重庆大学大数据与软件学院

作 业 报 告

|  |  |
| --- | --- |
| 实践项目 | ATM模拟编程作业 |
| 课程名称 | **计算机网络** |

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 郑桂桂 | 学号 | 20231478 |
| 姓名 | 汤燕妮 | 学号 | 20231474 |

|  |  |  |  |
| --- | --- | --- | --- |
| 教师 | 胡海波 | 班级 | 094003-002 |

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 2025.05.25 | 成绩 |  |

# 目的

通过作业二使同学们掌握所学TCP套接字编程方法，理解TCP套接字在服务器和客户端之间的应用层通信的流程和形式，巩固计算机网络的理论和知识。此外，通过作业的开展使同学们提高计算机网络程序设计和开发能力，熟悉所用编程语言的开发环境及调试过程，熟悉所用语言中的数据类型、数据结构、语法结构、运算方法等。基于作业一的内容进行程序设计开发，同学们也可以体会计算机网络通信协议(应用层)的制定、标准化等工作对实际应用的价值和意义。

# 作业内容

要求两位同学组队(参考石墨文档1-作业2分组)，基于模拟的应用层通信协议标准(参考石墨文档2-“RFC20232023”)完成ATM机客户端和服务器端的程序设计和开发。程序语言不限。

客户端具备GUI可以模拟ATM机的账户登录和存款取款等功能，同时应该包括错误信息的提示。服务器端应该需要通过后台数据库维护账户信息、具备记录日志的功能。

各组同学独立完成，不能抄袭。作业结束时，每组需提交作业报告，其中包含程序设计思路和软件源代码。

里程碑1：第2次实验课，完成组内程序设计开发和测试，尝试连接其他小组的服务器端。

里程碑2：第3次实验课，分析解决里程碑1中的问题，完成程序调试，通过测试脚本完成对其他所有小组服务器端的连接，记录结果。

里程碑3：第4次实验课，继续调试程序，最终提交文档、代码。

# 运行环境

//在本节介绍使用的系统环境、编码环境、程序语言等内容，包括TCP Socket 相关的理论和知识。

**操作系统**：Windows 11；

**程序语言**：Python；

**编码环境**: PyCharm 2023.3.7。

**相关知识**：

TCP Socket 是一种基于 TCP/IP 协议的网络通信机制，用于实现不同计算机之间的数据交换。其基本原理包括以下几个方面：

①IP地址和端口号：每台计算机都有唯一的 IP 地址，用于标识不同的计算机。而端口号则用于区分一个计算机上不同的进程或应用程序。

②三次握手建立连接：在进行 TCP Socket 通信之前，需要通过三次握手协议来建立连接。这个过程包括客户端向服务器发送请求，服务器接受请求并回复确认，最后客户端再回复确认。

③数据传输：TCP Socket 会将要传输的数据进行分段，并且对每个分段进行编号和校验，以保证数据的可靠性和完整性。同时，还会对数据进行流量控制和拥塞控制，确保网络质量和带宽合理利用。

④四次挥手断开连接：当数据传输完成后，需要通过四次挥手协议来断开连接。这个过程包括客户端向服务器发送请求断开连接，服务器回复确认，服务器向客户端发送断开连接请求，客户端回复确认，最终完成断开连接的操作。

# 程序设计

## 4.1设计思路

//在本节介绍程序的流程、客户端和服务器端的交互流程图等

图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。

图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。

表格

AI 生成的内容可能不正确。

## 4.2服务器端

//服务器端源代码，应加以文字进行简要说明

import socket  
import threading  
import logging  
import json  
  
# 配置日志  
logging.basicConfig(filename='server.log', level=logging.INFO,  
 format='%(asctime)s - %(levelname)s - %(message)s')  
  
# 用户数据文件  
USER\_FILE = 'users.json'  
  
# 加载用户数据  
def load\_users():  
 with open(USER\_FILE, 'r') as f:  
 return json.load(f)  
  
  
# 保存用户数据  
def save\_users(users):  
 with open(USER\_FILE, 'w') as f:  
 json.dump(users, f, indent=4)  
  
  
# 验证用户 ID 是否存在  
def is\_user\_exists(userid, users):  
 return userid in users  
  
  
# 初始化用户数据  
users = load\_users()  
  
  
def handle\_client(conn, addr):  
 logging.info(f"Connected by {addr}")  
 current\_user = None  
  
 try:  
 # 接收客户端发送的 HELO 请求  
 data = conn.recv(1024).decode().strip()  
 if data.startswith("HELO"):  
 userid = data.split()[1] # 提取 user1  
 logging.info(f"User ID received: {userid}")  
  
 # 验证用户 ID 是否存在  
 if is\_user\_exists(userid, users):  
 current\_user = userid  
 conn.sendall(b"500 AUTH REQUIRE")  
  
 # 接收客户端发送的 PASS 请求  
 data = conn.recv(1024).decode().strip()  
 if data.startswith("PASS"):  
 password = data.split()[1] # 提取 password  
 logging.info(f"Password {password}received for user {userid}")  
  
 # 验证密码  
 if password == users[userid]['password']:  
 conn.sendall(b"525 OK!")  
 logging.info(f"User {userid} authenticated successfully")  
  
 # 处理客户端请求  
 while True:  
 data = conn.recv(1024).decode().strip()  
  
 if not data: # 如果接收到空数据，表示客户端断开连接  
 logging.info(f"Client {addr} disconnected")  
 break  
 logging.info(f"Request received: {data}")  
  
 if data == "BALA":  
 # 查询余额  
 balance = users[userid]['balance']  
 conn.sendall(f"AMNT:{balance}\n".encode())  
 logging.info(f"Balance queried for user {userid}: {balance}")  
 elif data.startswith("WDRA"):  
 # 处理取款请求  
 try:  
 amount = int(data.split()[1])  
 if amount <= users[userid]['balance']:  
 users[userid]['balance'] -= amount  
 conn.sendall(b"525 OK!")  
 logging.info(  
 f"Withdrawal successful for user {userid}. Amount: {amount}, New Balance: {users[userid]['balance']}")  
 save\_users(users) # 更新用户数据文件  
 else:  
 conn.sendall(b"401 ERROR!")  
 logging.warning(  
 f"Insufficient balance for user {userid}. Requested amount: {amount}")  
 except (IndexError, ValueError):  
 conn.sendall(b"401 ERROR!")  
 logging.warning(f"Invalid WDRA request format: {data}")  
 elif data == "BYE":  
 logging.info(f"User {userid} disconnected")  
 conn.sendall(b"BYE")  
 break  
 else:  
 conn.sendall(b"401 ERROR!")  
 logging.warning(f"Unknown request: {data}")  
 else:  
 conn.sendall(b"401 ERROR!")  
 logging.warning(f"Authentication failed for user {userid}")  
 else:  
 conn.sendall(b"401 ERROR!")  
 logging.warning(f"Invalid password format for user {userid}")  
 else:  
 conn.sendall(b"401 ERROR!")  
 logging.warning(f"Invalid user ID: {userid}")  
 finally:  
 conn.close()  
 logging.info(f"Disconnected by {addr}")  
  
  
def start\_server():  
 with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:  
 s.bind(('0.0.0.0', 2525))  
 s.listen()  
 logging.info("Server started on port 2525")  
 while True:  
 conn, addr = s.accept()  
 threading.Thread(target=handle\_client, args=(conn, addr)).start()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 start\_server()

## 4.3客户端

import tkinter as tk  
from tkinter import messagebox  
import socket  
import logging  
  
# 配置日志记录  
logging.basicConfig(  
 level=logging.INFO,  
 format='%(asctime)s - %(levelname)s - %(message)s',  
 filename='atm\_client.log',  
 filemode='a'  
)  
  
client\_socket = None  
  
  
def send\_login():  
 global client\_socket  
 userid = userid\_entry.get()  
 password = password\_entry.get()  
  
 logging.info(f"Attempting to login with User ID: {userid}")  
  
 try:  
 client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
 client\_socket.connect(('192.168.138.161', 2525))  
 client\_socket.sendall(f"HELO {userid}".encode())  
 response = client\_socket.recv(1024).decode()  
  
 if response == "500 AUTH REQUIRE":  
 logging.info("Authentication required. Sending password.")  
 client\_socket.sendall(f"PASS {password}".encode())  
 response = client\_socket.recv(1024).decode()  
  
 if response == "525 OK!":  
 logging.info("Login successful.")  
 messagebox.showinfo("Login", "Login successful!")  
 show\_operations() # 显示操作界面  
 else:  
 logging.error("Invalid password.")  
 messagebox.showerror("Login", "Invalid password!")  
 else:  
 logging.error("Invalid user ID.",response)  
 messagebox.showerror("Login", "Invalid user ID!")  
 except Exception as e:  
 logging.error(f"Login failed: {e}")  
 messagebox.showerror("Error", f"Connection error: {e}")  
  
def show\_operations():  
 # 隐藏登录界面  
 userid\_label.grid\_forget()  
 userid\_entry.grid\_forget()  
 password\_label.grid\_forget()  
 password\_entry.grid\_forget()  
 login\_button.grid\_forget()  
  
 # 显示操作界面  
 bala\_button.grid(row=0, column=0, padx=10, pady=10)  
 wdra\_button.grid(row=0, column=1, padx=10, pady=10)  
 amount\_label.grid(row=1, column=0)  
 amount\_entry.grid(row=1, column=1)  
 bye\_button.grid(row=2, column=0, columnspan=2, pady=10)  
  
def send\_request(request):  
 global client\_socket  
 if not client\_socket:  
 logging.error("No active connection.")  
 messagebox.showerror("Error", "Not connected to server.")  
 return  
  
 logging.info(f"Sending request: {request}")  
 try:  
 if request == "BALA":  
 client\_socket.sendall(b"BALA")  
 response = client\_socket.recv(1024).decode()  
 logging.info(f"Received balance response: {response}")  
 messagebox.showinfo("Balance", response)  
 elif request == "WDRA":  
 amount = amount\_entry.get()  
 client\_socket.sendall(f"WDRA {amount}".encode())  
 response = client\_socket.recv(1024).decode()  
 if response.startswith("525 OK!"):  
  
 logging.info(f"Withdrawal successful.")  
 messagebox.showinfo("Withdrawal", f"取钱成功!\n")  
 else:  
 logging.error(f"Withdrawal failed: {response}")  
 messagebox.showerror("Withdrawal", response)  
  
  
 elif request == "BYE":  
 client\_socket.sendall(b"BYE")  
 response = client\_socket.recv(1024).decode()  
 logging.info(f"Received exit response: {response}")  
 messagebox.showinfo("Exit", response)  
 client\_socket.close()  
 app.quit()  
 except Exception as e:  
 logging.error(f"Request failed: {e}")  
 messagebox.showerror("Error", f"Request error: {e}")  
  
app = tk.Tk()  
app.title("ATM Client")  
  
# 登录界面  
userid\_label = tk.Label(app, text="User ID:")  
userid\_label.grid(row=0, column=0, padx=10, pady=10)  
userid\_entry = tk.Entry(app)  
userid\_entry.grid(row=0, column=1, padx=10, pady=10)  
  
password\_label = tk.Label(app, text="Password:")  
password\_label.grid(row=1, column=0, padx=10, pady=10)  
password\_entry = tk.Entry(app, show="\*")  
password\_entry.grid(row=1, column=1, padx=10, pady=10)  
  
login\_button = tk.Button(app, text="Login", command=send\_login)  
login\_button.grid(row=2, column=0, columnspan=2, pady=10)  
  
# 操作界面  
bala\_button = tk.Button(app, text="Check Balance", command=lambda: send\_request("BALA"))  
wdra\_button = tk.Button(app, text="Withdraw", command=lambda: send\_request("WDRA"))  
amount\_label = tk.Label(app, text="Amount:")  
amount\_entry = tk.Entry(app)  
bye\_button = tk.Button(app, text="Exit", command=lambda: send\_request("BYE"))  
  
app.mainloop()

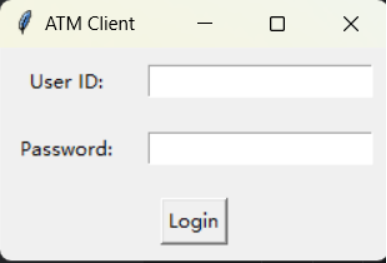
# 测试运行结果

//测试运行过程中的记录，包括截图和文字描述、问题分析、改进方案等。

## 5.1 组内测试

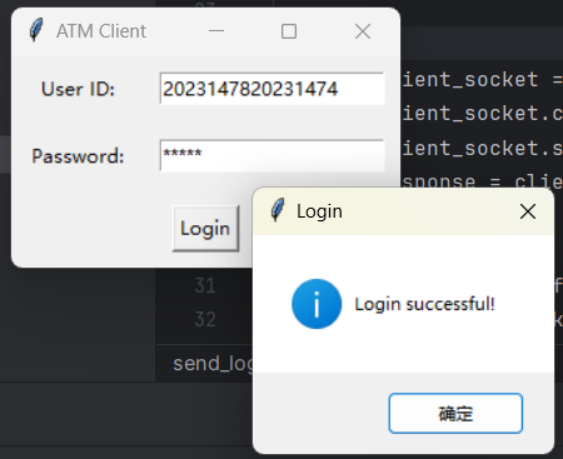
5.1.1 登录

5.1.1.1 登录界面



5.1.1.2 输入账号密码并成功连接

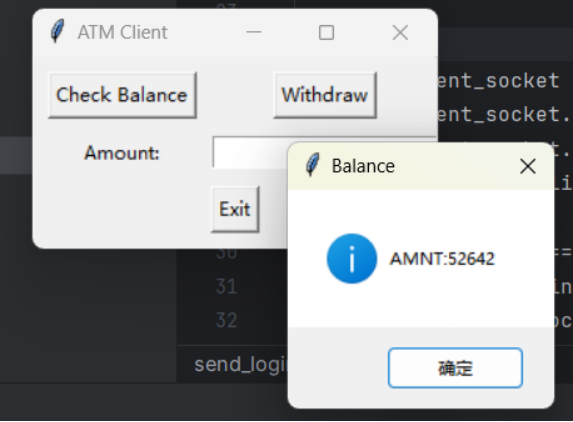
输入账号：“2023147820231474”和密码：“ pass1”,客户端显示“Login successful!”



5.1.2 取款

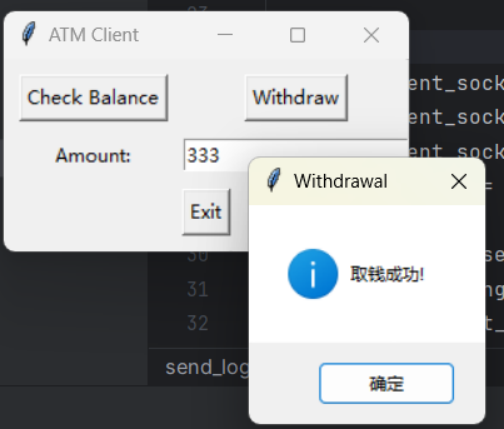
5.1.2.1 查询余额

点击“Check Balance”按钮,，查询余额为“52642 。



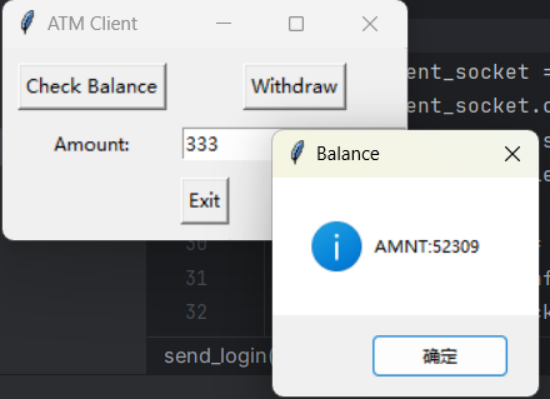
5.1.2.2 取款

在Amount处输入取款金额“333”，点击“Withdraw"，显示”取钱成功!"



5.1.2.3 取款成功

再次点击“Check Balance”按钮，查询余额为“52309 "。

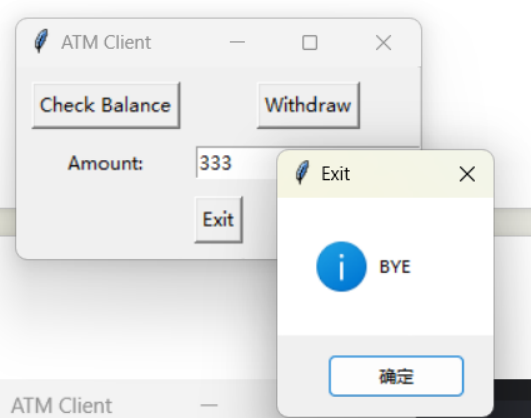


下面的存款余额正确更新 (52642-52309 =333)；

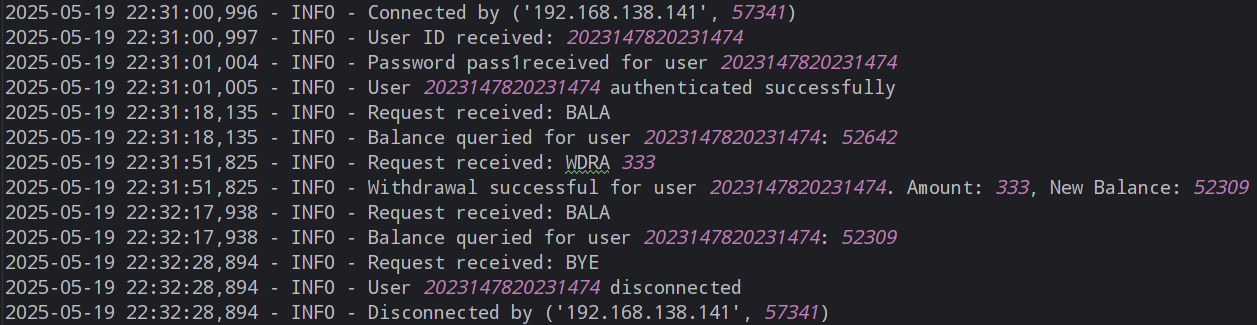
5.1..3 退出

在任意阶段点击“Exit”按钮即可退出该系统，关闭界面；

客户端有如下反馈：



5.1.4 日志文件



## 5.2 组间测试

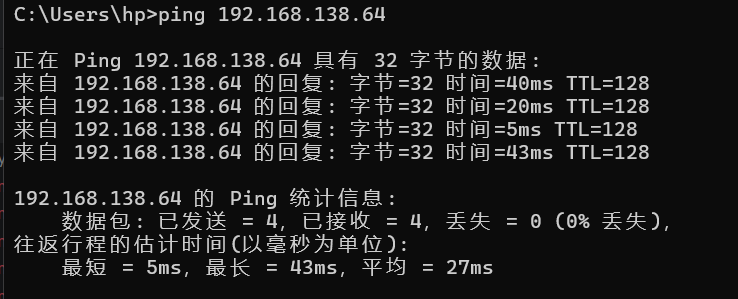
//在组内测试通过的基础上，可以在本组的客户端程序中写入自动化的测试脚本，连接其他小组的服务器进行测试。在测试结果的基础上逐渐改进客户端和服务器端程序。

5.2.1 测试对象：第24组 肖钦予20230906 陈子欣20230909

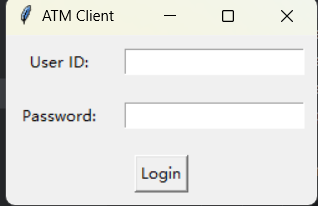


5.2.2 我方作为 Client，对方作为 Server

5.2.2.1 ping对方服务器



5.2.2.2 连接服务器

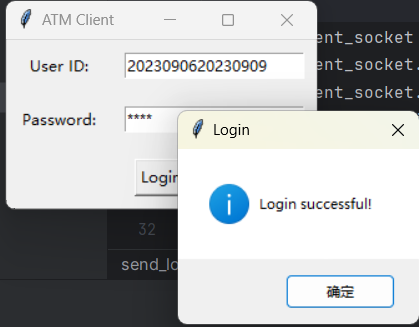
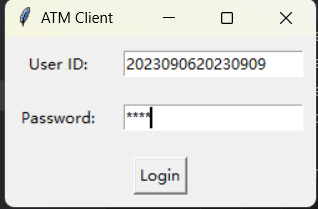


客户端运行后弹出登录界面，表明服务器连接成功，同时服务器端显示



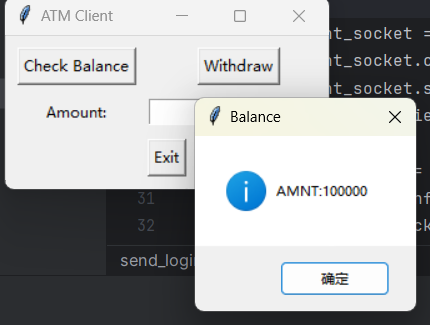
5.2.2.3 登录账号

客户端输入对方的账号（2023090620230909）和密码（1234）并登录，显示登陆成功



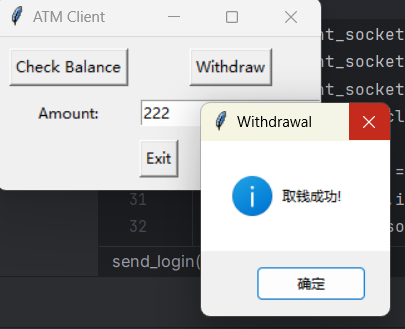
5.2.2.4 查询余额

查询余额，显示余额100000元

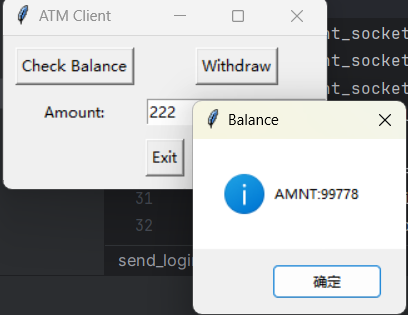


5.2.2.5 取款

客户端取款222元，并显示取款成功

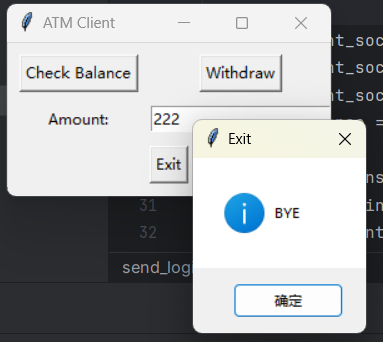


再次查询余额，显示99778元



5.2.2.5 退出登录

成功退出登录



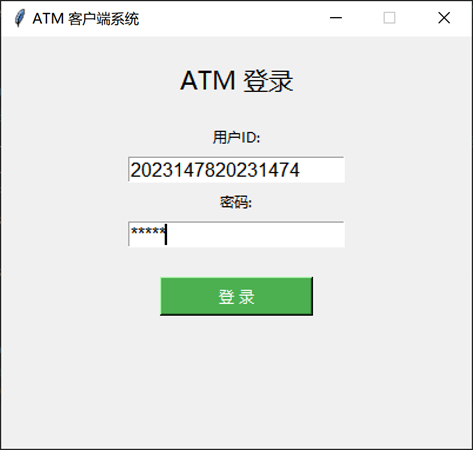
5.2.2.5 服务端日志



5.2.3 我方作为 Server，对方作为 Client

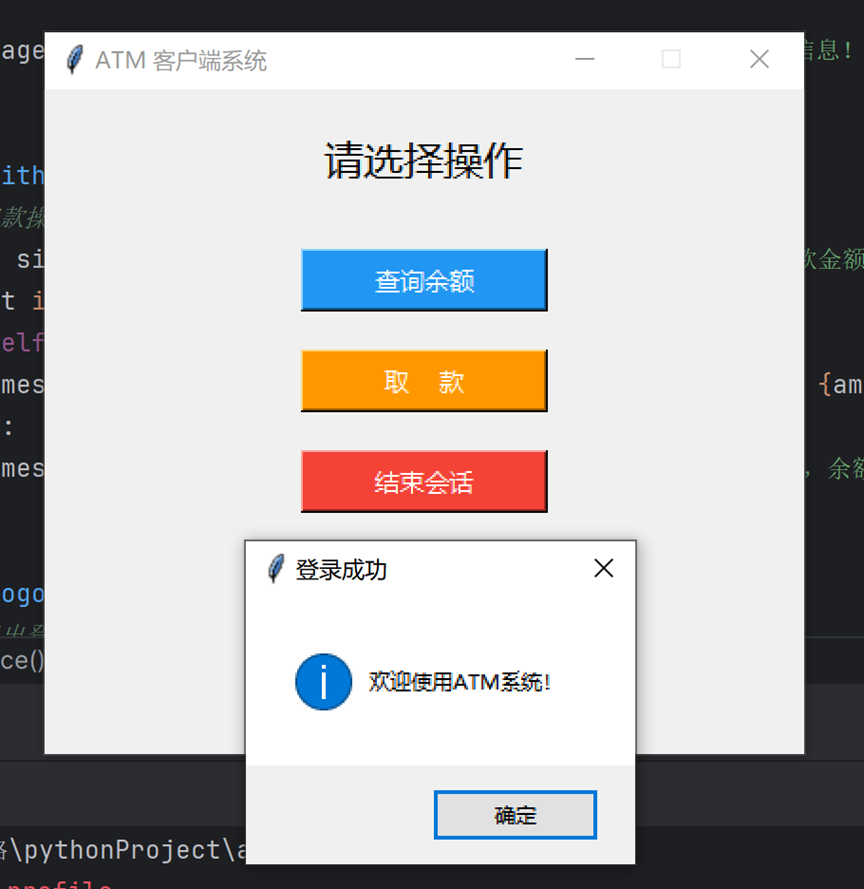
5.2.3.1 连接服务器

客户端开始运行后，弹出登录界面表示连接成功



5.2.3.2登录账号

客户端输入账号（2023147820231474）和密码（pass1）并登录，显示登录成功



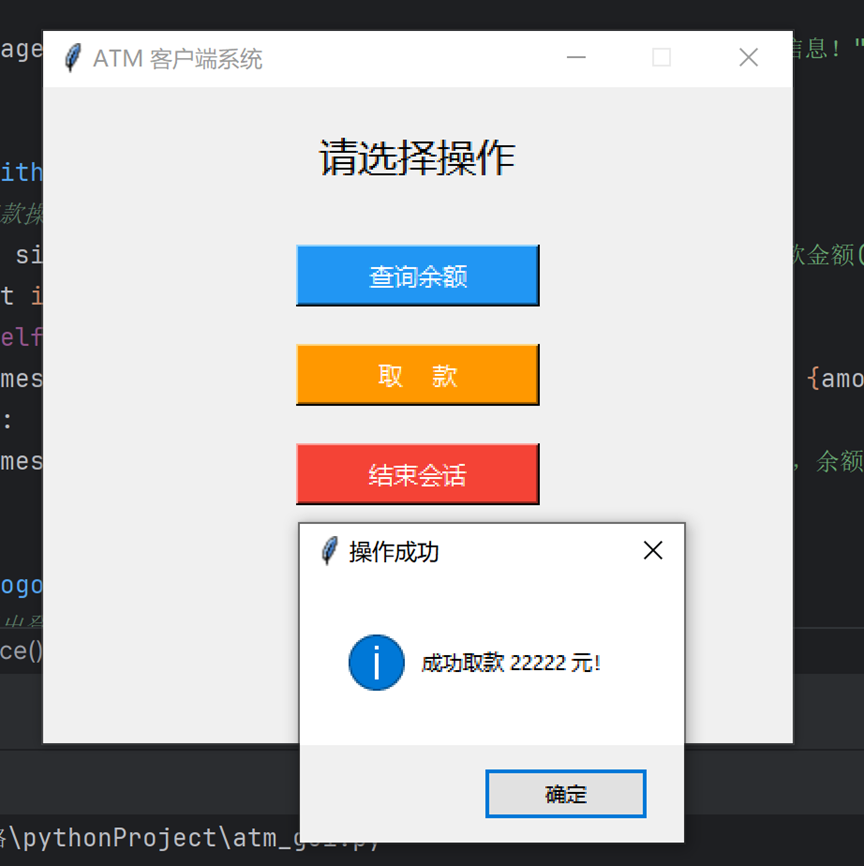
5.2.3.3查询余额

客户端查询余额，余额显示79312元



5.2.3.4取款

客户端取款22222元，显示取款成功

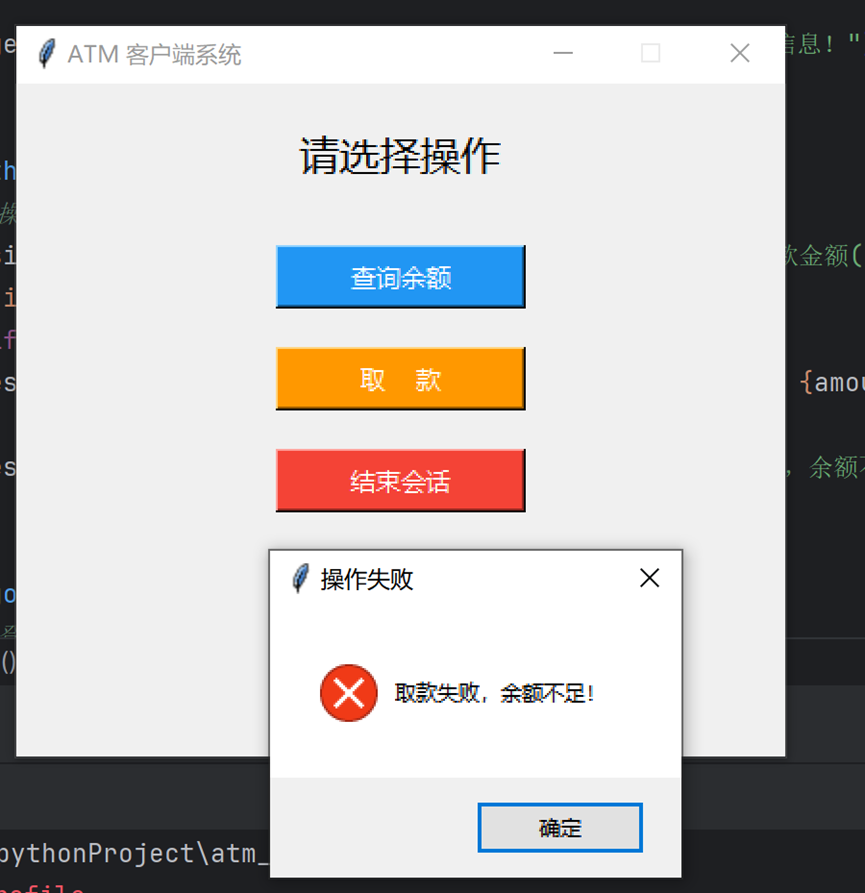


再次查询余额，余额显示57090元



5.2.3.5取款失败

客户端欲取款600000元，但显示取款失败

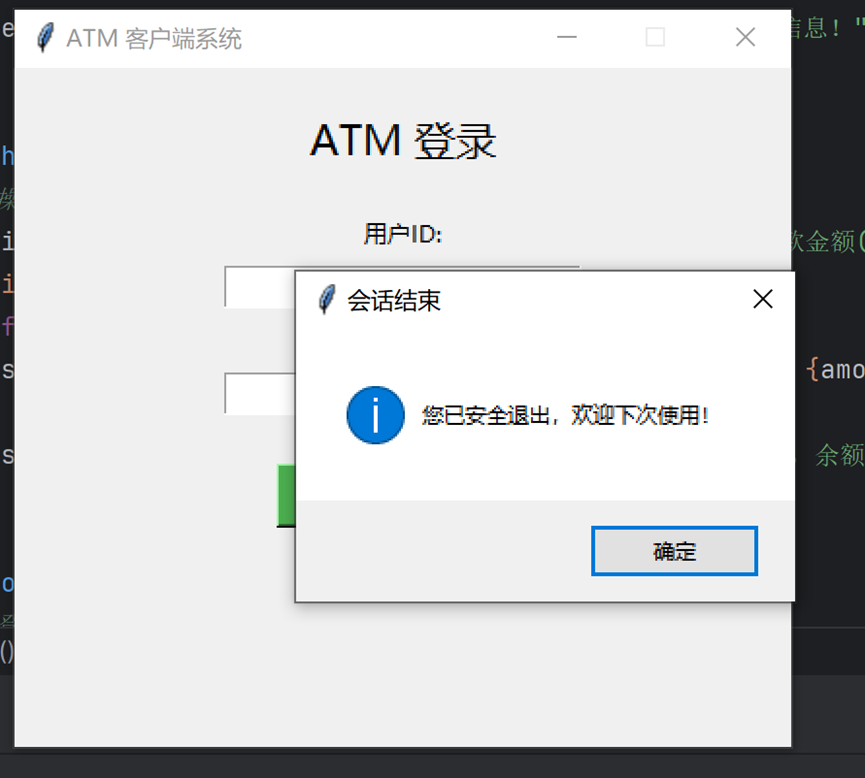


同时服务端日志显示

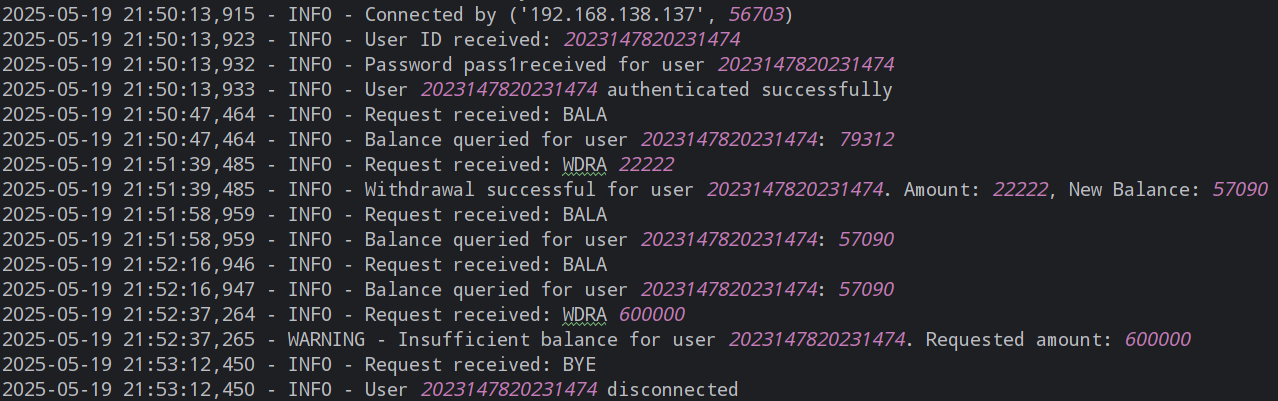


5.2.3.6退出登录

客户端结束对话，成功退出登录



5.2.3.7服务器日志



# 总结

//介绍一下开展作业1、2相关工作遇到的问题、解决的思路，以及相关的收获等。

问题1：

在进行组间测试时，多次发现账号或者密码输入错误，导致登录不成功，但实际上账号和密码输入正确。

解决思路1：

通过查看双方服务器与客户端的原始代码，发现组内测试时成员间统一了报文格式，所以组内测试可以通过。但是小组间存在报文格式不统一的问题，因此仅需要统一组间的报文格式便可以成功登录。

收获：

通过本次作业我们掌握了所学 TCP Socket 编程方法，理解了TCP Socket在服务器和客户端之间的应用层通信的流程和形式，巩固了计算机网络的理论和知识。