HTTP服务器/客户端实验

2020年9月29日

蔡润泽

本实验 `Github` 地址

实验内容

1、使用C语言分别实现最简单的HTTP服务器和HTTP客户端

- 服务器监听 `80` 端口，收到HTTP请求，解析请求内容，回复HTTP应答
- 对于本地存在的文件，返回 `HTTP 200 OK` 和相应文件
- 对于本地不存在的文件，返回 `HTTP 404 File Not Found`

2、服务器、客户端需要支持HTTP Get方法

3、服务器使用多线程支持多路并发

设计思路

1、已有代码框架已经能实现socket下的echo，而实现HTTP client和server首先要做的是实现HTTP GET请求与相应。

2、在查询 `HTTP GET` 请求格式后，了解到 `GET` 请求报文格式如下图：

因此，本设计在 `http-client.c` 文件里添加了字符串 `request_head` 来生成请求报文，并在程序运行时通过 `scanf` 获得用户输入的请求文件路径。在加上请求头部后， `Client` 将请求发送给 `Server` 。

3、`Server` 在收到 `HTTP GET` 请求格式后，了解到响应报文格式如下图：

因此，本设计在 `http-server.c` 文件里添加了 `int msg_handler(char * msg, char * path)` 函数来解析请求报文，并提取出请求文件的地址。

4、`Server` 在打开目标文件后，读取目标文件的内容，并将内容传递至 `return_message`，`return_message` 此外还会附上必要的响应头部属性最后`Server` 会将`return_message`发送给`Client`。

5、`Server` 为实现多线程发送文件的功能，本设计增加了 支持`pthread` 相关函数。通过 `pthread_detach` 请求处理线程来实现多路并发。相关代码如下：

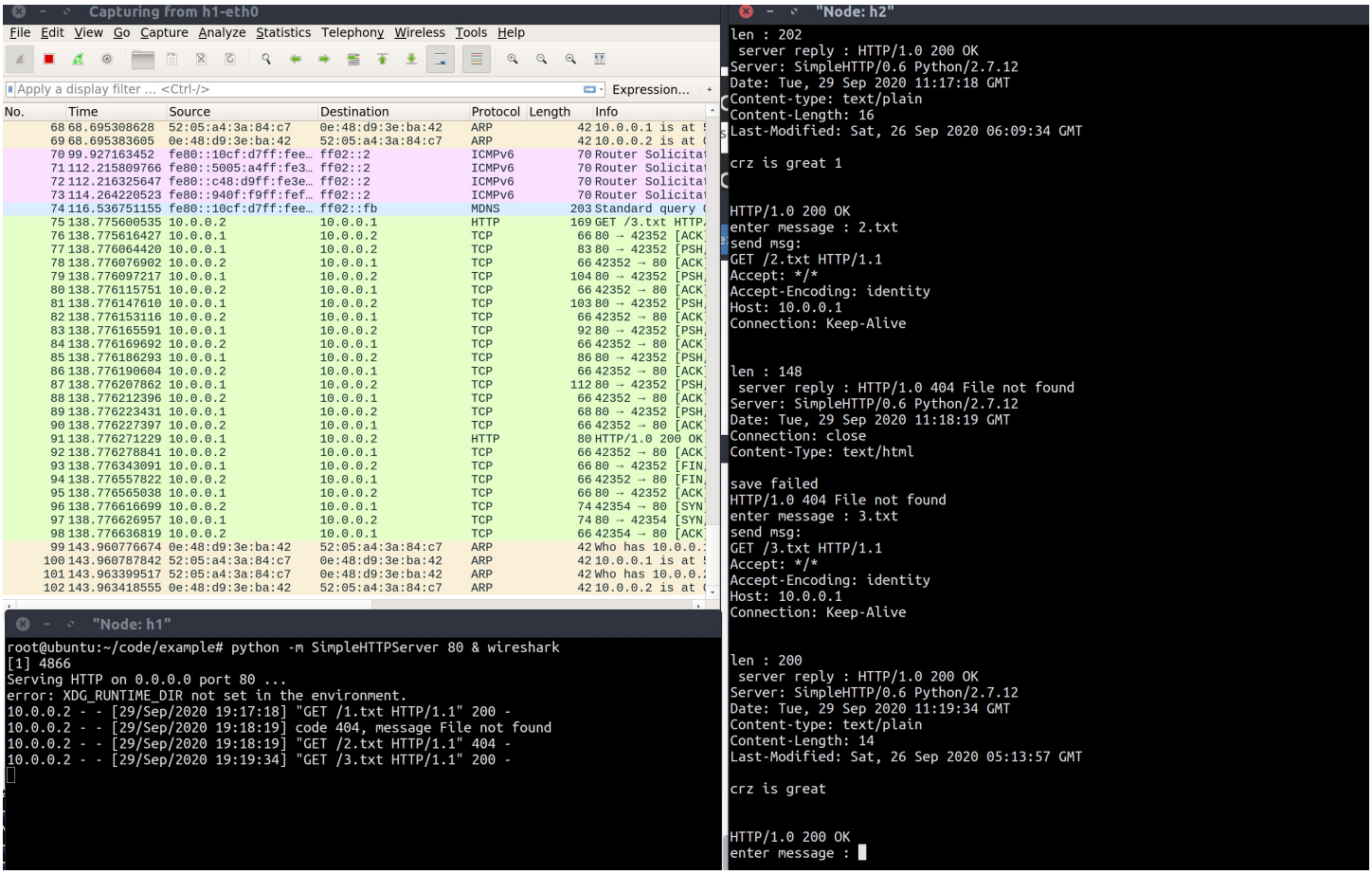
```
while(1) {
    pthread_t thread;
    pthread_create(&thread, NULL, handle_request, &s);
    pthread_detach(thread);
}
```

测试项目

1、所实现的HTTP服务器、客户端可以完成上述功能

使用客户端连续多次获取文件以及请求不存在的文件

本客户端通过与 `SimpleHTTPServer` 建立连接来测试客户端连续多次获得文件的能力，如下图：



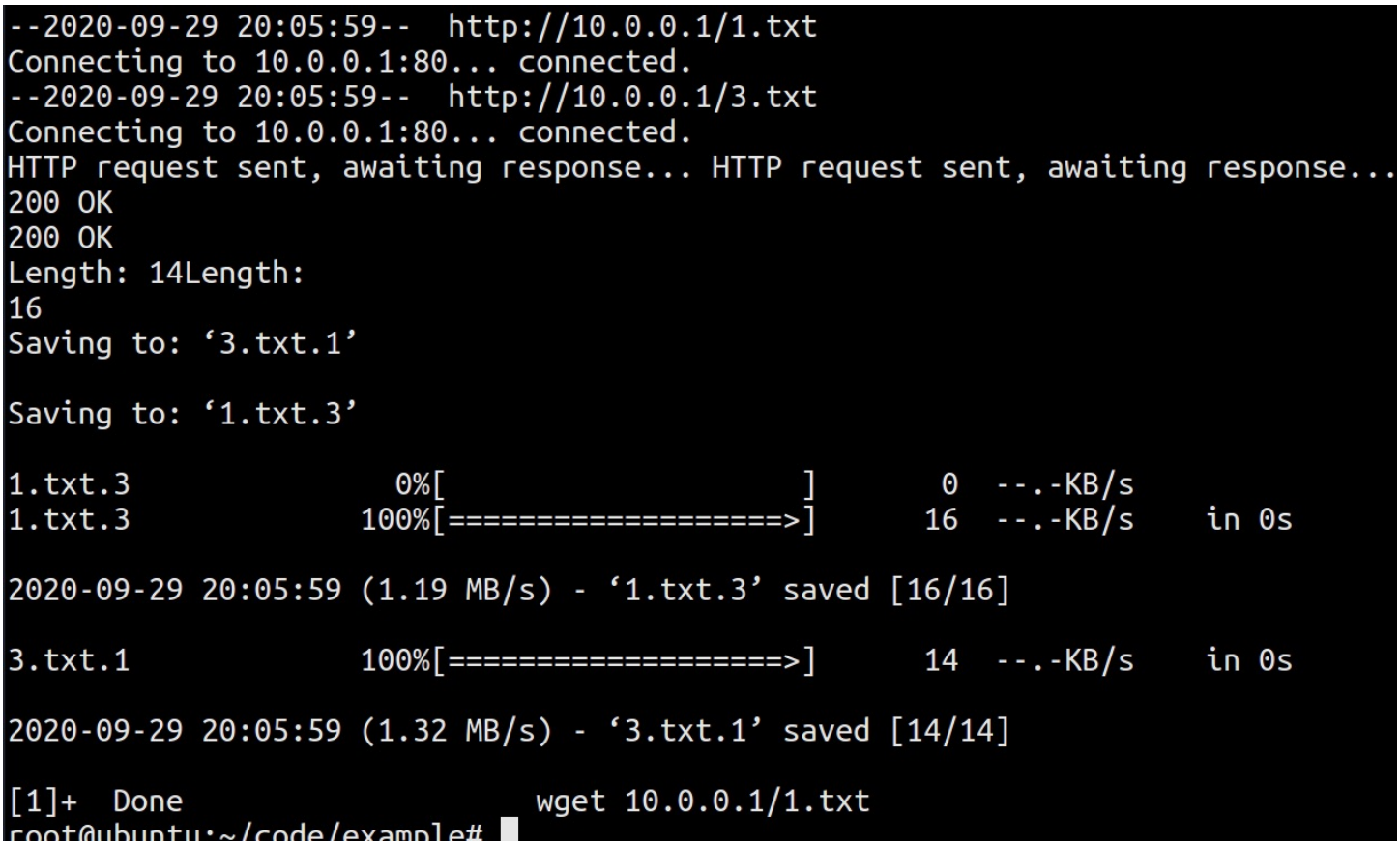
左侧 `Xterm` 界面显示了 `Client` 接收文件的反馈。`Client` 在控制台打印输出了请求的内容和相应的内容。

左侧 `Xterm` 里，`Client` 分别请求了 `1.txt`，`2.txt` 以及 `3.txt`。

`1.txt` 里的内容为 `crz is great 1`，`2.txt` 不存在，`3.txt` 里的内容为 `crz is great`。左边输出内容代表三个请求均反馈正确，且正常收取了 `1.txt` 和 `3.txt` 两个文件。接收的文件名在原文件名前增加 `recv_`，如下图：

同时启动多个客户端进程分别获取文件

通过 `pthread` 支持多线程多路并发，本设计支持同时启动多个客户端进程分别获取文件，具体的测试实现是借助 `wget 10.0.0.1/1.txt & wget 10.0.0.1/3.txt` 实现两个 `wget` 指令同时请求，测试结果如下：



由图可知， `1.txt` 和 `3.txt` 均被顺利接收。

2、使用”python -m SimpleHTTPServer 80”和wget分别替代服务器和客户端，对端可以正常工作

上述测试方式即可验证每一端均可正常运行。另外为了测试本设计中的 `Client` 和 `Server` 之间可以正常通行，增加了下述测试：

即多个`Client` 向 `Server`发送请求。结果如下：

```
Accept: */*
Accept-Encoding: identity
Host: 10.0.0.1
Connection: Keep-Alive


path: 1.txt
path: 3.txt
HTTP/1.1 200 OK
Content-Length: 16
Connection: Keep-Alive

crz is great 1


HTTP/1.1 200 OK
Content-Length: 14
Connection: Keep-Alive

crz is great
```

遇到的问题

在实验过程中，通过抓包发现，`Client` 向 `SimpleHTTPServer` 发送请求信号后， 如果多次发送请求， `SimpleHTTPServer` 在第一次正常返回响应后，TCP会发出RST信号，导致 `Client` 如果不重新建立连接就会发生异常。

因此本设计中，如果`Client` 向 `SimpleHTTPServer` 连续发包的话，需要在每次发包后重连。实现代码及其注释如下：

```
    /* The following part used depends on whether uses SimpleHttpServer.
    If you want to use SimpleHttpServer, you should enable this part
    because SimpleHttpServer will send TCP RST signal after once connection.
    */

    /*
    close(sock);
    sock = socket(AF_INET, SOCK_STREAM, 0);
    connect(sock, (struct sockaddr *)&server, sizeof(server));
    */
```

参考文献及网站

[1] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

[2] https://www.cnblogs.com/an-wen/p/11180076.html