Socket Programming: FTP 实验报告

吴佳龙 2018013418 软件83

一、功能简介

服务器

服务器使用 C 语言以及 Berkeley Socket API 完成，支持的主要功能如下：

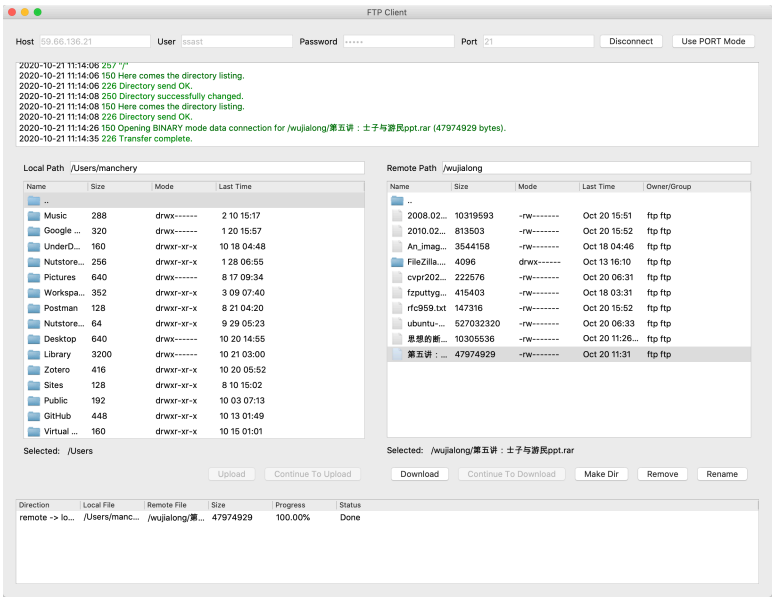
- 在指定端口和目录下运行 ftp 服务。
- 支持多个客户端同时连接而不阻塞。
- 支持 `USER, PASS, RETR, STOR, QUIT, SYST, TYPE, PORT, PASV, MKD, CWD, PWD, LIST, RMD, RNFR, RNTO, REST, DELE` 共 19 个命令。
- 支持大文件传输
- 支持断点续传（包括上传和下载）
- 能被下述自己实现的客户端连接并正常使用

客户端

客户端使用 Python 语言完成，图形界面使用 PyQt5，可运行在 MacOS 系统下。支持的主要功能如下：

- 实现了 `USER, PASS, RETR, STOR, QUIT, SYST, TYPE, PORT, PASV, MKD, CWD, PWD, LIST, RMD, RNFR, RNTO, REST, DELE` 命令。
- 能连接上述自己实现的服务器以及 vsftp 服务器并正常使用。
- 支持大文件传输
- 支持断点续传（包括上传和下载）
- 文件传输不阻塞 GUI

客户端图形界面如下：



其中**断点续传**功能的使用方法为：

- 在 local 和 remote 左右两个界面下选择两个文件，点击 Continue To Upload 则可以 server 选中的文件的尾部继续上传未传完的内容，点击 Continue To Download，则可以在 client 选中的文件的尾部继续下载未传完的内容。当选中的两个文件名不同时，会提示用户确认是否选择了正确的文件，但用户仍然选择按照上述的续传规则传送。

二、启动方式

服务器

Linux 系统下，进入 server 文件夹，运行如下命令（port 默认 21，root 默认 `\tmp`）

```
make serve
sudo ./server -port [port] -root [root]
```

客户端

依赖 PyQt5，进入 client 文件夹后：

```
python client.py
```

三、重点功能

这部分介绍重点功能的实现思路。

服务器的多客户端支持

服务器创建了一个 listen socket 之后，一旦与一台客户端握手之后建立了一个改客户端专属的 TCP 连接，则创建一个**线程**，并在该线程中处理来自该客户端的所有命令。

在创建线程时仅传入与客户端建立的 TCP 连接作为参数，因此所有线程之间互相不影响，能同时分别正常提供服务。

文件传输不阻塞客户端 GUI

当客户端被用户点击即将进行文件上传或者下载时，将利用已知的服务器地址、端口、用户、密码建立一个全新的连接，并使用 `USER` 和 `PASS` 命令重新登录（参见 `client.py` 中的 `clone_ftp_connection` 函数）。然后创建一个新的线程，在该线程中利用新创建的连接进行文件传输。

这相当于模拟了一个完全独立的“客户端”与服务器进行交互，因而**只要服务器支持多个客户端登录，即使服务器在一个线程中传输文件时是阻塞的**，依然可以做到文件传输不阻塞客户端 GUI。

断点续传

断点续传功能的使用方式如上所述，实现的重点仅需支持 `REST` 命令即可。

当需要在断点继续上传时，仅需要利用已经 `LIST` 得到的信息，获得 server 上目标文件的大小，并通过 `REST` 命令告知 server 从当前大小（也就是末尾）继续写入即可。

当需要在断点继续下载时，仅需要查看本地目标文件的大小，并通过 `REST` 告知 server 从本地已有大小的位置继续读取远程文件并传输即可。

四、实现难点及解决方案

获取本机 IP

在使用 `PORT` 和 `PASV` 命令时涉及到获取 server 和 client 自身的 IP 地址。解决方案为与公共 DNS 服务器 8.8.8.8 建立连接并从连接信息中获取到本机 IP。

代码参考了：[c++ - Get the IP address of the machine - Stack Overflow]

用 C 实现 `ls -l` 命令

我并没有在 `rfc959` 中找到关于 `LIST` 返回格式的明确规定，但是观察 `vsftp` 的返回数据，发现它的格式是与 `ls -l` 一致的，因此我也将自己的服务器的返回格式设定为 `ls -l` 的格式。因此我们需要用 C 实现 `ls -l` 的功能，返回与之相同的格式。

实现参考了 [linux 下用 c 实现 ls -l 命令 - Ritchie丶 - 博客园]。通过 `sys/stat.h` 中的相关函数和宏，如 `stat` 函数、`S_ISDIR` 宏等，可以获取到文件状态。通过 `getpwuid` 和 `getgrgid` 可以得到指定用户和用户组的信息。

server 端口占用

在调试 server 的时候需要频繁的重启 server，此时被占用的端口还未释放，会报错 `bind failed: Address already in use`，一般需要 1~2 分钟时间才能释放，这给开发效率造成了很大的困扰。

解决方案：设置 `SO_REUSEPORT` 选项，他能够支持多个进程或者线程绑定到同一端口。

根据 [The SO_REUSEPORT socket option [LWN.net]]：

> `SO_REUSEPORT` can be used with both TCP and UDP sockets. With TCP sockets, it allows multiple listening sockets—normally each in a different thread—to be bound to the same port.

client 关闭 sockect 导致 server 产生 SIGPIPE

当 server 正在传输时，client 意外关闭了 socket，此时 server 继续往该连接中写入数据，会在调用 `write` 函数时产生 SIGPIPE 信号，并造成 server 意外退出，这会影响其他客户端的正常使用。

解决方案参考了：[socket编程—— 服务器遇到Broken Pipe崩溃 - 静之深 - 博客园]。具体为通过 `signal(SIGPIPE, SIG_IGN);` 捕获 SIGPIPE 信号并忽略他，此时 `write` 函数会返回 `-1`，从而可以正常判断写入失败并结束传输。

总结

本次实验熟悉了 C 和 Python 两种语言下的 Socket 编程，熟悉了 FTP 的基本命令，也更深刻地理解了网络协议分层带来的便利与意义。