COMP3331 Assignment report
Jie Mei z5173405

Language version: python 3.7.3

Overall design idea:

Generally speaking, on the server side, I use the design concept of object-oriented programming. The entire command input from the customer service side is transmitted to the server side, and the server processes and feeds back a special string of the customer service side, and the customer service side prints. I have completed all commands except SHT.

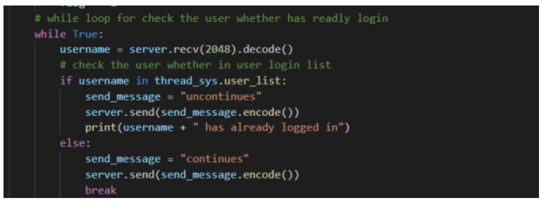
Server:

In the initial stage, I created the thread system class, the system saves all data content, it is the highest level to store the thread list. The thread list contains all the created message classes and file classes. When I parse the commands from the client, I will call my own functions and the functions in the thread class through the thread system class to operate.

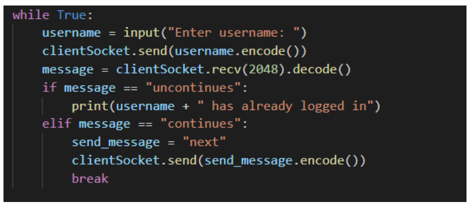
Client:

The client mainly transmits commands to the server and waits for the server's feedback to proceed to the next step. The main idea is divided into login phase and command process phase.

Login part:

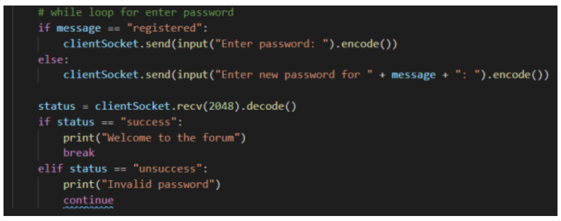


```python
# while loop for check the user whether has readly login
while True:
    username = server.recv(2048).decode()
    # check the user whether in user login list
    if username in thread_sys.user_list:
        send_message = "uncontinues"
        server.send(send_message.encode())
        print(username + " has already logged in")
    else:
        send_message = "continues"
        server.send(send_message.encode())
        break
```



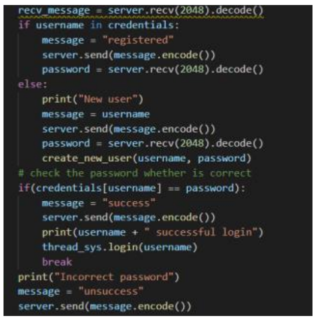
```python
while True:
    username = input("Enter username: ")
    clientSocket.send(username.encode())
    message = clientSocket.recv(2048).decode()
    if message == "uncontinues":
        print(username + " has already logged in")
    elif message == "continues":
        send_message = "next"
        clientSocket.send(send_message.encode())
        break
```

This loop mainly solves that when multiple customer services interact with the server, the client login detects whether the entered username has been logged in. If it has been logged in, the loop continues to know that the entered username is not in the login list.



```python
# while loop for enter password
if message == "registered":
    clientSocket.send(input("Enter password: ").encode())
else:
    clientSocket.send(input("Enter new password for " + message + ": ").encode())

status = clientSocket.recv(2048).decode()
if status == "success":
    print("Welcome to the forum")
    break
elif status == "unsuccess":
    print("Invalid password")
    continue
```



```python
recv_message = server.recv(2048).decode()
if username in credentials:
    message = "registered"
    server.send(message.encode())
    password = server.recv(2048).decode()
else:
    print("New user")
    message = username
    server.send(message.encode())
    password = server.recv(2048).decode()
    create_new_user(username, password)
# check the password whether is correct
if(credentials[username] == password):
    message = "success"
    server.send(message.encode())
    print(username + " successful login")
    thread_sys.login(username)
    break
print("Incorrect password")
message = "unsuccess"
server.send(message.encode())
```

This part of the implementation is the operation when the password is entered. If the password is wrong, it will re-enter the username and return to the initial position of the loop to know that the user is logged in correctly.

Command process:

The difficulty in this part lies in UPD and DWN. Due to the size limitation in recv(), we need to know the size of the transmitted file in order to upload and download correctly and completely. By querying the data, I get the size of the uploaded or downloaded file through os.path.getsize(). I didn't solve the SHT command correctly. The difficulty is that I don't know how to shut down the server and send server shutdown instructions to all logged-in users at the same time.