

# Socket应用编程实验报告

张翔雨 2018K8009929035

## 一、实验题目：Socket应用编程实验

## 二、实验内容

- 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
    - 所请求的文件在服务器程序当前目录中，返回HTTP 200 OK和相应文件
    - 所请求的文件不在服务器程序当前目录中，，返回HTTP 404 File Not Found
  - 服务器、客户端只需要支持HTTP Get方法，不需要支持Post等方法
  - 服务器使用多线程支持多路并发
- 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
  - 同时启动多个客户端进程分别获取文件
  - 使用客户端请求不存在的文件

## 三、实验流程

- 执行 `sudo python2 topo.py` 命令，生成包括两个端节点的网络拓扑
- 在主机h1上运行HTTP服务器程序，监听80端口
  - h1 # `./http-server 80`
- 在主机h2上运行HTTP客户端程序，获取服务器上某个文件
  - h2 # `./http-client http://10.0.0.1:80/test.html`

- 分别使用 `python -m SimpleHTTPServer 80` 和 `wget` 替代服务器和客户端程序，测试自己实现的是否正确

## 四、实验结果

### 1. Client客户端请求实验

#### (1) 关键代码说明

首先将程序的命令行参数切分得到请求的IP、端口号以及文件名字，这部分代码比较简单但比较长，不在实验报告中展示。得到之后建立socket文件描述符，并绑定监听地址时将对应参数填入。

```
server.sin_addr.s_addr = inet_addr(ip);
server.sin_family = AF_INET;
server.sin_port = htons(port);
```

然后申请建立连接，之后需要构造我们要传输的http请求的头部，代码如下，需要注意的是最后需要添加一个"`\r\n`"

```
strcat(message,"GET /");
strcat(message,filename);
strcat(message," HTTP/1.1\r\n");
strcat(message,"User-Agent: ZXY's simple client\r\n");
strcat(message,"Host: ");
strcat(message,ip);
strcat(message,"\r\nConnection:Keep-Alive");
strcat(message,"\r\n\r\n");
```

头部中包含的信息主要包括了请求的文件名字，使用的HTTP协议，访问的Host，连接方式以及放出请求方的名字。

成功发出消息后，就需要做接受应答的消息的准备，首先由于客户端和服务端位于同一目录下，为了做区别，我们需要生成一个与请求文件名有关但不同的接受到的文件的名字，比如请求文件名为1，生成的文件名即为1\_recv。

```
char recv_file[2000] = {0};
strcat(recv_file,filename);
strcat(recv_file,"_recv");
fd = fopen(recv_file, "w+");//如果文件不存在，则生成一个新的；如果文件存在，则清空重新写
```

当recv函数收到返回的消息后，我们需要对消息进行解析，其中有用的信息有获取的文件的文件的大小，因此我们需要找到header中"Content-Length:"这一行，得到文件大小。

```
sonchar = strstr(server_reply, "Content-Length:");
sonchar = sonchar + 15;
int k = 0;
int filesize = 0;
while(sonchar[k] != '\r')
{
    if(sonchar[k] >= '0' && sonchar[k] <= '9')
        filesize = filesize * 10 + sonchar[k] - '0';
    k ++;
}
```

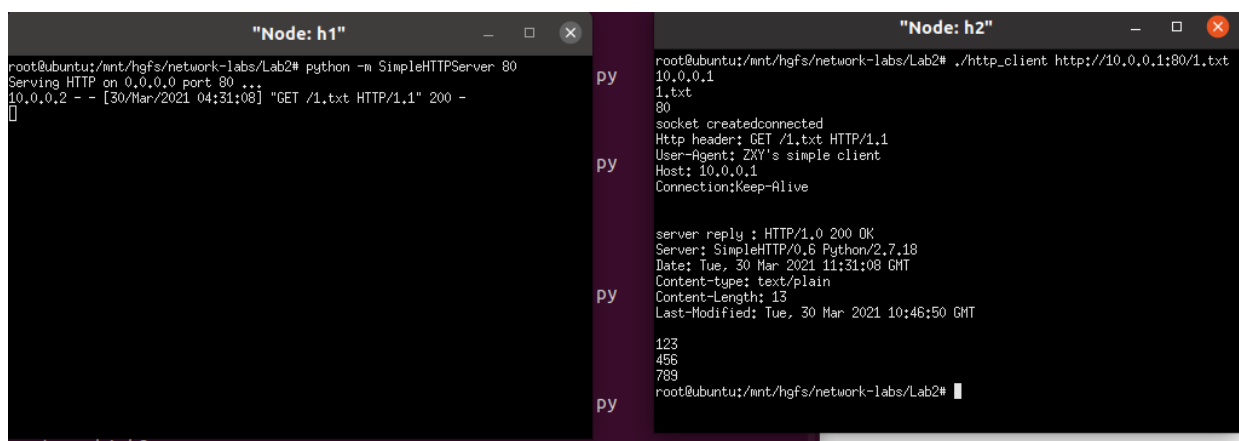
之后我们就需要找到http应答的正文，标志是"\r\n\r\n"，连续的两个换行出现代表头部结束了。得到正文后，将其根据文件大小写入文件即可。

```
sonchar = strstr(server_reply, "\r\n\r\n");
sonchar = sonchar + 4;
fwrite(sonchar, filesize, 1, fd);
fclose(fd);
```

这样我们就完成了一次http的请求与应答。

## (2) 测试结果

与simple http server进行交互：



```
"Node: h1"
root@ubuntu:/mnt/hgfs/network-labs/Lab2# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
10.0.0.2 - - [30/Mar/2021 04:31:08] "GET /1.txt HTTP/1.1" 200 -

"Node: h2"
root@ubuntu:/mnt/hgfs/network-labs/Lab2# ./http_client http://10.0.0.1:80/1.txt
10.0.0.1
1.txt
80
socket createdconnected
Http header: GET /1.txt HTTP/1.1
User-Agent: ZXY's simple client
Host: 10.0.0.1
Connection:Keep-Alive

server reply : HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/2.7.18
Date: Tue, 30 Mar 2021 11:31:08 GMT
Content-type: text/plain
Content-Length: 13
Last-Modified: Tue, 30 Mar 2021 10:46:50 GMT

123
456
789
root@ubuntu:/mnt/hgfs/network-labs/Lab2#
```

连续多次获取文件结果：

```

"Node: h1"
root@ubuntu:/mnt/hgfs/network-labs/Lab2# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
10.0.0.2 - - [30/Mar/2021 04:31:08] "GET /1.txt HTTP/1.1" 200 -
10.0.0.2 - - [30/Mar/2021 04:32:28] "GET /1.txt HTTP/1.1" 200 -
10.0.0.2 - - [30/Mar/2021 04:32:29] "GET /1.txt HTTP/1.1" 200 -
10.0.0.2 - - [30/Mar/2021 04:32:30] "GET /1.txt HTTP/1.1" 200 -
10.0.0.2 - - [30/Mar/2021 04:32:31] "GET /1.txt HTTP/1.1" 200 -
10.0.0.2 - - [30/Mar/2021 04:32:32] "GET /1.txt HTTP/1.1" 200 -
10.0.0.2 - - [30/Mar/2021 04:32:36] "GET /1.txt HTTP/1.1" 200 -
10.0.0.2 - - [30/Mar/2021 04:32:41] "GET /1.txt HTTP/1.1" 200 -
10.0.0.2 - - [30/Mar/2021 04:32:46] "GET /1.txt HTTP/1.1" 200 -
10.0.0.2 - - [30/Mar/2021 04:32:51] "GET /1.txt HTTP/1.1" 200 -
10.0.0.2 - - [30/Mar/2021 04:32:59] "GET /1.txt HTTP/1.1" 200 -
10.0.0.2 - - [30/Mar/2021 04:33:01] "GET /1.txt HTTP/1.1" 200 -
[]

"Node: h2"
123
456
789
root@ubuntu:/mnt/hgfs/network-labs/Lab2# ./http_client http://10.0.0.1:80/1.txt
10.0.0.1
1.txt
80
socket createdconnected
Http header: GET /1.txt HTTP/1.1
User-Agent: ZXY's simple client
Host: 10.0.0.1
Connection:Keep-Alive

server reply : HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/2.7.18
Date: Tue, 30 Mar 2021 11:32:28 GMT
Content-type: text/plain
Content-Length: 13
Last-Modified: Tue, 30 Mar 2021 10:46:50 GMT

123
456
789

```

正确性验证:

```

root@ubuntu:/mnt/hgfs/network-labs/Lab2# ./http_client http://10.0.0.1:80/1.txt
10.0.0.1
1.txt
80
socket createdconnected
Http header: GET /1.txt HTTP/1.1
User-Agent: ZXY's simple client
Host: 10.0.0.1
Connection:Keep-Alive

server reply : HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/2.7.18
Date: Tue, 30 Mar 2021 11:34:30 GMT
Content-type: text/plain
Content-Length: 13
Last-Modified: Tue, 30 Mar 2021 10:46:50 GMT

123
456
789
root@ubuntu:/mnt/hgfs/network-labs/Lab2# diff 1.txt 1.txt_recv
root@ubuntu:/mnt/hgfs/network-labs/Lab2#

```

## 2. Server服务器应答实验

### (1) 关键代码说明

服务器程序的参数只有端口号，获取到端口号之后流程与示例程序相同直到开始监听。之后accept每接受到一次连接请求便起一个线程处理以实现并发访问。多线程处理代码如下：

```

while(1)
{
    int c = sizeof(struct sockaddr_in);
    if ((cs = accept(s, (struct sockaddr *)&client, (socklen_t *)&c)) < 0)
    {
        printf("accept failed\n");
        continue;
    }
    //create a thread for each client
    pthread_t *pid;
    pid = (pthread_t *)malloc(sizeof(pthread_t));
    pthread_create(pid, NULL, (void *)func_server, (void *)cs);
}

```

之后单独分析处理函数 `func_server()` ,首先当我们收到一个HTTP请求的时候，首先要分析头部中的有用信息，首先是GET后的要获取的文件的名字，文件名字的截止标志是后面跟着的" HTTP/1.1\r\n"，做一次匹配后调用KMP算法获取截止位置即可。

```

    son_str = strstr(msg, "GET /");
    son_str = son_str + 5;
    int k = 0;
    k = KMP(son_str, " HTTP/1.1\r\n");
    filename[k] = 0;
    int x = 0;
    for (x = 0; x < k; x++)
        filename[x] = son_str[x];

```

之后要打开对应的文件，如果不存在，则要返回HTTP 404 File Not Found

```

    if((fd = fopen(filename, "r"))==NULL)//can't find the file
    {
        printf("No such file.\n");
        write(sock, "HTTP/1.1 404 file not found\r\n\r\n", strlen("HTTP/1.1 404 file not found\r\n\r\n"));
        return -1;
    }

```

找到文件后，首先要获取文件大小，利用的是 `ftell` 的方法，获取的文件大小后，组织HTTP应答header

```
strcat(ret_msg, "HTTP/1.1 200 ok\r\n");
strcat(ret_msg, "Server: ZXY's simple http server\r\n");
strcat(ret_msg, "Content-Type: text/plain\r\n");
strcat(ret_msg, "Content-Length: ");
strcat(ret_msg, size_str);
strcat(ret_msg, "\r\n\r\n");
```

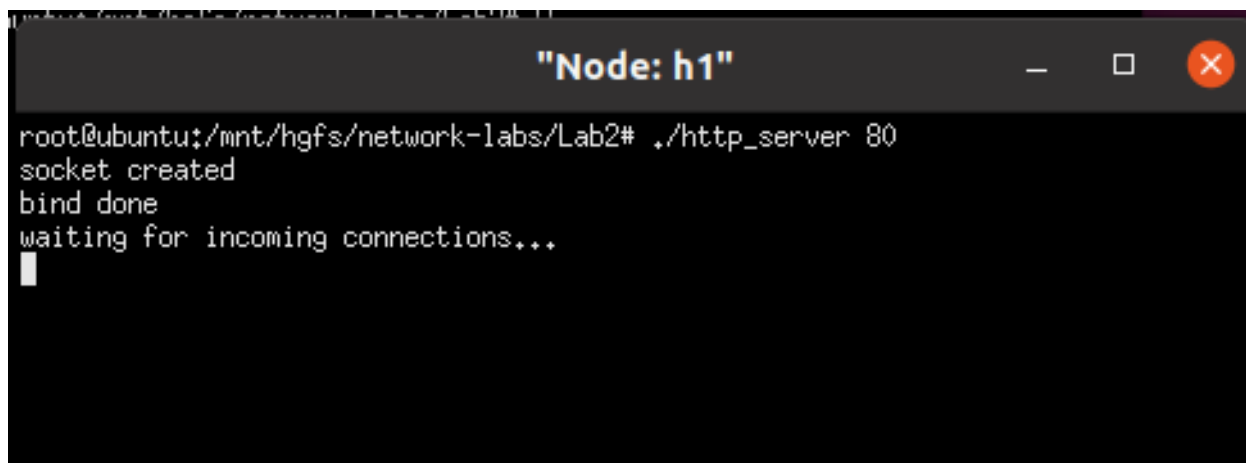
头部中包含的关键信息包括HTTP 200 OK以及文件大小，之后便是文件的具体内容，文件内容的读取通过fgets实现，由于fgets读到换行符会终止，所以用while循环逐行添加到报文的body中。

```
while (!feof(fd))
{
    memset(file_content, 0, sizeof(msg));
    fgets(file_content, sizeof(file_content), fd);
    strcat(ret_msg, file_content);
}
```

之后传递应答，完成服务器任务。

## (2) 测试结果

服务器启动：

A terminal window titled "Node: h1" with standard window controls. The terminal shows the command './http\_server 80' being executed in a directory '/mnt/hgfs/network-labs/Lab2#'. The output of the program is: 'socket created', 'bind done', and 'waiting for incoming connections...'. A cursor is visible on the line following the last output.

```
root@ubuntu:/mnt/hgfs/network-labs/Lab2# ./http_server 80
socket created
bind done
waiting for incoming connections...
█
```

请求存在的文件：

```
"Node: h1"
root@ubuntu:/mnt/hgfs/network-labs/Lab2# ./http_server 80
socket created
bind done
waiting for incoming connections...
connection accepted
GET /1.txt HTTP/1.1
User-Agent: Wget/1.20.3 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: 10.0.0.1
Connection: Keep-Alive

filename: 1.txt
13
client disconnected

"Node: h2"
root@ubuntu:/mnt/hgfs/network-labs/Lab2# wget http://10.0.0.1:80/1.txt
--2021-03-30 05:39:15-- http://10.0.0.1/1.txt
Connecting to 10.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 ok
Length: 13 [text/plain]
Saving to: '1.txt.4'

1.txt.4      100%[=====] 13 --.-KB/s  in 0s

2021-03-30 05:39:16 (1.43 MB/s) - '1.txt.4' saved [13/13]

root@ubuntu:/mnt/hgfs/network-labs/Lab2# diff 1.txt 1.txt.4
root@ubuntu:/mnt/hgfs/network-labs/Lab2#
```

请求不存在的文件：

```
filename: 1.txt
13
client disconnected
connection accepted
GET /test.txt HTTP/1.1
User-Agent: Wget/1.20.3 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: 10.0.0.1
Connection: Keep-Alive

filename: test.txt
No such file.

2021-03-30 05:39:16 (1.43 MB/s) - '1.txt.4' saved [13/13]

root@ubuntu:/mnt/hgfs/network-labs/Lab2# diff 1.txt 1.txt.4
root@ubuntu:/mnt/hgfs/network-labs/Lab2# wget http://10.0.0.1:80/test.txt
--2021-03-30 05:40:18-- http://10.0.0.1/test.txt
Connecting to 10.0.0.1:80... connected.
HTTP request sent, awaiting response... 404 file not found
2021-03-30 05:40:18 ERROR 404: file not found.

root@ubuntu:/mnt/hgfs/network-labs/Lab2#
```

并发访问：（由于wget只会显示一次获取的结果，这里展示前后目录下的文件以确保两次获取都完成了）

```
13
client disconnected
client disconnected
connection accepted
GET /3.txt HTTP/1.1
User-Agent: Wget/1.20.3 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: 10.0.0.1
Connection: Keep-Alive

filename: 3.txt
13
connection accepted
GET /1.txt HTTP/1.1
User-Agent: Wget/1.20.3 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: 10.0.0.1
Connection: Keep-Alive

filename: 1.txt
13
client disconnected
client disconnected

root@ubuntu:/mnt/hgfs/network-labs/Lab2# ls
1.txt http_client http_server Makefile topo.py
3.txt http_client.c http_server.c Socket用工程.md wget-log
root@ubuntu:/mnt/hgfs/network-labs/Lab2# wget http://10.0.0.1:80/1.txt & wget h
http://10.0.0.1:80/3.txt
[1] 5768
--2021-03-30 05:44:04-- http://10.0.0.1/3.txt
Connecting to 10.0.0.1:80...
Redirecting output to 'wget-log.1'.
connected.
HTTP request sent, awaiting response... 200 ok
Length: 13 [text/plain]
Saving to: '3.txt.1'

3.txt.1      100%[=====] 13 --.-KB/s  in 0s

2021-03-30 05:44:04 (1.58 MB/s) - '3.txt.1' saved [13/13]

root@ubuntu:/mnt/hgfs/network-labs/Lab2# ls
1.txt 3.txt.1 http_server Socket用工程.md wget-log.1
1.txt.1 http_client http_server.c topo.py
3.txt http_client.c Makefile wget-log
[1]+ Done wget http://10.0.0.1:80/1.txt
root@ubuntu:/mnt/hgfs/network-labs/Lab2#
```

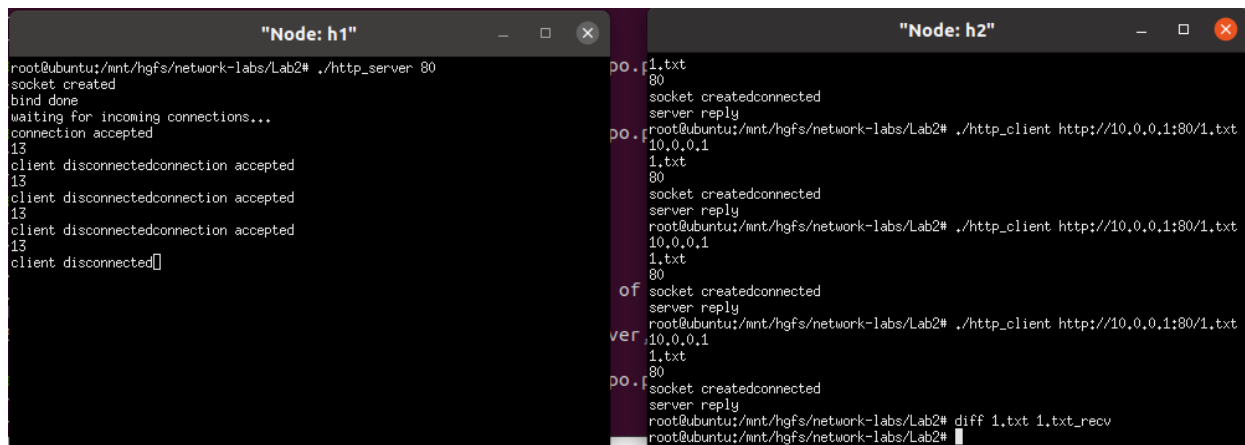
并发正确性验证：

```
[1]+ Done diff 1.txt.1 1.txt.1
root@ubuntu:/mnt/hgfs/network-labs/Lab2# diff 1.txt.1 1.txt
root@ubuntu:/mnt/hgfs/network-labs/Lab2# diff 3.txt.1 3.txt
root@ubuntu:/mnt/hgfs/network-labs/Lab2#
```

测试结果均符合预期。

### 3. server与client配合完成实验

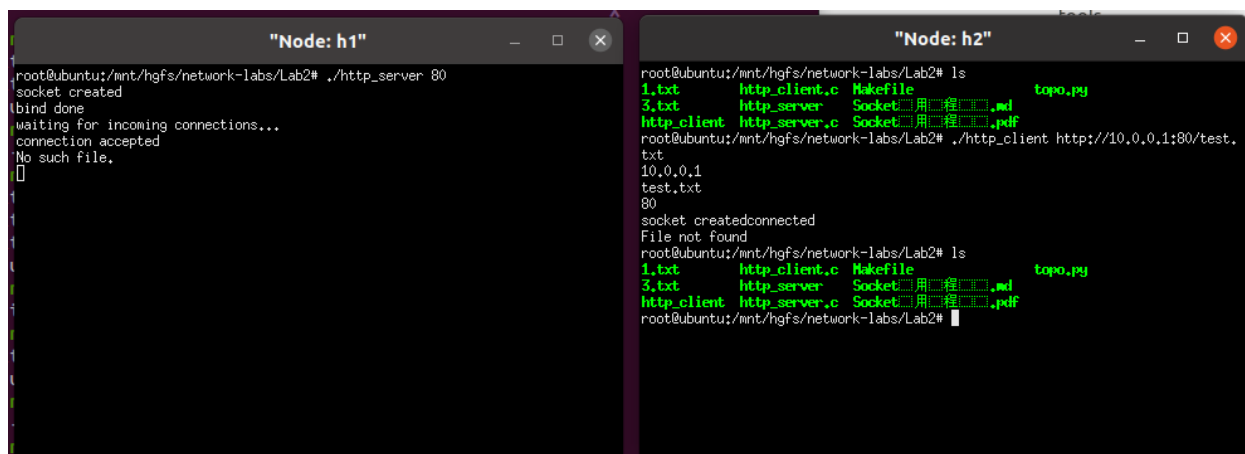
#### (1) client多次获取服务器文件



```
Node: h1
root@ubuntu:/mnt/hgfs/network-labs/Lab2# ./http_server 80
socket created
bind done
waiting for incoming connections...
connection accepted
13
client disconnected
connection accepted
13
client disconnected
connection accepted
13
client disconnected
[]

Node: h2
root@ubuntu:/mnt/hgfs/network-labs/Lab2# ./http_client http://10.0.0.1:80/1.txt
1.txt
10.0.0.1
1.txt
80
socket created
connected
server reply
root@ubuntu:/mnt/hgfs/network-labs/Lab2# ./http_client http://10.0.0.1:80/1.txt
10.0.0.1
1.txt
80
socket created
connected
server reply
root@ubuntu:/mnt/hgfs/network-labs/Lab2# diff 1.txt 1.txt_recv
root@ubuntu:/mnt/hgfs/network-labs/Lab2#
```

#### (2) 使用客户端请求不存在的文件

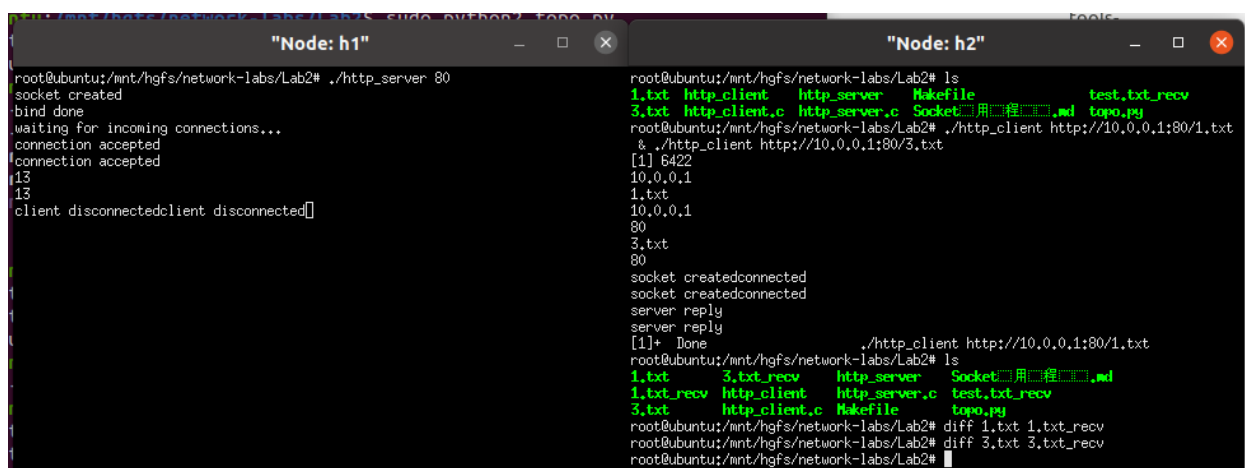


```
Node: h1
root@ubuntu:/mnt/hgfs/network-labs/Lab2# ./http_server 80
socket created
bind done
waiting for incoming connections...
connection accepted
No such file.
[]

Node: h2
root@ubuntu:/mnt/hgfs/network-labs/Lab2# ls
1.txt http_client.c Makefile topop.py
3.txt http_server Socket用程.md topo.py
http_client http_server.c Socket用程.pdf
root@ubuntu:/mnt/hgfs/network-labs/Lab2# ./http_client http://10.0.0.1:80/test.
txt
10.0.0.1
test.txt
80
socket created
connected
File not found
root@ubuntu:/mnt/hgfs/network-labs/Lab2# ls
1.txt http_client.c Makefile topop.py
3.txt http_server Socket用程.md topo.py
http_client http_server.c Socket用程.pdf
root@ubuntu:/mnt/hgfs/network-labs/Lab2#
```

正确报错，也未产生多余的文件。

#### (3) 使用多个客户端请求服务器上的文件



```
Node: h1
root@ubuntu:/mnt/hgfs/network-labs/Lab2# ./http_server 80
socket created
bind done
waiting for incoming connections...
connection accepted
connection accepted
13
13
client disconnected
client disconnected
[]

Node: h2
root@ubuntu:/mnt/hgfs/network-labs/Lab2# ls
1.txt http_client http_server Makefile test.txt_recv
3.txt http_client http_server.c Socket用程.md topo.py
http_client http_server.c Socket用程.pdf
root@ubuntu:/mnt/hgfs/network-labs/Lab2# ./http_client http://10.0.0.1:80/1.txt
[1] 6422
10.0.0.1
1.txt
10.0.0.1
80
3.txt
80
socket created
connected
socket created
connected
server reply
server reply
[1]+ Done ./http_client http://10.0.0.1:80/1.txt
root@ubuntu:/mnt/hgfs/network-labs/Lab2# ls
1.txt 3.txt_recv http_server Socket用程.md topo.py
1.txt_recv http_client http_server.c test.txt_recv
3.txt http_client.c Makefile
root@ubuntu:/mnt/hgfs/network-labs/Lab2# diff 1.txt 1.txt_recv
root@ubuntu:/mnt/hgfs/network-labs/Lab2# diff 3.txt 3.txt_recv
root@ubuntu:/mnt/hgfs/network-labs/Lab2#
```

发现两个请求都正确执行并处理，得到的文件也正确。



## 五、结果分析

### 1.客户端实验

在做客户端实验时发现，使用自己的程序向SimpleHTTPServer发请求时，有时会出现只能获得报文头部得不到报文body的情况，而使用自己的服务器程序时不会出现这个问题，猜测是自带的服务器连接会出现小的问题。

### 2.服务器实验

服务器实验开始时编译不通过，花了很长时间才想起使用pthread库在linux下编译时需要加链接库，需要修改makefile，因此浪费了时间。

## 六、实验总结

这次实验的难点主要在于http报文的撰写与处理，而socket应用部分由于给出的示例程序完整的展示了使用的过程，因此难度不是很大。在查询资料时发现，网上关于http报文头部具体如何书写的中文资料内容较少或过于复杂，对实验代码的撰写造成了很大的困难，我采取了比较笨的办法，将wget和SimpleHTTPServer发送的报文打印出来，再对照资料思考，结合产生了代码里的http头部，因此加深了我对HTTP报文的理解。