****

网络编程课程设计

实 验 报 告

|  |  |
| --- | --- |
| **学 院** | 网络空间安全学院 |
| **专 业** | 信息安全 |
| **班 级** | 一班 |
| **学 号** | 16272102 |
| **学生姓名** | 马皓然 |
| **教师姓名** | 胡伟通 |
| **完成日期** | 2018/12 |
| **成 绩** |  |

**实验三 不同网络架构下的文件下载服务协议设计**

1. **实验目的**
2. 熟悉不同网络架构的特点。
3. 掌握不同网络架构下的文件下载服务协议设计。
4. **实验内容**

使用TCP协议设计一个文件下载服务协议，客户端发送要下载的文件路径给服务器，服务器将对应的文件内容送给客户端，客户端将文件存储到本地磁盘。注意，当文件不存在时给出提示。要求，服务的实现分别采用以下三种方法实现：

(1)单线程，迭代服务器(依次服务每一个客户端)

(2)多线程，并发服务器

(3)异步方式(select模型或poll模型)

(4)asyncio库

例如：

$ python download\_server.py [-h] [-p port] host

$ python download\_client.py [-h] [-p port] host remote\_file local\_file

完成将服务器的文件remote\_file下载到本地，命名为local\_file

1. **程序设计思路**
2. **所选题目说明：**

使用tcp协议完成的文件下载服务，服务器端分别采用四种方法。

1. **网络应用拓扑结构**

客户端

单线程服务器

保持连接直到客户端关闭

客户端1

线程1

多线程服务器

客户端2

线程2

客户端1

Select监听

Select服务器

客户端2

客户端1

消息循环

Asyncio服务器

客户端2

1. **应用层协议设计**

{“字段名”: “值”}#

Json格式，报文以#结束

1. **所选用的Python库介绍**

Argparse 运行程序是输入参数

Socket socket通信

Json 处理数据格式

Os 查看文件是否存在

base64 处理图片视频等文件

asyncio asyncio模型需要

select select模型需要

1. **程序源代码**

**注意源代码要有详细的注释。 同学们提交的每个程序都应该遵循Honor Code（诚实代码保证）的要求。**

**请大家特别注意一定要在每个程序首部的注释中加上以下保证：**

**# 我真诚地保证：**

**# 我自己独立地完成了整个程序从分析、设计到编码的所有工作。**

**# 如果在上述过程中，我遇到了什么困难而求教于人，那么，我将在程序实习报告中**

**# 详细地列举我所遇到的问题，以及别人给我的提示。**

**# 在此，我感谢 XXX, …, XXX对我的启发和帮助。下面的报告中，我还会具体地提到**

**# 他们在各个方法对我的帮助。**

**# 我的程序里中凡是引用到其他程序或文档之处，**

**# 例如教材、课堂笔记、网上的源代码以及其他参考书上的代码段,**

**# 我都已经在程序的注释里很清楚地注明了引用的出处。**

**# 我从未没抄袭过别人的程序，也没有盗用别人的程序，**

**# 不管是修改式的抄袭还是原封不动的抄袭。**

**# 我编写这个程序，从来没有想过要去破坏或妨碍其他计算机系统的正常运转。**

**# 马皓然**

1. **服务器端源码**

#utiles.py

import socket, json, os, base64

PATH = "./srcfile/"

#从服务器上下载文件

def download(srcfile, dstfile):

if (os.path.getsize(file) > 65000):

return {"success":"false","error":"file is too big to transfer!"}

try:

#读取文件

fin = open(PATH + srcfile, 'rb')

content = fin.read()

fin.close()

return {"success":"true", "srcfile":srcfile, "dstfile":dstfile, "content":base64.b64encode(content).decode("ascii")}

#return {"success":"true", "srcfile":srcfile, "dstfile":dstfile, "content":content.decode("ascii")}

except:

return {"success":"false","error":"file is not exist!"}

#json格式发送数据

def sendData(sc, sendDataBuf):

# print("send {} to address {}".format(sendDataBuf, sc))

sendDataBuf = json.dumps(sendDataBuf) + "#"

try:

sc.sendall(sendDataBuf.encode("ascii"))

except TypeError:

sc.close()

except ConnectionResetError:

sc.close()

#json格式接收数据

def recvData(sc):

data = ""

more = ""

while more != "#":

try:

data += more

more = sc.recv(1)

more = more.decode("ascii")

except Exception as e:

return e

data = json.loads(data)

return data

#单线程

import argparse, socket, json, os, base64

import utils

HOST = "127.0.0.1"

PORT = 1060

PATH = "./srcfile/"

#单线程方式

def single(sc):

while True:

data = utils.recvData(sc)

try:

#客户端异常关闭

if( data["serverMode"] == "offline" ):

print("client {} offline".format(sc))

sc.close()

return

elif( data["serverMode"] == "single" ):

sendDataBuf = utils.download(data["srcfile"], data["dstfile"])

utils.sendData(sc, sendDataBuf)

else:

print(data)

sendDataBuf = {"success":"false","error":"this is single server, please choose single server mode"}

utils.sendData(sc, sendDataBuf)

except:

#客户端异常关闭

print(data)

sc.close()

return

def server():

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

sock.setblocking(True)

sock.bind((HOST,PORT))

sock.listen()

print("listening at", sock.getsockname())

while True:

sc, sockname = sock.accept()

print("Connected by {}".format(sockname))

single(sc)

sc.close()

sock.close()

if \_\_name\_\_ == "\_\_main\_\_":

server()

#多线程

import argparse, socket, json, os, base64, threading, utils

HOST = "127.0.0.1"

PORT = 1060

PATH = "./srcfile/"

#多线程方式

def multi(sc):

while True:

data = utils.recvData(sc)

try:

#客户端异常关闭

if( data["serverMode"] == "offline" ):

print("client {} offline".format(sc))

sc.close()

return

elif( data["serverMode"] == "multi" ):

print(data)

sendDataBuf = utils.download(data["srcfile"], data["dstfile"])

utils.sendData(sc, sendDataBuf)

else:

print(data)

sendDataBuf = {"success":"false","error":"this is multi server, please choose multi server mode"}

utils.sendData(sc, sendDataBuf)

except:

#客户端异常关闭

print(data)

sc.close()

return

def server():

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

sock.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)

sock.bind((HOST,PORT))

sock.listen()

print("listening at", sock.getsockname())

while True:

sc, sockname = sock.accept()

print("Connected by {}".format(sockname))

multiThread = threading.Thread(target = multi, args=(sc,))

multiThread.start()

sock.close()

if \_\_name\_\_ == "\_\_main\_\_":

server()

#select

import argparse, socket, json, os, base64, select

import utils

HOST = "127.0.0.1"

PORT = 1060

PATH = "./srcfile/"

socket\_list = []

#select模型

def selectModel(sc):

data = utils.recvData(sc)

try:

#客户端异常关闭

if( data["serverMode"] == "offline" ):

print("client {} offline".format(sc))

socket\_list.remove(sc)

sc.close()

return

elif( data["serverMode"] == "select" ):

sendDataBuf = utils.download(data["srcfile"], data["dstfile"])

utils.sendData(sc, sendDataBuf)

else:

print(data)

sendDataBuf = {"success":"false","error":"this is select server, please choose select server mode"}

utils.sendData(sc, sendDataBuf)

except:

#客户端异常关闭

print(data)

socket\_list.remove(sc)

sc.close()

return

def server():

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

sock.bind((HOST, PORT))

sock.listen(5)

socket\_list.append(sock)

while True:

try:

readable, writeable, exceptional = select.select(socket\_list, [], [])

except Exception as e:

print(e)

continue

for sc in readable:

if sc == sock:

conn, addr = sock.accept()

socket\_list.append(conn)

else:

try:

selectModel(sc)

except:

socket\_list.remove(sc)

if \_\_name\_\_ == "\_\_main\_\_":

server()

#asyncio

HOST = "127.0.0.1"

PORT = 1060

PATH = "./srcfile/"

def sendJson(sendDataBuf):

sendDataBuf = json.dumps(sendDataBuf) + "#"

sendDataBuf = sendDataBuf.encode("ascii")

return sendDataBuf

class ZenServer(asyncio.Protocol):

def connection\_made(self, transport):

self.transport = transport

self.address = transport.get\_extra\_info('peername')

self.data = ""

print('Accepted connection from {}'.format(self.address))

def data\_received(self, more):

try:

self.data += more.decode("ascii")

except Exception as e:

sendDataBuf = { "success": "error", "error": e }

self.transport.write(sendJson(sendDataBuf))

self.data = ""

if more.endswith(b"#"):

self.data = self.data[0:-1]

jsonData = json.loads(self.data)

if ( jsonData["serverMode"] == "asyncio" ):

sendDataBuf = utils.download(jsonData["srcfile"], jsonData["dstfile"])

self.transport.write(sendJson(sendDataBuf))

else:

sendDataBuf = {"success":"false","error":"this is asyncio server, please choose asyncio server mode"}

self.transport.write(sendJson(sendDataBuf))

self.data = ""

def connection\_lost(self, exc):

if exc:

print('Client {} error: {}'.format(self.address, exc))

elif self.data:

print('Client {} sent {} but then closed'

.format(self.address, self.data))

else:

print('Client {} closed socket'.format(self.address))

if \_\_name\_\_ == "\_\_main\_\_":

loop = asyncio.get\_event\_loop()

address = (HOST, PORT)

coro = loop.create\_server(ZenServer, \*address)

server = loop.run\_until\_complete(coro)

print('Listening at {}'.format(address))

try:

loop.run\_forever()

finally:

server.close()

loop.close()

1. **客户端源码**

#fileDownload.py

import argparse, socket, threading, json, base64

PATH = "./dstfile/"

BUFSIZE = 65535

HOST = "127.0.0.1"

PORT = 1060

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

try:

sock.connect((HOST, PORT))

except Exception as e:

print(e)

#整理格式

def formatserverMode(serverMode):

if serverMode == "单线程":

serverMode = "single"

elif serverMode == "多线程":

serverMode = "multi"

elif serverMode == "select":

serverMode == "select"

elif serverMode == "asyncio":

serverMode = "asyncio"

return serverMode

#发送源文件名

def sendsrcfile(serverMode, srcfile, dstfile):

sendDataBuf = { "serverMode":formatserverMode(serverMode), "srcfile":srcfile, "dstfile":dstfile }

return sendData(sendDataBuf)

#将文件保存到本地

def client\_download(dstfile, data):

fout = open(PATH + dstfile, 'wb')

fout.write(base64.b64decode(data.encode("ascii")))

fout.close()

#接收消息

def recvData():

data = ""

more = ""

while more != "#":

try:

data += more

more = sock.recv(1)

more = more.decode("ascii")

except ConnectionResetError:

return {"funSelec":"ConnectionResetError", "error":"server offline"}

except Exception as e:

return { "success": "error", "error": e }

data = json.loads(data)

return data

#发送消息

def sendData(data):

data = json.dumps(data) + "#"

try:

sock.sendall(data.encode("ascii"))

return "True"

except:

return "False"

def sendOffline():

sendDataBuf = { "serverMode":"offline" }

sendData(sendDataBuf)

#客户端

from tkinter import \*

from tkinter.messagebox import \*

from tkinter import ttk

import fileDownload\_client, threading

root = Tk()

root.title("藿香文件下载")

width = 400

height = 250

screenwidth = root.winfo\_screenwidth()

screenheight = root.winfo\_screenheight()

alignstr = '%dx%d+%d+%d' % (width, height, (screenwidth-width)/2, (screenheight-height)/2)

root.geometry(alignstr)

root.resizable(width=False,height=False)

#root.attributes("-alpha", 0.95) #窗口透明

class fileDownloadPage(Frame):

def \_\_init\_\_(self):

super().\_\_init\_\_()

#开启子线程，下载文件

deal\_Thread = threading.Thread(target = self.deal)

deal\_Thread.start()

self.serverMode = StringVar()

self.srcfile = StringVar()

self.dstfile = StringVar()

self.download()

def download(self):

#Label(root, text = "可下载文件列表").grid(row=0,column=0, columnspan=3, pady=5)

#self.messageList = Text(root, width=60, height=35, borderwidth=4, state="disabled")

#self.messageList.grid(row=1, columnspan=3, rowspan=8)

Label(root, text = "服务器模式").grid(row=2, column=3, pady=25, padx=35, sticky=W)

serverModelist = ttk.Combobox(root, width=21, textvariable=self.serverMode)

serverModelist["values"]=("单线程","多线程","select","asyncio")

serverModelist.current(0) #选择第一个

serverModelist.grid(row=2, column=4, sticky=E, padx=9)

Label(root, text = "源文件名").grid(row=3, column=3, pady=5, padx=35, sticky=W)

srcfileEntry = Entry(root, textvariable=self.srcfile, width=30).grid(row=3, column=4, pady=5)

Label(root, text = "目标文件名").grid(row=4, column=3, pady=5, padx=35, sticky=W)

dstfileEntry = Entry(root, textvariable=self.dstfile, width=30).grid(row=4, column=4, pady=20)

Button(root, text="下载", command=self.send, width=35).grid(row=5, column=3, columnspan=3, padx=70, pady=10)

def send(self):

serverMode = self.serverMode.get()

srcfile = self.srcfile.get()

dstfile = self.dstfile.get()

if srcfile == "":

showinfo(title="error", message="source filename can not be empty")

elif dstfile == "":

showinfo(title="error", message="destination filename can not be empty")

else:

if serverMode in ["单线程","多线程","select","asyncio"]:

flag = fileDownload\_client.sendsrcfile(serverMode, srcfile, dstfile)

if flag == "True":

showinfo(title="downloading", message="download {}, send request successfully".format(srcfile))

else:

showinfo(title="error", message="fail to connect server")

showinfo(title="error", message="current window will be closed")

root.destroy()

else:

showinfo(title="error", message="please choose serverMode from the list")

#下载文件线程

def deal(self):

while True:

data = fileDownload\_client.recvData()

if data["success"] == "true" :

fileDownload\_client.client\_download(data["dstfile"], data["content"])

showinfo(title="success", message="File {} is download to {} successfully".format(data["srcfile"], data["dstfile"]))

elif data["success"] == "false":

showinfo(title="error", message=data["error"])

def callback():

try:

fileDownload\_client.sendOffline()

except:

showinfo(title="error", message="fail to connect server")

showinfo(title="error", message="current window will be closed")

root.destroy()

fileDownloadPage()

root.protocol("WM\_DELETE\_WINDOW", callback)

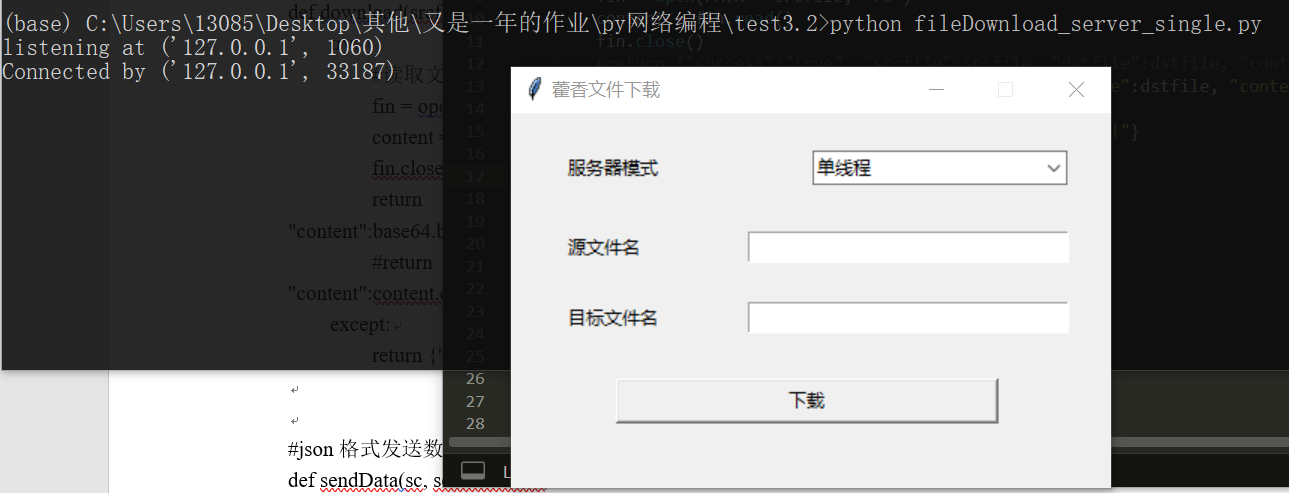
root.mainloop()

1. **程序测试方法及测试结果记录（不能光截图，要有相应的文字说明）**
2. **测试方法**

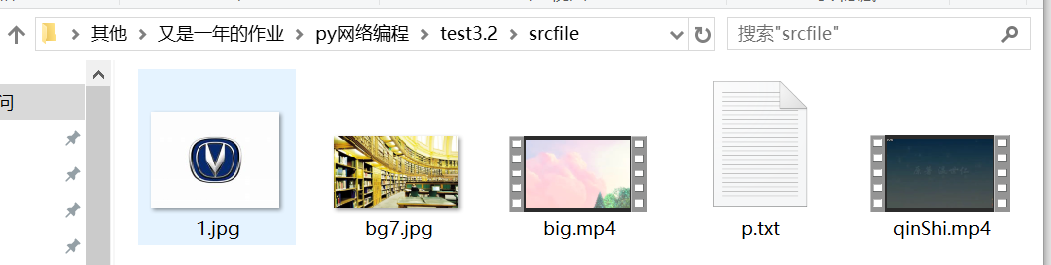
**下载文件，测试单线程服务器是否不能为第二个客户提供服务，以及多线程、select、asyncio服务器是否能为多个客户服务**

1. **测试流程**

**(1)单线程**



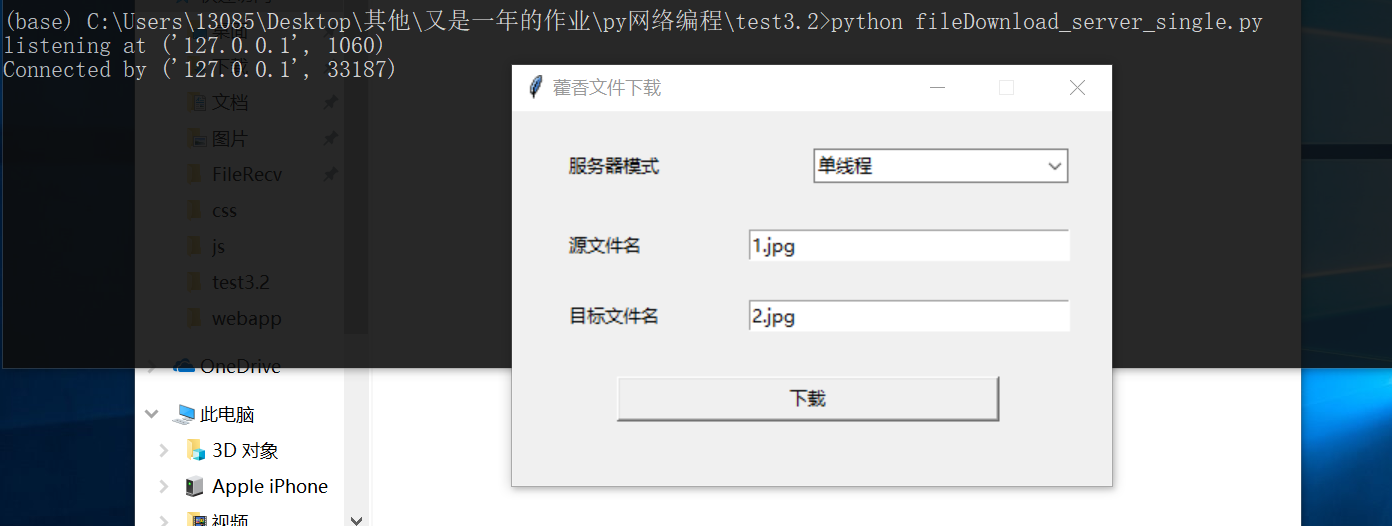
* 1. 源文件夹如下

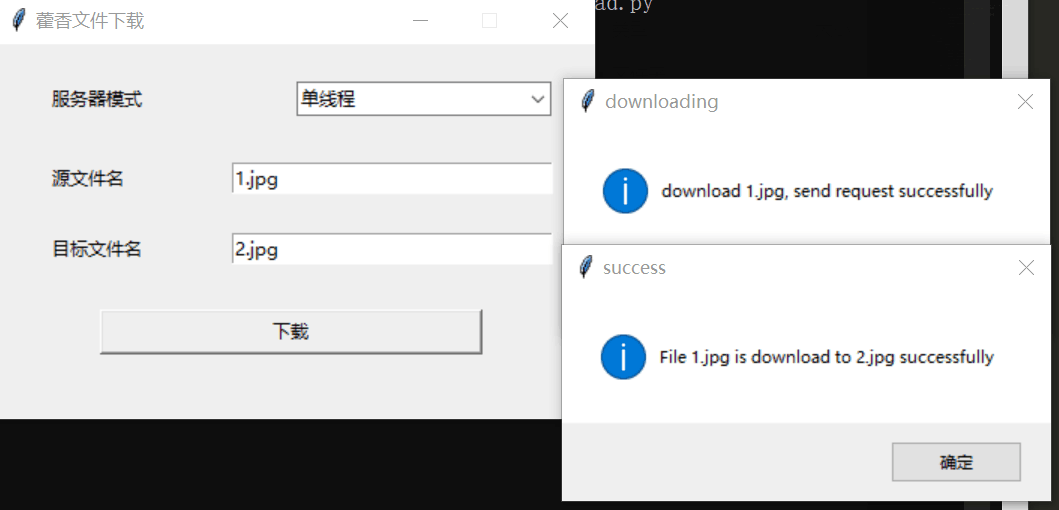


当前目标文件夹为空



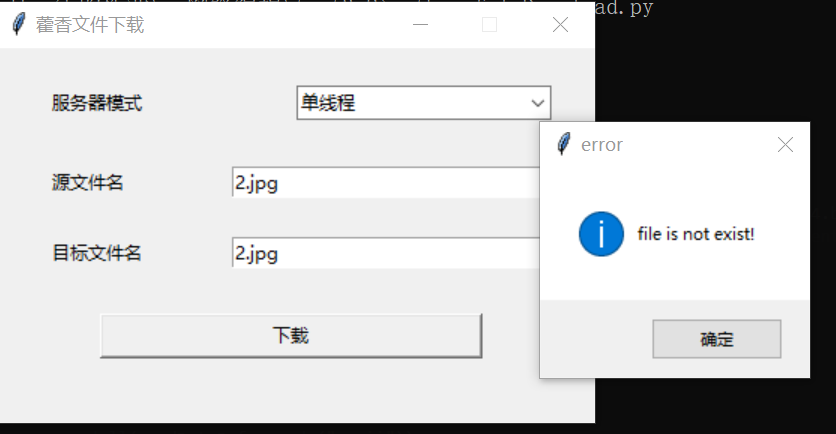
* 1. 下载1.jpg，命名为2.jpg



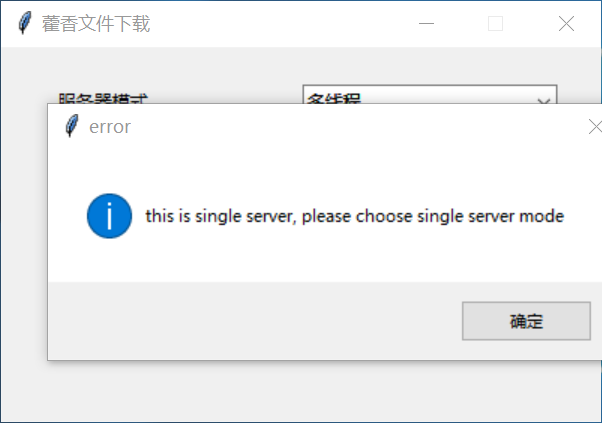




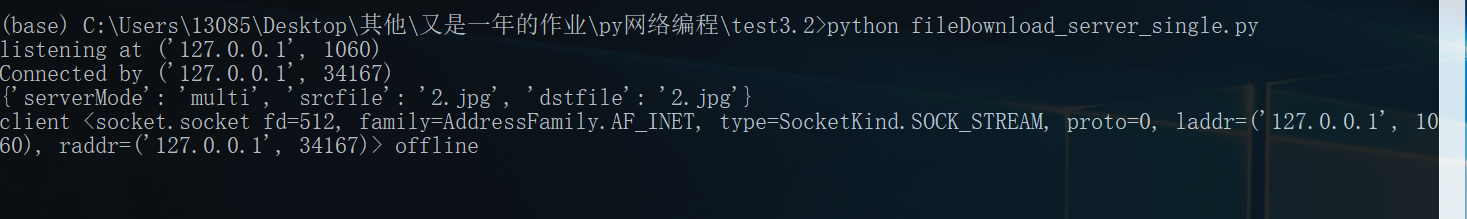
* 1. 下载不存在的文件



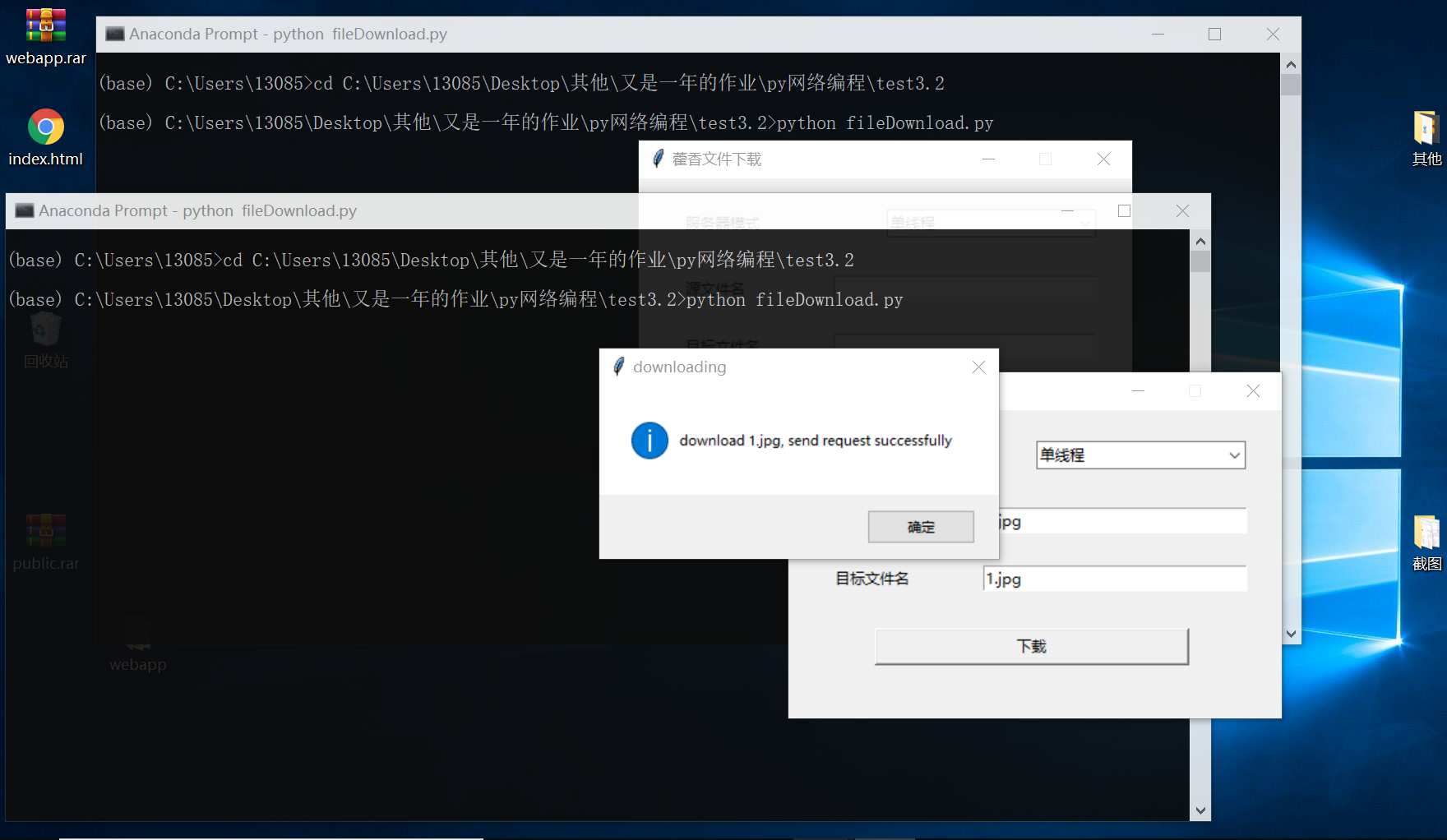
* 1. 模式选择错误



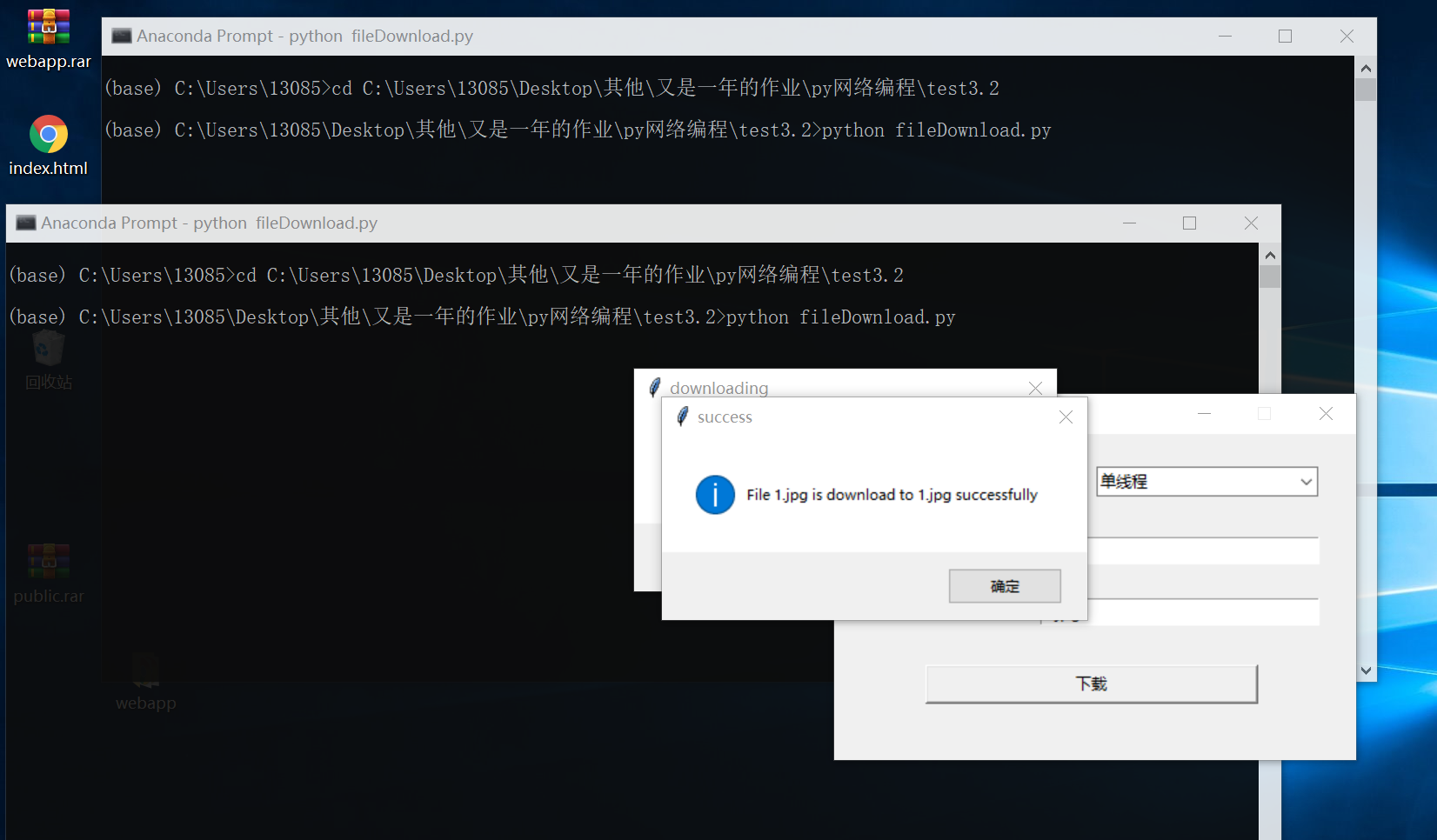
* 1. 客户关闭窗口



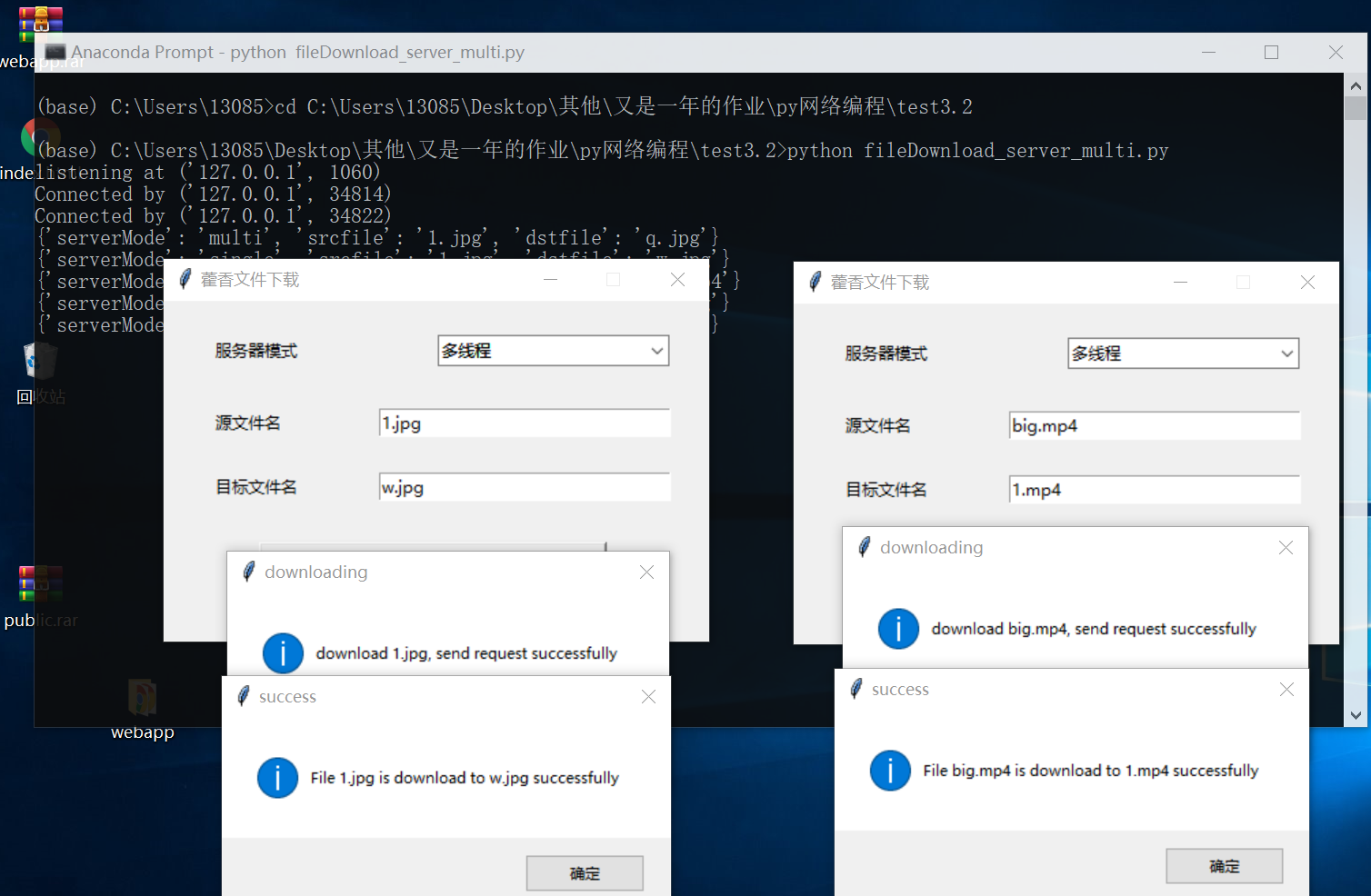
* 1. 连接两个客户端，第二个客户端请求下载文件，弹框表示请求信息发送成功，但服务器没有回应



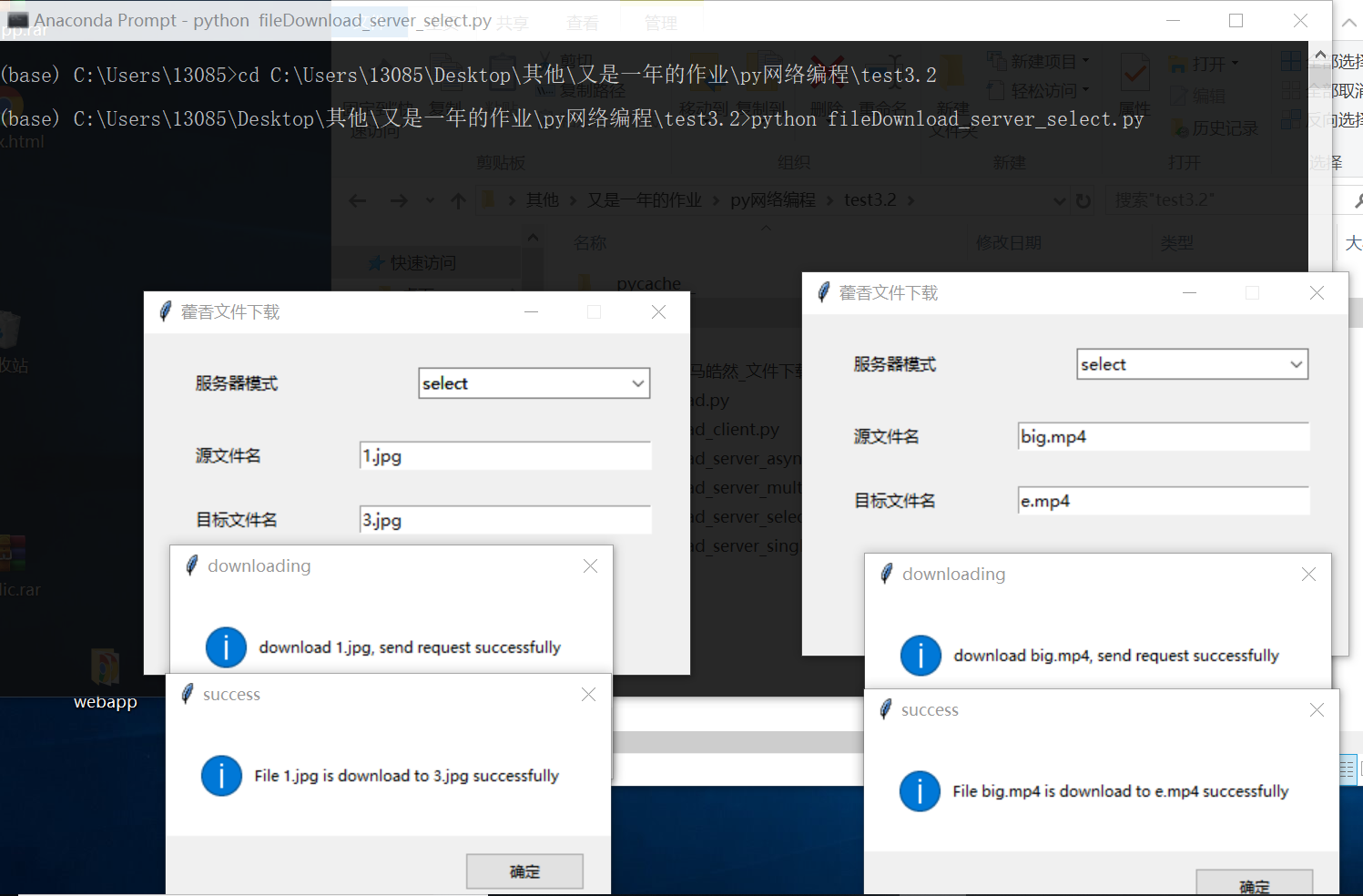
* 1. 退出第一个客户端，第二个客户下载成功



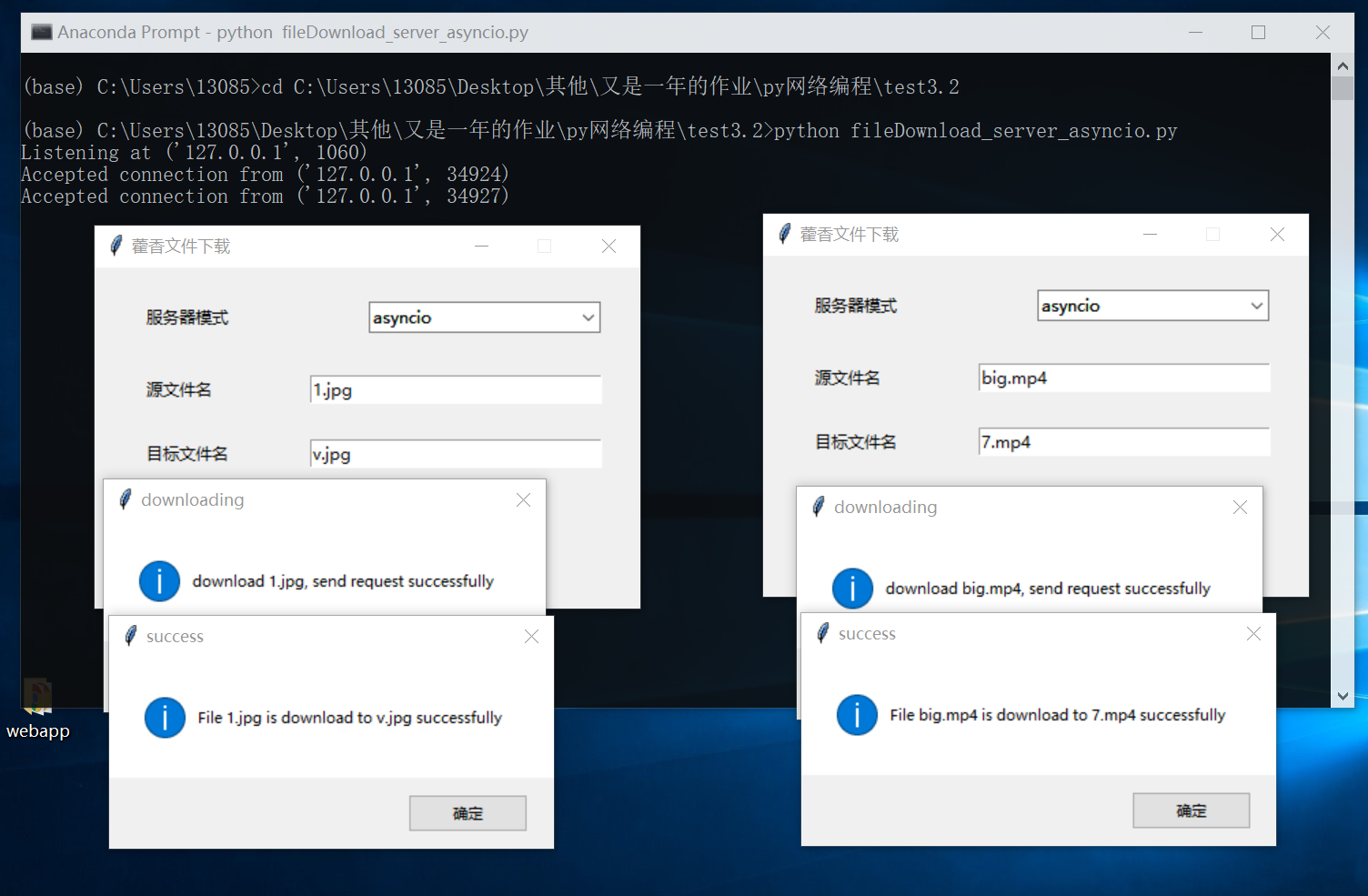
(2)多线程，两个客户端同时下载,big.mp4传输用时较长，在一个客户端接收big.mp4时，另一个客户端下载1.jpg成功



(3)select



(4)asyncio

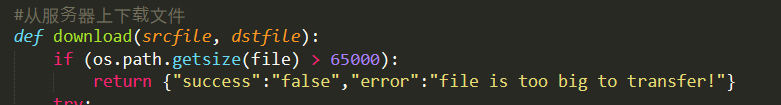


1. **实验分析总结及心得**

（结合所学知识对实验过程中观察到的实验结果进行分析总结，以便加深对知识的理解，并总结通过实验学到的知识或技术）

用json传输数据时，字段的值不能是字节串，而将图片以字节串读入编码成ascii会出错，所以中间垫了一步base64，先转base64再编码成ascii

对于文件过大的问题，直接在下载时判断大小，如果过大则不传输



如果在应用层不分片，传输层tcp是可靠传输服务，可以保证数据到达且不乱序