重庆大学大数据与软件学院

上 机 实 验 报 告

|  |  |
| --- | --- |
| 上机实践项目 | ATM程序 |
| 课程名称 | 计算机网络 |

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 杨家祥、许诺 | 成绩 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 学号 | 20221561、20221567 | 教师 | **胡海波** |

|  |  |  |  |
| --- | --- | --- | --- |
| 班级 | 软件工程4班 | 日期 | 2024/5/17 |

**《计算机网络》上机实验报告**

**开课实验室： 2024 年 5 月 17 日**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓 名 | | 杨家祥、许诺 | 年级、班级 | 2022级软件工程04班 | 学号 | **20221561、20221567** |
| 上机（项目）名称 | | | ATM程序编写 | | 指导教师 | 胡海波 |
| 教师评语 | 教师签名：  **年 月 日** | | | | | |

**一、上机目的**

套接字编程 目标：学习如何构建最简单的、基于C/S模式的通信程序

套接字：在主机端、由应用进程创建、由操作系统控制的接口，一个应用进程可以向另一个进程发送或接收报文

遵照RFC-20222022,使用Python编程语言设计一个ATM程序，拥有服务器端和客户端:

客户端：需要GUI

服务器端：不需要GUI，但需要读取数据文件、记录日志；默认端口号2525

**二、基本原理**

套接字编程

对应于传输层的服务，有两种类型的套接字

TCP：可靠的，面向字节流的传输，封装为TCP报文段(segment)

UDP：不可靠的传输，封装为UDP用户数据报(user datagram)

**三、使用的软件、硬件**

软件：VScode

硬件：Windows-PC

1. **上机操作步骤**

服务器端：

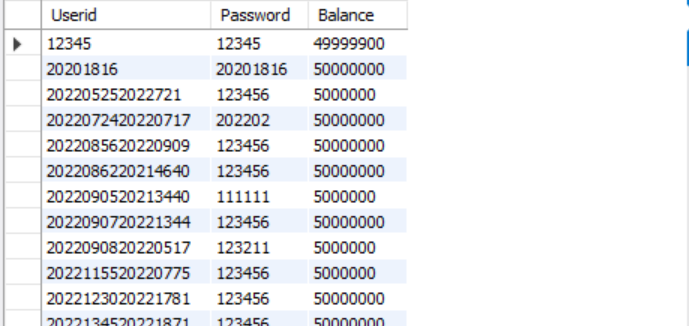
1. 数据库创建，将原始银行账户id，密码，余额等信息录入数据库
2. 创建基于TCP的套接字，用于接受客户端发来的信息，并进行响应
3. 基于客户端发送的报文进行判断，对其进行数据修改，以及结果回馈
4. 在进行数据修改的过程中，记录事件为日志

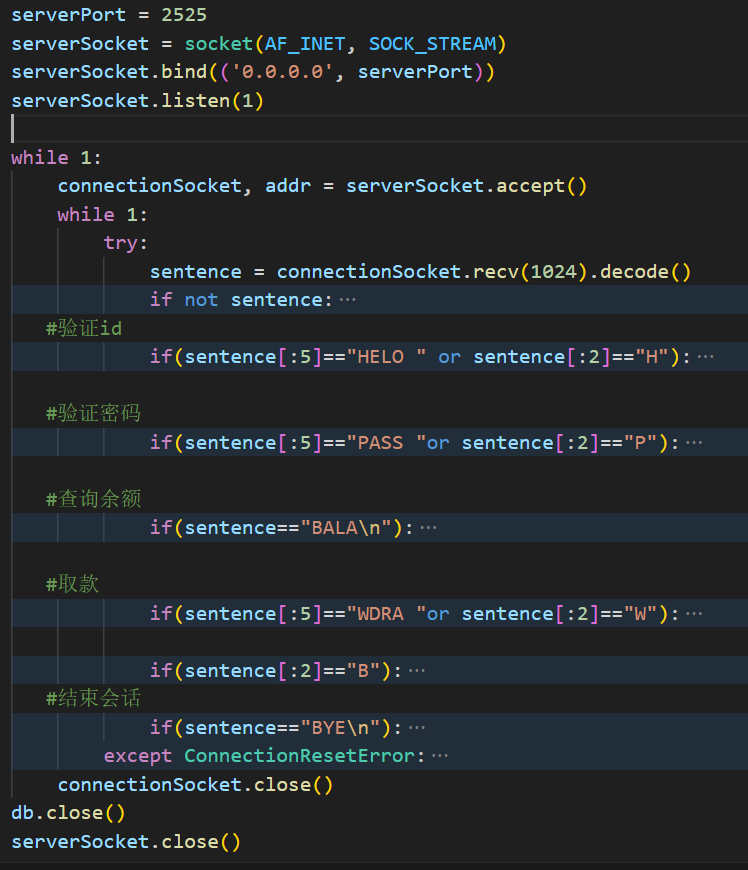
客户端：

1. 使用pyQT制作界面UI
2. 编写登录、余额查询、取款、退出函数
3. 绑定UI按钮和函数
4. 绑定服务器端，创建连接
5. **过程原始记录及分析(数据、图表、计算等)**

服务器端：

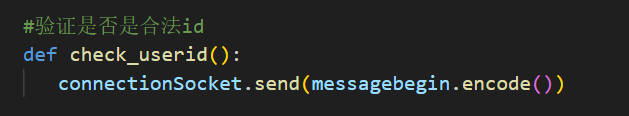
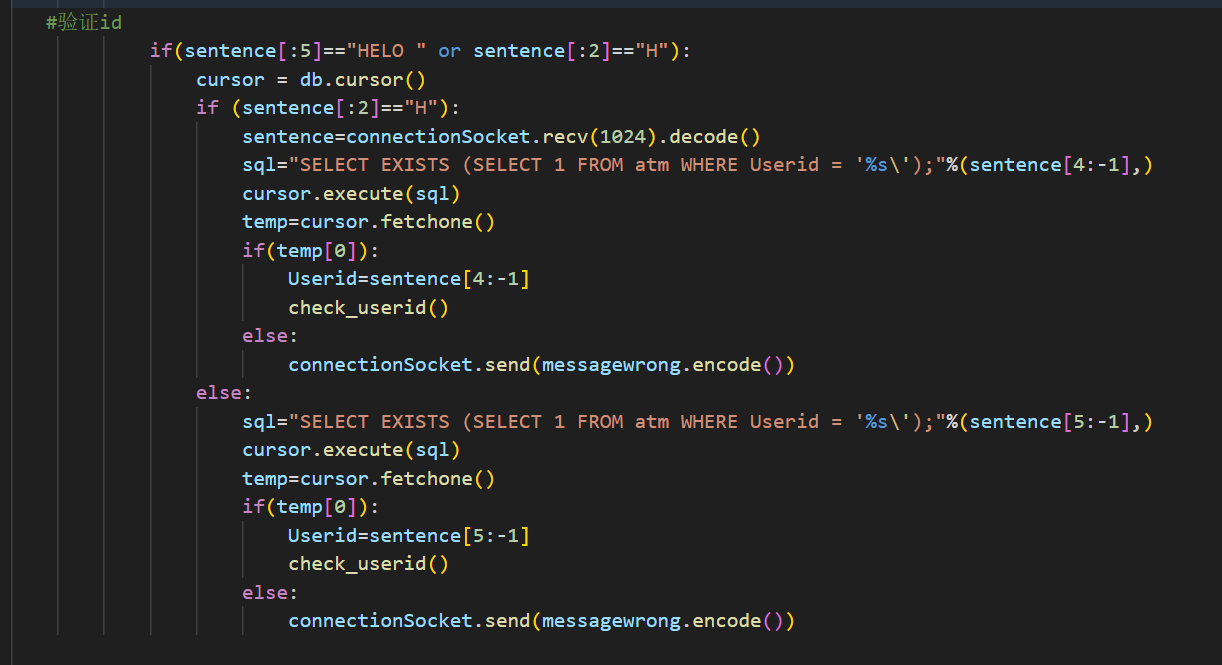
1. 利用MYSQL，将数据导入预先创建好的表，并将数据库与程序进行连接



1. 创建基于TCP的套接字，循环等待服务器端的连接，并循环接收本次连接的服务器端的发送的报文

对客户端发送的报文进行判断状态码，为HELO，进入验证id分支：

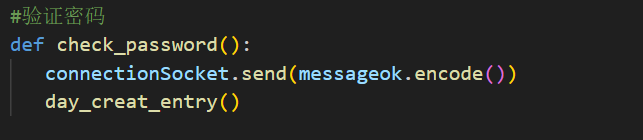
将收到的报文进行拆分，将获得的id记录，进数据库查找，若无对应id则发送401 ERROR，若有则发送500 AUTH REQUIRE，并继续接受后续报文

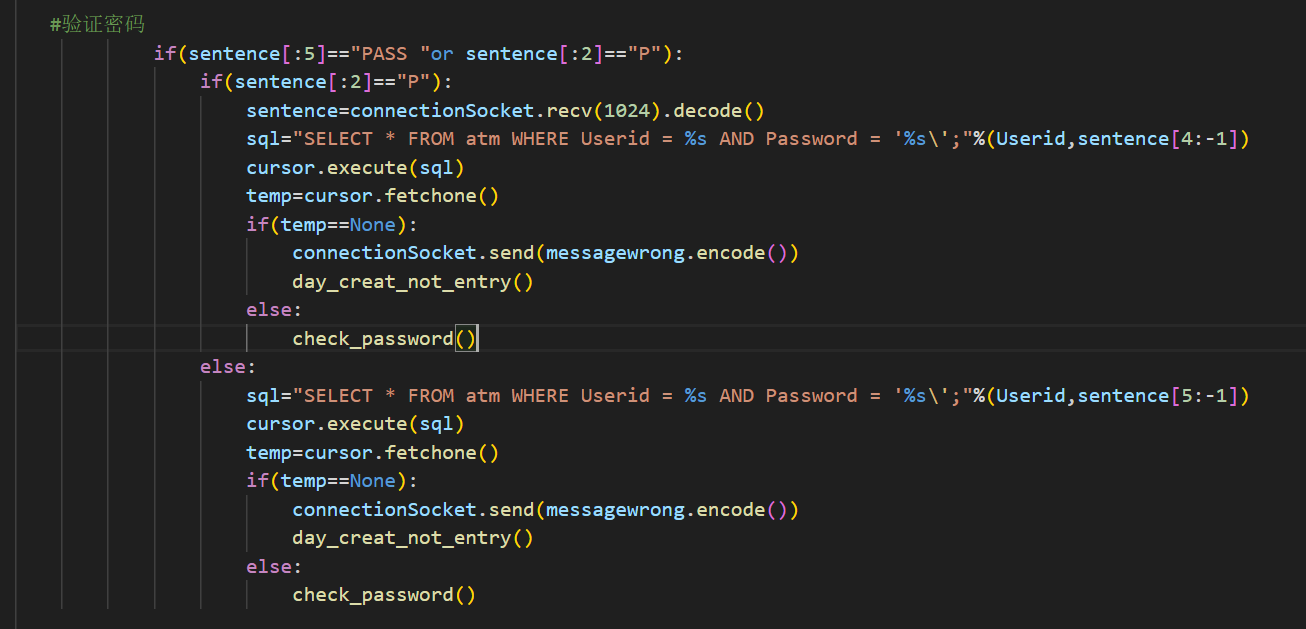


对客户端发送的报文进行判断状态码，为PASS，进入验证密码分支：

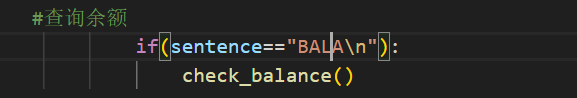
将收到的报文进行拆分，将获得的password记录，与之前记录的id对应进入数据库查询，若id

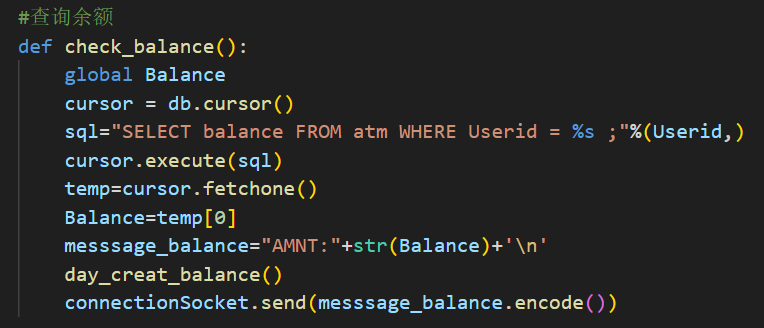
和password不符合，则返回报文401 ERROR，并将其记录为日志存储，反之进入功能页面，并返回报文525 OK，并继续接收后续报文



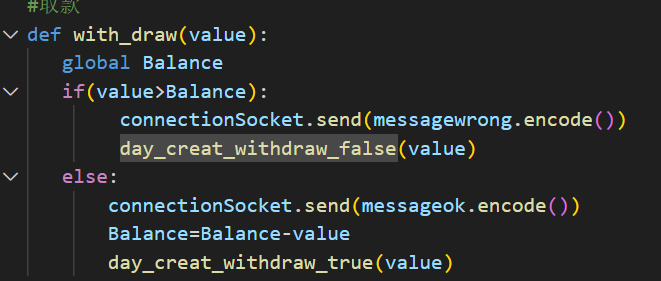
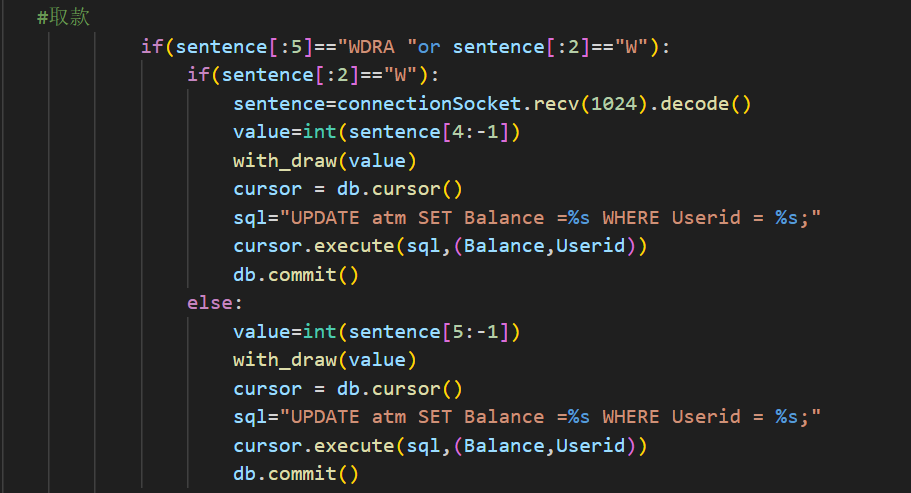


对客户端发送的报文进行判断状态码，为BALA，则进入查询余额分支，调用check\_balance函数进行日志记录

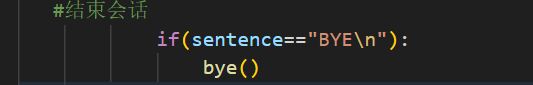


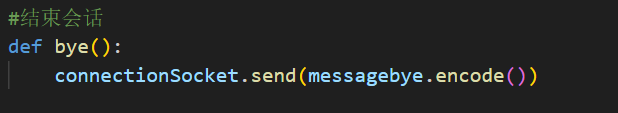
对客户端发送的报文进行分析状态码，为WDRA，则进入取款分支：

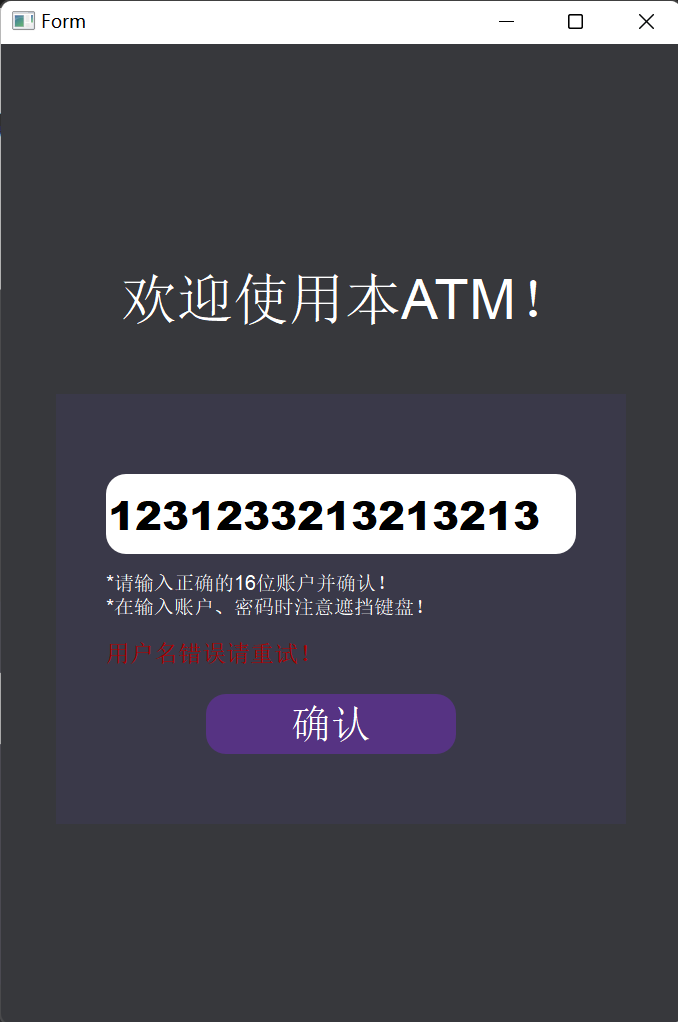
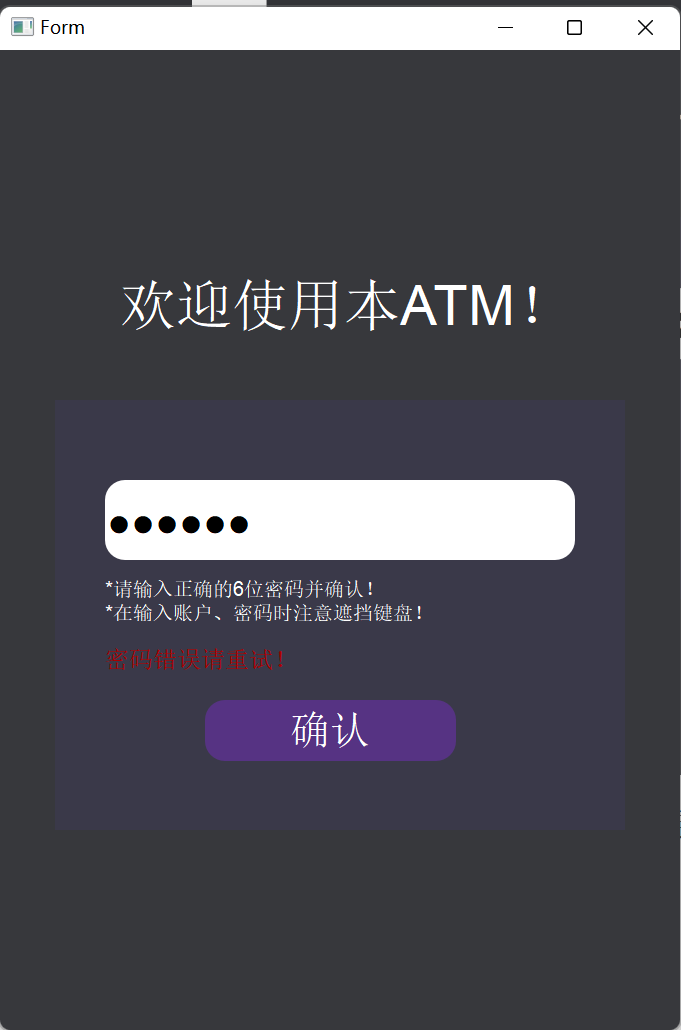
将报文继续拆分，获得用户希望取得的金额value，通过之前记录的id进数据库查询，若value小于余额则取款成功，对数据库进行数据改写，并反馈525 OK，并记录日志，若value对于余额，则取款失败401 ERROR，并记录日志

****

对客户端发送的报文进行判断状态码，为BYE，则进入结束会话分支：



客户端：

1. 建立连接，打开程序，输入用户名，如果用户名存在，则跳转输入密码界面，如果不存在，则提示用户名错误请重试。
2. 输入密码，如果密码正确，则跳转主界面，如果错误，则提示密码错误请重试。
3. 进入主界面，点击查询余额，则跳转余额界面；点击取款，则进入取款界面；点击退出，则退出程序，断开连接。



1. **问题及解决方法**

服务器端：

1.对于数据库的数据操作及查询并不能有效的进行

解决：创建了游标，并且在对数据库进行操作后进行了事务提交（do.commit）

2.在与一个客户端结束会话后，程序无法继续运行，无法接受后续客户端的连接

解决：创建了双层循环，外层用来循环接受客户端的连接，内层用来循环接收单次连接中客户端发送的报文，当与某一客户端结束会话后，结束内层循环

3.在与其他组进行交叉测试时，发现以Python和java两种编程语言编写的程序，在进行报文发送时，Python端只能接收到单次报文的头字母，若要获得后续报文，则需再次接受

解决：原理没有清楚，暴力增加判断语句，将所有情况考虑在内

客户端：

1.在与其他组进行交叉测试时，发现以Python和java两种编程语言编写的程序，在进行报文发送时，Python端只能接收到单次报文的头字母，若要获得后续报文，则需再次接受

解决：原理没有清楚，暴力增加判断语句，将所有情况考虑在内

1. **源码**

服务器端：

from socket import \*

from datetime import datetime

import pymysql

Userid='0'

global Balance

Balance=0

#连接数据库

db = pymysql.connect(

    host="localhost",

    port=3306,

    user='root',    #在这里输入用户名

    password='zasx1554',     #在这里输入密码

    charset='utf8mb4',

    database='computer'     #指定操作的数据库

    )

print ('The server is ready to receive')

#账户密码余额等基础信息

#日志的创建

def day\_creat\_entry():

    current\_time = datetime.now()

    with open('C:\\Users\\Administrator\\Desktop\\计算机网络\\实验\\ATM\\day\_record.txt', 'a') as file:

        file.write(str(current\_time)+' '+str(Userid)+' '+"进入使用了ATM机!\n")

    file.close()

def day\_creat\_not\_entry():

      current\_time = datetime.now()

      with open('C:\\Users\\Administrator\\Desktop\\计算机网络\\实验\\ATM\\day\_record.txt', 'a') as file:

         file.write(str(current\_time)+' '+str(Userid)+' '+"尝试进入使用ATM机,但是失败了\n")

      file.close()

def day\_creat\_balance():

      current\_time = datetime.now()

      with open('C:\\Users\\Administrator\\Desktop\\计算机网络\\实验\\ATM\\day\_record.txt', 'a') as file:

         file.write(str(current\_time)+' '+str(Userid)+' '+"查询了余额！\n")

      file.close()

def day\_creat\_withdraw\_true(value):

      current\_time = datetime.now()

      with open('C:\\Users\\Administrator\\Desktop\\计算机网络\\实验\\ATM\\day\_record.txt', 'a') as file:

         file.write(str(current\_time)+' '+str(Userid)+' '+"取出了"+str(value)+"\n")

      file.close()

def day\_creat\_withdraw\_false(value):

      current\_time = datetime.now()

      with open('C:\\Users\\Administrator\\Desktop\\计算机网络\\实验\\ATM\\day\_record.txt', 'a') as file:

         file.write(str(current\_time)+' '+str(Userid)+' '+"尝试取出"+str(value)+"但是失败了！\n")

      file.close()

#各种回复报文段

messagebegin="500 AUTH REQUIRE\n"

messagewrong="401 ERROR!\n"

messageok="525 OK!\n"

messagebye="BYE\n"

#验证是否是合法id

def check\_userid():

   connectionSocket.send(messagebegin.encode())

#验证密码

def check\_password():

   connectionSocket.send(messageok.encode())

   day\_creat\_entry()

#查询余额

def check\_balance():

    global Balance

    cursor = db.cursor()

    sql="SELECT balance FROM atm WHERE Userid = %s ;"%(Userid,)

    cursor.execute(sql)

    temp=cursor.fetchone()

    Balance=temp[0]

    messsage\_balance="AMNT:"+str(Balance)+'\n'

    day\_creat\_balance()

    connectionSocket.send(messsage\_balance.encode())

#取款

def with\_draw(value):

   global Balance

   if(value>Balance):

        connectionSocket.send(messagewrong.encode())

        day\_creat\_withdraw\_false(value)

   else:

       connectionSocket.send(messageok.encode())

       Balance=Balance-value

       day\_creat\_withdraw\_true(value)

#结束会话

def bye():

    connectionSocket.send(messagebye.encode())

serverPort = 2525

serverSocket = socket(AF\_INET, SOCK\_STREAM)

serverSocket.bind(('0.0.0.0', serverPort))

serverSocket.listen(1)

while 1:

    connectionSocket, addr = serverSocket.accept()

    while 1:

        try:

            sentence = connectionSocket.recv(1024).decode()

            if not sentence:

                break

   #验证id

            if(sentence[:5]=="HELO " or sentence[:2]=="H"):

                cursor = db.cursor()

                if (sentence[:2]=="H"):

                    sentence=connectionSocket.recv(1024).decode()

                    sql="SELECT EXISTS (SELECT 1 FROM atm WHERE Userid = '%s\');"%(sentence[4:-1],)

                    cursor.execute(sql)

                    temp=cursor.fetchone()

                    if(temp[0]):

                        Userid=sentence[4:-1]

                        check\_userid()

                    else:

                        connectionSocket.send(messagewrong.encode())

                else:

                    sql="SELECT EXISTS (SELECT 1 FROM atm WHERE Userid = '%s\');"%(sentence[5:-1],)

                    cursor.execute(sql)

                    temp=cursor.fetchone()

                    if(temp[0]):

                        Userid=sentence[5:-1]

                        check\_userid()

                    else:

                        connectionSocket.send(messagewrong.encode())

   #验证密码

            if(sentence[:5]=="PASS "or sentence[:2]=="P"):

                if(sentence[:2]=="P"):

                    sentence=connectionSocket.recv(1024).decode()

                    sql="SELECT \* FROM atm WHERE Userid = %s AND Password = '%s\';"%(Userid,sentence[4:-1])

                    cursor.execute(sql)

                    temp=cursor.fetchone()

                    if(temp==None):

                        connectionSocket.send(messagewrong.encode())

                        day\_creat\_not\_entry()

                    else:

                        check\_password()

                else:

                    sql="SELECT \* FROM atm WHERE Userid = %s AND Password = '%s\';"%(Userid,sentence[5:-1])

                    cursor.execute(sql)

                    temp=cursor.fetchone()

                    if(temp==None):

                        connectionSocket.send(messagewrong.encode())

                        day\_creat\_not\_entry()

                    else:

                        check\_password()

   #查询余额

            if(sentence=="BALA\n"):

                check\_balance()

   #取款

            if(sentence[:5]=="WDRA "or sentence[:2]=="W"):

                if(sentence[:2]=="W"):

                    sentence=connectionSocket.recv(1024).decode()

                    value=int(sentence[4:-1])

                    with\_draw(value)

                    cursor = db.cursor()

                    sql="UPDATE atm SET Balance =%s WHERE Userid = %s;"

                    cursor.execute(sql,(Balance,Userid))

                    db.commit()

                else:

                    value=int(sentence[5:-1])

                    with\_draw(value)

                    cursor = db.cursor()

                    sql="UPDATE atm SET Balance =%s WHERE Userid = %s;"

                    cursor.execute(sql,(Balance,Userid))

                    db.commit()

            if(sentence[:2]=="B"):

                sentence=connectionSocket.recv(1024).decode()

                if(sentence[:1]=="A"):

                    check\_balance()

                else:

                    bye()

   #结束会话

            if(sentence=="BYE\n"):

                bye()

        except ConnectionResetError:

            print("ConnectionResetError: 远程主机强制关闭了连接")

            break

    connectionSocket.close()

db.close()

serverSocket.close()

客户端：

from PyQt5 import uic  
from PyQt5.QtWidgets import QApplication  
import socket  
from PyQt5.QtGui import QColor  
  
# 确定服务器IP以及Port  
serverName = '10.234.107.112'  
serverPort = 2525  
  
client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
client\_socket.connect((serverName, serverPort))  
class window:  
  
 def \_\_init\_\_(self):  
 # 从文件中加载UI  
 self.password\_ui = uic.loadUi("password\_input.ui")  
 self.password\_ui.pushButton\_2.clicked.connect(self.password\_verification)  
  
 self.userid\_ui = uic.loadUi("userid\_input.ui")  
 self.userid\_ui.pushButton\_1.clicked.connect(self.userid\_verification)  
  
 self.main\_ui = uic.loadUi("main\_window.ui")  
 self.main\_ui.pushButton\_3.clicked.connect(self.balance\_inquiry)  
 self.main\_ui.pushButton\_4.clicked.connect(self.withdraw\_ui)  
 self.main\_ui.pushButton\_5.clicked.connect(self.exit)  
  
 self.withdraw\_ui = uic.loadUi("withdraw.ui")  
 self.withdraw\_ui.pushButton\_7.clicked.connect(self.withdraw)  
 self.withdraw\_ui.pushButton\_8.clicked.connect(self.back1)  
  
 self.balance\_ui = uic.loadUi("balance.ui")  
 self.balance\_ui.pushButton\_6.clicked.connect(self.back2)  
  
 # userid验证  
 def userid\_verification(self):  
 userid = self.userid\_ui.userid\_lineedit.text()  
 message = "HELO " + userid + '\n'  
  
 # 向服务器发送请求  
 client\_socket.send(message.encode())  
 response1 = client\_socket.recv(1024).decode()  
 if response1 == '5':  
 response2 = client\_socket.recv(1024).decode()  
 response = response1 + response2  
 print(response)  
  
 # 如果验证成功，跳转输入password界面,否则提示userid非法！  
 if response == ("500 AUTH REQUIRE" + '\n'):  
 self.userid\_ui.label\_3.clear()  
 self.userid\_ui.close()  
 self.password\_ui.show()  
 else:  
 self.userid\_ui.label\_3.clear()  
 self.userid\_ui.label\_3.setText("用户名错误请重试！")  
 else:  
 print(response1)  
 if response1 == ("500 AUTH REQUIRE" + '\n'):  
 self.userid\_ui.label\_3.clear()  
 self.userid\_ui.close()  
 self.password\_ui.show()  
 else:  
 self.userid\_ui.label\_3.clear()  
 self.userid\_ui.label\_3.setText("用户名错误请重试！")  
  
 # password验证  
 def password\_verification(self):  
 password = self.password\_ui.password\_lineedit.text()  
 message = "PASS " + password + '\n'  
  
 client\_socket.send(message.encode())  
 response1 = client\_socket.recv(1024).decode()  
 print(response1)  
 if response1 == '5':  
 response2 = client\_socket.recv(1024).decode()  
 response = response1 + response2  
  
  
 if response == ("525 OK!" + '\n'):  
 self.password\_ui.label\_3.clear()  
 self.password\_ui.close()  
 self.main\_ui.show()  
 else:  
 self.password\_ui.label\_3.clear()  
 self.password\_ui.label\_3.setText("密码错误请重试！")  
 else:  
 if response1 == ("525 OK!" + '\n'):  
 self.password\_ui.label\_3.clear()  
 self.password\_ui.close()  
 self.main\_ui.show()  
 else:  
 self.password\_ui.label\_3.clear()  
 self.password\_ui.label\_3.setText("密码错误请重试！")  
  
 # 查询余额  
 def balance\_inquiry(self):  
 message = "BALA" + '\n'  
  
 client\_socket.send(message.encode())  
 orgin\_balance1 = client\_socket.recv(1024).decode()  
 if orgin\_balance1 == 'A':  
 orgin\_balance2 = client\_socket.recv(1024).decode()  
 orgin\_balance = orgin\_balance1 + orgin\_balance2  
 balance = orgin\_balance.split(':')[1]  
  
 self.balance\_ui.show()  
 self.balance\_ui.label\_2.setText(f'该账户可用余额为：\n\n {balance}元')  
 self.main\_ui.hide()  
 else:  
 if 'AMNT:' in orgin\_balance1:  
 balance = orgin\_balance1.split(':')[1]  
  
 self.balance\_ui.show()  
 self.balance\_ui.label\_2.setText(f'该账户可用余额为：\n\n {balance}元')  
 self.main\_ui.hide()  
  
 def withdraw\_ui(self):  
 self.withdraw\_ui.show()  
 self.main\_ui.hide()  
  
 def exit(self):  
 message = "BYE" + '\n'  
  
 client\_socket.send(message.encode())  
 response1 = client\_socket.recv(1024).decode()  
 if response1 == 'B':  
 response2 = client\_socket.recv(1024).decode()  
 response = response1 + response2  
  
 if response == ("BYE" + '\n'):  
 client\_socket.close()  
 self.main\_ui.close()  
 else:  
 if response1 == ("BYE" + '\n'):  
 client\_socket.close()  
 self.main\_ui.close()  
  
 def back(self):  
 self.main\_ui.show()  
 self.balance\_ui.close()  
  
 # 取款  
 def withdraw(self):  
 amount = self.withdraw\_ui.amount\_lineedit.text()  
 message = "WDRA " + amount + '\n'  
  
 client\_socket.send(message.encode())  
 response1 = client\_socket.recv(1024).decode()  
 if response1 == '5':  
 response2 = client\_socket.recv(1024).decode()  
 response = response1 + response2  
  
 if response == ("525 OK!" + '\n'):  
 self.withdraw\_ui.label\_3.setText("取款成功！")  
 text\_color1 = QColor("green")  
 self.withdraw\_ui.label\_3.setStyleSheet("color: {}".format(text\_color1.name()))  
 self.withdraw\_ui.amount\_lineedit.clear()  
 else:  
 self.withdraw\_ui.label\_3.setText("取款失败或余额不足！")  
 text\_color2 = QColor("red")  
 self.withdraw\_ui.label\_3.setStyleSheet("color: {}".format(text\_color2.name()))  
 self.withdraw\_ui.amount\_lineedit.clear()  
 else:  
 if response1 == ("525 OK!" + '\n'):  
 self.withdraw\_ui.label\_3.setText("取款成功！")  
 text\_color1 = QColor("green")  
 self.withdraw\_ui.label\_3.setStyleSheet("color: {}".format(text\_color1.name()))  
 self.withdraw\_ui.amount\_lineedit.clear()  
 else:  
 self.withdraw\_ui.label\_3.setText("取款失败或余额不足！")  
 text\_color2 = QColor("red")  
 self.withdraw\_ui.label\_3.setStyleSheet("color: {}".format(text\_color2.name()))  
 self.withdraw\_ui.amount\_lineedit.clear()  
  
 def back1(self):  
 self.withdraw\_ui.close()  
 self.main\_ui.show()  
  
 def back2(self):  
 self.balance\_ui.close()  
 self.main\_ui.show()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QApplication([])  
 user\_window = window()  
 user\_window.userid\_ui.show()  
 app.exec\_()