

计算机网络 课程基础实验二

应用协议与数据包分析实验(Wireshark)

计科210X 甘晴void 202108010XXX

一、实验目的

通过本实验，学习采用**Socket**（套接字）设计简单的网络数据收发程序，理解应用数据包是如何通过传输层进行传送的。

二、实验内容

Socket（套接字）是一种抽象层，应用程序通过它来发送和接收数据，就像应用程序打开一个文件句柄，将数据读写到稳定的存储器上一样。一个**socket**允许应用程序添加到网络中，并与处于同一个网络中的其他应用程序进行通信。一台计算机上的应用程序向**socket**写入的信息能够被另一台计算机上的另一个应用程序读取，反之亦然。

不同类型的**socket**与不同类型的底层协议族以及同一协议族中的不同协议栈相关联。现在**TCP/IP**协议族中的主要**socket**类型为流套接字（**sockets sockets**）和数据报套接字

（**datagram sockets**）。流套接字将**TCP**作为其端对端协议（底层使用**IP**协议），提供了一个可信赖的字节流服务。一个**TCP/IP**流套接字代表了**TCP**连接的一端。数据报套接字使用**UDP**协议（底层同样使用**IP**协议），提供了一个"尽力而为"（**best-effort**）的数据报服务，应用程序可以通过它发送最长65500字节的个人信息。一个**TCP/IP**套接字由一个互联网地址，一个端对端协议（**TCP**或**UDP**协议）以及一个端口号唯一确定。

（1）问题与解决

①获取本机**ip**地址

手动查询

```
cmd > ipconfig
```

获取信息如下

```
C:\Users\y>ipconfig
```

windows IP 配置

以太网适配器 以太网 2:

媒体状态 : 媒体已断开连接
连接特定的 DNS 后缀 :

以太网适配器 VirtualBox Host-Only Network:

连接特定的 DNS 后缀 :
IPv4 地址 : 192.168.56.1
子网掩码 : 255.255.255.0
默认网关. :

无线局域网适配器 本地连接* 10:

媒体状态 : 媒体已断开连接
连接特定的 DNS 后缀 :

无线局域网适配器 本地连接* 11:

媒体状态 : 媒体已断开连接
连接特定的 DNS 后缀 :

无线局域网适配器 WLAN:

连接特定的 DNS 后缀 :
本地链接 IPv6 地址. : fe80::77e3:b329:3430:3e8d%4
IPv4 地址 : 192.168.1.111
子网掩码 : 255.255.255.0
默认网关. : 192.168.1.1

以太网适配器 蓝牙网络连接:

媒体状态 : 媒体已断开连接
连接特定的 DNS 后缀 :

重点关注无线局域网适配器WLAN的IPV4地址

192.168.1.111

这个就是本机的IP地址

自动查询

还有一种方法，可以通过python自动实现，具体原理是 创建一个socket连接到一个公共的主机（例如: Google DNS服务器）来获取本机IP地址。代码如下。

```
import socket

def get_local_ip():
    try:
        # 创建一个socket连接到一个公共的主机来获取本机IP地址
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        sock.connect(("8.8.8.8", 80))    #Google DNS服务器: 8.8.8.8 或者 8.8.4.4
        local_ip = sock.getsockname()[0]
        sock.close()
        return local_ip
    except Exception as e:
        print("获取本机IP地址时发生错误:", str(e))
        return None

if __name__ == "__main__":
    local_ip = get_local_ip()
    if local_ip:
        print(f"本机IP地址是: {local_ip}")
    else:
        print("无法获取本机IP地址。")
```

②地端口号以及管理员权限导致的问题

较低的端口号可能

```
E:\anaconda\envs\python3-11\python.exe E:\python_files\计网-实验
2\UDPserver.py
Traceback (most recent call last):
  File "E:\python_files\计网-实验2\UDPserver.py", line 11, in
<module>
    server_socket.bind((server_host, server_port))
PermissionError: [winError 10013] 以一种访问权限不允许的方式做了一个访问套
接字的尝试。
进程已结束,退出代码1
```

③防火墙

需要允许通过防火墙的请求

④python报错

```
E:\anaconda\envs\python3-11\python.exe E:\python_files\计网-实验
2\TCPserver-threadpool-plus.py
Traceback (most recent call last):
  File "E:\python_files\计网-实验2\TCPserver-threadpool-plus.py",
line 64, in <module>
    main()
  File "E:\python_files\计网-实验2\TCPserver-threadpool-plus.py",
line 48, in main
    server.bind((server_host, server_port))
OSError: [WinError 10048] 通常每个套接字地址(协议/网络地址/端口)只允许使用一
次。

进程已结束,退出代码1
```

连续停止开始并使用统一端口号可能出现该情况。这是因为此时上一个python的进程还没有完全分离，该端口还被占用。此时需要改用另一个端口号即可。

(2) 需要实现的目标

- 1 采用TCP进行数据发送的简单程序
- 2 采用UDP进行数据发送的简单程序
- 3 多线程、线程池对比

- 当一个客户端向一个已经被其他客户端占用的服务器发送连接请求时，虽然其在连接建立后即可向服务器端发送数据，服务器端在处理完已有客户端的请求前，却不会对新的客户端作出响应。

并行服务器：可以单独处理每一个连接，且不会产生干扰。并行服务器分为两种：一客户一线程和线程池。

每个新线程都会消耗系统资源：创建一个线程将占用CPU周期，而且每个线程都有自己的数据结构（如，栈）也要消耗系统内存。另外，当一个线程阻塞（*block*）时，JVM将保存其状态，选择另外一个线程运行，并在上下文转换（*context switch*）时恢复阻塞线程的状态。随着线程数的增加，线程将消耗越来越多的系统资源。这将最终导致系统花费更多的时间来处理上下文转换和线程管理，更少的时间来对连接进行服务。那种情况下，加入一个额外的线程实际上可能增加客户端总服务时间。

我们可以通过限制总线程数并重复使用线程来避免这个问题。与为每个连接创建一个新的线程不同，服务器在启动时创建一个由固定数量线程组成的线程池（*thread pool*）。当一个新的客户端连接请求传入服务器，它将交给线程池中的一个线程处理。当该线程处理完这个客户端后，又返回线程池，并为下一次请求处理做好准备。如果连接请求到达服务器时，线程池中的所有线程都已经被占用，它们则在一个队列中等待，直到有空闲的线程可用。

- 4 写一个简单的chat程序，并能互传文件。

（3）使用python语言实现

★在实现之前需要注意将所有的ip地址与端口号正确分配

所有python可执行文件及说明陈列如下

- get_ip.py （获取本机ip）
- TCPclient.py （TCP客户端）
- TCPserver.py （TCP服务端-基础）
- TCPserver-multithreading.py （TCP服务端-多线程-一客户一线程）
- TCPserver-threadpool.py （TCP服务端-线程池）
- TCPserver-threadpool-plus.py （TCP服务端-线程池-改进）
- UDPclient.py （UDP客户端）
- UDPserver.py （UDP服务端）
- chatClient.py （chat程序客户端）
- chatServer.py （chat程序服务端）

2.1 采用TCP进行数据发送的简单程序

TCPserver

```
import socket

# 创建TCP服务器套接字
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# 绑定服务器地址和端口
server_host = '192.168.1.110' # 服务器主机
server_port = 15000 # 服务器端口
server_socket.bind((server_host, server_port))

# 开始监听客户端连接
server_socket.listen(1) # 最多允许一个客户端连接

print(f"等待客户端连接在 {server_host}:{server_port}...")
client_socket, client_address = server_socket.accept()
print(f"连接来自: {client_address}")

while True:
    # 接收客户端发送的数据
    data = client_socket.recv(1024)
    if not data:
        break
    print(f"接收到的数据: {data.decode('utf-8')}")

    # 回复客户端
    response = "服务器已收到您的消息"
    client_socket.send(response.encode('utf-8'))

# 关闭连接
client_socket.close()
server_socket.close()
```

TCPclient

```
import socket
```

```

# 创建TCP客户端套接字
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# 服务器地址和端口
server_host = '192.168.1.110' # 服务器主机
server_port = 15000 # 服务器端口

# 连接到服务器
client_socket.connect((server_host, server_port))

while True:
    message = input("请输入要发送的消息: ")
    if message == 'exit':
        break

    # 发送消息到服务器
    client_socket.send(message.encode('utf-8'))

    # 接收服务器的响应
    response = client_socket.recv(1024)
    print(f"服务器响应: {response.decode('utf-8')}")

# 关闭连接
client_socket.close()

```

2.2采用UDP进行数据发送的简单程序

UDPserver

```

import socket

# 创建UDP服务器套接字
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# 服务器地址和端口
server_host = '192.168.1.110' # 监听所有网络接口
server_port = 15001 # 端口号

# 绑定服务器地址和端口
server_socket.bind((server_host, server_port))

```

```

print(f"UDP服务器已启动，等待数据传入在 {server_host}:{server_port}...")

while True:
    # 接收客户端发送的数据
    data, client_address = server_socket.recvfrom(1024)
    print(f"接收到的数据: {data.decode('utf-8')} 来自 {client_address}")

    # 回复客户端
    response = "服务器已收到您的消息"
    server_socket.sendto(response.encode('utf-8'), client_address)

```

UDPclient

```

import socket

# 创建UDP客户端套接字
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# 服务器地址和端口
server_host = '192.168.1.110' #
server_port = 15001 # 与服务器端相同的端口号

while True:
    message = input("请输入要发送的消息: ")
    if message == 'exit':
        break

    # 发送消息到服务器
    client_socket.sendto(message.encode('utf-8'), (server_host,
server_port))

    # 接收服务器的响应
    data, server_address = client_socket.recvfrom(1024)
    print(f"服务器响应: {data.decode('utf-8')} 来自 {server_address}")

# 关闭连接
client_socket.close()

```


2.3多线程、线程池对比

①多线程（一客户一线程）

```
import socket
import threading

# 处理客户端连接的函数
def handle_client(client_socket, client_address):
    try:
        while True:
            data = client_socket.recv(1024)
            if not data:
                break
            print(f"来自客户端 {client_address[0]}: {client_address[1]} 的消息: {data.decode('utf-8')}")

            # 服务器处理数据逻辑（范例置空）

            # 向客户端发送响应数据
            response = "服务器已收到您的消息"
            client_socket.send(response.encode('utf-8'))
        except Exception as e:
            print(f"来自客户端 {client_address[0]}: {client_address[1]} 的连接发生异常: {str(e)}")
        finally:
            # 关闭客户端连接
            client_socket.close()

def main():
    server_host = '192.168.1.110' # 监听所有网络接口
    server_port = 15007 # 选择一个未被占用的端口号

    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((server_host, server_port))
    server.listen(5) # 最多允许5个客户端排队等待连接

    print(f"等待客户端连接在 {server_host}:{server_port}...")

    while True:
        client_socket, client_address = server.accept()
```

```

        print(f"客户端已连接: {client_address[0]}:
{client_address[1]}")

        # 创建一个新线程来处理客户端连接
        client_handler = threading.Thread(target=handle_client,
args=(client_socket, client_address))
        client_handler.start()

if __name__ == "__main__":
    main()

```

②线程池

该线程池可以实现至多5个客户端的同时连接，但在同时出现第6个客户端时，将不对该客户端进行回应。一旦前5个客户端中有结束连接的，第6个客户端就立刻加入。

```

import socket
import concurrent.futures

# 处理客户端连接的函数
def handle_client(client_socket, client_address):
    try:
        while True:
            data = client_socket.recv(1024)
            if not data:
                break
            print(f"客户端 {client_address[0]}:{client_address[1]} 发
来数据: {data.decode('utf-8')}")

            # 服务器处理数据逻辑（范例置空）

            # 向客户端发送响应数据
            response = "服务器已收到您的消息"
            client_socket.send(response.encode('utf-8'))
    except Exception as e:
        print(f"发生异常: {str(e)}")
    finally:
        # 关闭客户端连接

```

```

        client_socket.close()

def main():
    server_host = '192.168.1.110' # 服务器主机
    server_port = 15004 # 服务器端口

    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((server_host, server_port))
    server.listen(5) # 最多允许5个客户端排队等待连接

    print(f"等待客户端连接在 {server_host}:{server_port}...")

    # 创建一个线程池
    with concurrent.futures.ThreadPoolExecutor(max_workers=5) as
executor:
        while True:
            client_socket, client_address = server.accept()
            print(f"客户端已连接: {client_address[0]}:
{client_address[1]}")

            # 使用线程池处理客户端连接
            executor.submit(handle_client, client_socket,
client_address)

if __name__ == "__main__":
    main()

```

③线程池改进

增加的功能是，当目前可同时连接的客户端数到达上限的时候，再次连入的客户端由一个新的线程处理，该线程向该客户端发送拒绝告知，并结束连接。由该线程告知所有在此时想要接入的新客户端。

在当前有空余可连接空间的时候，客户端正常连接。

不足在于，由于想公用之前写的TCPclient作为客户端，对其不做改变。有一点无法实现，即被拒绝的客户端无法在有空闲时自动接入连接，因为这需要对客户端的逻辑进行更改。在下面的程序中，客户端在被拒绝后接收到来自服务器的拒绝要求后直接被断开连接，需要增加一个客户端在被服务器主动断开连接之后的处理，才能解决这个问题。这里我就不做讨论了。

```

import socket
import concurrent.futures
import threading

# 最大允许的客户端连接数
MAX_CLIENTS = 5

# 存储当前活跃客户端连接的列表
active_clients = []

# 线程锁用于保护 active_clients 列表
count_lock = threading.Lock()

# 全局计数器用于跟踪已连接的客户端数量
connected_clients = 0

# 处理客户端连接的函数
def handle_client(client_socket, client_address):
    global connected_clients

    try:
        with count_lock:
            connected_clients += 1
            # print(f"connected_clients= {connected_clients} ")
            if connected_clients > MAX_CLIENTS:
                # 如果连接数超过最大限制，向客户端发送拒绝通知
                print("服务器连接已满，已拒绝一个新的连接")
                response = "服务器连接已满，拒绝连接。"
                client_socket.send(response.encode('utf-8'))
                return
            else:
                active_clients.append(client_socket)

        while True:
            data = client_socket.recv(1024)
            if not data:
                break
            print(f"客户端 {client_address[0]}:{client_address[1]} 发
来数据: {data.decode('utf-8')}")

    # 服务器处理数据逻辑（范例置空）

```

```

        # 向客户端发送响应数据
        response = "服务器已收到您的消息"
        client_socket.send(response.encode('utf-8'))
    except Exception as e:
        print(f"发生异常: {str(e)}")
    finally:
        with count_lock:
            connected_clients -= 1
            active_clients.remove(client_socket)
            client_socket.close()

def main():
    server_host = '192.168.1.110' # 监听所有网络接口
    server_port = 15009 # 选择一个未被占用的端口号

    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((server_host, server_port))
    server.listen(MAX_CLIENTS+1) # 最多允许5个客户端排队等待连接

    print(f"等待客户端连接在 {server_host}:{server_port}...")

    # 创建一个线程池
    with
    concurrent.futures.ThreadPoolExecutor(max_workers=MAX_CLIENTS+1) as
    executor:
        while True:
            client_socket, client_address = server.accept()
            print(f"客户端已连接: {client_address[0]}:
{client_address[1]}")

            # 使用线程池处理客户端连接
            executor.submit(handle_client, client_socket,
client_address)

if __name__ == "__main__":
    main()

```

2.4 写一个简单的chat程序，并能互传文件。

要求写一个简单的chat程序，能够实现互相通信与文件传输。实现这个需要一个服务端和若干客户端，客户端之间可以通过服务端互相联系。

①程序思想

服务端需要能够接受客户端发送的请求，并且对请求做出判断。这个请求是发送的消息还是请求文件操作。如果是请求文件操作，是要上传文件还是要下载文件，文件如果存在就传输这个文件给客户端下载，如果文件不存在就向用户端发送错误报告。

客户端需要能够处理用户的终端输入，并且根据约定好的解析方式解析出用户的意图，使用与服务端约定好的方式去表示用户的意图，再发送给服务端。从服务端接收反馈，再通过用户能够看懂的方式反馈给客户。

传输文件的时候要对文件进行分段传输，因为文件通常都大于我们一次可以传输的范围。

②实现效果

用户视角可用操作：

- 输入用户名并加入聊天室
- 直接输入并回车 #发送普通消息
- 使用/exit #退出客户端
- 使用/file upload +文件路径 #上传文件（若文件存在，即上传；若文件不存在，本地报错）
- 使用/file download +文件名 #下载文件（若文件存在，即下载；若文件不存在，即服务器返回报错）
- 使用/file list #查看服务器可供下载的文件列表（★这是在开发过程中额外实现的功能）
- 接收其他任意用户加入聊天室的消息
- 接收其他任意用户发送的消息
- 接收其他任意用户上传文件成功的消息
- 接收其他任意用户退出聊天室的消息

服务器视角可用操作：

- 监听所有用户发送的普通消息
- 接收所有用户发送的文件
- 向请求文件的用户发送文件
- 显示用户加入以及退出的消息

显示系统时（获取发生事件时的服务器时间与用户时间）

③程序设计

chatServer

主进程:

开启服务端，在地址、端口上等待客户端，

维护一个客户端列表，记录每个客户端的名字

开启永真循环，若当前用户数量未达到上限，接收到用户请求接入时通过线程池分配线程与用户进行连接。

子线程handle_client(client_socket):

在永真循环内接收消息

- 若消息为"/exit": 从客户端列表中删除该用户，调用broadcast函数向其他用户广播该用户离开聊天室的消息
- 若消息以"FILE:"开头: 解析客户端发送的【"FILE:"+"|" + file_name + "|" + str(file_size)】，获得文件名和文件大小，调用receive_file(client_socket, file_name, file_size)函数接收文件
- 若消息以"DOWN:"开头: 解析客户端发送的【"DOWN:" + "|" + file_name】，获得文件名，在服务端储存文件夹内查找该文件是否存在。若不存在，返回报错；若存在，调用send_requested_file(client_socket, file_name, file_path)向客户端发送该文件
- 若消息为"/file list": 调用list_files_in_directory()函数遍历服务端储存文件夹内存在的所有文件名，并使用pickle包编码（因为列表形式的结果无法直接发送）。先向客户端发送【"LIST:"】提示客户端做好准备，然后发送结果
- 否则: 表示普通消息。向所有用户转发该普通消息

函数receive_file(client_socket, file_name, file_size):

- 接收来自客户端的【"FILE:" + "|" + file_name + "|" + str(file_size)】并分离出文件名和文件大小
- 分段分次从服务端接收文件并保存到指定路径

函数send_requested_file(client_socket, file_name, file_path):

- 从文件路径获取文件大小
- 向客户端发送【"FILE:" + "|" + file_name + "|" + str(file_size)】提醒做好准备
- 分段分次向客户端发送文件

函数**broadcast(message)**:

- 使用**for**循环向列表内所有在线的客户端发送消息

函数**list_files_in_directory(directory)**:

- 使用**for**循环遍历文件夹获取存在的文件名

chatClient

主进程:

获取用户名，连接服务器。

之后创建子线程，开启永真循环，充当接收端，负责从服务器接收消息:

- 若消息为空: 表示与服务器断开了连接
- 若消息以"**FILE:**"开头: 表示文件传递协议，调用**download(message)**下载文件
- 若消息为"**LIST:**": 表示文件列表请求，直接处理解序列化并向终端输出服务器可供下载文件列表

子线程**send_message()**:

充当类似于**shell**的工作，读入用户终端输入的内容，进行解析处理后向服务器发送

- 若输入为"**/exit**": 向服务器发送退出请求，并退出
- 若输入为"**/file upload** "开头: 先验证文件是否存在，若存在调用**send_file(file_path)**发送文件，否则本地报错
- 若输入为"**/file download** "开头: 调用**download_query(file_name)**向服务器确认该文件是否存在
- 若输入为"**/file list** ": 向服务器发送"**/file list** "
- 否则: 为普通消息，直接向服务器发送

函数**send_file(file_path)**:

- 通过文件路径获得文件名和文件大小
- 向服务器发送【"**FILE:**"+"|"+**file_name**+"|"+**str(file_size)**】引导服务端接受文件
- 分段分次向服务端发送文件

函数**download_query(file_name)**:

- 向服务器发送【"DOWN:" + "|" + file_name】查询文件是否可以下载

函数download(message):

- 接收来自服务器的【"FILE:" + "|" + file_name + "|" + str(file_size)】并分离出文件名和文件大小
- 分段分次从服务端接收文件并保存到指定路径

④技术亮点

I 显示时间

使用这个包与这个方式获取时间，一般人对时间的毫秒不感兴趣，因此规范化格式。

```
import datetime #获取系统时间
current_time = datetime.datetime.now()
time = current_time.strftime("%Y-%m-%d %H:%M:%S")
```

II 显示服务器可供下载文件列表

善用os.path包，这个包提供了关于系统路径的很有用的工具

另外传输时没办法直接传输表格，因此需要使用pickle包进行序列化与解序列化

```
import pickle #序列化列表为字节数据

#服务端：序列化并发送
file_names_bytes = pickle.dumps(file_names)
client_socket.send(file_names_bytes)

#客户端：接收并解序列化
file_names_bytes = client_socket.recv(1024)
file_names = pickle.loads(file_names_bytes)
```

III 客户端采用线程

客户端采用主进程接收，线程发送的模式，收发更加清晰。

⑤程序源码

chatServer

```
import socket
import threading
import os
import concurrent.futures
import pickle #序列化列表为字节数据
import datetime #获取系统时间

current_time = datetime.datetime.now()
time = current_time.strftime("%Y-%m-%d %H:%M:%S")
# 建立服务器并等待客户端
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #
创建服务器套接字
server_host = '192.168.1.110' # 服务器主机
server_port = 15004 # 服务器端口
server_socket.bind((server_host, server_port))
server_socket.listen(6)
# 用于存储已连接的客户端套接字和用户名的字典
client_sockets = {}
client_usernames = {}
MAX_CONNECTIONS = 5 # 设置最大连接数
executor =
concurrent.futures.ThreadPoolExecutor(max_workers=MAX_CONNECTIONS)
# 创建线程池
print(f"{time} server>>等待客户端连接在 {server_host}:
{server_port}...")

# 处理客户端消息的函数
def handle_client(client_socket):
    base_path = "E:\python_files\计网-实验2\服务端保存文件"
    while True:
        try:
            # 接收消息
            message = client_socket.recv(1024).decode("utf-8")
            current_time = datetime.datetime.now()
            time = current_time.strftime("%Y-%m-%d %H:%M:%S")
            if message == "/exit": #客户端退出
                # 客户端断开连接
                del client_sockets[client_socket]
```

```

        username = client_usernames[client_socket]
        del client_usernames[client_socket]
        broadcast(f"{time} server>>>{username} 离开了聊天室")
        print(f"{time} server>>>{username} 离开了聊天室")
        break
    elif message.startswith("FILE:"):    #客户端上传文件
        # 客户端要上传文件
        recv = message.split("|")
        file_name,file_size = recv[1],recv[2]
        receive_file(client_socket, file_name, file_size)
    elif message.startswith("DOWN:"):    #客户端请求下载
        # 处理文件下载请求
        recv = message.split("|")
        file_name = recv[1]
        file_path = os.path.join(base_path, file_name)
        # 请求的文件是否在本地存在
        if os.path.isfile(file_path) != True:
            client_socket.send(f"{time} server>>>错误:
{file_path}: 文件不存在".encode("utf-8"))
            print(f"{time} server>>>错误: {file_path}: 文件不存
在")

            continue
        # 请求的文件在本地存在, 开始发送
        send_requested_file(client_socket, file_name,
file_path)
    elif message == "/file list":    #客户端请求查看文件列表
        file_names =
list_files_in_directory(os.path.normpath(base_path))
        file_names_bytes = pickle.dumps(file_names)
        inform = "LIST:"
        client_socket.send(inform.encode("utf-8"))
        client_socket.send(file_names_bytes)
    else:
        # 否则, 将消息广播给其他客户端
        username = client_usernames[client_socket]
        broadcast(f"{time} {username}: {message}")
        print(f"{time} {username}: {message}")
except:
    # 处理异常, 例如客户端断开连接
    break

# 用于接收文件的函数

```

```

def receive_file(client_socket, file_name, file_size):
    current_time = datetime.datetime.now()
    time = current_time.strftime("%Y-%m-%d %H:%M:%S")
    base_path = "E:\python_files\计网-实验2\服务端保存文件"
    save_path = os.path.join(base_path, file_name)
    try:
        recv_size = 0
        file = open(save_path, 'wb')
        flag = True
        while flag:
            if int(file_size) > recv_size:
                data = client_socket.recv(1024)
                recv_size += len(data)
                file.write(data)
            else:
                flag = False
        file.close()
        print(f"{time} server>>文件上传成功")
        broadcast(f"{time} server>>>
{client_usernames[client_socket]} 上传了文件: {file_name}")
    except Exception as e:
        print(f"{time} server>>>{client_usernames[client_socket]}文
件传输失败: {e}")
        client_socket.send(f"{time} server>>>
{client_usernames[client_socket]}文件传输失败: {e}".encode("utf-8"))
        # 发送错误消息给客户端

# 处理客户端下载文件请求
def send_requested_file(client_socket, file_name, file_path):
    current_time = datetime.datetime.now()
    time = current_time.strftime("%Y-%m-%d %H:%M:%S")
    file_size = os.stat(file_path).st_size
    try:
        inform = ("FILE:" + "|" + file_name + "|" + str(file_size))
        client_socket.send(inform.encode("utf-8"))
        send_size = 0
        file = open(file_path, 'rb')
        flag = True
        while flag:
            if send_size + 1024 > file_size:
                data = file.read(file_size - send_size)

```

```

        flag = False
    else:
        data = file.read(1024)
        send_size += 1024
        client_socket.send(data)
    file.close()
    print(f"{time} server>>向 {client_usernames[client_socket]}
发送文件成功")
    except Exception as e:
        print(f"{time} server>>向 {client_usernames[client_socket]}
发送文件失败: {e}")
        client_socket.send(f"{time} server>>文件发送失
败".encode("utf-8")) # 发送错误消息给客户端

# 广播消息给所有客户端
def broadcast(message):
    for client_socket in client_sockets:
        client_socket.send(message.encode("utf-8"))

# 获取当前文件列表
def list_files_in_directory(directory):
    file_list = []
    for filename in os.listdir(directory):
        if os.path.isfile(os.path.join(directory, filename)):
            file_list.append(filename)
    return file_list

# 主循环, 等待客户端连接
while True:
    current_time = datetime.datetime.now()
    time = current_time.strftime("%Y-%m-%d %H:%M:%S")
    if len(client_sockets) >= MAX_CONNECTIONS:
        # 达到最大连接数时, 拒绝新连接
        client_socket, _ = server_socket.accept()
        client_socket.send(f"{time} server>>>连接已满, 请稍后再
试.".encode("utf-8"))
        client_socket.close()
    else:
        client_socket, client_address = server_socket.accept()
        username = client_socket.recv(1024).decode("utf-8")
        client_sockets[client_socket] = client_address

```

```

        client_usernames[client_socket] = username
        print(f"{time} server>>>连接来自 {client_address}, 用户名:
{username}<<<")
        client_socket.send(f"{time} server>>>连接成功! 开始聊天
吧".encode("utf-8"))
        broadcast(f"{time} server>>>{username} 加入了聊天室")

# 使用线程池处理客户端消息
executor.submit(handle_client, client_socket)

```

chatClient

```

import socket
import threading
import os
import pickle #序列化列表为字节数据
import datetime #获取系统时间

# 创建客户端套接字
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# 服务器地址和端口
server_host = '192.168.1.110' # 服务器主机
server_port = 15004 # 服务器端口
client_socket.connect((server_host, server_port))

# 输入用户名
username = input("请输入您的用户名: ")
client_socket.send(username.encode("utf-8"))

# 处理本地输入
def send_message():
    while True:
        current_time = datetime.datetime.now()
        time = current_time.strftime("%Y-%m-%d %H:%M:%S")
        message = input()
        if message.lower() == "/exit": # 退出客户端
            client_socket.send(message.lower().encode("utf-8"))
            client_socket.close()
            break
        elif message.startswith("/file upload "): # 上传文件
            file_path = os.path.normpath(message[13:])

```

```

        if os.path.isfile(file_path) != True:
            print(f"{time} local>>错误: {file_path}: 文件不存在")
            continue
        send_file(file_path)
    elif message.startswith("/file download "): # 下载文件
        file_name = message[15:]
        download_query(file_name)
    elif message.lower() == "/file list ": # 查看文件列表
        client_socket.send(message.lower().encode("utf-8"))
    else: # 发送普通消息
        client_socket.send(message.encode("utf-8"))

# 上传文件
def send_file(file_path):
    current_time = datetime.datetime.now()
    time = current_time.strftime("%Y-%m-%d %H:%M:%S")
    try:
        file_name = os.path.basename(file_path)
        file_size = os.stat(file_path).st_size
        inform = ("FILE:"+"|" +file_name+"|" +str(file_size))
        client_socket.send(inform.encode("utf-8"))
        send_size = 0
        file = open(file_path, 'rb')
        flag = True
        while flag:
            if send_size + 1024 > file_size:
                data = file.read(file_size-send_size)
                flag = False
            else:
                data = file.read(1024)
                send_size += 1024
            client_socket.send(data)
        file.close()
        print(f"{time} local>>文件上传成功")
    except:
        print(f"{time} local>>文件传输失败")

# 发送下载文件请求
def download_query(file_name):
    current_time = datetime.datetime.now()
    time = current_time.strftime("%Y-%m-%d %H:%M:%S")
    try:

```

```

        inform = ("DOWN:" + "|" + file_name)
        client_socket.send(inform.encode("utf-8"))
    except:
        print(f"{time} local>>下载文件请求发送失败")

def download(message):
    current_time = datetime.datetime.now()
    time = current_time.strftime("%Y-%m-%d %H:%M:%S")
    recv = message.split("|")
    # 格式("FILE:" + "|" + file_name + "|" + str(file_size))
    file_name, file_size = recv[1], recv[2]
    base_path = "E:\python_files\计网-实验2\客户端保存文件"
    save_path = os.path.join(base_path, file_name)
    recv_size = 0
    file = open(save_path, 'wb')
    flag = True
    while flag:
        if int(file_size) > recv_size:
            data = client_socket.recv(1024)
            recv_size += len(data)
            file.write(data)
        else:
            flag = False
    print(f"{time} local>>文件 {file_name} 下载成功")
    file.close()

# 创建一个线程来处理用户输入的消息
message_thread = threading.Thread(target=send_message)
message_thread.daemon = True # 将线程设置为守护线程
message_thread.start()

# 接收并显示服务器发送的消息
while True:
    current_time = datetime.datetime.now()
    time = current_time.strftime("%Y-%m-%d %H:%M:%S")
    message = client_socket.recv(1024).decode("utf-8")
    if not message:
        # 如果消息为空, 表示与服务器断开连接
        print(f"{time} local>>与服务器断开连接")
        break
    if message.startswith("FILE:"):

```



```

        download(message)
    elif message == "LIST:":
        file_names_bytes = client_socket.recv(1024)
        file_names = pickle.loads(file_names_bytes)
        if not file_names:
            print("当前服务器:无文件")
        else:
            print("当前服务器文件列表:")
            for file_name in file_names:
                print(file_name)
    else:
        print(message)

```

⑥程序测试

I 使用正确的IP地址与端口号

首先使用get_ip.py获得本机ip

```

E:\anaconda\envs\python3-11\python.exe E:\python_files\计网-实验
2\get_ip.py
本机IP地址是: 192.168.56.103

进程已结束,退出代码0

```

将此ip地址替换客户端/服务端程序段中的ip地址

注意端口号要用较高的, 以避免需要管理员权限。若开机第一次运行, 无需考虑更换端口号。

II 测试普通消息

运行chatServer.py, 启动服务端

```

E:\anaconda\envs\python3-11\python.exe E:\python_files\计网-实验
2\chatServer.py
2023-10-27 22:16:44 server>>等待客户端连接在 192.168.56.103:15004...

```

运行chatClient.py, 启动一个客户端, 用户名为wolf, 服务端返回连接成功消息。

```
E:\anaconda\envs\python3-11\python.exe E:\python_files\计网-实验
2\chatClient.py
请输入您的用户名: wolf
2023-10-27 22:16:44 server>>>连接成功! 开始聊天吧2023-10-27 22:16:44
server>>>wolf 加入了聊天室
```

使用客户端wolf, 发送消息“hello,I am wolf.”

```
# 服务端
E:\anaconda\envs\python3-11\python.exe E:\python_files\计网-实验
2\chatServer.py
2023-10-27 22:16:44 server>>等待客户端连接在 192.168.56.103:15004...
2023-10-27 22:16:44 server>>>连接来自 ('192.168.56.103', 59606), 用户
名: wolf<<<
2023-10-27 22:19:50 wolf: hello,I am wolf.

#客户端
E:\anaconda\envs\python3-11\python.exe E:\python_files\计网-实验
2\chatClient.py
请输入您的用户名: wolf
2023-10-27 22:16:44 server>>>连接成功! 开始聊天吧2023-10-27 22:16:44
server>>>wolf 加入了聊天室
hello,I am wolf.
2023-10-27 22:19:50 wolf: hello,I am wolf.
```

消息发送成功

III 测试多用户状态下普通消息

运行chatClient.py, 启动一个客户端, 用户名为void, 服务端返回连接成功消息。

使用void发送消息“hello,I am void.”

从wolf端可看到void进入聊天室以及发送消息

```
E:\anaconda\envs\python3-11\python.exe E:\python_files\计网-实验
2\chatClient.py
请输入您的用户名: wolf
2023-10-27 22:16:44 server>>>连接成功! 开始聊天吧2023-10-27 22:16:44
server>>>wolf 加入了聊天室
hello,I am wolf.
2023-10-27 22:19:50 wolf: hello,I am wolf.
2023-10-27 22:18:38 server>>>void 加入了聊天室
2023-10-27 22:22:26 void: hello,I am void.
```

IV 测试文件列表与文件传输

在目录下有如下文件

- 文件夹: 服务端保存文件
- 文件夹: 客户端保存文件
- 8-BIT OPTIMIZERS VIA BLOCK-WISE QUANTIZATION.pdf
- A甘晴void.txt
- chatClient.py
- chatServer.py

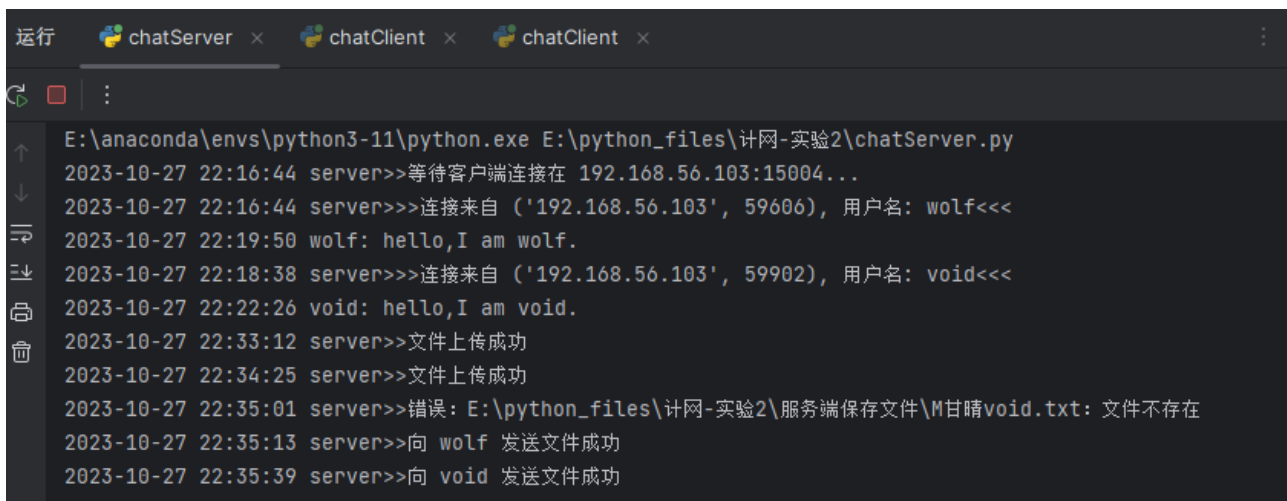
测试步骤如下:

- 使用 void 端请求查看服务器可供下载的文件列表
- 使用 void 端先上传一个不存在的文件“M甘晴void.txt”
- 使用 void 端上传一个存在的文件“A甘晴void.txt”
- 使用 void 端请求查看服务器可供下载的文件列表
- 使用 wolf 端请求查看服务器可供下载的文件列表
- 使用 wolf 端上传一个存在的文件“8-BIT OPTIMIZERS VIA BLOCK-WISE QUANTIZATION.pdf”
- 使用 wolf 端请求查看服务器可供下载的文件列表
- 使用 wolf 端请求下载一个不存在的文件“M甘晴void.txt”
- 使用 wolf 端请求下载一个存在的文件“A甘晴void.txt”
- 使用 void 端请求下载一个存在的文件“8-BIT OPTIMIZERS VIA BLOCK-WISE QUANTIZATION.pdf”

测试结果及截图如下: (截图更好看一些)

服务端

```
E:\anaconda\envs\python3-11\python.exe E:\python_files\计网-实验
2\chatServer.py
2023-10-27 22:16:44 server>>等待客户端连接在 192.168.56.103:15004...
2023-10-27 22:16:44 server>>>连接来自 ('192.168.56.103', 59606), 用户
名: wolf<<<
2023-10-27 22:19:50 wolf: hello,I am wolf.
2023-10-27 22:18:38 server>>>连接来自 ('192.168.56.103', 59902), 用户
名: void<<<
2023-10-27 22:22:26 void: hello,I am void.
2023-10-27 22:33:12 server>>文件上传成功
2023-10-27 22:34:25 server>>文件上传成功
2023-10-27 22:35:01 server>>错误: E:\python_files\计网-实验2\服务端保存
文件\M甘晴void.txt: 文件不存在
2023-10-27 22:35:13 server>>向 wolf 发送文件成功
2023-10-27 22:35:39 server>>向 void 发送文件成功
```

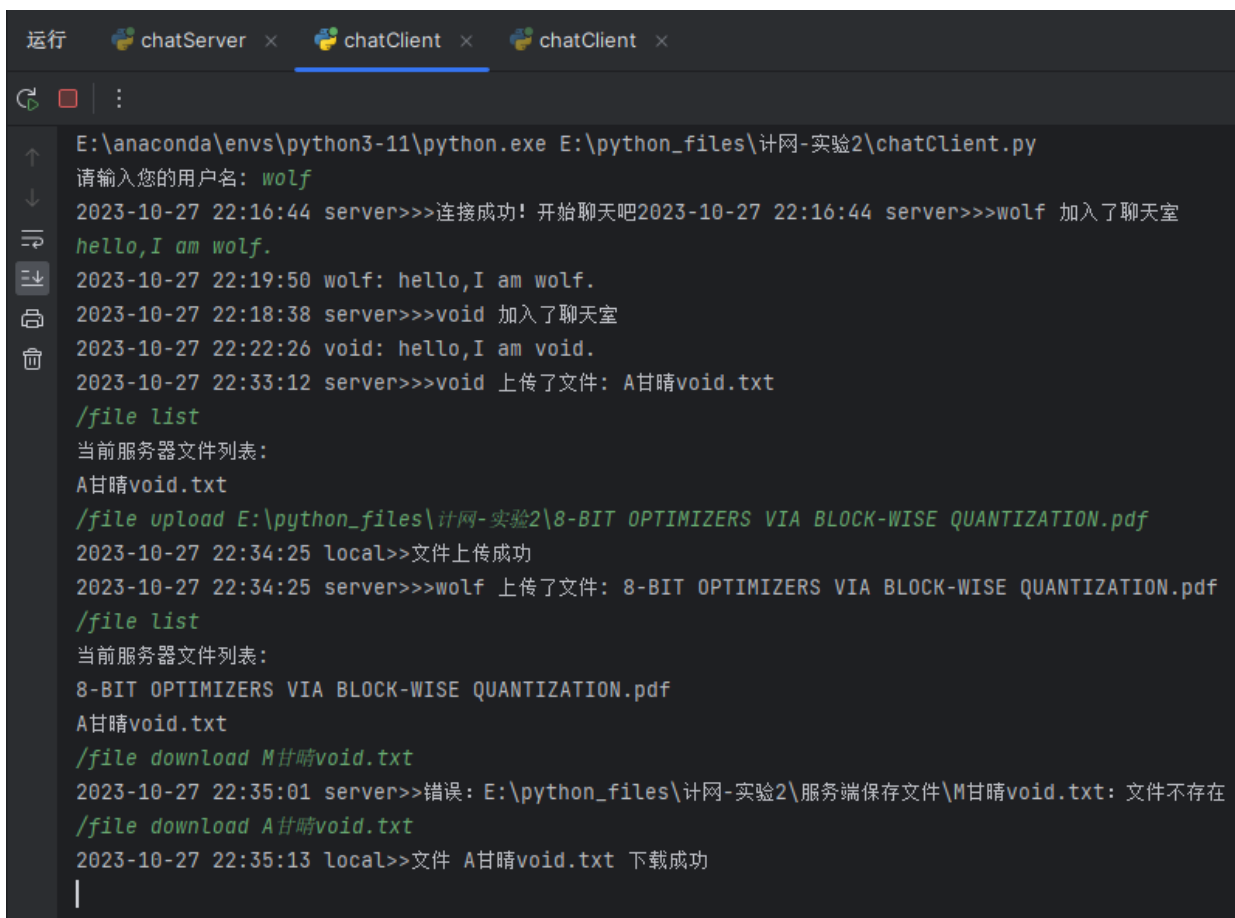


```
运行 chatServer x chatClient x chatClient x
E:\anaconda\envs\python3-11\python.exe E:\python_files\计网-实验2\chatServer.py
2023-10-27 22:16:44 server>>等待客户端连接在 192.168.56.103:15004...
2023-10-27 22:16:44 server>>>连接来自 ('192.168.56.103', 59606), 用户名: wolf<<<
2023-10-27 22:19:50 wolf: hello,I am wolf.
2023-10-27 22:18:38 server>>>连接来自 ('192.168.56.103', 59902), 用户名: void<<<
2023-10-27 22:22:26 void: hello,I am void.
2023-10-27 22:33:12 server>>文件上传成功
2023-10-27 22:34:25 server>>文件上传成功
2023-10-27 22:35:01 server>>错误: E:\python_files\计网-实验2\服务端保存文件\M甘晴void.txt: 文件不存在
2023-10-27 22:35:13 server>>向 wolf 发送文件成功
2023-10-27 22:35:39 server>>向 void 发送文件成功
```

客户端wolf

```
E:\anaconda\envs\python3-11\python.exe E:\python_files\计网-实验
2\chatClient.py
请输入您的用户名: wolf
2023-10-27 22:16:44 server>>>连接成功! 开始聊天吧2023-10-27 22:16:44
server>>>wolf 加入了聊天室
hello,I am wolf.
2023-10-27 22:19:50 wolf: hello,I am wolf.
2023-10-27 22:18:38 server>>>void 加入了聊天室
2023-10-27 22:22:26 void: hello,I am void.
```

```
2023-10-27 22:33:12 server>>>void 上传了文件: A甘晴void.txt
/file list
当前服务器文件列表:
A甘晴void.txt
/file upload E:\python_files\计网-实验2\8-BIT OPTIMIZERS VIA BLOCK-
WISE QUANTIZATION.pdf
2023-10-27 22:34:25 local>>文件上传成功
2023-10-27 22:34:25 server>>>wolf 上传了文件: 8-BIT OPTIMIZERS VIA
BLOCK-WISE QUANTIZATION.pdf
/file list
当前服务器文件列表:
8-BIT OPTIMIZERS VIA BLOCK-WISE QUANTIZATION.pdf
A甘晴void.txt
/file download M甘晴void.txt
2023-10-27 22:35:01 server>>错误: E:\python_files\计网-实验2\服务端保存
文件\M甘晴void.txt: 文件不存在
/file download A甘晴void.txt
2023-10-27 22:35:13 local>>文件 A甘晴void.txt 下载成功
```

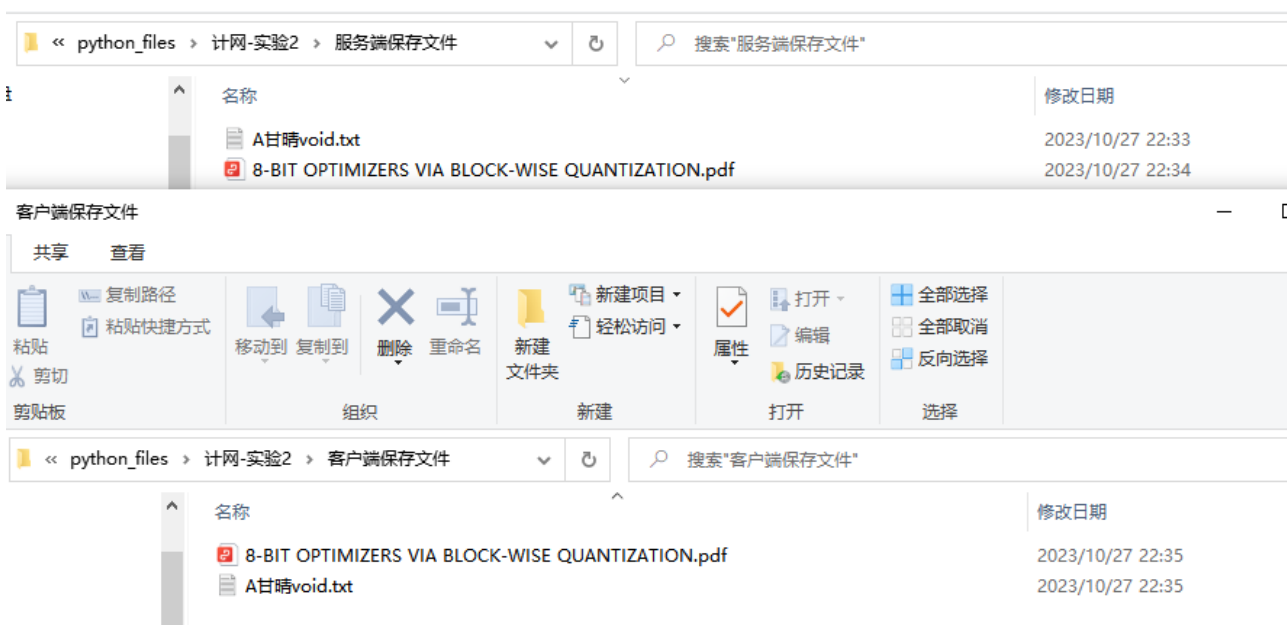


```
运行 chatServer x chatClient x chatClient x
E:\anaconda\envs\python3-11\python.exe E:\python_files\计网-实验2\chatClient.py
请输入您的用户名: wolf
2023-10-27 22:16:44 server>>>连接成功! 开始聊天吧2023-10-27 22:16:44 server>>>wolf 加入了聊天室
hello,I am wolf.
2023-10-27 22:19:50 wolf: hello,I am wolf.
2023-10-27 22:18:38 server>>>void 加入了聊天室
2023-10-27 22:22:26 void: hello,I am void.
2023-10-27 22:33:12 server>>>void 上传了文件: A甘晴void.txt
/file list
当前服务器文件列表:
A甘晴void.txt
/file upload E:\python_files\计网-实验2\8-BIT OPTIMIZERS VIA BLOCK-WISE QUANTIZATION.pdf
2023-10-27 22:34:25 local>>文件上传成功
2023-10-27 22:34:25 server>>>wolf 上传了文件: 8-BIT OPTIMIZERS VIA BLOCK-WISE QUANTIZATION.pdf
/file list
当前服务器文件列表:
8-BIT OPTIMIZERS VIA BLOCK-WISE QUANTIZATION.pdf
A甘晴void.txt
/file download M甘晴void.txt
2023-10-27 22:35:01 server>>错误: E:\python_files\计网-实验2\服务端保存文件\M甘晴void.txt: 文件不存在
/file download A甘晴void.txt
2023-10-27 22:35:13 local>>文件 A甘晴void.txt 下载成功
|
```

```
E:\anaconda\envs\python3-11\python.exe E:\python_files\计网-实验
2\chatClient.py
请输入您的用户名: void
2023-10-27 22:18:38 server>>>连接成功! 开始聊天吧2023-10-27 22:18:38
server>>>void 加入了聊天室
hello,I am void.
2023-10-27 22:22:26 void: hello,I am void.
/file list
当前服务器:无文件
/file upload E:\python_files\计网-实验2\M甘晴void.txt
2023-10-27 22:32:24 local>>错误: E:\python_files\计网-实验2\M甘晴
void.txt: 文件不存在
/file upload E:\python_files\计网-实验2\A甘晴void.txt
2023-10-27 22:33:12 local>>文件上传成功
2023-10-27 22:33:12 server>>>void 上传了文件: A甘晴void.txt
/file list
当前服务器文件列表:
A甘晴void.txt
2023-10-27 22:34:25 server>>>wolf 上传了文件: 8-BIT OPTIMIZERS VIA
BLOCK-WISE QUANTIZATION.pdf
/file download 8-BIT OPTIMIZERS VIA BLOCK-WISE QUANTIZATION.pdf
2023-10-27 22:35:39 local>>文件 8-BIT OPTIMIZERS VIA BLOCK-WISE
QUANTIZATION.pdf 下载成功
```

```
运行 chatServer x chatClient x chatClient x
E:\anaconda\envs\python3-11\python.exe E:\python_files\计网-实验2\chatClient.py
请输入您的用户名: void
2023-10-27 22:18:38 server>>>连接成功! 开始聊天吧2023-10-27 22:18:38 server>>>void 加入了聊天室
hello,I am void.
2023-10-27 22:22:26 void: hello,I am void.
/file list
当前服务器:无文件
/file upload E:\python_files\计网-实验2\M甘晴void.txt
2023-10-27 22:32:24 local>>错误: E:\python_files\计网-实验2\M甘晴void.txt: 文件不存在
/file upload E:\python_files\计网-实验2\A甘晴void.txt
2023-10-27 22:33:12 local>>文件上传成功
2023-10-27 22:33:12 server>>>void 上传了文件: A甘晴void.txt
/file list
当前服务器文件列表:
A甘晴void.txt
2023-10-27 22:34:25 server>>>wolf 上传了文件: 8-BIT OPTIMIZERS VIA BLOCK-WISE QUANTIZATION.pdf
/file download 8-BIT OPTIMIZERS VIA BLOCK-WISE QUANTIZATION.pdf
2023-10-27 22:35:39 local>>文件 8-BIT OPTIMIZERS VIA BLOCK-WISE QUANTIZATION.pdf 下载成功
|
```

再看看两个文件夹内的情况，可以发现文件的上传与下载确实成功完成了。



可以发现测试是成功的，每个细节都完美地回应了设想。

V 测试客户端退出

在这之前可以让wolf和void再说几句话

之后让客户端void使用“/exit”命令主动退出

客户端void使用/exit主动退出之后，wolf端和服务端都在终端显示了void退出的消息。（具体可以看整体数据和截图）

之后再让wolf端退出，最后结束服务器。

VI 全部测试过程终端命令行与截图

服务端

```
E:\anaconda\envs\python3-11\python.exe E:\python_files\计网-实验
2\chatServer.py
2023-10-27 22:16:44 server>>等待客户端连接在 192.168.56.103:15004...
2023-10-27 22:16:44 server>>>连接来自 ('192.168.56.103', 59606), 用户
名: wolf<<<
2023-10-27 22:19:50 wolf: hello,I am wolf.
2023-10-27 22:18:38 server>>>连接来自 ('192.168.56.103', 59902), 用户
名: void<<<
2023-10-27 22:22:26 void: hello,I am void.
2023-10-27 22:33:12 server>>文件上传成功
2023-10-27 22:34:25 server>>文件上传成功
2023-10-27 22:35:01 server>>错误: E:\python_files\计网-实验2\服务端保存
文件\M甘晴void.txt: 文件不存在
2023-10-27 22:35:13 server>>向 wolf 发送文件成功
2023-10-27 22:35:39 server>>向 void 发送文件成功
2023-10-27 22:45:22 void: OK,well computer network is really fun.
2023-10-27 22:46:20 wolf: You get it. Now I manifest dense interest
in computer network.
2023-10-27 22:46:51 void: wish you get a good mark and achieve your
score, good bye.
2023-10-27 22:47:03 wolf: good bye.
2023-10-27 22:47:08 server>>void 离开了聊天室
2023-10-27 22:48:40 server>>wolf 离开了聊天室
```

进程已结束,退出代码-1


```
运行 chatServer x chatClient x chatClient x
E:\anaconda\envs\python3-11\python.exe E:\python_files\计网-实验2\chatServer.py
2023-10-27 22:16:44 server>>等待客户端连接在 192.168.56.103:15004...
2023-10-27 22:16:44 server>>>连接来自 ('192.168.56.103', 59606), 用户名: wolf<<<
2023-10-27 22:19:50 wolf: hello,I am wolf.
2023-10-27 22:18:38 server>>>连接来自 ('192.168.56.103', 59902), 用户名: void<<<
2023-10-27 22:22:26 void: hello,I am void.
2023-10-27 22:33:12 server>>文件上传成功
2023-10-27 22:34:25 server>>文件上传成功
2023-10-27 22:35:01 server>>错误: E:\python_files\计网-实验2\服务端保存文件\M甘晴void.txt: 文件不存在
2023-10-27 22:35:13 server>>向 wolf 发送文件成功
2023-10-27 22:35:39 server>>向 void 发送文件成功
2023-10-27 22:45:22 void: OK,well computer network is really fun.
2023-10-27 22:46:20 wolf: You get it. Now I manifest dense interest in computer network.
2023-10-27 22:46:51 void: wish you get a good mark and achieve your score, good bye.
2023-10-27 22:47:03 wolf: good bye.
2023-10-27 22:47:08 server>>void 离开了聊天室
2023-10-27 22:48:40 server>>wolf 离开了聊天室

进程已结束,退出代码-1
```

客户端wolf

```
E:\anaconda\envs\python3-11\python.exe E:\python_files\计网-实验2\chatClient.py
请输入您的用户名: wolf
2023-10-27 22:16:44 server>>>连接成功! 开始聊天吧2023-10-27 22:16:44
server>>>wolf 加入了聊天室
hello,I am wolf.
2023-10-27 22:19:50 wolf: hello,I am wolf.
2023-10-27 22:18:38 server>>>void 加入了聊天室
2023-10-27 22:22:26 void: hello,I am void.
2023-10-27 22:33:12 server>>>void 上传了文件: A甘晴void.txt
/file list
当前服务器文件列表:
A甘晴void.txt
/file upload E:\python_files\计网-实验2\8-BIT OPTIMIZERS VIA BLOCK-
WISE QUANTIZATION.pdf
2023-10-27 22:34:25 local>>文件上传成功
2023-10-27 22:34:25 server>>>wolf 上传了文件: 8-BIT OPTIMIZERS VIA
BLOCK-WISE QUANTIZATION.pdf
/file list
当前服务器文件列表:
8-BIT OPTIMIZERS VIA BLOCK-WISE QUANTIZATION.pdf
A甘晴void.txt
```

```
/file download M甘晴void.txt
2023-10-27 22:35:01 server>>错误: E:\python_files\计网-实验2\服务端保存
文件\M甘晴void.txt: 文件不存在
/file download A甘晴void.txt
2023-10-27 22:35:13 local>>文件 A甘晴void.txt 下载成功
2023-10-27 22:45:22 void: OK,well computer network is really fun.
You get it. Now I manifest dense interest in computer network.
2023-10-27 22:46:20 wolf: You get it. Now I manifest dense interest
in computer network.
2023-10-27 22:46:51 void: wish you get a good mark and achieve your
score, good bye.
good bye.
2023-10-27 22:47:03 wolf: good bye.
2023-10-27 22:47:08 server>>>void 离开了聊天室
/exit
Traceback (most recent call last):
  File "E:\python_files\计网-实验2\chatClient.py", line 108, in
<module>
    message = client_socket.recv(1024).decode("utf-8")
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ConnectionAbortedError: [winError 10053] 你的主机中的软件中止了一个已建立
的连接。

进程已结束,退出代码1
```

```
运行 chatServer x chatClient x chatClient x
E:\anaconda\envs\python3-11\python.exe E:\python_files\计网-实验2\chatClient.py
请输入您的用户名: wolf
2023-10-27 22:16:44 server>>>连接成功! 开始聊天吧2023-10-27 22:16:44 server>>>wolf 加入了聊天室
hello,I am wolf.
2023-10-27 22:19:50 wolf: hello,I am wolf.
2023-10-27 22:18:38 server>>>void 加入了聊天室
2023-10-27 22:22:26 void: hello,I am void.
2023-10-27 22:33:12 server>>>void 上传了文件: A甘晴void.txt
/file list
当前服务器文件列表:
A甘晴void.txt
/file upload E:\python_files\计网-实验2\8-BIT OPTIMIZERS VIA BLOCK-WISE QUANTIZATION.pdf
2023-10-27 22:34:25 local>>文件上传成功
2023-10-27 22:34:25 server>>>wolf 上传了文件: 8-BIT OPTIMIZERS VIA BLOCK-WISE QUANTIZATION.pdf
/file list
当前服务器文件列表:
8-BIT OPTIMIZERS VIA BLOCK-WISE QUANTIZATION.pdf
A甘晴void.txt
/file download M甘晴void.txt
2023-10-27 22:35:01 server>>错误: E:\python_files\计网-实验2\服务端保存文件\M甘晴void.txt: 文件不存在
/file download A甘晴void.txt
2023-10-27 22:35:13 local>>文件 A甘晴void.txt 下载成功
2023-10-27 22:45:22 void: OK,well computer network is really fun.
You get it. Now I manifest dense interest in computer network.
2023-10-27 22:46:20 wolf: You get it. Now I manifest dense interest in computer network.
2023-10-27 22:46:51 void: wish you get a good mark and achieve your score, good bye.
good bye.
2023-10-27 22:47:03 wolf: good bye.
2023-10-27 22:47:08 server>>>void 离开了聊天室
/exit
Traceback (most recent call last):
  File "E:\python_files\计网-实验2\chatClient.py", line 108, in <module>
    message = client_socket.recv(1024).decode("utf-8")
              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ConnectionAbortedError: [WinError 10053] 你的主机中的软件中止了一个已建立的连接。

进程已结束,退出代码1
```

客户端void

```
E:\anaconda\envs\python3-11\python.exe E:\python_files\计网-实验
2\chatClient.py
请输入您的用户名: void
2023-10-27 22:18:38 server>>>连接成功! 开始聊天吧2023-10-27 22:18:38
server>>>void 加入了聊天室
hello,I am void.
2023-10-27 22:22:26 void: hello,I am void.
/file list
当前服务器:无文件
/file upload E:\python_files\计网-实验2\M甘晴void.txt
```

```
2023-10-27 22:32:24 local>>错误: E:\python_files\计网-实验2\M甘晴
void.txt: 文件不存在
/file upload E:\python_files\计网-实验2\A甘晴void.txt
2023-10-27 22:33:12 local>>文件上传成功
2023-10-27 22:33:12 server>>>void 上传了文件: A甘晴void.txt
/file list
当前服务器文件列表:
A甘晴void.txt
2023-10-27 22:34:25 server>>>wolf 上传了文件: 8-BIT OPTIMIZERS VIA
BLOCK-WISE QUANTIZATION.pdf
/file download 8-BIT OPTIMIZERS VIA BLOCK-WISE QUANTIZATION.pdf
2023-10-27 22:35:39 local>>文件 8-BIT OPTIMIZERS VIA BLOCK-WISE
QUANTIZATION.pdf 下载成功
OK,well computer network is really fun.
2023-10-27 22:45:22 void: OK,well computer network is really fun.
2023-10-27 22:46:20 wolf: You get it. Now I manifest dense interest
in computer network.
wish you get a good mark and achieve your score, good bye.
2023-10-27 22:46:51 void: wish you get a good mark and achieve your
score, good bye.
2023-10-27 22:47:03 wolf: good bye.
/exit
Traceback (most recent call last):
  File "E:\python_files\计网-实验2\chatClient.py", line 108, in
<module>
    message = client_socket.recv(1024).decode("utf-8")
              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ConnectionAbortedError: [WinError 10053] 你的主机中的软件中止了一个已建立
的连接。

进程已结束,退出代码1
```

```
运行 chatServer x chatClient x chatClient x
E:\anaconda\envs\python3-11\python.exe E:\python_files\计网-实验2\chatClient.py
请输入您的用户名: void
2023-10-27 22:18:38 server>>>连接成功! 开始聊天吧2023-10-27 22:18:38 server>>>void 加入了聊天室
hello,I am void.
2023-10-27 22:22:26 void: hello,I am void.
/file list
当前服务器:无文件
/file upload E:\python_files\计网-实验2\M甘晴void.txt
2023-10-27 22:32:24 local>>错误: E:\python_files\计网-实验2\M甘晴void.txt: 文件不存在
/file upload E:\python_files\计网-实验2\A甘晴void.txt
2023-10-27 22:33:12 local>>文件上传成功
2023-10-27 22:33:12 server>>>void 上传了文件: A甘晴void.txt
/file list
当前服务器文件列表:
A甘晴void.txt
2023-10-27 22:34:25 server>>>wolf 上传了文件: 8-BIT OPTIMIZERS VIA BLOCK-WISE QUANTIZATION.pdf
/file download 8-BIT OPTIMIZERS VIA BLOCK-WISE QUANTIZATION.pdf
2023-10-27 22:35:39 local>>文件 8-BIT OPTIMIZERS VIA BLOCK-WISE QUANTIZATION.pdf 下载成功
OK,well computer network is really fun.
2023-10-27 22:45:22 void: OK,well computer network is really fun.
2023-10-27 22:46:20 wolf: You get it. Now I manifest dense interest in computer network.
wish you get a good mark and achieve your score, good bye.
2023-10-27 22:46:51 void: wish you get a good mark and achieve your score, good bye.
2023-10-27 22:47:03 wolf: good bye.
/exit
Traceback (most recent call last):
  File "E:\python_files\计网-实验2\chatClient.py", line 108, in <module>
    message = client_socket.recv(1024).decode("utf-8")
              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ConnectionAbortedError: [WinError 10053] 你的主机中的软件中止了一个已建立的连接。

进程已结束,退出代码1
|
```

⑦程序总结与展望

感觉还是有必要给我花了很多时间做的这个chat小程序做一个总结的。

从程序功能上来看,应该是比较好地回应了需求,甚至新增了新的功能,比如列表的查看等,但这些功能也是在编写现有功能的时候觉得有必要加上去。确实是越编写,就越觉得有必要新增功能,换句话说,只有在写的时候才知道有什么新的功能是可以去优化用户体验的。

在程序实现上,发送与接收文件的分段的部分参考了A橙_学长,因为我原来怎么也想不到发送文件要考虑大多数情况下,尤其是文件很大的情况下,它一次是难以完全发送完毕的。这个时候如果不分段,就不能够正常发送。

我的程序仍然有很多没有实现,比如

- 线程的锁
- 并发处理

- 可视化（使用pyqt或者tk）
- 在服务端已满状态下向新用户发送拒绝（这个在之前2.3线程池实现了，但这里原理有点不一样，故未作处理）
- 验证用户身份（数据库介入，涉及到注册、登录等验证）
- 其他更加复杂的功能

其实可以发现这很像暑期CS课程组教我们的非常简单的全栈实现，在计网的基础上与别的学科有了交叉的融合。实际上我跟同学开玩笑说，这个chat小程序的最终终极进化对象应该是QIQC,也就是QQ，甚至实现更多的功能。

但是显然作为一门核心课的其中一个实验，做到这个程度我认为无论是从花的时间还是精力来讲，都已经足够了。因此我主要是从计网的角度去实现套接字的编程，并未涉及到其他融合的角度。我认为如果有足够的时间，我能做到，这也是很有趣的，然而现在我还有其他更多的内容需要学习。

三、实验感悟

能自己实现简单的套接字编程。

最初实现第一个TCP传输的时候，看到接收到的结果真的很开心！我感觉这个真的非常有趣，同时及也对于计算机网络的学习更加感兴趣了。

实现简单的chat程序的时候遇到了瓶颈，特别是在传输文件的部分，真的搞不懂文件的传输还需要分段，但幸运的是向A橙_学长学习到了经验，解决了这个问题。另外，这个chat程序确实倾注了很多心血，感觉最后能实现预期的结果真的很开心。

最后感谢老师与助教学姐。

2023.10.27晚 于天马学生公寓