FTP 实验报告

姚远　软件 23　2021010082

1　服务器介绍

1.1　运行方法及支持命令

FTP 服务器运行方法:

```
cd server/src
make
sudo ./server -root /tmp -port 21 [-remote]
```

其中 root 指定了服务器的根目录，后续所有文件操作均在此目录下执行，默认值为 /tmp；port 指定了服务器运行的端口号，默认值为 21；-remote 为可选项，表示服务器和客户端不在同一台机器上运行。

支持的命令：USER, PASS, RETR, STOR, QUIT, SYST, TYPE, PORT, PASV, MKD, CWD, PWD, LIST, RMD, REST（用于断点续传）, SIZE（客户端通过此命令获取服务器上的文件大小，为了 STOR 的断点续传）

1.2　重要命令介绍及其错误处理

USER & PASS 命令：格式为 USER anonymous，后面接 PASS email@666.com。按照文档要求，服务器仅实现了匿名登录，不支持其他形式的登录。同时，PASS 命令需要紧跟在 USER 命令之后，否则返回客户端 503 Login with USER first。登录之后，设置变量 logged_in 为 1，表示已经登录，后续所有操作均需登录，否则返回 530 Login required。

PORT 命令：格式为 PORT h1,h2,h3,h4,p1,p2，其中 h1.h2.h3.h4 表示客户端的 IP 地址，p1*256 + p2 表示数据连接端口号，指定客户端的 IP 地址和端口，用于后续的主动模式数据传输。在解析并验证了输入的 IP 地址和端口号后，服务器会检查 IP 地址的格式是否正确，端口号是否在有效范围内（0-255）。此外，服务器还会通过 getpeername 函数验证客户端 IP 地址与控制连接的 IP 地址是否一致，如果不一致，返回错误提示 533 IP addresses do not match。若验证成功，服务器启用 PORT 模式并返回 200 PORT command successful，表示命令执行成功。

PASV 命令：格式为 PASV，允许客户端连接到服务器指定的 IP 地址和随机生成的端口进行数据传输。服务器首先获取自身的 IP 地址，并在动态/私有端口范围（49152-65535）内随机选择一个端口。然后，服务器为被动模式创建一个 passive_socket，并绑定到生成的 IP 地址和端口。如果成功绑定并开始监听，服务器会返回响应 227 Entering Passive Mode (h1,h2,h3,h4,p1,p2)。如果在获取 IP 地址、绑定端口或开始监听时发生错误，服务器会返回相应的错误消息，提示客户端进入被动模式失败。

RETR 命令：RETR 命令用于下载服务器上的文件，格式为 RETR filename。在执行该命令前，客户端必须先通过 PORT 或 PASV 命令建立数据连接。服务器会验证请求的文件路径，确保文件在服务器根目录下（保证安全性，不可通过../等方式攻击破坏服务器）并且可访问。根据客户端请求的传输模式（ASCII 或二进制），服务器以相应的方式打开文件。在数据传输过程中，服务器使用多线程处理，将文件传输任务分配给独立的线程执行（这部分单独写在了 filetr.c 中），这样可以避免服务器因文件传输而阻塞，确保其他客户端请求能够被及时处理。

STOR 命令：上传本地文件到服务器，格式为 STOR filename。与 RETR 相似，不过多赘述。

LIST 命令：用于列出服务器当前工作目录下的文件和目录信息，格式为 LIST。客户端必须先通过 PORT 或 PASV 命令建立数据连接。服务器在执行该命令时，会首先验证当前目录是否位于服务器的根目录下（保证安全性），然后进行数据传输。为了安全执行 ls -l 命令，服务器使用 fork 创建一个子进程，在子进程中执行 ls 命令，并通过管道将输出结果传回父进程，父进程从管道中读取输出并通过数据连接发送给客户端。在实现服务器时并未使用这种方式实现，后续在完成 Client-GUI 的过程中，发现需要频繁使用到 LIST 命令（为了实时显示服务器的目录结构），所以使用 fork 和管道的方式确保了服务器的稳定性和并发处理能力，避免阻塞主进程，同时能够安全地执行系统命令并将结果传递给客户端。

CWD & PWD & MKD & RMD 命令：这四个命令分别用于切换目录、显示当前目录、创建目录和删除目录。为了确保服务器的安全性，这些命令在操作前都会对路径进行严格验证，每个命令都会解析客户端提供的路径，使用 realpath 将其转换为绝对路径，并确保该路径始终在服务器的根目录 root_dir 下，防止越权访问或操作服务器其他文件系统区域。

大文件传输 & 多客户端：大文件传输已过测试。为了支持并发的多客户端连接，服务器为每个客户端请求创建一个新的子进程。通过调用 fork()，父进程继续监听新的客户端连接，而子进程负责处理当前客户端的请求。子进程在完成与客户端的交互后关闭套接字并退出，确保服务器的主进程不会被单个客户端阻塞。

由于篇幅限制，ABOR / QUIT / SYST / TYPE 等使用和实现较为简单的命令未做说明。

1.3 附加功能

功能 1：文件传输不阻塞服务器。在 STOR 和 RETR 命令中，通过创建独立的传输线程处理 STOR 和 RETR 命令。每当客户端请求文件传输时，服务器会启动一个新的线程，该线程负责从客户端接收或向客户端发送文件数据。传输线程独立于主服务器线程运行，这样主线程可以继续监听其他客户端的请求，不会因文件传输而被阻塞。线程结束后，自动关闭文件和数据连接，并根据传输结果向客户端发送响应。效果：在服务器执行 STOR 命令的时候，可以继续输入 SYST / TYPE I 等命令并立即得到返回值，同时，多个客户端可以同时下载或上传文件。demo 视频可以见 client 的附加功能 3 的视频，此视频的环境是 client 与自己的 server 进行通信，下载视频的时候可以执行其他命令既能说明不会阻塞 GUI，也能说明文件传输不会阻塞服务器。

功能 2：断点续传。针对断点续传功能，实现了 REST 和 SIZE 命令。REST 命令允许客户端指定文件的重启位置（偏移量），并在后续文件传输时从该位置继续传输。服务器通过 sscanf 提取并验证偏移量，保存到 restart_position 变量中，确认后返回成功响应。SIZE 命令则用于客户端获取服务器上指定文件的大小，以便进行断点续传的准备。demo 视频见 demo_server_21.mp4 和 demo_server_22.mp4（使用 wsl 中自带的 ftp 客户端进行测试）。

2 客户端介绍

2.1 运行方法及支持命令

FTP 客户端运行方法:

```
cd client/src
python client.py -ip 127.0.0.1 -port 21 [-remote]
```

其中 ip 指定了服务器的地址，默认值为 127.0.0.1；port 指定了服务器运行的端口号，默认值为 21；-remote 为可选项，表示服务器和客户端不在同一台机器上运行。带有-remote 选项时，客户端会打印出自己的公网 ip 地址，方便用户在执行 PORT 命令的时候输入本机 ip。

支持的命令：USER, PASS, RETR, STOR, QUIT, SYST, TYPE, PORT, PASV, MKD, CWD, PWD, LIST, RMD, REST, SIZE（最后两个命令与服务器端做适配）

2.2 重要命令介绍及必要功能

客户端使用 python 语言完成，整体逻辑较为简单，且命令与服务器原理相同，由于篇幅限制，在此不过多介绍。大文件传输在后续的 GUI 的 demo 视频中可以看出。可以登录助教提供的 std_server。

2.3 附加功能

功能 1（实现不完整，只是提出困惑）：断点续传。客户端实现了 SIZE 命令和 REST 命令，可以支持断点续传。但是，如何判断 RETR / STOR 命令下何时进行断点续传令我困惑。如果将 SIZE 和 REST 命令开放给用户随便使用，那么随意的 REST 将会导致上传或下载的文件由用户自定义的拼接而成，感觉这是不可取的。

功能 2: 用户友好 GUI。图形界面写在了一个单独的文件 client_gui.py 中, 通过调用 client.py 中的相关函数实现功能（避免在完成 GUI 的过程中导致评测失败），使用 PyQt 绘制了简洁明了的 GUI。demo 视频见 demo_client_2.mp4。

功能 3：文件传输不阻塞 GUI。客户端通过使用 QThread 实现文件传输的多线程处理，在用户发起上传或下载操作时，创建一个独立的 FileTransferThread 线程，在该线程中执行上传或下载操作。传输过程中，主线程（GUI 线程）不会被阻塞，依然可以响应用户的其他操作。文件传输完成后，线程通过信号 completed 通知主线程，并根据传输结果更新 UI。demo 视频见 demo_client_3.mp4。

3 遇到的问题和想法

在进行测试的时候，我发现在服务端和客户端都能通过评测脚本的情况下，我的客户端不能与自己的服务端在 PASV 模式下执行 RETR/STOR/LIST。经过自己的客户端与助教提供的 std_server 逐行调试下,我发现 std_server 的对于 RETR(后两个同理)的处理不是像 cr.yp.to/ftp.html (数据通道打开返回的是标记号码是 150, 无论 PORT 还是 PASV 模式) 中那样的, 而是在 PORT 模式下返回 125 Data connection already open. Transfer starting（意味着 data_socket 已经打开），在 PASV 模式下返回 150 File status okay. About to open data connection（意味着 data_socket

还未打开，即将打开）。我服务器原本的处理是在打开 data_socket 后发送 150 标记，但是这样会导致我的客户端在发出 RETR 命令之后就会一直等待 150 标记，从而导致在 PASV 模式下，server 会阻塞在 data_socket = accept(passive_socket, NULL, NULL) 从而致使在 PASV 模式下无法运行 RETR, LIST, STOR 等命令，所以我将发送标记的位置针对 PORT 和 PASV 模式下做了不同的区分。代码位置见 server/src/control.c 的 262-295 行。