

**FTP实验**

学号： 2022011599

专业： 计算机科学与技术

姓名： 黄思源

# 目录

[1 实验概述 1](#_Toc4307)

[1.1 实验目的 1](#_Toc14397)

[1.2 实验内容 1](#_Toc20280)

[2 FTP 实验实现 1](#_Toc10383)

[2.1 基本功能介绍 1](#_Toc12265)

[2.2 套接口函数的应用 2](#_Toc21806)

[2.3 文件操作函数的应用 2](#_Toc4884)

[2.4 客户端关键函数与功能描述 2](#_Toc28983)

[2.5 服务器功能实现与关键函数 3](#_Toc27772)

[3 FTP实验结果展示 5](#_Toc25104)

[4 Web 服务器实现 9](#_Toc4575)

[4.1 技术架构 9](#_Toc15555)

[4.2 核心功能 10](#_Toc2386)

[4.3 界面展示 11](#_Toc1890)

[5 爬虫部分实现 11](#_Toc16302)

[5.1 技术架构 11](#_Toc9794)

[5.2核心功能 12](#_Toc3917)

[6 思考题解答 12](#_Toc23629)

[6.1 为何 FTP 使用两个 TCP 连接？ 12](#_Toc407)

[6.2 主动模式与被动模式的区别？ 13](#_Toc16071)

[6.3 下载大量小文件速度慢的原因及改进？ 13](#_Toc19007)

[7 实验结论 13](#_Toc16427)

[8 附录 13](#_Toc12574)

1 实验概述

## 1.1 实验目的

本次实验旨在通过在 Linux 系统上使用 C/C++ 语言实现 FTP 客户端与服务器，深入理解 FTP 协议的原理与细节，掌握 Socket 接口编程和 TCP/IP 网络应用开发技能。同时，基于 Web 技术实现文件传输系统，并开发简易网络爬虫，提升综合网络编程能力。

## 1.2 实验内容

实现 FTP 客户端与服务器，支持控制通道与数据通道分离，实现get、put、pwd、dir等基本命令，支持主动 / 被动模式及登录验证。

使用 Python Flask 框架搭建 Web 文件传输系统，实现文件上传、下载、目录管理等功能。

基于 Socket 和 BeautifulSoup 开发网络爬虫，抓取指定网页的文本和图片资源并保存。

# 2 FTP 实验实现

## 2.1 基本功能介绍

本实验实现的 FTP 系统遵循 RFC959 规范，客户端与服务器通过 TCP 协议建立连接：

控制通道：固定端口（服务器 21 端口），用于传输 FTP 命令（如登录、文件操作指令）。

数据通道：动态端口，用于文件传输或目录列表获取，支持主动模式（服务器主动连接客户端）和被动模式（客户端主动连接服务器）。

支持的核心命令包括：

connect <host> <port>：连接到服务器

login <username> <password>：登录

ls/dir：列出当前目录内容

cd <directory>：切换目录

get <filename>：从服务器下载文件

put <filename>：上传文件到服务器

passive：设置客户端为被动模式（默认）

active：设置客户端为主动模式

quit：断开连接并退出

## 2.2 套接口函数的应用

socket()： 创建套接字，指定通信域（AF\_INET）、套接字类型（SOCK\_STREAM，表示 TCP 连接）和协议。

bind()： 将套接字绑定到本地的 IP 地址和端口号。服务器端使用此函数绑定到固定的监听端口（如 21），被动模式下服务器的数据连接监听套接字也会绑定到随机端口。

listen()： 服务器端套接字进入监听状态，等待客户端连接请求。

accept()： 服务器端接受客户端的连接请求，返回一个新的已连接套接字，用于与特定客户端进行通信。在被动模式下，服务器的数据连接监听套接字也会使用此函数接受客户端的数据连接请求。

connect()： 客户端使用此函数向服务器发起连接请求，建立控制连接。在主动模式下，服务器会使用此函数向客户端发起数据连接。在被动模式下，客户端会使用此函数向服务器发起数据连接。

send() / recv()： 用于在已建立的套接字上发送和接收数据。控制连接用于发送命令和接收响应，数据连接用于传输文件数据或目录列表数据。

setsockopt()： 用于设置套接字选项，例如 SO\_REUSEADDR 和 SO\_REUSEPORT，以允许端口重用，避免 TIME\_WAIT 状态导致的端口占用问题。

close()： 关闭套接字，释放相关资源。

## 2.3 文件操作函数的应用

文件传输：

使用fopen()、fread()、fwrite()实现文件读写，结合sendfile()优化大文件传输效率。

传输前通过stat()获取文件元数据（如大小、修改时间）。

目录操作：

使用opendir()、readdir()遍历目录，chdir()切换工作目录，mkdir()创建目录。

## 2.4 客户端关键函数与功能描述

客户端程序 (ftp\_client/client.cpp) 提供了一个交互式命令行界面，负责解析用户输入并与服务器进行通信。

main() 函数： 客户端程序的入口点，负责套接字初始化、连接服务器、用户登录流程，以及进入主循环以接收用户输入并调用相应的命令处理函数。

send\_command(int sock, const std::string& command)： 核心通信函数，负责向服务器的控制连接发送完整的 FTP 命令（自动添加 \r\n 终止符），并等待接收服务器的响应字符串。

open\_passive\_data\_connection\_client(const std::string& ip, int port)： 在被动模式下，客户端根据服务器返回的 IP 和端口，主动发起数据连接。

open\_active\_data\_listener\_client(std::string& client\_ip, int& client\_port)： 在主动模式下，客户端打开一个监听套接字，等待服务器来连接，并将监听的 IP 和端口返回给服务器。

handle\_list\_command(int sock)： 处理用户输入的 ls 或 dir 命令。根据当前设置的模式（主动或被动），先与服务器协商数据连接（发送 PASV 或 PORT），然后发送 LIST 命令，并通过建立的数据连接接收目录列表数据并显示。

handle\_get\_command(int sock, const std::string& filename)： 处理 get <filename> 命令。协商数据连接后，发送 RETR <filename> 命令，并通过数据连接接收文件内容并保存到本地文件。

handle\_put\_command(int sock, const std::string& filename)： 处理 put <filename> 命令。协商数据连接后，发送 STOR <filename> 命令，并通过数据连接读取本地文件内容并发送到服务器。

set\_passive\_mode() / set\_active\_mode()： 客户端内部的辅助函数，用于切换客户端的数据传输模式标志位，影响后续数据传输命令的行为。

quit\_command(int sock)： 发送 QUIT 命令给服务器，接收服务器的 221 Goodbye 响应，然后关闭控制连接并终止客户端程序。

login\_user(int sock, const std::string& username, const std::string& password)： 负责客户端的登录流程，发送 USER 和 PASS 命令，并根据服务器响应判断登录是否成功。

change\_directory(int sock, const std::string& path)： 处理 cd <directory> 命令，发送 CWD <directory> 命令给服务器。

## 2.5 服务器功能实现与关键函数

服务器程序 (ftp\_server/server.cpp) 负责监听客户端连接，管理会话状态，并响应客户端的 FTP 命令。

main() 函数： 服务器程序的入口点。负责创建主监听套接字（21 端口），绑定地址，进入监听状态。在主循环中，它使用 accept() 接收新的客户端连接，并为每个新连接创建一个独立的线程 (client\_handler\_thread) 来处理。

client\_handler\_thread(void \*socket\_desc)： 每个客户端连接的处理线程函数。它负责初始化 ClientState，发送欢迎消息，然后进入循环，持续接收、解析客户端命令，并调用相应的处理函数。

ClientState 结构体： 用于存储每个客户端的会话状态，包括其控制套接字、当前工作目录、登录状态、用户名、数据连接相关信息（如 IP、端口、模式、数据套接字文件描述符）。

send\_response(int client\_socket, const std::string& response)： 向指定客户端的控制连接发送 FTP 响应字符串。

handle\_user(ClientState& client\_state, const std::string& username)： 处理 USER 命令，保存客户端提供的用户名。

handle\_pass(ClientState& client\_state, const std::string& password)： 处理 PASS 命令，根据预设的用户数据库验证用户名和密码，更新客户端的登录状态。

open\_passive\_data\_listener(int control\_socket, std::string& server\_ip, int& data\_port)： 处理 PASV 命令。服务器打开一个随机高位端口并监听，将服务器的 IP 和该端口信息格式化后返回给客户端。

accept\_passive\_data\_connection(int data\_listener\_sock)： 在被动模式下，服务器在 open\_passive\_data\_listener 建立的监听套接字上接受客户端的数据连接请求。

open\_active\_data\_connection(int control\_socket, const std::string& client\_ip, int client\_port)： 处理 PORT 命令。服务器根据客户端提供的 IP 和端口，主动发起数据连接。

handle\_list\_data(ClientState& client\_state)： 处理 LIST 命令。根据当前模式，建立数据连接，遍历 client\_state.current\_dir 中的文件和目录，将列表数据通过数据连接发送给客户端。

handle\_retr\_data(ClientState& client\_state, const std::string& filename)： 处理 RETR 命令。建立数据连接，打开指定文件，将文件内容通过数据连接发送给客户端。

handle\_stor\_data(ClientState& client\_state, const std::string& filename)： 处理 STOR 命令。建立数据连接，在服务器端创建或打开文件，从数据连接接收文件内容并写入文件。

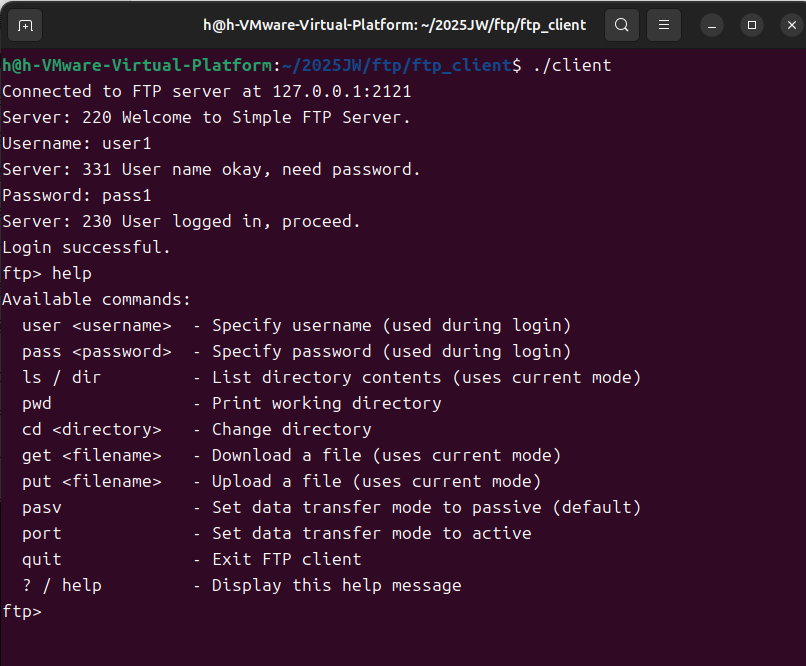
handle\_pwd(int client\_socket, const std::string& current\_dir)： 处理 PWD 命令，返回客户端当前在服务器上的工作目录。

handle\_cwd(int client\_socket, std::string& current\_dir, const std::string& path)： 处理 CWD 命令，改变客户端在服务器上的工作目录。

handle\_quit(ClientState& client\_state)： 处理 QUIT 命令，发送 221 Goodbye 响应，并负责清理客户端资源，关闭套接字。

# 3 FTP实验结果展示

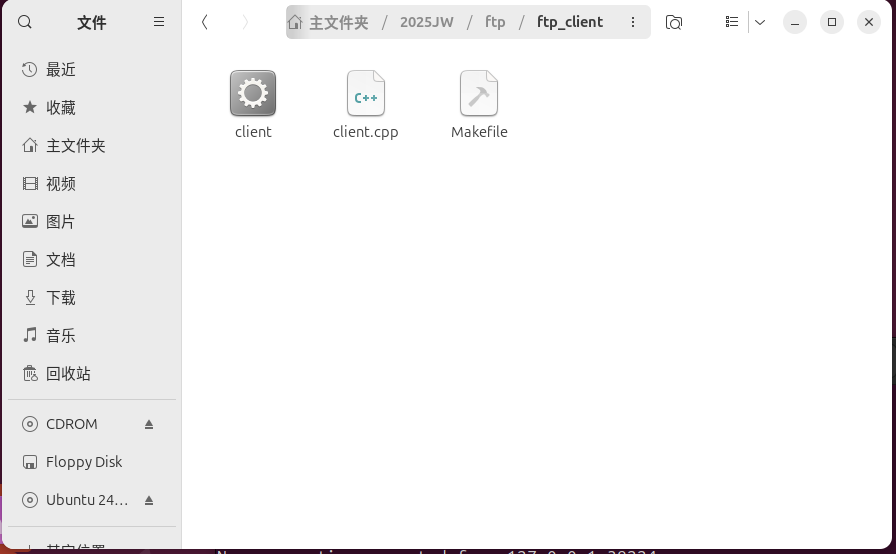
所有命令展示（使用help可以得到所有指令的提示）



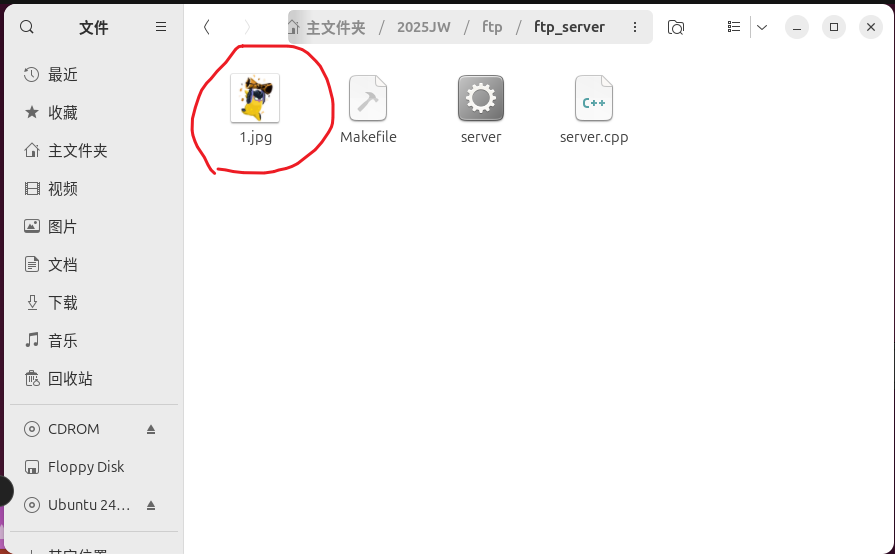
下面只演示关键功能（get、put）部分，其他功能可以自行下载我的代码尝试。

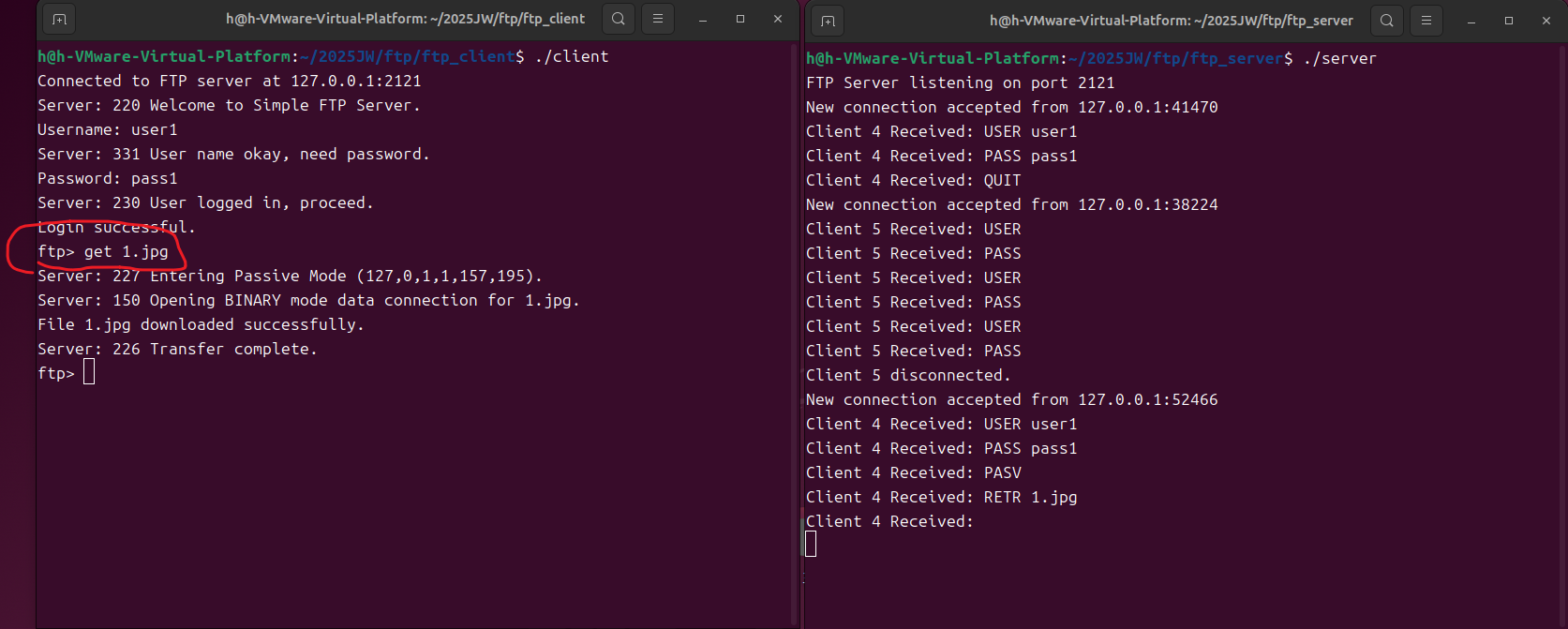
Get之前

客户端的文件

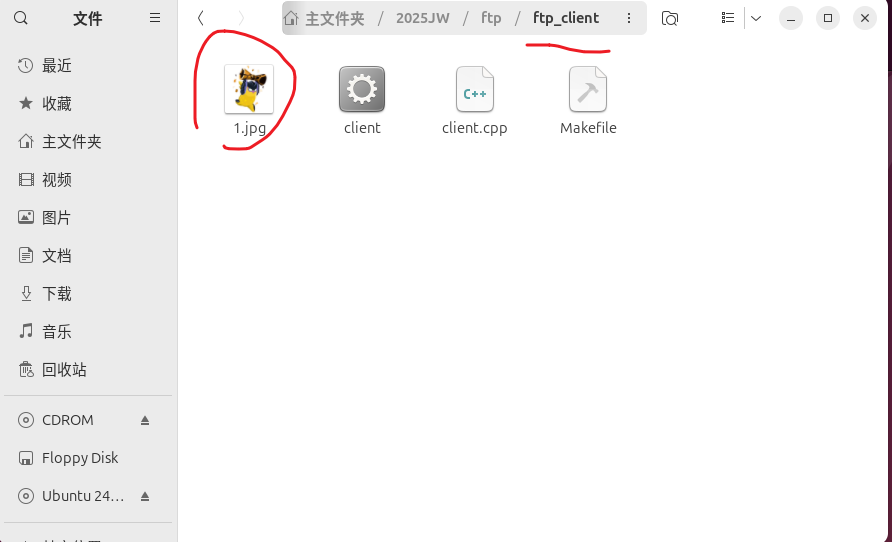


服务端的文件，有一个1.jpg，接下来我们会在客户端通过get指令获取它



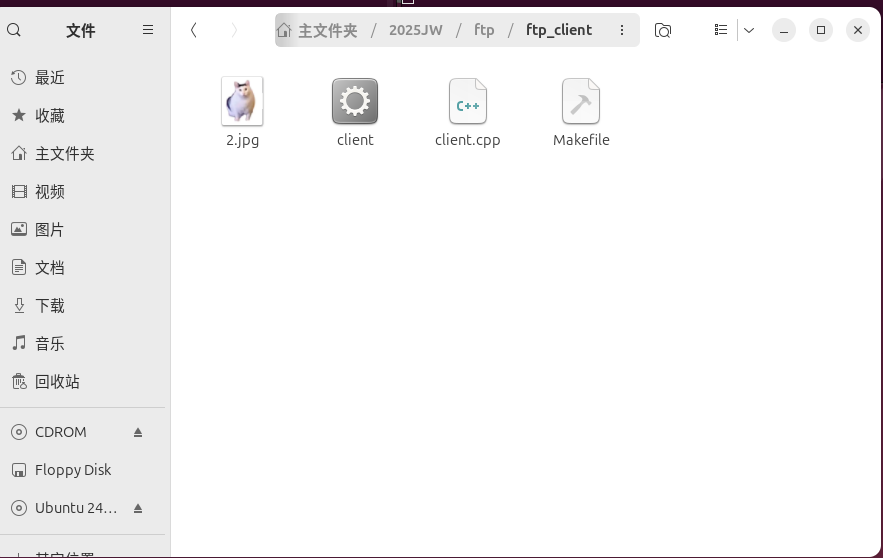


可以看到，输入get 1.jpg指令后，客户端获取到该图片了。

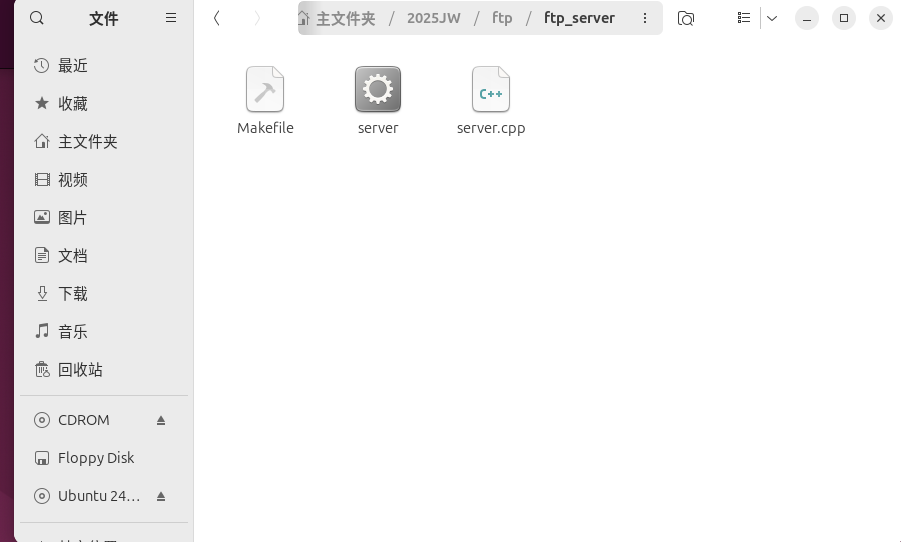


同理，下面演示put。

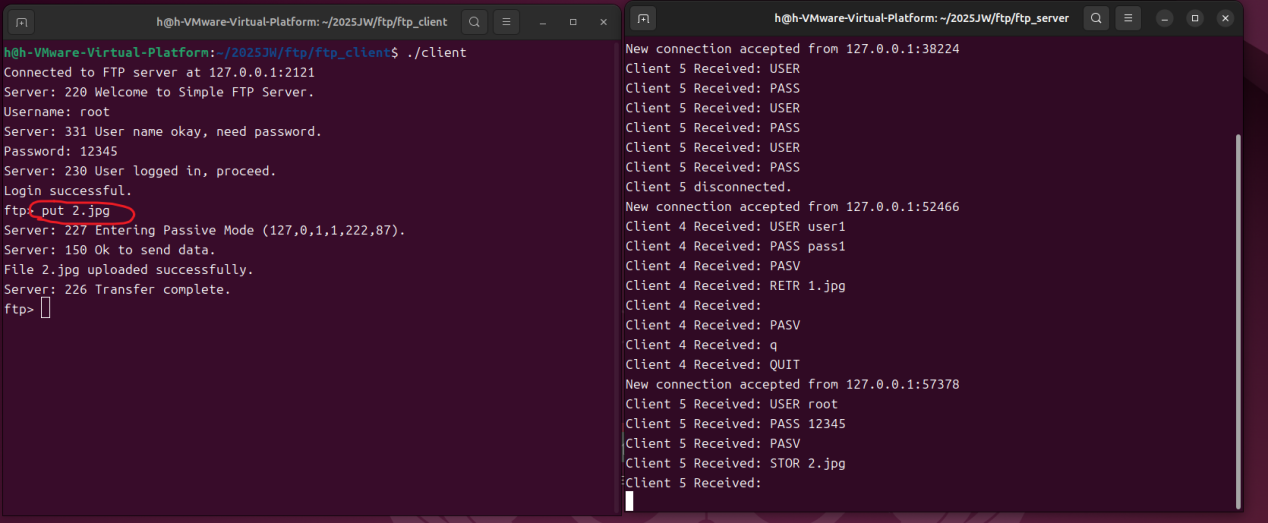
这是put之前，

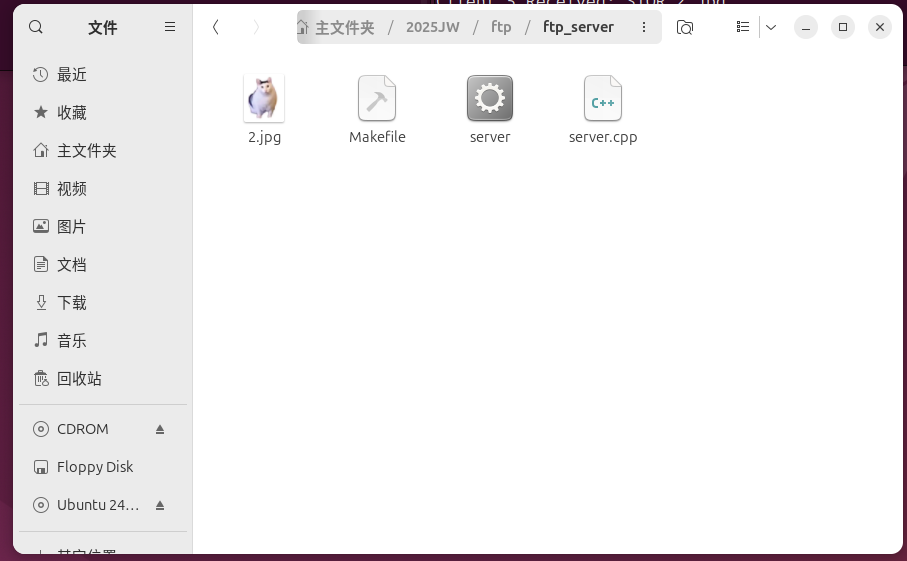
客户端的文件

服务器端的文件



我们会通过客户端的put指令，将2.jpg文件从客户端传给服务器端。





# 4 Web 服务器实现

## 4.1 技术架构

后端：Python Flask，实现轻量级 Web 服务。

前端：HTML/CSS+JavaScript，提供文件上传下载界面（upload.html、listfile.jsp）。

## 4.2 核心功能

文件列表与目录导航：

显示服务器指定 uploads 目录下当前路径的文件和子目录列表。

支持点击子目录名称进入下一级目录，并通过“返回上一级”链接回到父目录。

动态调整显示路径，提供清晰的用户界面。

文件上传：

通过 Web 表单接收用户选择的文件。

将文件安全地保存到服务器上的当前浏览目录中。

支持文件类型过滤 (ALLOWED\_EXTENSIONS) 和文件大小限制 (MAX\_CONTENT\_LENGTH)。

文件下载：

为列表中的文件提供明确的“下载”按钮。

点击下载按钮时，强制浏览器将文件下载到本地，而不是直接在浏览器中打开。

目录创建：

允许用户在当前目录下创建新的空文件夹。

对输入的文件夹名称进行安全处理，防止非法字符。

文件/目录删除：

为文件和空目录提供删除功能。

在删除前有 JavaScript 确认提示，增强用户体验和安全性。

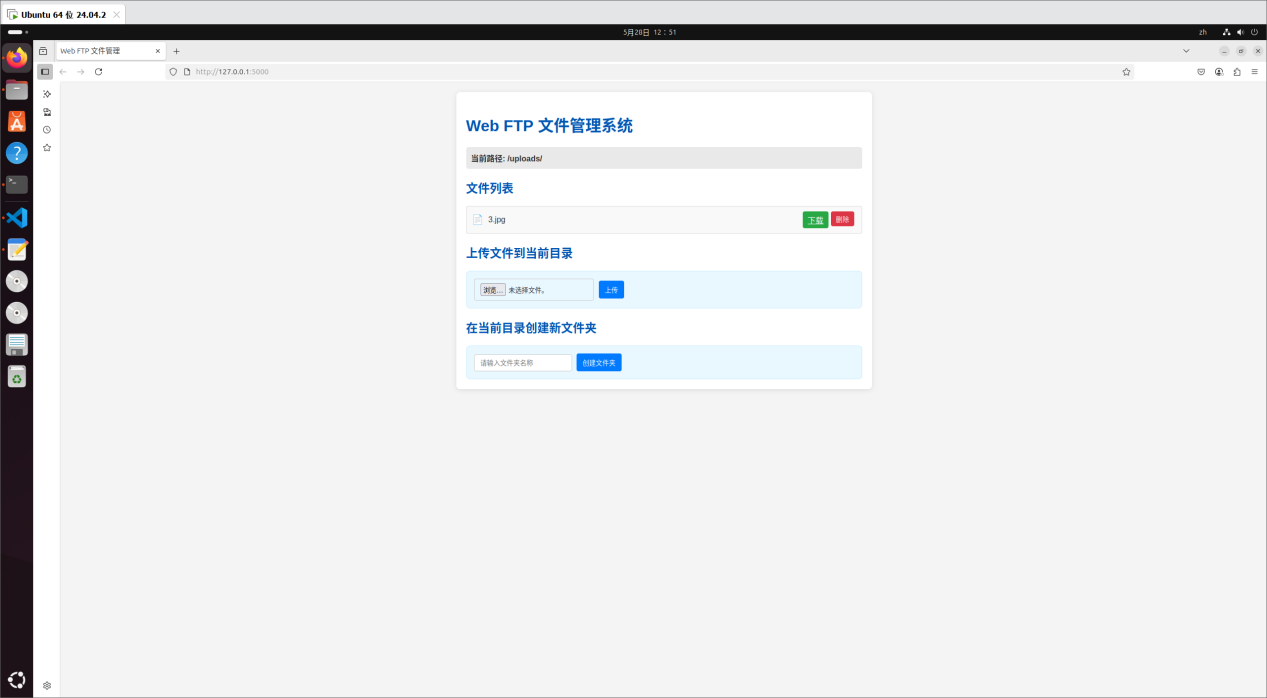
安全防护：

secure\_filename：清洗文件名，防止恶意文件名导致的问题。

safe\_path\_join：严格检查所有文件操作的路径，确保操作始终限制在 UPLOAD\_FOLDER 内部，有效防止了路径遍历攻击（如 ../ 访问服务器其他文件）。

用户反馈：

通过 Flask 的 flash 消息机制，在页面顶部显示操作成功、失败或警告等提示信息。

4.3 界面展示  


# 5 爬虫部分实现

## 5.1 技术架构

核心网络通信： Python socket 模块直接进行 TCP 连接，发送和接收原始 HTTP 请求和响应数据。

HTTP 协议实现： 手动构建 HTTP GET 请求头，并解析服务器返回的原始 HTTP 响应（包括状态行、响应头和响应体）。

HTML 解析： BeautifulSoup4 库用于解析 HTML 响应体，方便地提取网页中的文本内容和指定标签（如 <img>）的属性。

XML 解析器： lxml 是 BeautifulSoup4 默认推荐和使用的解析器之一，提供高性能的解析能力。

文件系统操作： Python 的 os 模块和内置的文件操作（open, write）用于将抓取到的数据（文本、图片）保存到本地文件。

URL 处理： urllib.parse 模块（虽然代码中可能没直接显示但通常会用到）用于解析和构建 URL。

## 5.2核心功能

底层 HTTP GET 请求：

通过 Socket 连接到目标 Web 服务器的 80 端口（HTTP 默认端口）。

手动构造并发送符合 HTTP/1.0 或 HTTP/1.1 规范的 GET 请求字符串。

原始 HTTP 响应接收与解析：

通过 Socket 接收服务器返回的原始字节流。

手动解析响应头和响应体，分离 HTTP 头部和实际内容。

处理可能的编码问题（如 UTF-8）。

网页文本内容提取：

使用 BeautifulSoup 解析 HTML 响应体，提取可见的纯文本内容。

图片 URL 提取与下载：

在解析后的 HTML 中查找所有 <img> 标签。

提取图片的 src 属性（URL）。

如果图片 URL 是 HTTP 协议，则再次通过 Socket 发送 GET 请求下载图片数据。

将下载的图片保存到本地指定目录。

本地数据存储：

将抓取到的网页文本内容保存为 .txt 文件。

将下载的图片保存为原始格式的文件。

错误处理：

处理网络连接失败、HTTP 状态码异常等情况。

（根据实现细节）可能包含对重定向、相对路径 URL 的处理。

# 6 思考题解答

## 6.1 为何 FTP 使用两个 TCP 连接？

控制与数据分离：控制通道传输命令和响应（如用户认证、文件操作指令），数据通道专注于文件传输，提高协议灵活性和效率。

并发处理：可在不中断控制连接的情况下，并行处理多个数据传输任务。

## 6.2 主动模式与被动模式的区别？

主动模式：服务器主动连接客户端数据端口（默认 20 端口），适用于服务器位于公网、客户端在内网的场景，但可能被防火墙拦截。

被动模式：客户端主动连接服务器临时开放的数据端口，适用于客户端位于 NAT 或防火墙后的场景，兼容性更强。

## 6.3 下载大量小文件速度慢的原因及改进？

原因：每个文件传输需新建 / 关闭数据连接，TCP 三次握手和四次挥手带来额外开销。

改进方法：

使用MLSD/MLST等批量目录查询命令减少交互次数；

启用 TCP\_NODELAY 选项禁用 Nagle 算法，减少小包延迟；

采用二进制传输模式（bin命令）替代 ASCII 模式，避免字符转换开销。

# 7 实验结论

通过本次实验，深入掌握了 FTP 协议的底层原理与 Socket 编程实现，理解了网络应用中协议分层和连接管理的设计思想。同时，通过 Web 文件系统和网络爬虫的开发，熟悉了前后端分离架构和 HTTP 协议的实际应用。实验中遇到的防火墙配置、端口冲突等问题，通过查阅 RFC 文档和调试 Socket 状态得以解决，进一步提升了问题分析与解决能力。

# 8 附录

完整项目代码可见本人的GitHub 项目链接。

[https://github.com/FishCoderovo/2025JW\_Network\_Projects](https://github.com/FishCoderovo/2025JW_Network_Projects" \t "https://www.doubao.com/chat/_blank)

ftp/目录：FTP 客户端与服务器代码；

web/目录：Web 文件传输系统代码；

crawler/目录：网络爬虫代码。