简单FTP服务器搭建（python版）

1. 测试环境

- `centos7(3.10.0-693.el7.x86_64)`
- `python2.7.5`

2. 服务端代码

```
import os
import socket

FTP_DIR = "./ftpserver"
def handle_client(client_socket):
    welcome_message = "220 Welcome to the FTP server\r\n"
    client_socket.send(welcome_message.encode())

    while True:
        try:
            command = client_socket.recv(1024).decode().strip()
            print("Received command:", command)

            if command == "QUIT":
                client_socket.send("221 Goodbye\r\n".encode())
                client_socket.close()
                break
            elif command.startswith("STOR"):
                filename = command[5:].strip()
                file_path = os.path.join(FTP_DIR, filename)
                print('filename:',filename,file_path)

                try:
                    with open(file_path, "wb") as file:
                        client_socket.send("150 File status okay; about to open
data connection\r\n".encode())
                        while True:
                            data = client_socket.recv(1024)
                            if not data:
                                break
                            file.write(data)
                    client_socket.send("226 File uploaded
successfully\r\n".encode())
                    print("File uploaded:", file_path)
                except IOError:
                    client_socket.send("550 Failed to upload file\r\n".encode())
                    print("Failed to upload file:", file_path)

            else:
                client_socket.send("500 Unknown command\r\n".encode())

        except socket.error as e:
            print("Client disconnected:", e)
            client_socket.close()
```

```
            break


def start_ftp_server(host, port):
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((host, port))
    server_socket.listen(1)
    print("FTP server started on {}:{}".format(host, port))

    while True:
        client_socket, client_address = server_socket.accept()
        print("Accepted connection from:", client_address)

        handle_client(client_socket)

    server_socket.close()

if __name__ == "__main__":
    HOST = 'localhost'
    PORT = 22000

    start_ftp_server(HOST, PORT)
```

1. 定义了常量 `FTP_DIR`，表示文件上传的目标目录。
2. 定义了 `handle_client` 函数，用于处理客户端连接。在函数中，首先发送欢迎消息给客户端，然后进入循环接收客户端发送的命令。
3. 如果客户端发送的命令是 `QUIT`，则关闭客户端连接。
4. 如果客户端发送的命令以 `STOR` 开头，则表示客户端要上传文件。从命令中提取文件名，并构建文件的完整路径。然后使用文件读写模式打开文件，并循环接收客户端发送的数据，并将数据写入文件中。
5. 如果文件上传成功，发送上传成功的消息给客户端，并打印上传成功的文件路径。如果发生IO错误（例如文件无法打开），发送上传失败的消息给客户端，并打印上传失败的文件路径。
6. 如果客户端发送的命令不是 `QUIT` 或以 `STOR` 开头，则发送未知命令的消息给客户端。
7. 定义了 `start_ftp_server` 函数，用于启动 FTP 服务器。在函数中，创建一个服务器套接字，绑定到指定的主机和端口，并开始监听连接。
8. 当有客户端连接时，接受连接，并调用 `handle_client` 函数处理客户端连接。
9. 如果遇到键盘中断（Ctrl+C），关闭服务器套接字。

将该代码保存为一个文件（例如 `server.py`），在虚拟机中运行该文件。服务器将在指定的主机和端口上启动，并等待客户端连接和文件上传。

代码中的 FTP 目录 `FTP_DIR` 默认设置为 `./ftpserver`，表示在当前工作目录下创建一个名为 `ftpserver` 的文件夹用于存储上传的文件。你可以根据需要修改该目录的路径。

3. 客户端代码

```
# -*- coding: utf-8 -*-
import socket
import sys

SERVER_HOST = 'localhost'
SERVER_PORT = 22000
FILE_PATH = sys.argv[1]
```

```
def upload_file(server_host, server_port, file_path):
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((server_host, server_port))
    response = client_socket.recv(1024).decode()
    print(response)

    with open(file_path, 'rb') as file:
        filename = file_path.split('/')[-1]
        print('filename:',filename)
        command = 'STOR ' + filename
        client_socket.send(command.encode())
        response = client_socket.recv(1024).decode()
        print(response)
        if response.startswith('150'):
            while True:
                data = file.read(1024)
                if not data:
                    break
                client_socket.send(data)

    client_socket.close()


if __name__ == '__main__':
    upload_file(SERVER_HOST, SERVER_PORT, FILE_PATH)
```

首先，导入了`socket`模块，以便进行网络通信。然后定义了三个常量：

- `SERVER_HOST`：FTP服务器的主机名或IP地址。
- `SERVER_PORT`：FTP服务器的端口号。
- `FILE_PATH`：要上传的文件的路径，这里使用了`sys.argv[1]`获取命令行参数。

接下来，定义了`upload_file`函数，它接受FTP服务器的主机名、端口号和文件路径作为参数。在函数内部，首先创建了一个客户端套接字，并使用`connect`方法连接到FTP服务器。然后接收并打印了来自服务器的欢迎消息。

接下来，使用`open`函数打开文件，使用`split`函数从文件路径中提取文件名，并构建了一个以"STOR "为前缀的命令，然后发送给服务器。然后接收并打印了来自服务器的响应。

随后，进入一个循环，从文件中读取数据，并使用`send`方法将数据发送给服务器，直到文件读取完毕。最后关闭客户端套接字。

在`__name__ == '__main__'`的条件下，调用`upload_file`函数，并传递FTP服务器的主机名、端口号和文件路径作为参数，从而完成文件的上传过程。

4. 测试

1. 新建测试目录，测试文件

```
[root@localhost ~]# mkdir ftpserver
[root@localhost ~]# cat >> file.txt
123test ftp 文件上传功能
[root@localhost ~]# ls -ll ftpserver/
总用量 0
[root@localhost ~]# ll file.txt
-rw-r--r-- 1 root root 43 7月   5 22:02 file.txt
```

2.启动服务端

```
root@localhost ~]# python service.py
FTP server started on localhost:21000
```

3.启动客户端

```
[root@localhost ~]# python client.py  file.txt
220 Welcome to the FTP server

('filename:', 'file.txt')
150 File status okay; about to open data connection
```

4.服务端现象

```
[root@localhost ~]# python service.py
FTP server started on localhost:22000
('Accepted connection from:', ('127.0.0.1', 35878))
('Received command:', u'STOR file.txt')
('filename:', u'file.txt', u'./ftpserver/file.txt')
('File uploaded:', u'./ftpserver/file.txt')
('Received command:', u'')
```

5.`ftpserver`目录

```
[root@localhost ~]# ll ftpserver/
总用量 4
-rw-r--r-- 1 root root 43 7月   5 22:25 file.txt
[root@localhost ~]# cat f
file.txt   ftpserver/
[root@localhost ~]# cat ftpserver/file.txt
ear
23

563
123test ftp 文件上传功能
```

5. summary

当开发FTP服务器和客户端时，有几个注意事项需要考虑：

1. 协议规范：FTP协议有一定的规范和约定，包括命令格式、响应码和响应格式等。在开发过程中，要确保遵循FTP协议规范，以确保服务器和客户端之间能够正确地进行通信。
2. 套接字编程：FTP使用TCP作为传输协议，因此在服务器和客户端的代码中需要使用套接字编程。确保正确地创建、连接和关闭套接字，并处理套接字相关的异常情况。
3. 并发处理：FTP服务器需要能够处理多个客户端同时连接的情况。可以使用多线程、多进程或异步编程来实现并发处理，以确保服务器能够同时处理多个客户端的请求。
4. 路径处理：FTP涉及文件传输，因此需要处理文件路径。在服务器端，确保正确处理文件的存储路径，避免安全风险和文件冲突。在客户端，处理本地文件的路径和远程服务器上的文件路径，确保正确的文件传输。
5. 错误处理：在服务器和客户端的代码中，要考虑到可能出现的错误情况，并进行适当的错误处理。例如，处理套接字错误、文件读写错误、权限错误等，并向用户提供有用的错误信息。
6. 用户认证和安全性：FTP服务器通常需要进行用户认证，以确保只有授权用户才能访问和传输文件。在服务器代码中，要实现用户认证机制，并确保传输过程的安全性，例如使用SSL/TLS进行加密。
7. 日志记录：为了追踪和排查问题，可以在服务器和客户端的代码中添加适当的日志记录功能。记录关键事件、错误信息和调试信息，以便进行故障排除和性能优化。

本文简单的介绍了ftp服务端和客户端的搭建，有需要的话，可以根据需求进行拓展