

**北 京 科 技 大 学**

**计算机网络课程设计报告**

学　　院： \_\_计算机与通信工程学院\_\_

班　　级： \_\_\_\_\_\_\_\_信安182\_\_\_\_\_\_\_\_\_

姓　　名： \_\_\_\_\_\_\_\_\_梁逸飞\_\_\_\_\_\_\_\_\_\_

学　　号： \_\_\_\_\_\_\_\_41824141\_\_\_\_\_\_\_\_\_

成　　绩： \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

指导教师签字： \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**年 月**

**北京科技大学实验报告**

学院：计算机与通信工程学院 专业：信息安全 班级： 信安182

姓名：梁逸飞 学号：41824141 实验日期：2021年 4月

**实验名称：**计算机网络课程设计 实验任务二——实现同时与多个客户端通信的服务器小程序

**实验目的：**

在实验一的基础上，理解TCP/IP的通信原理与过程，搭建跨平台通信的C/S架构体系网络。同时理解并掌握并发的原理，实现并发服务器。

**实验仪器：**

实验硬件设备：华硕飞行堡垒， Intel(R) Dual Band Wireless-AC 8265，安卓手机

实验软件要求：win10， kivy虚拟机，pycharm， python3.6

**实验原理：**

实验原理：

基于SOCKET开发简单的C/S文本传输程序，socket是通信系统双方建立的一个标准约定，即“接头暗号”，其内部封装有各种网络通信所需要的功能函数。系统中的应用层功能的实现是需要通过传输层进行服务中转的，分布式的系统决定了网络系统中同一时间内应用程序可能是多个的，但是传输层只能有一个，此时传输层就需要同时为多个应用程序进程提供并发服务。socket接口能够为不同的应用程序进程和网络连接创建各自的身份标识，规范应用程序与TCP/IP协议之间的数据交互。应用程序进程间的网络通信和连接的身份主要通过目标IP地址、传输层协议和端口号三个参数标识。每个socket都绑定有IP地址、端口、端口号这3个参数。

并发服务器的实现有多种方法，可以采用多进程服务器、多线程服务器、单进程非阻塞服务器、单进程轮询服务器、单进程select、协程等方法，主要了解并发的实现机制与原理，方法不同实现机制也不同。

**实验内容与步骤：**

**题目：基于SOCKET开发简单的C/S文本传输程序**

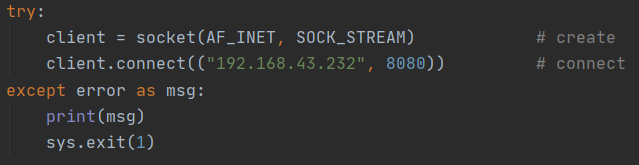
（1）需求分析

选择python语言，在实验一的基础框架上，对服务器端程序加以修改完善，使用并发服务器的模式，实现同时与多个客户端（含PC端、Android手机或苹果设备）通信的服务器小程序或APP，使他们之间能并发地实现文字、短消息、图片、音频或视频文件的传输。

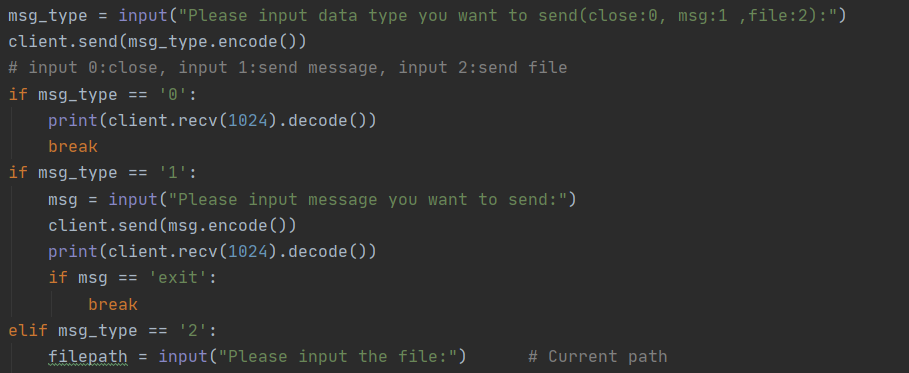
（2）设计方案

首先是客户端。

客户端的连接过程与实验一相同，但由于是python实现，所以代码量要少很多。



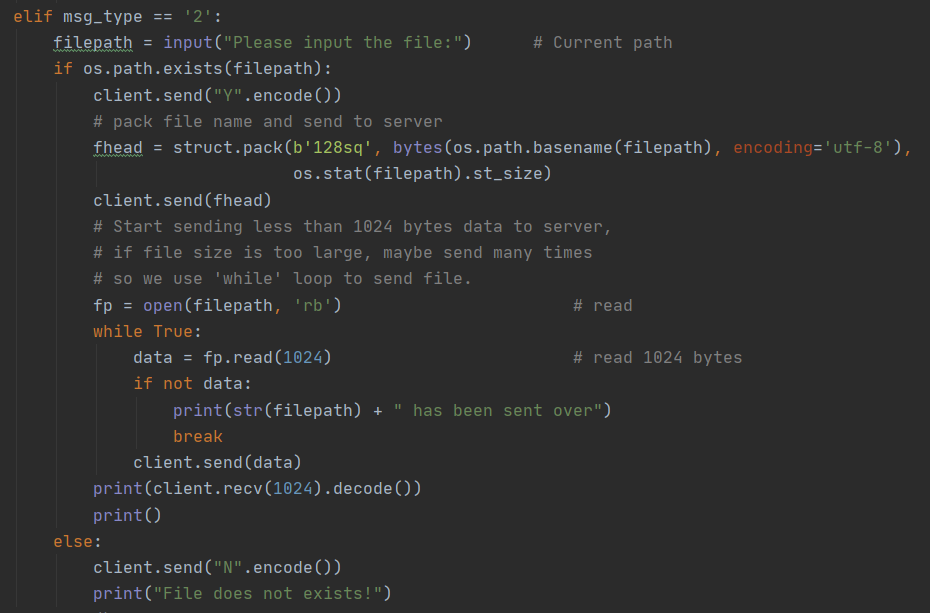
客户端在实验一的基础上增添了新的功能，即传送文件。由于文件传送较为复杂，与文本传送分开实现。当客户端连接上服务器的时候，首先输入传送的内容，0为退出，1为传送文字，2为传送文件。输入传送数据的类型，将该类型发送给服务器，服务器根据类型来接受对应的数据。



如果输入0，则直接断开连接。

如果输入1，则直接将输入的文字信息发送给服务器，此时可以输入exit断开连接。

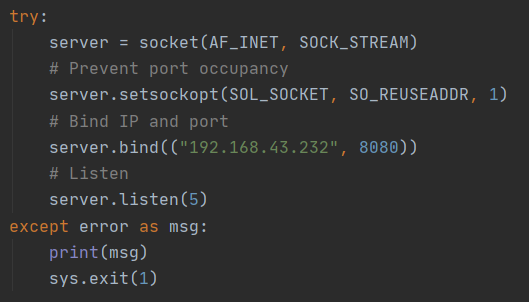
如果输入2，则需要输入客户端传送的文件名（文件路径），然后通过文件读取，读取文件的内容然后发送给服务器。



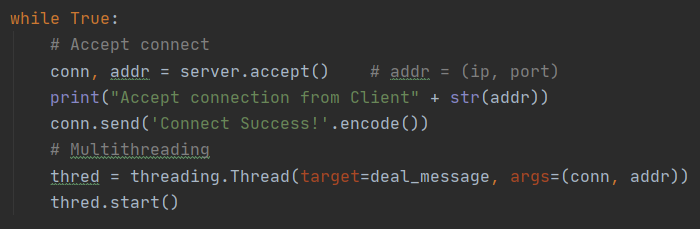
这里采用struct库函数用128sq格式对文件进行打包，发送给服务器，服务器可以根据包获取该文件的文件名和文件大小。客户端这里限制读取的文件大小为1024个字节，由于不知道传送文件的大小，所以要采用循环的方式，传送多次，当文件数据读取完成传输完成后再退出该循环。

然后是服务器。

服务器的建立和实验一相同。

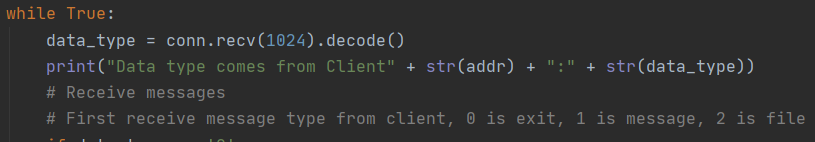


由于要实现并发服务器，这里我采用的是多线程并发服务器。

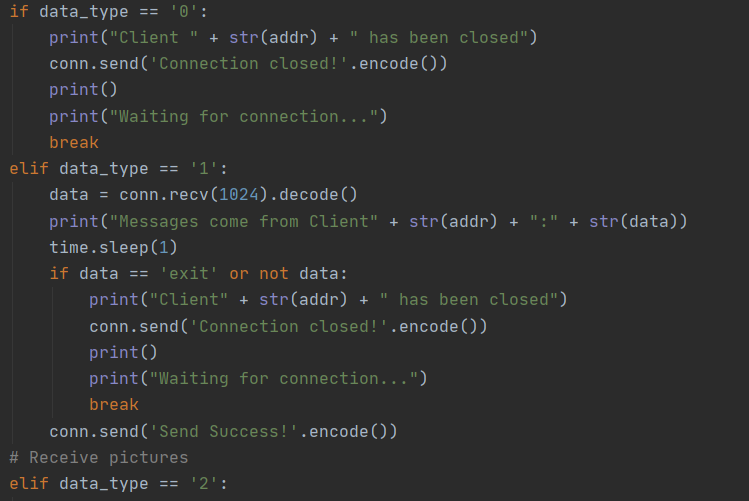


然后服务器要处理客户端发来的消息。

服务器首先要接受客户端发送过来的传送数据类型。



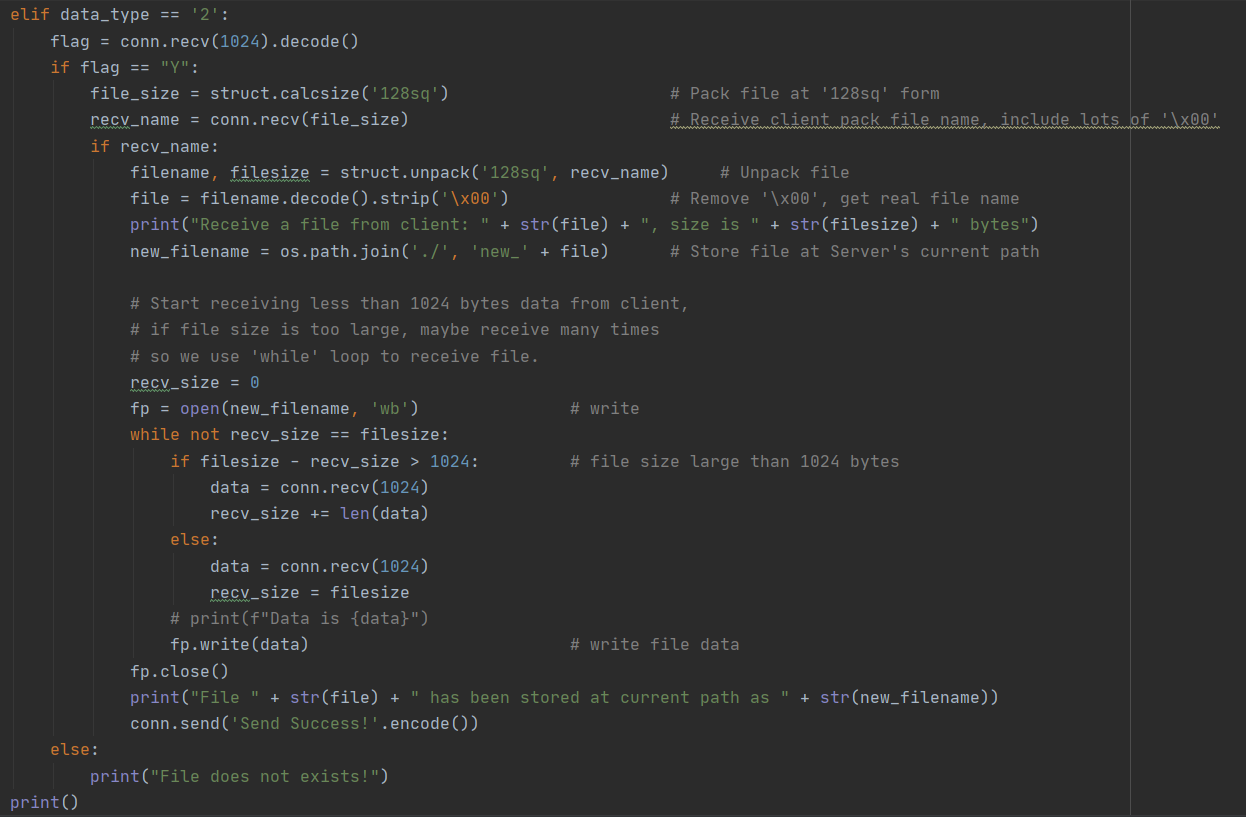
如果是0，则断开客户端的连接，如果是1，则接受文字，如果是2，则接受文件。



接受文字内容较为简单，直接接受并显示，判断如果是exit则断开连接。

如果是2，则需要接受文件，首先根据客户端发来的包，然后解包获取文件名和文件大小。解包过后的文件名会有许多的后缀\x00，需要去掉处理才能获取真正的文件名。

然后循环写文件，对应客户端的循环发送数据，并且保存到服务器的当前路径下。

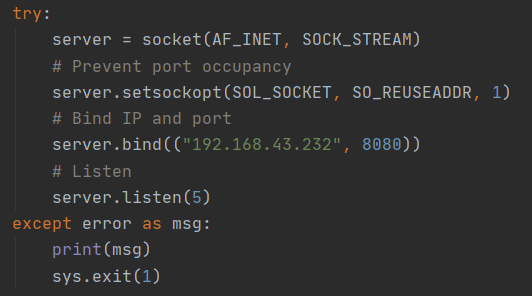


（3）异常处理

服务器客户端建立和传送的每个过程都可能会在不同的地方出现异常，所以需要对可能出现异常的地方进行异常检测，以防止程序运行造成的恶性后果。

①socket建立过程异常处理

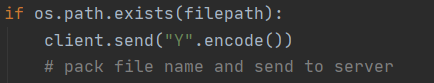
不管是服务器还是客户端，都采用try的方式进行socket建立，如有异常，则抛出except信息，并及时终止程序。



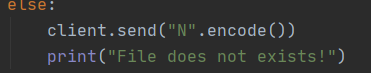
②文件传输异常处理

文件传输可能会输入一个不存在的文件，这时按照正常流程就会发生错误。

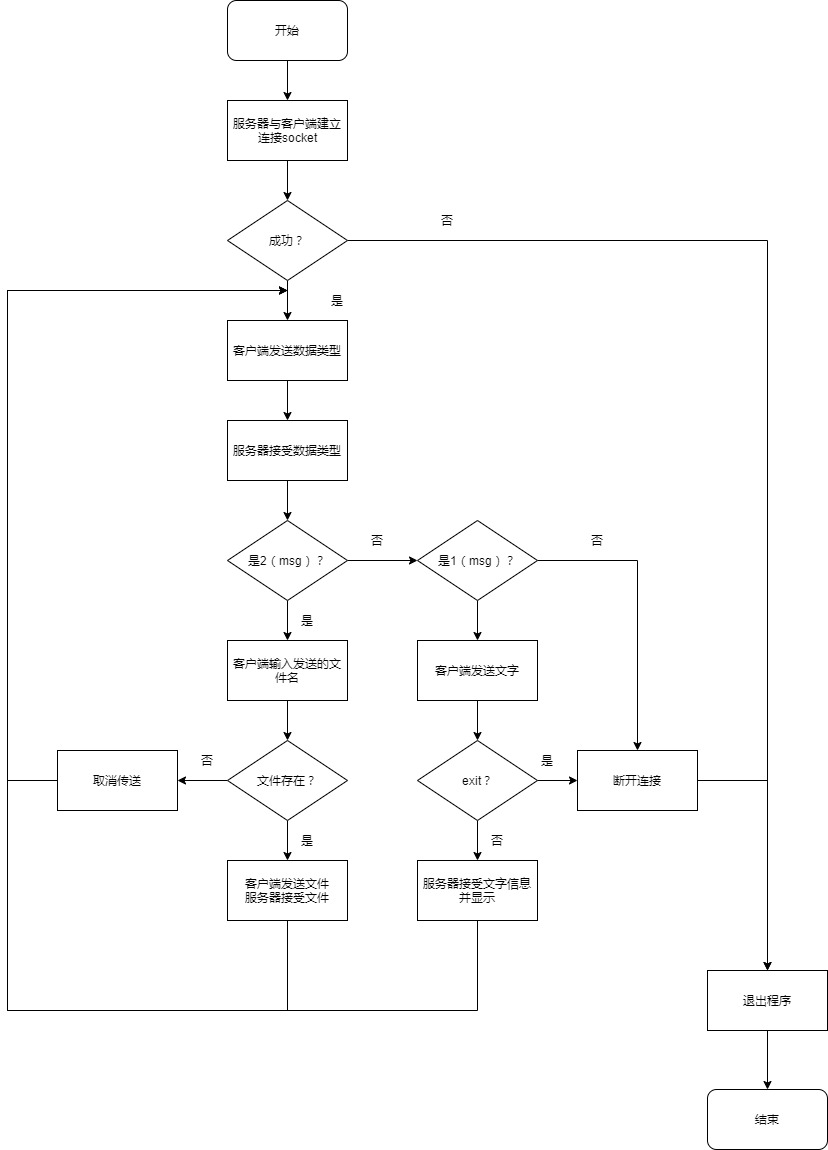
在客户端首先判断输入的文件是否存在，如果存在，则发送标志字符“Y”，告诉服务器这个文件是存在的要发送，服务器做好接受准备。



如果不存在，则向服务器发送标志字符“N”，告诉服务器这个文件是不存在的，服务器取消接受准备。



（4）程序结构和流程图



（5）制作小程序

需要用到kivy虚拟机。将两个python文件（一个服务器程序，一个客户端程序）传入虚拟机中，通过虚拟机中命令工具bulidozer android debug将python交叉编译成apk小程序。

（5）测试方案

打开手机热点，电脑连接手机热点，并且服务器绑定此时电脑的IP地址，保证在同一局域网。

①先启动服务器，再启动客户端1。

②连接成功后客户端1输入0，观察现象。

③连接成功后客户端1输入1，“hello Server， I am Client1”，观察现象。

④连接成功后在客户端1路径下创建1.txt文件，输入2，1.txt，观察现象。

⑤同时连接客户端1和客户端2，客户端1输入1，“hello Server， I am Client-1”，客户端2输入1，“hello Server， I am Client-2”，观察现象。

⑥安卓手机启动客户端，输入1，“hello Server， I am Android”，观察现象。

**实验数据：**

输入数据：

①无输入，仅连接。

②连接成功后客户端1中输入0。

③连接成功后客户端1输入1，“hello Server， I am Client1”。

④连接成功后客户端1输入2，1.txt。

⑤同时连接后客户端1输入1，“hello Server， I am Client-1”，客户端2输入1，“hello Server， I am Client-2”。

⑥安卓手机客户端，输入1，“hello Server， I am Android”。

预期输出数据：

①服务器和客户端均显示连接成功。

②服务器客户端断开连接。

③服务器显示“hello Server， I am Client1”。

④服务器路径下有new\_1.txt文件。

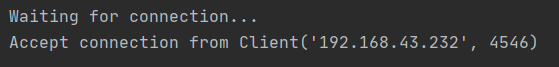
⑤服务器同时显示“hello Server， I am Client-1”和“hello Server， I am Client-2”。

⑥服务器显示“hello Server， I am Android”。

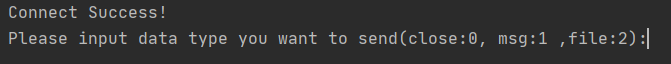
**实验结果与分析：**

（1）程序运行结果

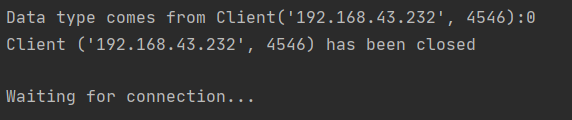
①服务器



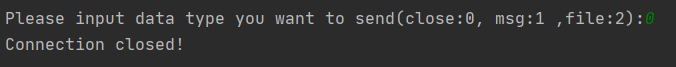
客户端1



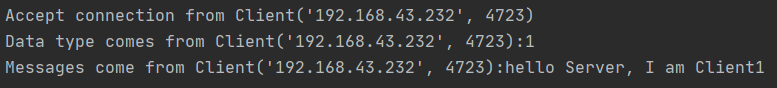
②服务器



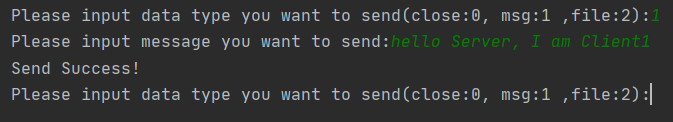
客户端1



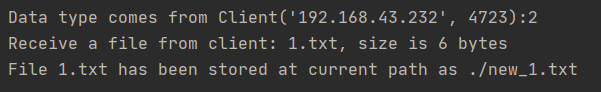
③服务器



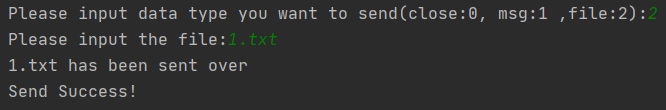
客户端1



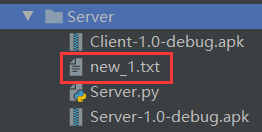
④服务器



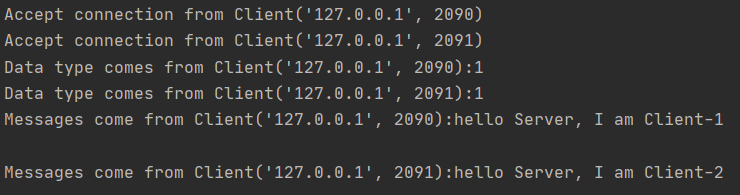
客户端1



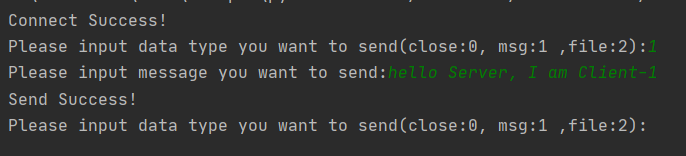
服务器1路径下



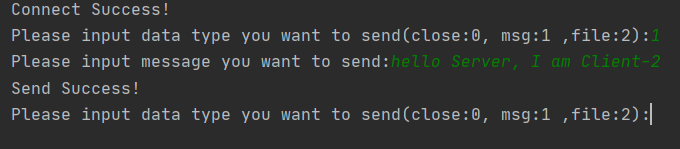
⑤服务端



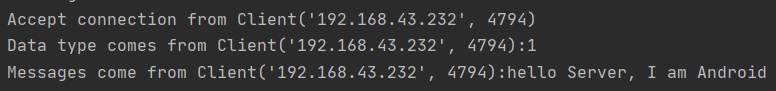
客户端1



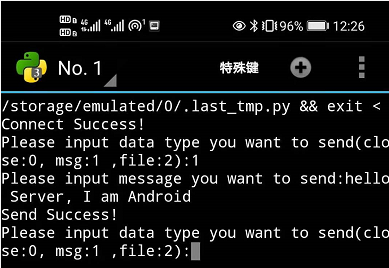
客户端2



⑥服务端



安卓客户端



（2）运行结果分析

通过观察运行结果发现，服务器与客户端能正常连接并发送文字与文件内容，能同时接收不同客户端的消息，说明服务器已经实现了并发。

电脑启动服务器，在手机上运行客户端，同样能实现手机与电脑的连接，并且也能发送消息。

运行结果说明我们已经实现了并发的跨平台的C/S。

（3）实验结论

通过本次实验了解并掌握了并发的概念，能通过代码实现并发服务器。通过kivy虚拟机能够将python代码交叉编译成apk小程序在安卓手机上运行，实现跨平台的通信。

（4）讨论

本次实验基于实验一，弥补了实验一单进程的缺点，实现了并发服务器，同时还能跨平台进行通信。

**重要程序代码：**

|  |
| --- |
| *'''  Server.py  Using multithreading concurrent blokcing '''* from socket import \* import os import threading import time import sys import struct # pack file   def socket\_server():  try:  server = socket(AF\_INET, SOCK\_STREAM)  # Prevent port occupancy  server.setsockopt(SOL\_SOCKET, SO\_REUSEADDR, 1)  # Bind IP and port  server.bind(("192.168.43.232", 8080)) # 127.0.0.1  # Listen  server.listen(5)  except error as msg:  print(msg)  sys.exit(1)   print("Waiting for connection...")   while True:  # Accept connect  conn, addr = server.accept() # addr = (ip, port)  print("Accept connection from Client" + str(addr))  conn.send('Connect Success!'.encode())  # Multithreading  thred = threading.Thread(target=deal\_message, args=(conn, addr))  thred.start()   def deal\_message(conn, addr):  while True:  data\_type = conn.recv(1024).decode()  print("Data type comes from Client" + str(addr) + ":" + str(data\_type))  # Receive messages  # First receive message type from client, 0 is exit, 1 is message, 2 is file  if data\_type == '0':  print("Client " + str(addr) + " has been closed")  conn.send('Connection closed!'.encode())  print()  print("Waiting for connection...")  break  elif data\_type == '1':  data = conn.recv(1024).decode()  print("Messages come from Client" + str(addr) + ":" + str(data))  time.sleep(1)  if data == 'exit' or not data:  print("Client" + str(addr) + " has been closed")  conn.send('Connection closed!'.encode())  print()  print("Waiting for connection...")  break  conn.send('Send Success!'.encode())  # Receive pictures  elif data\_type == '2':  flag = conn.recv(1024).decode()  if flag == "Y":  file\_size = struct.calcsize('128sq') # Pack file at '128sq' form  recv\_name = conn.recv(file\_size) # Receive client pack file name, include lots of '\x00'  if recv\_name:  filename, filesize = struct.unpack('128sq', recv\_name) # Unpack file  file = filename.decode().strip('\x00') # Remove '\x00', get real file name  print("Receive a file from client: " + str(file) + ", size is " + str(filesize) + " bytes")  new\_filename = os.path.join('./', 'new\_' + file) # Store file at Server's current path   # Start receiving less than 1024 bytes data from client,  # if file size is too large, maybe receive many times  # so we use 'while' loop to receive file.  recv\_size = 0  fp = open(new\_filename, 'wb') # write  while not recv\_size == filesize:  if filesize - recv\_size > 1024: # file size large than 1024 bytes  data = conn.recv(1024)  recv\_size += len(data)  else:  data = conn.recv(1024)  recv\_size = filesize  # print(f"Data is {data}")  fp.write(data) # write file data  fp.close()  print("File " + str(file) + " has been stored at current path as " + str(new\_filename))  conn.send('Send Success!'.encode())  else:  print("File does not exists!")  print()  conn.close()   if \_\_name\_\_ == '\_\_main\_\_':  socket\_server() |

服务器Server.oy

|  |
| --- |
| *'''  Client.py '''* from socket import \* import os import sys import struct   def socket\_client():  try:  client = socket(AF\_INET, SOCK\_STREAM) # create  client.connect(("192.168.43.232", 8080)) # connect  except error as msg:  print(msg)  sys.exit(1)   print(client.recv(1024).decode())  while True:  msg\_type = input("Please input data type you want to send(close:0, msg:1 ,file:2):")  client.send(msg\_type.encode())  # input 0:close, input 1:send message, input 2:send file  if msg\_type == '0':  print(client.recv(1024).decode())  break  if msg\_type == '1':  msg = input("Please input message you want to send:")  client.send(msg.encode())  print(client.recv(1024).decode())  if msg == 'exit':  break  elif msg\_type == '2':  filepath = input("Please input the file:") # Current path  if os.path.exists(filepath):  client.send("Y".encode())  # pack file name and send to server  fhead = struct.pack(b'128sq', bytes(os.path.basename(filepath), encoding='utf-8'),  os.stat(filepath).st\_size)  client.send(fhead)  # Start sending less than 1024 bytes data to server,  # if file size is too large, maybe send many times  # so we use 'while' loop to send file.  fp = open(filepath, 'rb') # read  while True:  data = fp.read(1024) # read 1024 bytes  if not data:  print(str(filepath) + " has been sent over")  break  client.send(data)  print(client.recv(1024).decode())  print()  else:  client.send("N".encode())  print("File does not exists!")  client.close()   if \_\_name\_\_ == '\_\_main\_\_':  socket\_client() |

客户端Client.py

**指导教师评语：（空白，半页A4）**