Lab: Socket Programming tcp部份报告

作者：阙铭淏

邮箱：quemh22@mails.tsinghua.edu.cn

学号：2022012746

日期：2024.10.20

一. 简介

本报告记录了我在实现FTP客户端与服务器过程中，尤其是PORT模式下的调试过程和解决方案。该项目旨在构建一个功能齐全的FTP服务器和客户端，能够处理RETR、STOR、USER、PASS、LIST等多种指令，并支持主动（PORT）和被动（PASV）两种数据传输模式。

在实现过程中，我重点关注了如何正确处理套接字创建、连接建立以及客户端与服务器之间的数据传输，尤其是在PORT模式下，遇到了较大的挑战。

二. 实现概述

FTP客户端和服务器均使用C语言编写，利用标准的套接字编程技术实现。为了提高灵活性，实现了PORT（主动模式）和PASV（被动模式），使服务器和客户端可以通过不同的方式建立数据连接。

1. **客户端命令**: 客户端支持多种基础的FTP命令，如USER、PASS、RETR、STOR、LIST、CWD等，用户可以根据需求选择使用PORT或PASV模式与服务器进行数据传输。
2. **服务器响应**: 服务器会针对客户端的请求进行相应处理，并在遇到无效命令、路径错误、连接错误等情况时返回合适的错误信息。
3. **多线程支持**: 服务器使用pthread库实现了多线程功能，能够同时处理多个客户端的连接，每个连接会创建一个独立线程，从而确保多个客户端之间的数据传输互不干扰。

三. 调试难点

问题 1: PORT模式下连接被拒绝

现象: 在PORT模式下，客户端遇到 `425 Can't connect to specified address` 错误。服务器试图连接客户端提供的IP和端口，但无法成功建立数据连接。

原因分析: 问题出现在客户端对端口的监听处理上。在PORT模式下，客户端需要主动监听指定的端口以接受服务器的连接。在我的初始实现中，端口绑定出现了错误，或者端口未正确打开，导致服务器无法连接。

解决方案:

- 修正`create_socket_PORT()`函数，确保客户端能够正确绑定指定端口，并启动监听等待服务器的连接。

- 在调试过程中，通过输出IP和端口的值，确保客户端正在监听正确的地址。
- 在客户端和服务器端增加日志，追踪连接尝试的每一步，以检查IP或端口是否匹配。

反思: 该问题让我更深入理解了`PORT`和`PASV`模式的区别。在`PORT`模式下，客户端需要主动承担建立数据连接的责任，而`PASV`模式则是服务器等待客户端的连接。

问题 2: RETR/STOR时文件为空

现象: 在`PORT`模式下，执行`RETR`或`STOR`命令时，客户端收到或发送的文件为空，但服务器显示传输已完成。

原因分析: 问题出在客户端过早关闭了数据套接字，导致数据连接在数据真正传输之前被断开，结果传输的文件为空。

解决方案:

- 修改逻辑，确保客户端只有在数据传输完全结束后才关闭数据连接。增加逻辑，等待文件传输完成后再关闭套接字。
- 在传输过程中增加额外检查，确保服务器发送`150`响应开始传输，并在传输结束时发送`226`响应。

反思: 该问题强调了在FTP数据传输中，套接字管理的重要性。客户端和服务器都需要小心处理数据连接的开启与关闭，确保传输过程不会中途中断。

四. 支持的FTP指令

命令	功能	处理
USER	输入用户名	输入 anonymous：331 Guest login ok, send your complete e-mail address as password. 其他：430 Invalid username.
PASS	输入密码	密码正确登录成功: 230 login success. 密码错误: 530 Not logged in, password error. 未登录: 503 Bad sequence of commands.
SYST	返回系统信息	215 UNIX Type: L8
TYPE	设置 type	输入 Type I: 200 Type set to I 其他: 501 Type Error
PORT	主动建立连接	成功建立连接: 200 PORT command successful 指令不合法: 501 Invalid command arguments 创建错误: 425 Connect Error()
PASV	被动建立连接	227 Entering Passive Mode(h1,h2,h3,h4,p1,p2)

命令	功能	处理
RETR	从服务端上获取文件	成功传输: 150 Opening BINARY mode data connection for <file_name>, (<file_size> bytes)，传输结束后 : 226 Transfer complete. 执行命令前未选择连接模式 : 503 Bad sequence of commands. 连接错误: 425 Data Connection Failed to connect. 文件不存在: 551 File not found.
STOR	向服务端发送文件	成功传输: 150 Opening BINARY mode data connection for <file_name>，传输结束后 : 226 Transfer complete. Transferred: <file_size> Bytes. Average speed: KB/s. 执行命令前未选择连接模式 : 503 Bad sequence of commands. 连接错误: 425 Data Connection Failed to connect. 文件创建错误: 551 File created error. 客户端文件不存在: 551 File not found.
LIST	列出目录下所有文件	开始获取文件列表: 150 Opening data channel for directory list. 传输结束后: 226 List complete. 执行命令前未选择连接模式 : 503 Bad sequence of commands. 连接错误: 425 Data Connection Failed to connect.
MKD	创建文件夹	成功: 257 Directory created successfully. 失败: 504 Failed to create directory.
CWD	切换目录	成功: 250 CWD command ok. <new_path> is current directory. 原路径有误: 504 Invalid directory. 目标路径有误: 550 CWD command failed: directory not found.
PWD	返回当前目录	成功: 257 <cur_dic> is current directory. 失败: 550 Failed to get current directory.
RMD	删除文件	成功: 250 Directory deleted successfully. 失败: 550 Delete directory failed.

五．结论

通过此次项目，我对套接字编程有了更深的理解，特别是在实现FTP协议时遇到的复杂数据连接处理问题。虽然在PORT模式的实现上遇到了较大的挑战，但通过反复调试与验证，我最终使得FTP客户端和服务器能够在PORT和PASV模式下正常工作，完成文件传输、目录列表和用户认证等操作。

六．收获与感悟

1. **深入理解FTP协议**: 本项目让我深入理解了PORT和PASV两种模式的不同，尤其是在套接字管理方面的差异。虽然PASV模式相对简单，但PORT模式下，服务器主动连接客户端的复杂性增加了实现的难度。
2. **调试技巧**: 在调试过程中，详细的日志记录和逐步测试是解决问题的关键。通过输出IP地址、端口号、连接状态等调试信息，我能够快速定位问题。

3. **时间控制**: 项目调试过程中，我深刻体会到了代码开发需要时间积累，急于求成反而会导致逻辑混乱。通过合理的时间管理，我最终能够在截止日期前完成项目的所有功能。

七．未来改进

1. **增强错误处理**: 当前的实现在处理超时、连接中断等异常情况时，仍有提升空间。增加更健壮的错误处理机制将有助于提高系统的稳定性。

2. **安全性改进**: 目前FTP协议存在安全隐患，因为它以明文形式传输数据。未来可以考虑实现加密协议（如FTPS），以提升实际应用中的数据安全性。

3. **性能优化**: 在处理大文件传输或高并发情况下，代码的性能仍有优化空间。进一步提高传输效率和服务器的处理能力将是下一步改进的方向。