# XH-2XH-2云南大学软件学院期末课程报告

Final Course Report

School of Software, Yunnan University

**个人成绩**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **学号** | **姓名** | **成绩** |
| **1** | **20211060203** | **周楷翔** |  |
| **2** | **20211060143** | **李俊杰** |  |
| **3** | **20211060245** | **陈俊宏** |  |

学　　期: 　 **2023** 年 **春** 季学期

课程名称: 　 　 计算机网络

任课教师: 　 李海

实践题目: FTP服务器的设计和实现

组长姓名: 　 周楷翔

联系电话: 　 19169379730

电子邮件: 　 1990506872@qq.com

完成提交时间：2023 年 6 月 22 日

**项目成员及分工**

**（工作量总和为100%）**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **学 号** | **姓 名** | **工作内容** | **工作量（%）** |
| **1** | **20211060203** | **周楷翔** | 上传文件‘upload’命令、下载文件‘get’命令 | 40% |
| **2** | **20211060143** | **李俊杰** | 切换目录‘cd’命令、浏览当前目录‘ls’等命令、查询当前路径‘cwd’命令 | 30% |
| **3** | **20211060245** | **陈俊宏** | 登录验证功能、Socket监听模块、线程创建模块、初始化用户文件路径 | 30% |

《FTP服务器的设计和实现》成绩考核表

**年级： 2021级 专业： 人工智能 学号： 20211060203 姓名： 周楷翔**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **指标内容** | **分值** | **指标内涵及评估标准** | | | | **得**  **分** |
| **A** | **B** | **C** | **D** |
| **构思与选题 （C, 15分）** | | | | | |  |
| 选题意义 | 5 | 意义重大 | 意义较大 | 意义一般,属于简单的开发 | 无意义 |  |
| 技术路线的可行程度 | 10 | 合理可行  具体且有创新 | 合理可行,具体 | 基本合理可行 | 不够合理 |  |
| **设计 （D, 20分）** | | | | | |  |
| 设计内容 | 10 | 内容非常丰富 | 内容较丰富 | 内容一般 | 内容欠缺 |  |
| 解决的关键技术问题 | 10 | 准确，范围合适  重点突出 | 基本准确 | 部分关键 | 未抓住关键 |  |
| **实现 （I, 30分）** | | | | | |  |
| 项目完成的技术水平（个人） | 10 | 难度很大  达到较高水平 | 难度较大  超出一般水平 | 难度一般  达到普通水平 | 难度小  很容易实现 |  |
| 小组成员的工作量 | 10 | 高出平均要求工作量15%以上 | 高出平均要求工作量 | 达到平均要求  工作量 | 低于平均要求工作量 |  |
| 团队精神 | 10 | 团队合作精神强 | 合作情况良好 | 合作情况一般 | 合作不好 |  |
| **运作 （O, 15分）** | | | | | |  |
| 达到预期目标的程度 | 15 | 组织优秀  完全达到 | 组织良好  较好达到 | 组织一般  基本达到 | 组织混乱  未能达到 |  |
| **文档总结情况（20分）** | | | | | |  |
| 文字表达 | 5 | 文字表达非常好 | 文字表达较好 | 文字表达  一般 | 文字表达差  意思不明了 |  |
| 文档制作 | 5 | 制作非常专业化 | 制作良好 | 制作一般 | 制作效果差 |  |
| 报告质量 | 5 | 报告非常完整 | 报告比较完整 | 完整程度一般 | 报告不完整 |  |
| 口头表达能力 | 5 | 整体效果很好 | 整体效果良好 | 整体效果一般 | 整体效果差 |  |
| 综合得分（满分100分） | |  | | | | |
| 评语 | |  | | | | |

《FTP服务器的设计和实现》成绩考核表

**年级： 2021级 专业： 人工智能 学号： 20211060143 姓名： 李俊杰**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **指标内容** | **分值** | **指标内涵及评估标准** | | | | **得**  **分** |
| **A** | **B** | **C** | **D** |
| **构思与选题 （C, 15分）** | | | | | |  |
| 选题意义 | 5 | 意义重大 | 意义较大 | 意义一般,属于简单的开发 | 无意义 |  |
| 技术路线的可行程度 | 10 | 合理可行  具体且有创新 | 合理可行,具体 | 基本合理可行 | 不够合理 |  |
| **设计 （D, 20分）** | | | | | |  |
| 设计内容 | 10 | 内容非常丰富 | 内容较丰富 | 内容一般 | 内容欠缺 |  |
| 解决的关键技术问题 | 10 | 准确，范围合适  重点突出 | 基本准确 | 部分关键 | 未抓住关键 |  |
| **实现 （I, 30分）** | | | | | |  |
| 项目完成的技术水平（个人） | 10 | 难度很大  达到较高水平 | 难度较大  超出一般水平 | 难度一般  达到普通水平 | 难度小  很容易实现 |  |
| 小组成员的工作量 | 10 | 高出平均要求工作量15%以上 | 高出平均要求工作量 | 达到平均要求  工作量 | 低于平均要求工作量 |  |
| 团队精神 | 10 | 团队合作精神强 | 合作情况良好 | 合作情况一般 | 合作不好 |  |
| **运作 （O, 15分）** | | | | | |  |
| 达到预期目标的程度 | 15 | 组织优秀  完全达到 | 组织良好  较好达到 | 组织一般  基本达到 | 组织混乱  未能达到 |  |
| **文档总结情况（20分）** | | | | | |  |
| 文字表达 | 5 | 文字表达非常好 | 文字表达较好 | 文字表达  一般 | 文字表达差  意思不明了 |  |
| 文档制作 | 5 | 制作非常专业化 | 制作良好 | 制作一般 | 制作效果差 |  |
| 报告质量 | 5 | 报告非常完整 | 报告比较完整 | 完整程度一般 | 报告不完整 |  |
| 口头表达能力 | 5 | 整体效果很好 | 整体效果良好 | 整体效果一般 | 整体效果差 |  |
| 综合得分（满分100分） | |  | | | | |
| 评语 | |  | | | | |

《FTP服务器的设计和实现》成绩考核表

**年级： 2021级 专业： 人工智能 学号： 20211060245 姓名： 陈俊宏**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **指标内容** | **分值** | **指标内涵及评估标准** | | | | **得**  **分** |
| **A** | **B** | **C** | **D** |
| **构思与选题 （C, 15分）** | | | | | |  |
| 选题意义 | 5 | 意义重大 | 意义较大 | 意义一般,属于简单的开发 | 无意义 |  |
| 技术路线的可行程度 | 10 | 合理可行  具体且有创新 | 合理可行,具体 | 基本合理可行 | 不够合理 |  |
| **设计 （D, 20分）** | | | | | |  |
| 设计内容 | 10 | 内容非常丰富 | 内容较丰富 | 内容一般 | 内容欠缺 |  |
| 解决的关键技术问题 | 10 | 准确，范围合适  重点突出 | 基本准确 | 部分关键 | 未抓住关键 |  |
| **实现 （I, 30分）** | | | | | |  |
| 项目完成的技术水平（个人） | 10 | 难度很大  达到较高水平 | 难度较大  超出一般水平 | 难度一般  达到普通水平 | 难度小  很容易实现 |  |
| 小组成员的工作量 | 10 | 高出平均要求工作量15%以上 | 高出平均要求工作量 | 达到平均要求  工作量 | 低于平均要求工作量 |  |
| 团队精神 | 10 | 团队合作精神强 | 合作情况良好 | 合作情况一般 | 合作不好 |  |
| **运作 （O, 15分）** | | | | | |  |
| 达到预期目标的程度 | 15 | 组织优秀  完全达到 | 组织良好  较好达到 | 组织一般  基本达到 | 组织混乱  未能达到 |  |
| **文档总结情况（20分）** | | | | | |  |
| 文字表达 | 5 | 文字表达非常好 | 文字表达较好 | 文字表达  一般 | 文字表达差  意思不明了 |  |
| 文档制作 | 5 | 制作非常专业化 | 制作良好 | 制作一般 | 制作效果差 |  |
| 报告质量 | 5 | 报告非常完整 | 报告比较完整 | 完整程度一般 | 报告不完整 |  |
| 口头表达能力 | 5 | 整体效果很好 | 整体效果良好 | 整体效果一般 | 整体效果差 |  |
| 综合得分（满分100分） | |  | | | | |
| 评语 | |  | | | | |

目录

[云南大学软件学院期末课程报告 1](#_Toc138343446)

[目录 1](#_Toc138343447)

[**摘要** 2](#_Toc138343448)

[**1.** **各功能模块介绍** 2](#_Toc138343449)

[1.1 登录和创建客户端线程 2](#_Toc138343450)

[1.2 切换及查看当前目录目录 4](#_Toc138343451)

[1.3 上传和下载 6](#_Toc138343452)

[**2. FTP功能演示** 9](#_Toc138343453)

[2.1 登录FTP服务器 9](#_Toc138343454)

[2.2 查看当前路径、切换路径、查看当前目录文件 9](#_Toc138343455)

[2.3 上传和下载文件 10](#_Toc138343456)

[**3. 总结** 10](#_Toc138343457)

**摘要**

本文介绍了一个基于Python语言和socket、thread、os、sys等模块实现的多线程FTP服务器。该服务器支持用户登录、切换目录、查看当前目录文件、上传和下载文件等功能。

在该FTP服务器中，使用socket模块建立TCP连接，通过thread模块实现多线程处理客户端请求。在客户端连接到服务器时，服务器会要求客户端输入用户名和密码进行登录。如果用户名和密码正确，则服务器会返回一个欢迎信息，并允许客户端执行其他操作。如果用户名和密码不正确，则服务器会拒绝客户端请求。

一旦客户端登录成功，就可以执行一系列操作。使用os模块可以实现切换目录、查看当前目录文件等功能。使用socket模块可以实现上传和下载文件的功能，客户端可以向服务器上传文件，也可以从服务器下载文件。

总的来说，这个FTP服务器是一个基于Python语言和socket、thread、os、sys等模块实现的轻量级多线程FTP服务器，可以实现基本的FTP功能，并且具有一定的扩展性和可定制性。。

**关键词：FTP; Python; Socket; 服务器; 客户端**

1. **各功能模块介绍**

1.1 登录和创建客户端线程

**Server.py**

# 创建Socket，监听本地端口，等待客户端连接  
sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
sock.bind(('10.100.229.195', 5000))  
sock.listen(5)  
while True:  
 conn, addr = sock.accept()  
 # 验证客户端输入的用户名和密码是否正确  
 with open("S:/Python/ComputerNetwork/users.json", "r", encoding="UTF-8") as users:  
 users = json.load(users)  
 userinfo = None  
 userId, userPwd = conn.recv(1024).decode().split(',')  
 for i in range(len(users)):  
 if userId == users[i]["username"] and userPwd == users[i]["password"]:  
 userinfo = users[i]  
 break  
 if userinfo:  
 conn.send(b'ok')  
 # 为每个客户端连接创建并启动一个线程，参数为连接、客户端地址、客户主目录  
 home = userinfo["home"]  
 t = threading.Thread(target=server, args=(conn, addr, home))  
 t.daemon = True  
 t.start()  
 else:  
 conn.send(b'error')

该段代码创建一个Socket服务器，监听本地端口，等待客户端连接。当客户端连接时，程序会验证客户端输入的用户名和密码是否正确。如果验证通过，程序会为每个客户端连接创建并启动一个线程，参数为连接、客户端地址、客户主目录。如果验证不通过，程序会发送一个错误信息给客户端。

**Client.py**

if \_\_name\_\_ == '\_\_main\_\_':  
 if len(sys.argv) != 2:  
 print('Usage:{0} serverIPAddress'.format(sys.argv[0]))  
 exit()  
 serverIP = sys.argv[1]  
 if re.match(r'^\d{1,3}.\d{1,3}.\d{1,3}.\d{1,3}$', serverIP):  
 main(serverIP)  
 else:  
 print('服务器地址不合法')  
 exit()

def main(serverIP):  
 sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
 sock.connect((serverIP, 5000))  
 userId = input('请输入用户名：')  
 # 使用getpass模块的getpass()方法获取密码，不回显  
 userPwd = getpass.getpass('请输入密码：')  
 message = userId + ',' + userPwd  
 sock.send(message.encode())  
 login = sock.recv(100)  
 # 验证是否登录成功  
 if login == b'error':  
 print('用户名或密码错误')  
 return

这段代码是一个Client.py程序的入口，用于解析命令行参数并调用主函数。程序首先检查命令行参数的数量，如果不是2个，则打印使用说明并退出程序。如果命令行参数数量正确，则将第二个参数解析为服务器的IP地址。程序接着使用正则表达式检查服务器的IP地址是否合法。如果服务器的IP地址合法，则调用主函数main(serverIP)。如果服务器的IP地址不合法，则打印错误信息并退出程序。

1.2 切换及查看当前目录目录

**Server.py**

data = conn.recv(100).decode().lower()  
# 显示客户端输入的每一条命令  
print(data)  
# 客户端退出  
if data in ('quit', 'q'):  
 break  
# 查看当前文件夹的文件列表  
elif data in ('list', 'ls', 'dir'):  
 files = str(os.listdir(os.getcwd()))  
 files = files.encode()  
 conn.send(struct.pack('I', len(files)))  
 conn.send(files)  
# 切换至上一级目录  
elif ''.join(data.split()) == 'cd..':  
 cwd = os.getcwd()  
 newCwd = cwd[:cwd.rindex('\\')]  
 # 考虑根目录的情况  
 if newCwd[-1] == ':':  
 newCwd += '\\'  
 # 限定用户主目录  
 if newCwd.lower().startswith(home):  
 os.chdir(newCwd)  
 conn.send(b'ok')  
 else:  
 conn.send(b'error')  
# 查看当前目录  
elif data in ('cwd', 'cd'):  
 conn.send(str(os.getcwd()).encode())  
elif data.startswith('cd '):  
 # 指定最大分隔次数，考虑目标文件夹带有空格的情况  
 # 只允许使用相对路径进行跳转  
 data = data.split(maxsplit=1)  
 if len(data) == 2 and os.path.isdir(data[1]) \  
 and data[1] != os.path.abspath(data[1]):  
 os.chdir(data[1])  
 conn.send(b'ok')  
 else:  
 conn.send(b'error')

1. `data = conn.recv(100).decode().lower()`：从客户端接收数据，最多接收100个字节，然后将其转换为小写字母格式，并将其赋值给变量`data`。

2. `print(data)`：打印出客户端输入的每一条命令。

3. `if data in ('quit', 'q'):`：如果客户端输入的命令是`quit`或`q`，则退出循环，关闭连接。

4. `elif data in ('list', 'ls', 'dir'):`：如果客户端输入的命令是`list`、`ls`或`dir`，则获取当前文件夹中的文件列表，并将其发送给客户端。

5. `elif ''.join(data.split()) == 'cd..'`：如果客户端输入的命令是`cd..`，则将当前目录切换到上一级目录。

6. `elif data in ('cwd', 'cd'):`：如果客户端输入的命令是`cwd`或`cd`，则将当前目录发送给客户端。

7. `elif data.startswith('cd '):`：如果客户端输入的命令以`cd `开头，则将其解析为切换到指定目录。

8. `if len(data) == 2 and os.path.isdir(data[1]) and data[1] != os.path.abspath(data[1]):`：如果输入的命令是`cd`加上一个目录名，并且这个目录名是一个存在的目录，并且不是绝对路径，则切换到该目录。

这段代码实现了如查看当前目录、切换目录、查看文件列表等基本命令。同时考虑了一些特殊情况，如目录名中带有空格、不能跳出用户主目录等。

**Client.py**

while True:  
 # 接收客户端命令，其中##>是提示符  
 command = input('>> ').lower().strip()  
 # 没有输入任何有效字符，提前进入下一次循环，等待用户继续输入  
 if not command:  
 continue  
 # 向服务端发送命令  
 command = ' '.join(command.split())  
 sock.send(command.encode())  
 # 退出  
 if command in ('quit', 'q'):  
 break  
 # 查看文件列表  
 elif command in ('list', 'ls', 'dir'):  
 loc\_size = struct.unpack('I', sock.recv(intSize))[0]  
 files = eval(sock.recv(loc\_size).decode())  
 for item in files:  
 print(item)  
 # 切换至上一级目录  
 elif ''.join(command.split()) == 'cd..':  
 print(sock.recv(100).decode())  
 # 查看当前工作目录  
 elif command in ('cwd', 'cd'):  
 print(sock.recv(1024).decode())  
 # 切换至子文件夹  
 elif command.startswith('cd '):  
 print(sock.recv(100).decode())

客户端通过socket模块与FTP服务器进行通信，实现了一些基本的FTP命令。具体分析如下：

1. command = input('>> ').lower().strip() 通过input函数获取用户输入的命令，并将其转换为小写字符，去掉前后的空格。

2. if not command: 如果用户没有输入任何有效字符，就进入下一次循环，等待用户继续输入。

3. command = ' '.join(command.split()) 将用户输入的命令中的多余空格去掉，只保留一个空格。

4. sock.send(command.encode()) 将处理后的命令通过socket发送给FTP服务器。

5. if command in ('quit', 'q'): 如果用户输入quit或q命令，就退出程序。

6. elif command in ('list', 'ls', 'dir'): 如果用户输入list、ls或dir命令，就向FTP服务器发送list命令，接收FTP服务器返回的文件列表，并逐个输出文件名。

7. elif ''.join(command.split()) == 'cd..': 如果用户输入cd..命令，就向FTP服务器发送cd..命令，切换至上一级目录，并输出FTP服务器返回的信息。

8. elif command in ('cwd', 'cd'): 如果用户输入cwd或cd命令，就向FTP服务器发送pwd命令，获取当前工作目录，并输出FTP服务器返回的信息。

9. elif command.startswith('cd '): 如果用户输入以cd开头的命令，就向FTP服务器发送cd命令，切换至指定的子文件夹，并输出FTP服务器返回的信息。

1.3 上传和下载

**Server.py**

# 从客户端上传文件  
elif data.startswith('upload '):  
 isFileExist = conn.recv(20)  
 # 文件不存在  
 if isFileExist != b'ok':  
 print('error')  
 # 文件存在，开始下载  
 else:  
 print('uploading.', end='')  
 fp = open(data.split()[1], 'wb')  
 while True:  
 print('.', end='')  
 data = conn.recv(4096)  
 if data == b'overxxxx':  
 break  
 fp.write(data)  
 conn.send(b'ok')  
 fp.close()  
 print('ok')  
  
# 下载文件  
elif data.startswith('get '):  
 data = data.split(maxsplit=1)  
 # 检查文件是否存在  
 if len(data) == 2 and os.path.isfile(data[1]):  
 conn.send(b'ok')  
 fp = open(data[1], 'rb')  
 while True:  
 content = fp.read(4096)  
 # 发送文件结束  
 if not content:  
 conn.send(b'overxxxx')  
 break  
 # 发送文件内容  
 conn.send(content)  
 if conn.recv(10) == b'ok':  
 continue  
 fp.close()  
 else:  
 conn.send(b'no')

具体分析如下：

1. 从客户端上传文件

当客户端发送"upload "开头的命令时，服务器会接收到该命令并判断文件是否存在。如果文件不存在，则会返回"error"；如果文件存在，则会开始上传文件。上传过程中，服务器会不断接收客户端发送的数据，并写入文件中，直到接收到"overxxxx"表示文件传输结束。

2. 下载文件

当客户端发送"get "开头的命令时，服务器会接收到该命令并判断文件是否存在。如果文件存在，则会返回"ok"并开始发送文件内容；如果文件不存在，则会返回"no"。发送文件内容时，服务器会不断读取文件内容，并发送给客户端，直到文件传输结束。在发送文件内容前，服务器会等待客户端发送确认信息"ok"，以保证文件内容能够正确传输。

**Client.py**

# 上传文件到服务器  
elif command.startswith('upload '):  
 data = command.split(maxsplit=1)  
 # 检查文件是否存在  
 if len(data) == 2 and os.path.isfile(data[1]):  
 sock.send(b'ok')  
 print('uploading.', end='')  
 fp = open(data[1], 'rb')  
 while True:  
 content = fp.read(4096)  
 # 发送文件结束  
 if not content:  
 sock.send(b'overxxxx')  
 break  
 # 发送文件内容  
 sock.send(content)  
 if sock.recv(10) == b'ok':  
 continue  
 fp.close()  
 print('ok')  
 else:  
 print('error')  
 sock.send(b'no')  
  
# 从服务器下载文件  
elif command.startswith('get '):  
 isFileExist = sock.recv(20)  
 # 文件不存在  
 if isFileExist != b'ok':  
 print('error')  
 # 文件存在，开始下载  
 else:  
 print('downloading.', end='')  
 fp = open(command.split()[1], 'wb')  
 while True:  
 print('.', end='')  
 data = sock.recv(4096)  
 if data == b'overxxxx':  
 break  
 fp.write(data)  
 sock.send(b'ok')  
 fp.close()  
 print('ok')

具体分析如下：

1. 上传文件到服务器

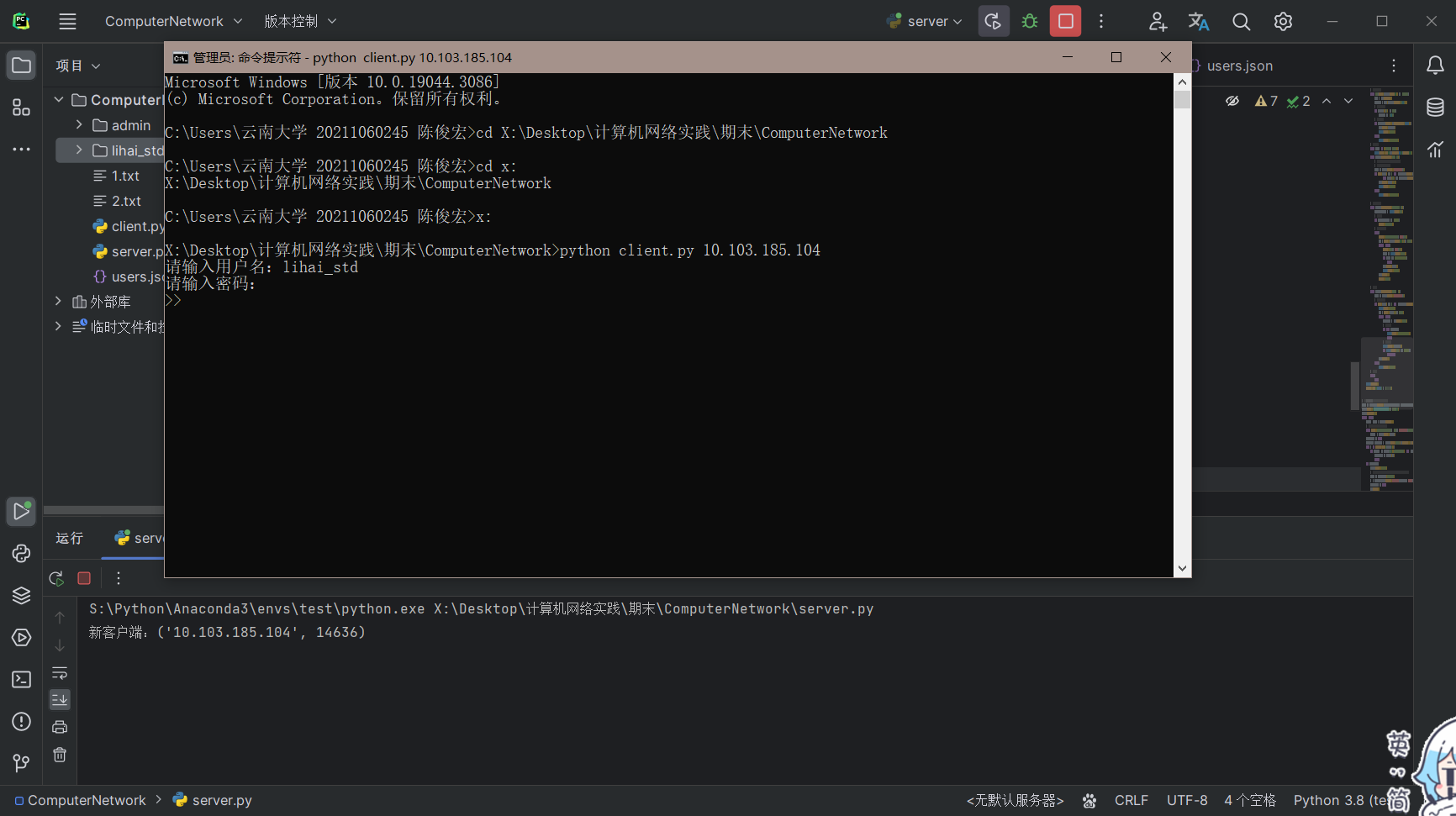
客户端会首先检查待上传的文件是否存在，如果存在则向服务器发送"ok"，表示可以开始上传文件。然后客户端打开文件，每次读取4096字节的内容并将其发送给服务器，直到文件读取结束。在每次发送文件内容之后，客户端会等待服务器返回"ok"，表示已经成功接收该部分内容，然后才会继续发送下一部分内容。当整个文件发送完毕后，客户端向服务器发送"overxxxx"，表示文件发送结束。

2. 从服务器下载文件

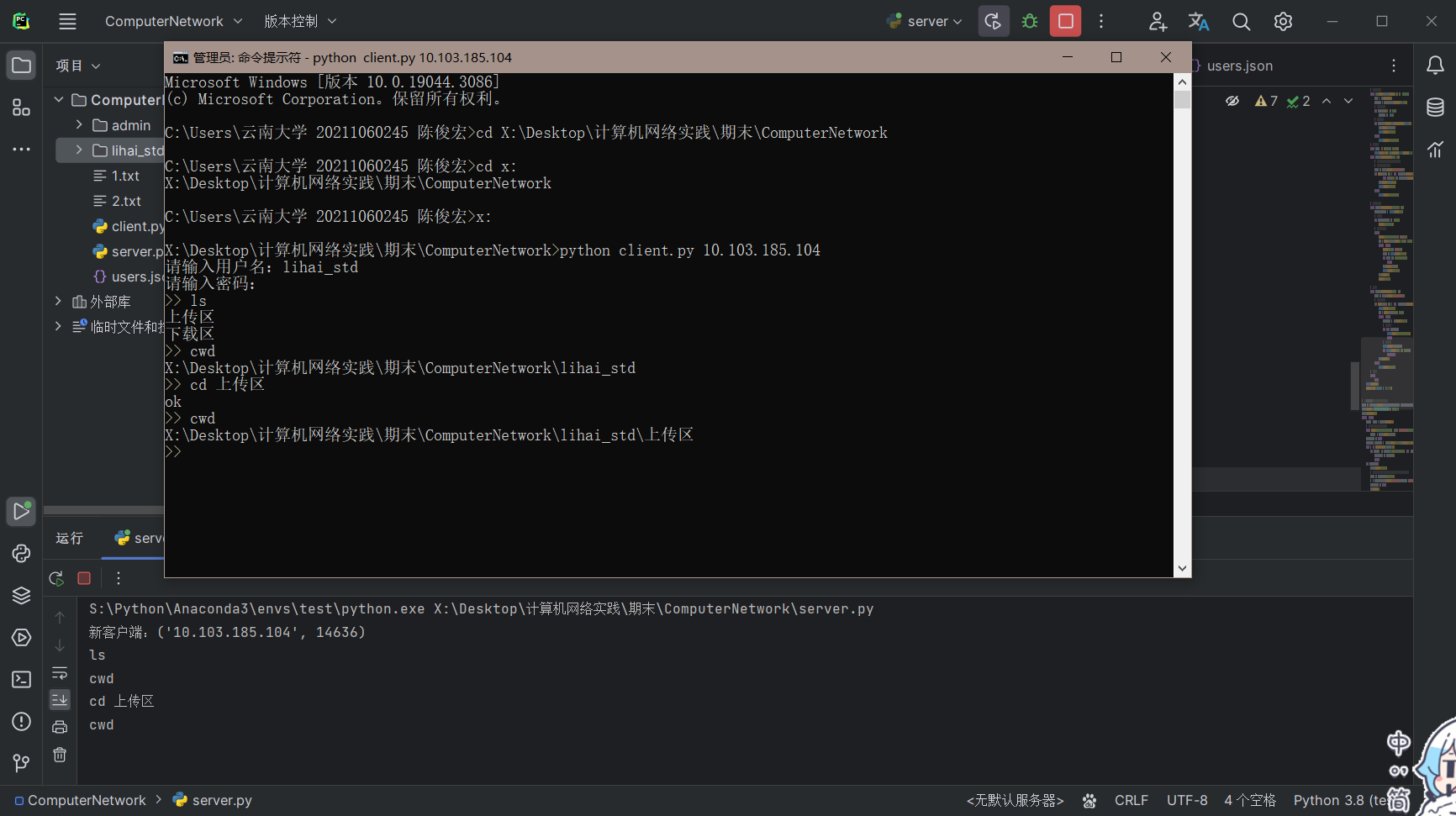
客户端会首先向服务器发送要下载的文件名，并等待服务器的响应。如果文件不存在，则服务器会返回"no"，客户端会打印"error"并退出下载操作。如果文件存在，则服务器会返回"ok"，客户端会打开一个新文件，每次接收4096字节的数据并将其写入文件中，直到接收到服务器发送的"overxxxx"，表示文件接收结束。在每次发送数据之后，客户端会向服务器发送"ok"，表示已经成功接收该部分数据。当整个文件接收完毕后，客户端会关闭文件并打印"ok"。

**2. FTP功能演示**

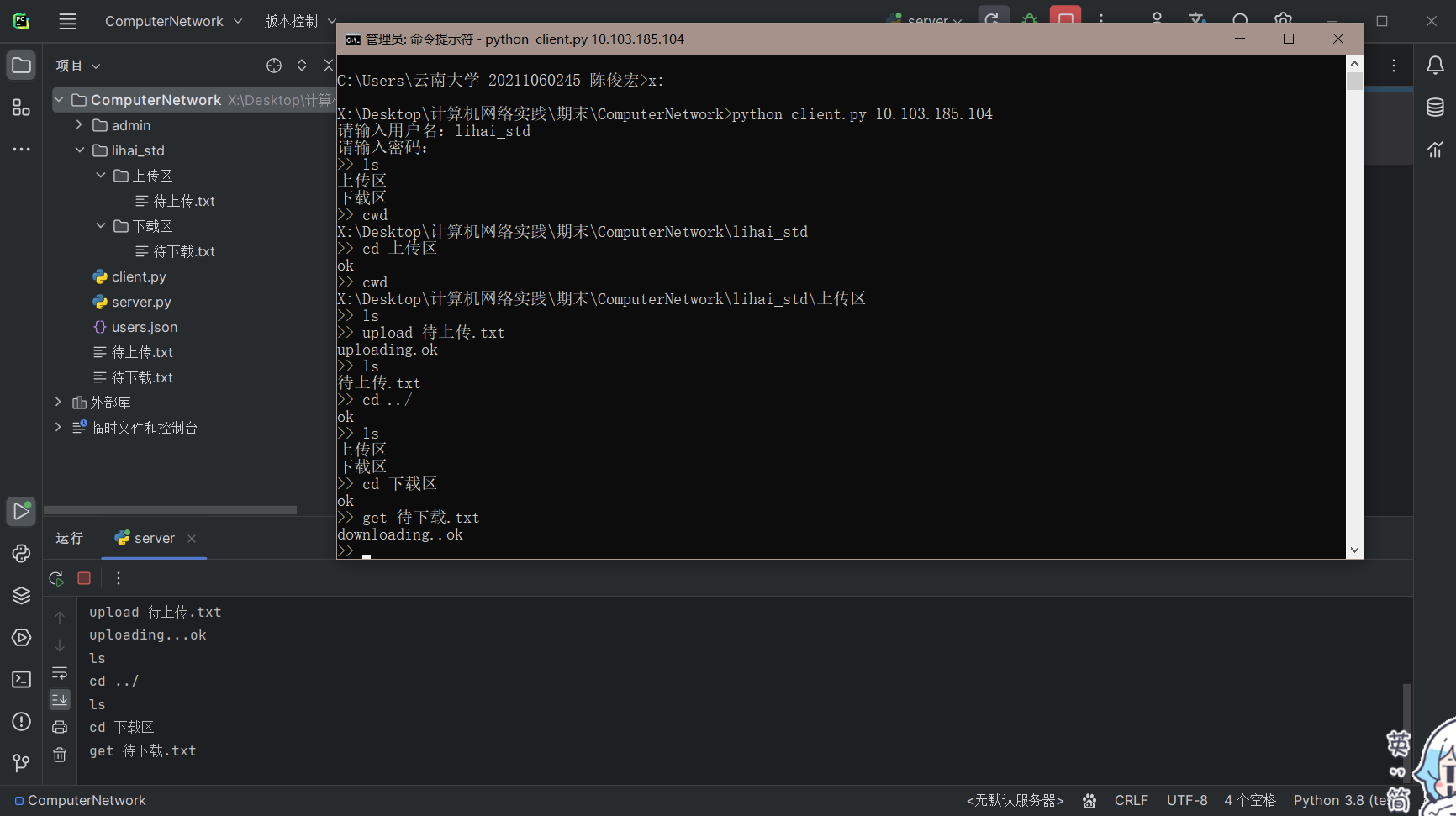
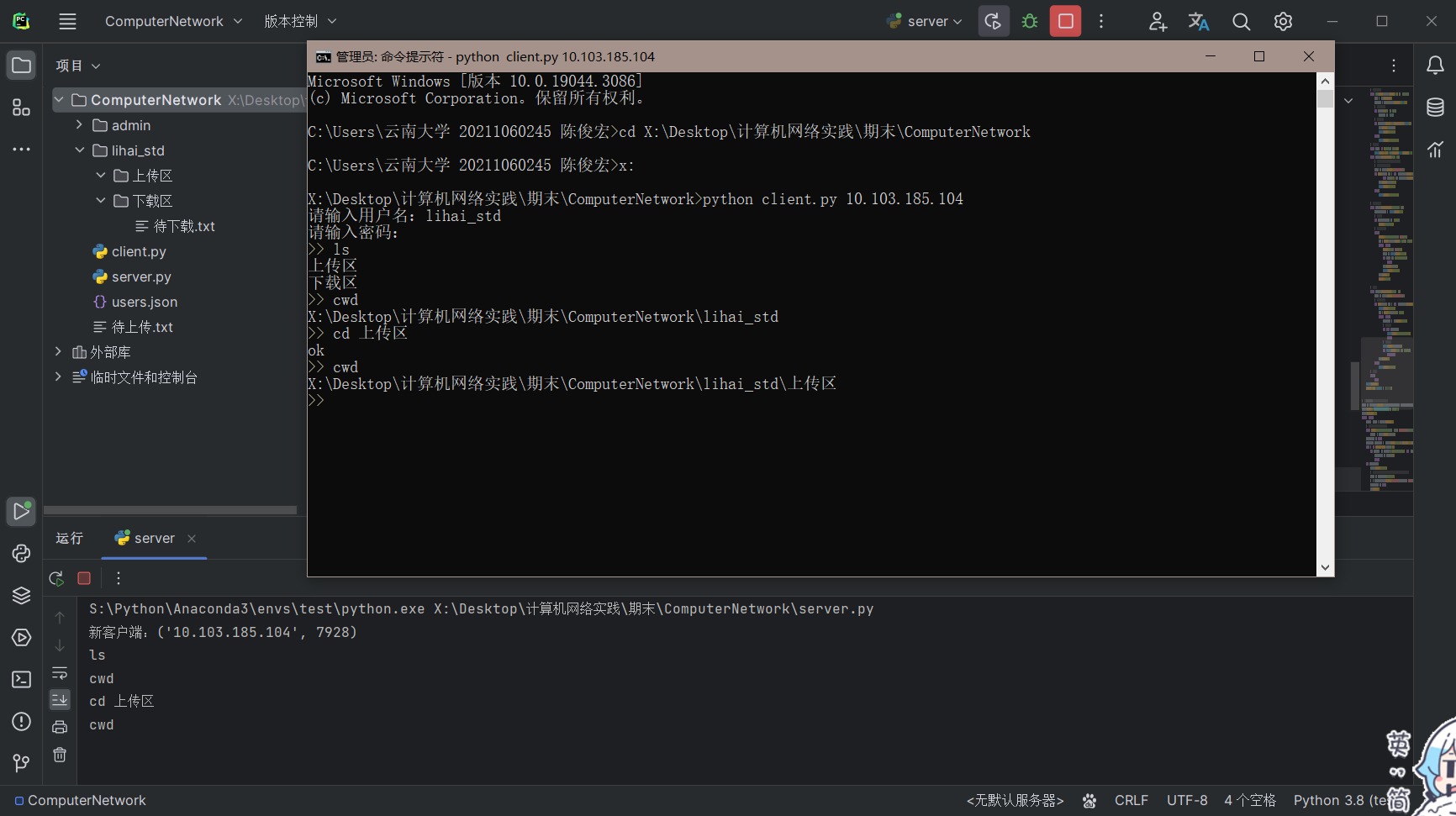
2.1 登录FTP服务器



2.2 查看当前路径、切换路径、查看当前目录文件



2.3 上传和下载文件



**3. 总结**

在FTP服务器的设计和实现过程中，我们需要考虑多方面的因素。首先，我们需要选择合适的编程语言和相关模块。在本次对话中，我们选择了Python作为编程语言，并使用了socket模块、json模块、正则表达式等相关模块。

其次，我们需要了解FTP协议的基本原理和实现方式。FTP协议是一种基于TCP/IP协议的应用层协议，主要用于文件传输。在FTP服务器的实现过程中，我们需要考虑如何实现基本的FTP命令，如查看当前目录、切换目录、查看文件列表等。

此外，我们还需要考虑一些特殊情况，如目录名中带有空格、不能跳出用户主目录等。在本次实验中，我们实现了一个基于Python实现的多线程FTP服务器和客户端的核心代码，该代码实现了上述基本命令，并考虑了上述特殊情况。

综上所述，FTP服务器的设计和实现需要综合考虑多方面的因素，包括编程语言和相关模块的选择、FTP协议的基本原理和实现方式、特殊情况的处理等。同时，我们还需要将FTP服务器的设计和实现置于更广泛的背景和需求之中，以便更好地满足用户的需求。