FTP 实验报告

李青林，5110307074

1 实验概况

本实验利用 Ruby 编程语言在 Unix-like 系统上完成了一个文件传输协议（FTP）的简单实现, 利用 Socket 接口在应用层上通过 TCP 协议实现了 FTP 协议.

2 实验内容

2.1 Server

Server 需首先开启一个 TCPSocket 监听某一端口 (默认为 21, 可配置), 默认的根目录为同目录下的 root 文件夹. 对于每一个请求, 开启一个进程处理该请求. 该进程将开启一个死循环, 若接收到的信息 (命令) 不为空, 则将该命令交给一个 CMDHandler 类处理, 否则跳出死循环.

CMDHandler 类负责具体处理每条客户端发来的命令, 维护用户当前目录等信息, 目前支持如下命令

1. **CWD** 更改 CMDHandler 类记录的当前目录
2. **PWD** 根据 CMDHandler 类记录的当前目录, 返回该信息给用户
3. **PORT** 根据 PORT 命令中的相关信息, 连接 Client 端的相应端口
4. **RETR** 读取相应文件, 利用 PORT 命令建立的数据连接将文件传送给 Client 端
5. **LIST** 读取当前目录下地文件列表, 利用 PORT 命令建立的数据连接将文件列表发送给 Client 端
6. **QUIT** 返回 Goodbye 并关闭所有 TCP 连接
7. **USER** 目前仅支持你们登录, 即对所有 USER 命令返回登录成功
8. **SYST** 返回当前系统为 UNIX
9. **DELE** 调用相应 API 尝试删除文件, 并将结果返回给用户
10. **MDTM** 调用相应 API 尝试获取文件修改时间, 并将结果返回给用户
11. **MKD** 调用相应 API 尝试建立目录, 并将结果返回给用户
12. **RMD** 调用相应 API 尝试删除目录, 并将结果返回给用户
13. **SIZE** 调用相应 API 尝试获取文件大小, 并将结果返回给用户
14. **STOR** 从 PORT 命令建立的数据连接中获取文件数据, 并保存在本地
15. **RNFR** 记录要改名的文件, 将该文件存在与否的信息返回给用户
16. **RNTO** 调用相应 API 尝试文件改名, 将改名结果返回给用户

2.2 Client

Client 通过建立一个 TCPSocket，连接相应 Server 的相应端口，并进入一个死循环，每次接受用户输入的命令，并将该命令转交给 Handler 类执行.

Handler 类负责具体处理用户输入的命令，为了方便用户使用，我自己根据 Unix-like 系统的文件操作命令及系统 FTP 的相应命令设计了如下一套客户端命令

1. **pwd, cd, user, mkdir, rmdir, rm** 这几条命令的处理方式类似，都是发送相应的 FTP 命令及参数给 Server，并显示执行结果. 其对应的 FTP 命令分别为 PWD,CWD,USER,MKD,RMD,DELE.
2. **mv** 对于 mv 命令，首先发送一条 RNFR 命令，若执行成功，则发送一条 RNTO 命令，并将执行结果返回给用户.
3. **ls** 首先发送一条 PORT 命令，与 Server 建立一个数据连接，利用该连接获取文件列表并返回给客户，最后关闭数据连接.
4. **get** 首先发送一条 PORT 命令，与 Server 建立一个数据连接，利用该连接获取文件内容并存储在本地，返回给客户执行结果，最后关闭数据连接.
5. **put** 首先发送一条 PORT 命令，与 Server 建立一个数据连接，读取文件内容，并利用该连接将文件内容发送给 Server，将执行结果返回给用户，最后关闭数据连接.
6. **bye** 发送 QUIT 给 Server，关闭所有连接并退出客户端.

3 实验总结

通过这次实验, 我理解了 FTP 协议的原理和协议细节, 学习了利用 Socket 接口设计实现简单应用层协议, 掌握了 TCP 网络应用程序的基本设计方法.