FTP 实验报告

2015013222 陈超

一、 实验内容

本次实验是有关 socket 的编程，先实现一个简单的 udp 操作，再完成一个基本的 ftp 服务端与客户端。

二、 实验环境

开发环境

- 操作系统：ubuntu 16.04
- IDE：Clion, Pycharm
- 编程语言：C，python

三、 FTP 功能介绍与实现

1、 USER 和 PASS

使用 USER [username]和 PASS [password]登录服务器。用户名只允许匿名，密码可任意输入。

由于之前在完成 udp 的过程中了解到 socket 的一些基本实现方法。因此，这两个命令实际上就是对客户端传来的命令加以判断，根据判断返回不同的消息。最初实现了正常登录，但不符合标准的 FTP，后来加入了错误码，并删除了一些凭空想象的逻辑。

2、 SYST 和 TYPE

SYST 返回“215 UNIX Type:L8” ,TYPE I 返回”200 Type set to I”

这两个命令非常简单，只是单纯地返回相应的字符串。

3、 PASV 和 PORT

PASV 和 PORT [ip, port]用于设置服务器的模式。若为 PORT 模式，客户端打开一个 socket 等待服务端连接，若为 PASV 模式，则服务端发送 ip 和 port 等待客户端连接。

做完整个实验才发现，PASV 和 PORT，自我感觉是最难的部分。曾一度让我花费整天时间调试因其引起的 bug，其实当用户发来一个 PASV 或者 PORT 时，建立一个 socket 进行监听或者等待连接并不困难，难的是在用户传来 RETR 或者 STOR 时的连接问题。针对 PASV 还比较好处理，服务端 accept 不需要管太多。而 PORT 模式中，客户端 accept 时，我发生了阻塞的问题。我一开始尝试使用线程的方式解决。后来发现只要逻辑正确，可以不用线程便能实现。

4、 RETR 和 STOR

RETR [filename]用于从服务器上下载文件，STOR [filename]用于上传文件到服务器。

对于文件下载 RETR，服务端首先判断是否处于被动或主动模式，其次判断所要下载的文件是否存在，在两者都满足的情况下返回 150，接着读取文件、传输文件。完成后返回 226。不满足则返回相应的错误码。

对于文件上传 STOR，服务端首先判断是否处于被动或者主动模式，若

处于，则创建或打开文件，并返回 150，接着接收文件信息，完成后返回 226，反之返回相应错误码。

5、 LIST

LIST 用于展示服务器当前工作目录的文件信息，发送给客户端。

客户端发送 LIST 指令，服务端首先判断是否处于被动或主动模式，若处于，则返回 150，接着发送当前工作目录的文件信息， 完成后返回 226.

6、 MKD 和 RMD

MKD [dirname] 用于在服务端工作目录中建立一个指定文件夹，

RMD [dirname] 则用于删除服务端工作目录中的指定空文件夹

这两个命令虽然是相对了，但实现起来却不是相对的。MKD 在收到客户端请求后发送 257，接着建立相应文件夹，最后再返回错误码。RMD 收到客户端请求后删除相应文件夹，并返回错误码。

7、 CWD

CWD [path]用于修改服务器当前工作路径

8、 QUIT

QUIT 用于退出登录，客户端发送 QUIT 后，服务器退出命令执行的循环过程，结束与客户端的对话。

9、 DELE

DELE [filename]用于删除服务器工作路径下的指定文件

10、 RNFR 和 RNTO

RNFO [filename]用于记录一个待修改文件名的文件

RNTO [filename]用于修改文件名

四、 FTP 操作

服务端：

命令行通过以下命令运行服务端

sudo ./server [-port PORT] [-root ROOT]

可自行设置端口和工作路径，默认情况下为：

- port : 21
- root : /tmp

客户端：

命令行通过以下命令运行客户端

sudo ./server [-ip IP] [-port PORT]

可自行设置 ip 和端口，默认情况下为：

- ip：127.0.0.1
- port：21

五、 实验特色

1、 良好的代码框架结构与风格

代码部分分为多个文件模块。对重要的功能要点实现了函数封装，对于文件的描述、函数功能、参数的介绍均加有必要的注释。

server 端包括 server(主程序), file(文件操作),directory(目录操作), mode(模式设置),handle(细节函数的实现)五个模块。

client 端包括 client(主程序),file(文件操作),mode(模式操

作),handle(细节函数的实现)。

2、 多用户请求

使用 fork 函数，通过多个子进程对应多个客户端的方式，使得服务端能够同时接收多个用户的请求。

3、 文件目录操作

用户登录一个服务器，可以进行文件目录的创建，删除，方便地查看服务端当前工作目录的文件信息，也可修改服务器工作路径。同时，支持文件删除与重命名。

六、 实验中遇到的问题

1、 autograte 测试

在 autograte 测试的时候,遇到了一些因为逻辑不对导致的阻塞以及对特殊情况处理不当导致服务端不断出现“broken pipes”,发生崩溃。但这些问题毕竟在代码中比较明显，很快就得到了解决。让我印象最深的是在做 RETR 测试的时候提示找不到文件，我发现 autograte 创建的文件带有权限，于是我尝试使用自己创建的文件进行传输，并修改了代码中相应的文件名,发现测试成功,这使我认为是我的文件权限问题没有考虑到。然而经过查找相关资料，fopen 函数本身就支持多种权限。最后经过一点点调试，才发现没有处理“/r/n”。导致文件名错误。

2、 两次 send 一次 recv

在实现 RETR 和 STOR 的过程中，出现了偶然性客户端阻塞的问题，经过调试发现，服务端需要连续两次发送消息给客户端。而客户端有一定几率一次性就接收到了两条消息，导致第二次 recv 接收不到任何消息，从而发生阻塞。因此，对第一次 recv 接受来的消息，以”/r/n“为标记判断有几条消息。从而决定是否进行第二次 recv。

七、 实验总结

本次实验我的收获很大。学习到了有关 socket 编程的知识。其间也踩到了不少坑。比如前面提到的关于 send 和 recv 收发的问题，端口号要加 htons，主动模式下 accept 发生阻塞等等。在这些问题上我花费了较长时间专门研究了一下。也理解了其中的原理。同时，这次实验也提高了我的代码调试能力。在最后提交之前，我对整个项目进行了封装、注释。总之，这次实验使我对 FTP 有了一个比较清晰全面的认识。