Network_02_attachment

· 系统概述：

运行环境为Python3.7，初步实现于iPadOS操纵系统下Juno平台实现；后续优化使用Windows操纵系统下Jupyter Notebook实现。程序共包含四个文件，分别为基于TCP和UDP的客户端和服务端，具体程序如下表所示：

表1 程序文件列表

文件名	文件概述
Socket_Client_TCP.ipynb	基于TCP的客户端程序
Socket_Server_TCP.ipynb	基于TCP的服务端程序
Socket_Client_UDP.ipynb	基于UDP的客户端程序
Socket_Server_UDP.ipynb	基于UDP的服务端程序

· 主要数据结构

```
struct sockaddr {
        unsigned short sa_family; /* address family, AF_xxx */
        char sa_data[14]; /* 14 bytes of protocol address */
};
```

此数据结构用做 bind、connect、recvfrom、sendto 等函数的参数，指明地址信息。

· 主要算法

将服务端建立 socket、发接收数据句段写为 while Ture 的死循环，保证其在管理方不有意停止程序的过程中始终接收客户端的数据。

· 用户使用

服务端：首先将主机地址与端口号绑定，便开始监听客户端请求。

客户端：指定需要发送连接请求的 IP 地址，收到服务端的确认信号后即可键入登录信息，信息核对正确后便可传输数据。

·Socket 通信实验-Python 实现源代码

基于TCP：

客户端：Client

```
def socket_client():
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(('127.0.0.1',6666))
    except socket.error as msg:
        print(msg)
        sys.exit(1)
    print(s.recv(1024))
    while 1:
        data = input("please input work: ").encode()
        s.send(data)
        print('reveived from server', s.recv(1024))
        if data == 'exit':
            break
    s.close

if __name__ == '__main__':
    socket_client()
```

服务端：Server

```
def socket_server():
    try:
        # 首先调用socket函数创建一个socket
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        # 防止socket server重启端口被占用（socket.error:[Errno 98] Address already in use）
        s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        # 调用bind函数将其与本机地址以及一个端口号绑定
        s.bind(('127.0.0.1', 6666))
        # listen在相应的socket上监听
        s.listen(10)
    except socket.error as msg:
        print(msg)
        sys.exit(1)
    print("waiting connection...")

    while 1:
        conn, addr = s.accept()
        t = threading.Thread(target=deal_data, args=(conn, addr))
        t.start()
# 处理数据
```

```
def deal_data(conn, addr):
    print("Accept new connection from {0}".format(addr))
    conn.send(("Welcome to the server.").encode())
    while 1:
        data = conn.recv(1024)
        print('{0} client send data is {1}'.format(addr, data.decode()))
        time.sleep(1)
        if data == 'exit' or not data:
            print(f'{addr} connection close.')
            conn.send(bytes('Connetion closed!'), 'UTF-8')
            break
        conn.send(bytes(f'Hello, {data}', 'UTF-8'))
    conn.close()

if __name__ == '__main__':
    socket_server()
```

基于UDP：

客户端：Client

```
serverName='127.0.0.1'
serverPort=12345
clientSocket=socket(AF_INET,SOCK_DGRAM)
while 1:
    message=input().encode()
    clientSocket.sendto(message,(serverName,serverPort))
    modifiedMessage,serverAddress=clientSocket.recvfrom(2048)
    print(modifiedMessage)
clientSocket.close()
```

服务端：Server

```
from socket import *
serverPort=12345
serverSocket=socket(AF_INET,SOCK_DGRAM)
serverSocket.bind(('127.0.0.1',serverPort))
print('ready')
while 1:
    message,clientAddress=serverSocket.recvfrom(2048)
    modifiedMessage=message.upper()
    print(modifiedMessage)
    serverSocket.sendto(modifiedMessage,clientAddress)
```