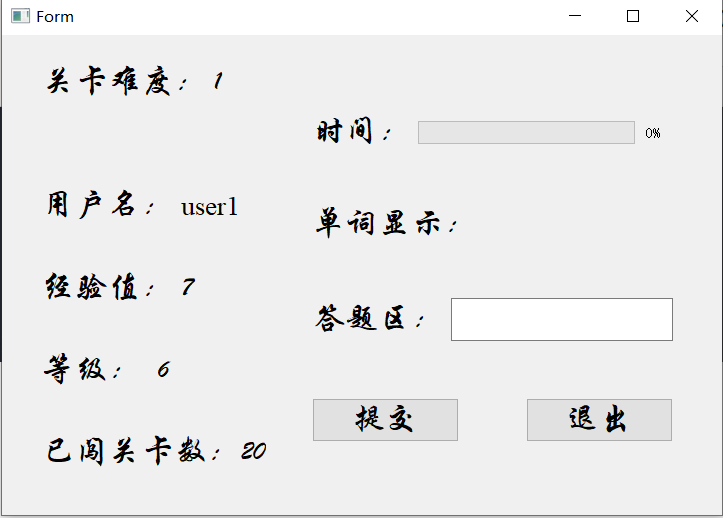
查询原理较为简单，根据不同的查询规则，向服务器发送不同的查询申请。根据服务器返回的信息，发送不同的信号，并创建表格。

链接所有所需的信号。在退出该界面后，仅仅需要关闭tcp链接即可。



Ⅲ.玩家游戏界面(实现于gamepage.cpp gamepage.h)

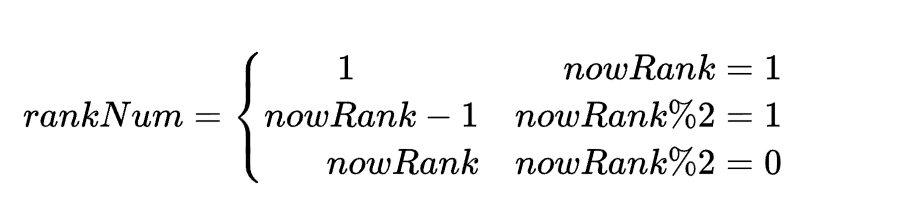


玩家游戏界面通过向服务器发送请求，获取需要记忆的单词，而后在客户端进行本地运算，计算出单词的难度，记忆的时间，答对后获取的经验以及每一关需要正确拼写单词的数量。以下是计算公式：

单词难度：

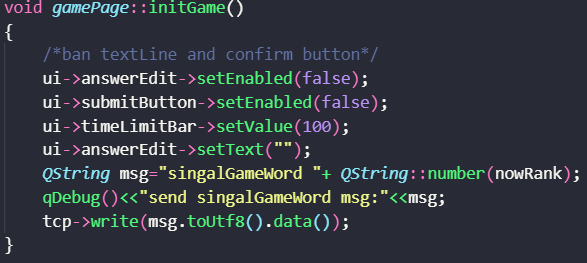
记忆时间：

获取经验：

通关需要单词数：

游戏实现流程如下：

①游戏初始化，向服务器发送需要一个单词的请求



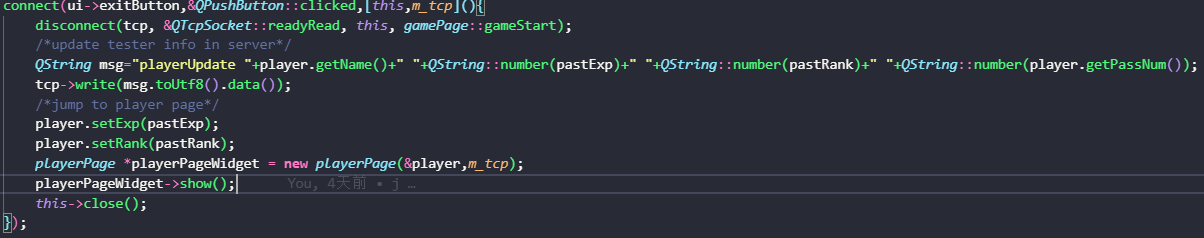
②收到单词后进行上述计算，然后加载进度条提示用户剩余记忆时间。存在两个计时器：processBarTimer和memoryTimer。当memoryTimer计时结束后，单词消失，用户可以输入记忆的单词。processBarTimer将记忆时间均分，每次计时结束均匀递减。



③当点击提交按钮时，进行判断：输入正确则回到步骤①，否则弹窗提示，重新开始这一关。已经获得的经验和等级重置。



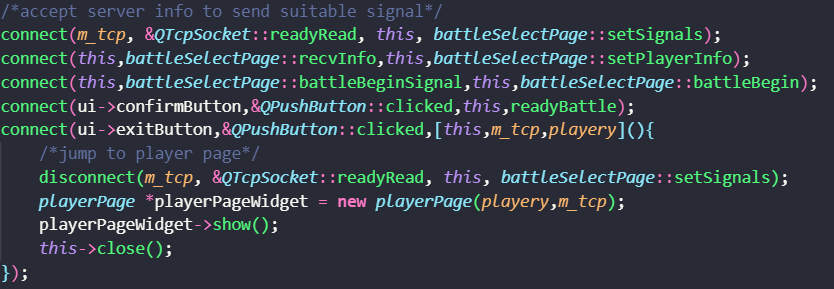
④当玩家打算退出游戏后，需要对所有数据进行更新。



Ⅳ.双人对战模式

①点击对战按钮，首先进入挑选玩家界面。向服务器发送请求，获取当前在线所有玩家，并以表格形式展现。当选中玩家后，点击确认按钮将向服务器发送请求对战要求。服务器向对手发送邀请，如果双方均同意则跳转到对战界面(实现于battleselectpage.cpp battleselectpage.h)。





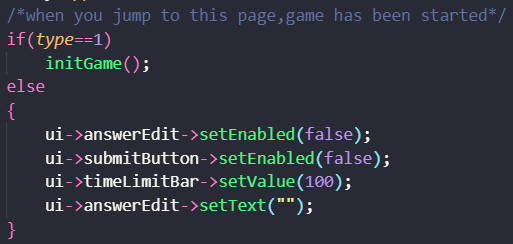
由于需要接受服务器发回的不同数据，因此设计该方法，根据不同数据发出不同信号。





②对战模式双方需要对同一个单词进行回答。（实现于battlepage.h battlepage.cpp y）因此进行如下设计：

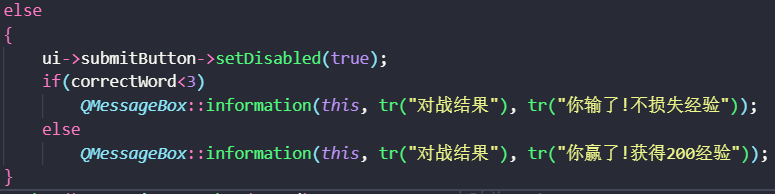
<1> 挑战者首先发送一个需要单词请求，服务端向两个客户端发送该单词。



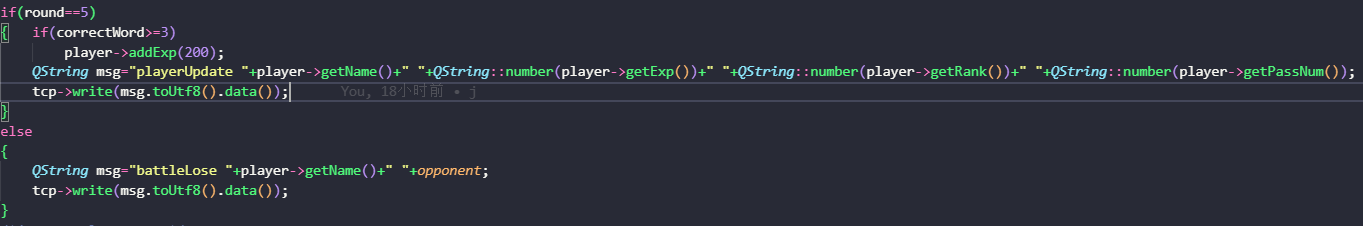
<2> 之后比较两人手速，最先答对的人发送需要单词请求，更新两人题目。



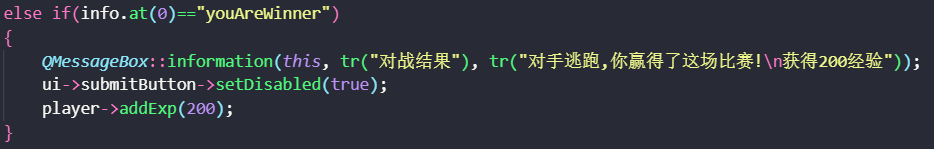
<3>游戏进行5轮，答对题目多者胜。该判断基于本地：本地存在一个变量currentWord，答对单词后自增。当收到五个单词后进入判定。currentWord小于3就会终止游戏，弹出游戏失败指令。

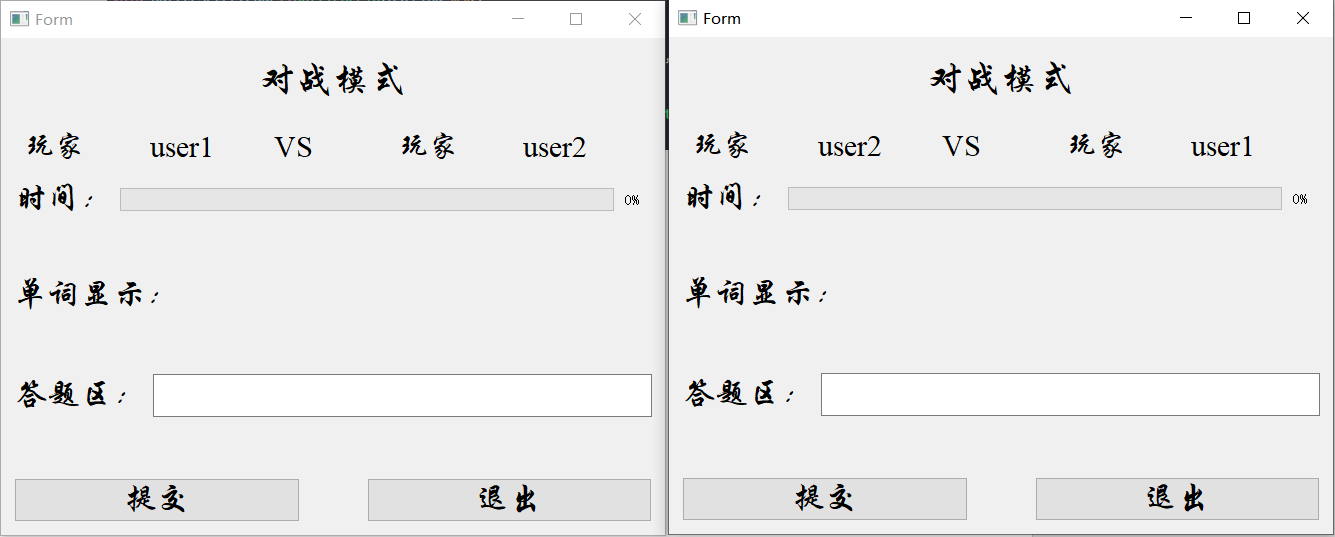


<4> 之后退出对战模式后更新角色经验等级。



<5> 同时做了逃跑模式，当获得单词数目小于5时点击退出视为逃跑。向服务端发送战败信息，服务端接收后向对手发送获胜信息。





(4)出题者操作界面(实现于testerpage.h testerpage.cpp)

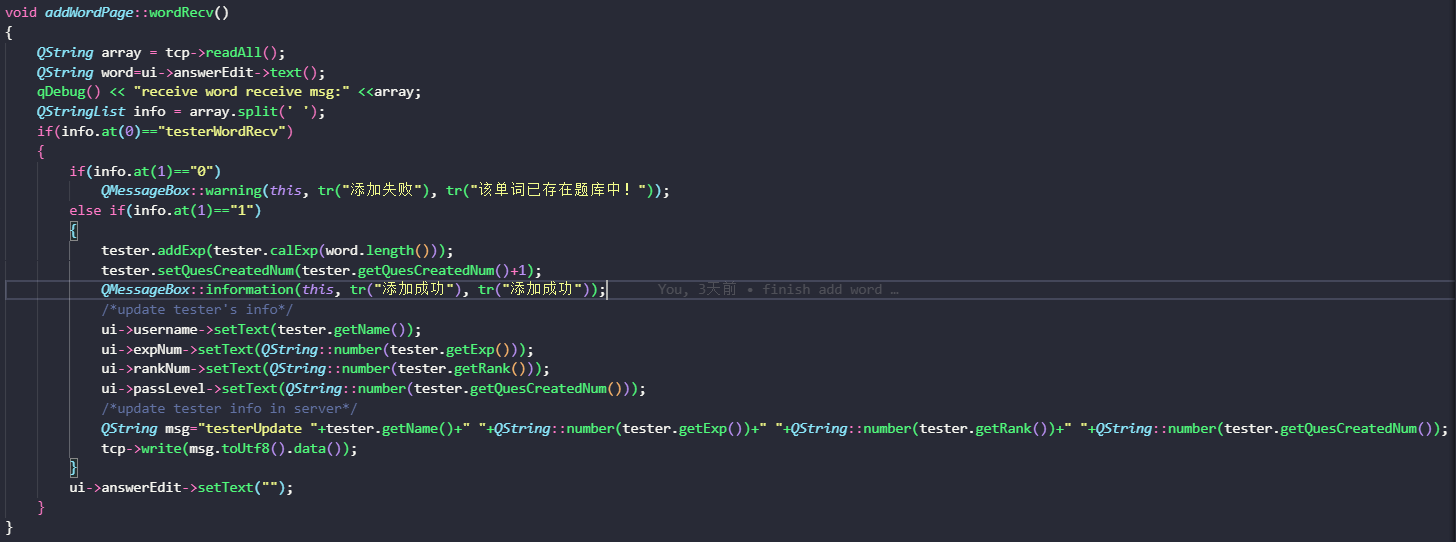
出题者查询和玩家查询使用同一界面，不多赘述。仅仅介绍出题界面实现方法。



Ⅰ.出题界面(实现于addwordpage.cpp addwordpage.h)

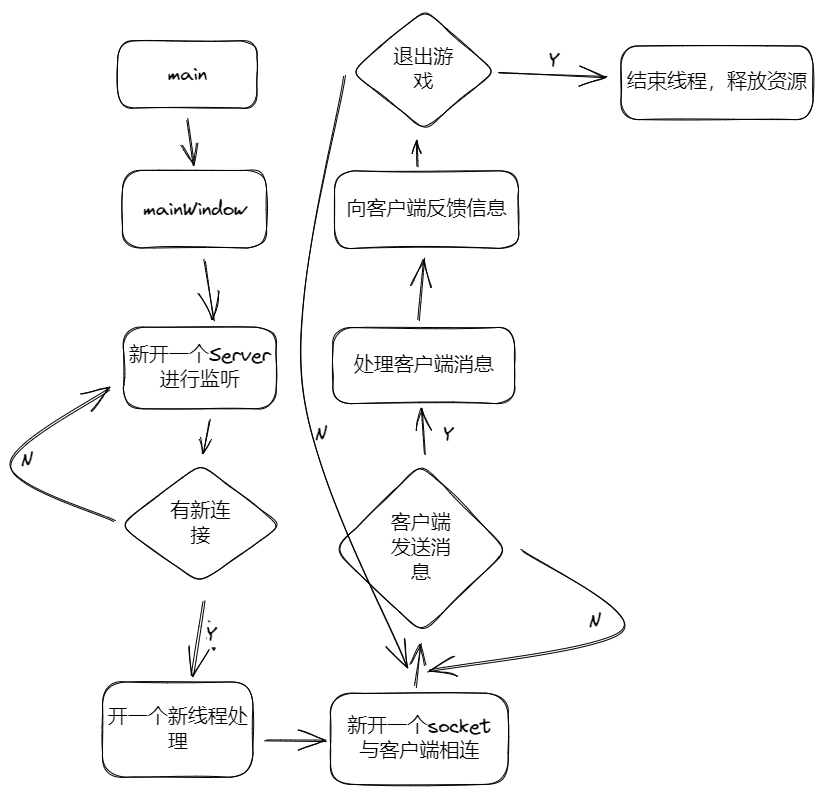


点击提交后将单词和难度发往服务端，接受服务端返回的数据。成功添加更新本地出题者数据，向服务端发送更新数据。



3.服务端功能实现

1.总体架构



2.数据库(实现于dbUtil.h dbUtil.cpp)

数据表如下设置：

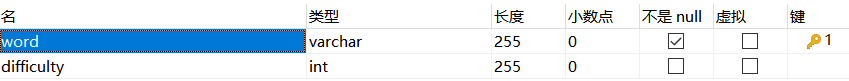
Player：



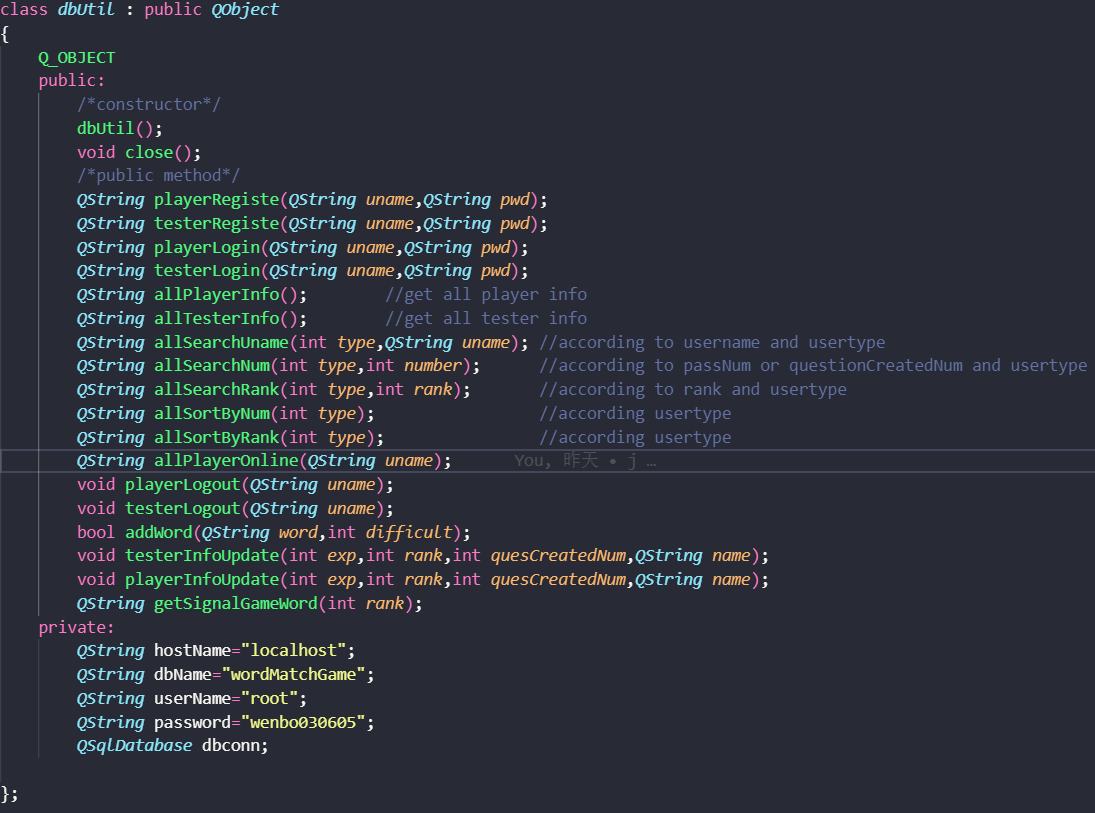
Tester：



Vocabulary：



这部分主要实现服务器对工程所有的数据的处理，从而实现对客户端的应答， 这部分通过 qt 的数据 mysql 接口完成，处理过程就是类似的数据库的管理操作 （插入、更新、删除、查找等等），这部分的所有功能全部封装在 dbUtil 这类中。 所有函数如下：（相关操作就不再赘述，全是数据库操作）

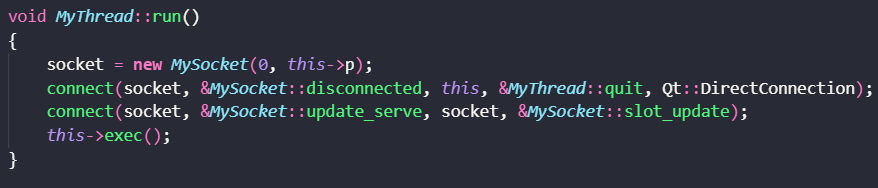


3 多线程

服务器端这边：myserver.cpp 会监听是否有客户端登录，如果有的话自动 调用这函数：virtual void incomingConnection(qintptr socketDescriptor); 只需要这个函数中为这个客户端新开一个进程就可以实现多客户端模式了，每个 客户端都会在单独的进程中运行，互不干扰。

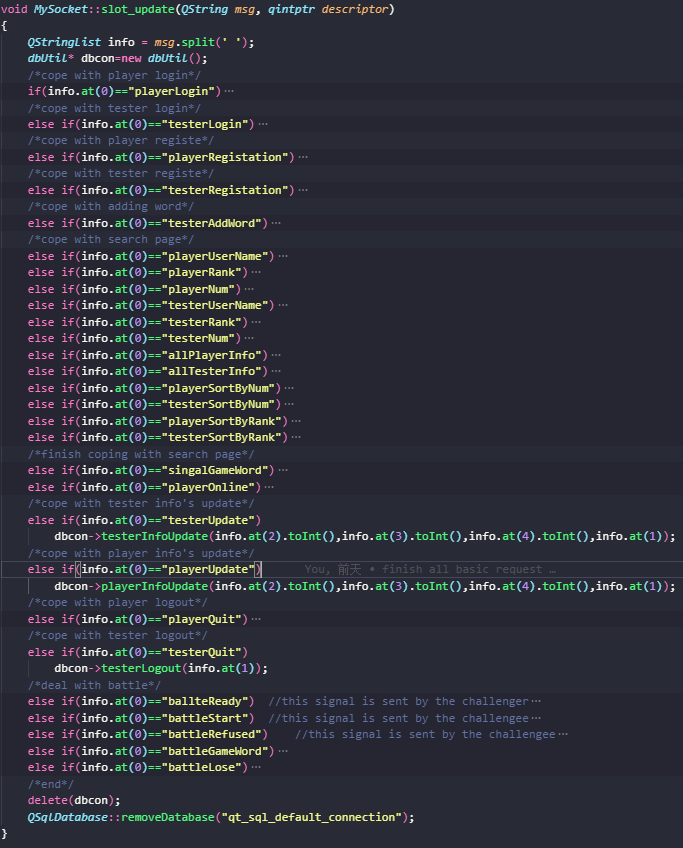


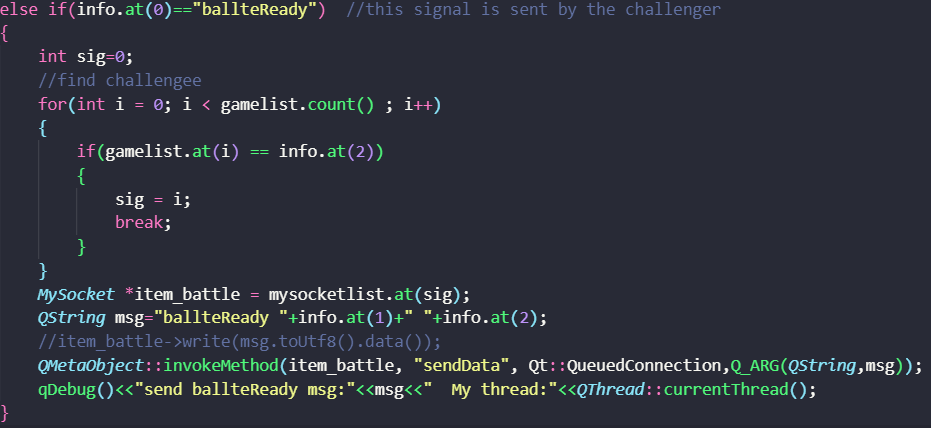
另外处理线程的函数 mythread.cpp，因为每次连接上一个客户端就会开辟 一个线程，在新开的线程中都会重写 void MyThread::run()，在里面会新开一个 socket 作为每个客户端通信用的工具：



4 通信格式

处理通信 socket 的函数是 mysocket.cpp，只要服务器接收到消息就会出发 void MySocket::slot\_update(QString msg)，这个更新函数，用来处理“通信格式”：以下是所有格式截图：



为了解释怎样处理通信格式，我举一个简单的例子：比如玩家A向玩家B发起对战请求。首先在全局mysocketlist中找到玩家B的socket，然后跨线程调用sendData，最终将这些信息写入通信套接字发给客户端。

五、实验总结

(一) 技术学习

1.练习了数据库增删改查

2.学习了如何在qt框架中使用了mysql（下载源码编译对应动态链接库）

3.学习了qt框架的使用，信号/槽机制的练习

4.跨线程调用QTcpSocket::write

(二) 收获

通过这次的程序设计实践，我巩固了许多有关 C++ 的知识，包括类的封装、继承、虚函数、重载等等。这进一步加深了我对“面向对象”编程范式的理解，即通过对特定对象的操作来实现预期的效果。这种编程思想在软件开发中非常重要，因为它通过对象连接程序和信息，从而使程序可以从文件、数据库或其他数据源中获取相关信息，并将其赋值给对象，然后我们可以对这个对象进行操作以实现所需的功能。这种思想使得我们在编写程序、设计软件时更加清晰地思考，以更高效、更普遍的方式编写出可维护、可扩展、可重用的软件。除此之外，通过这次实践，我也深入了解了客户端-服务器模式的通信以及多线程编程。在这种模式下，客户端向服务器发送请求，服务器作出相应的响应，并通过套接字 socket 进行通信。总之，这次实践让我从多个方面提高了编程技能和理解，也让我更加深入地认识了面向对象编程和客户端-服务器通信模式的重要性。