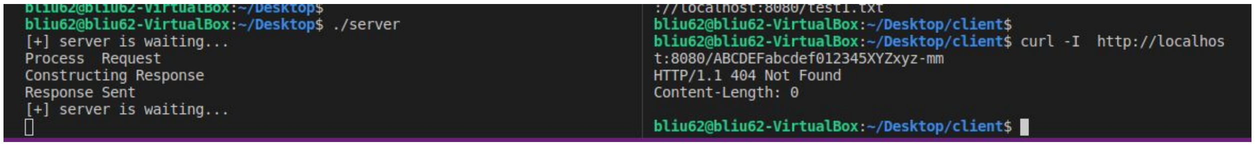
Testing

Connection check

First, check the connection between server and client



```
bliu62@bliu62-VirtualBox:~/Desktop$
bliu62@bliu62-VirtualBox:~/Desktop$ ./server
[+] server is waiting...
Process  Request
Constructing Response
Response Sent
[+] server is waiting...
```

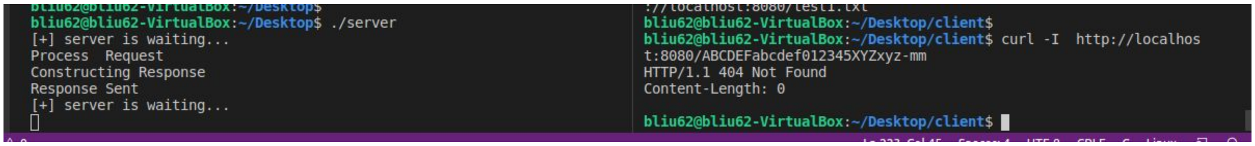
```
://localhost:8080/test1.txt
bliu62@bliu62-VirtualBox:~/Desktop/client$
bliu62@bliu62-VirtualBox:~/Desktop/client$ curl -I  http://localhos
t:8080/ABCDEFabcdef012345XYZxyz-mm
HTTP/1.1 404 Not Found
Content-Length: 0

bliu62@bliu62-VirtualBox:~/Desktop/client$
```

Give the command on the client side, the server responses and connection is successful.

Function check

Head

Server **Client**



```
bliu62@bliu62-VirtualBox:~/Desktop$
bliu62@bliu62-VirtualBox:~/Desktop$ ./server
[+] server is waiting...
Process  Request
Constructing Response
Response Sent
[+] server is waiting...
```

```
://localhost:8080/test1.txt
bliu62@bliu62-VirtualBox:~/Desktop/client$
bliu62@bliu62-VirtualBox:~/Desktop/client$ curl -I  http://localhos
t:8080/ABCDEFabcdef012345XYZxyz-mm
HTTP/1.1 404 Not Found
Content-Length: 0

bliu62@bliu62-VirtualBox:~/Desktop/client$
```

Send the HEAD message to the server; the server responded the message according to the status of file

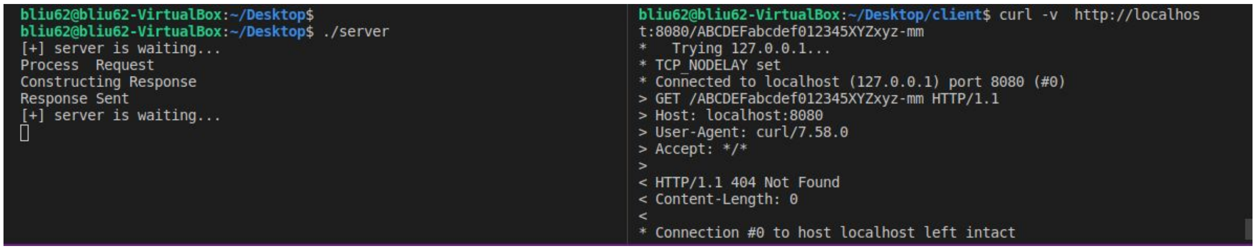
Send 403 if the file is forbidden

404 if the file is not Found

500 if Internal server error occurs

200 if the file exists and readable.

GET



Server

```
bliu62@bliu62-VirtualBox:~/Desktop$
bliu62@bliu62-VirtualBox:~/Desktop$ ./server
[+] server is waiting...
Process  Request
Constructing Response
Response Sent
[+] server is waiting...
```

Client

```
bliu62@bliu62-VirtualBox:~/Desktop/client$ curl -v  http://localhos
t:8080/ABCDEFabcdef012345XYZxyz-mm
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /ABCDEFabcdef012345XYZxyz-mm HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 404 Not Found
< Content-Length: 0
<
* Connection #0 to host localhost left intact
```

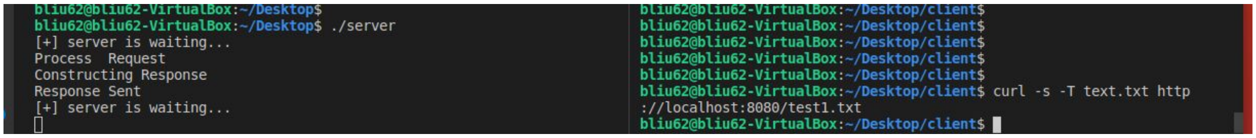
Send the GET message to the server; the server responds to the message according to the status of file and sends the content of file to the Client side. We send the data of file to a new file,and use DIFF to test whether there is difference between the file we receive and file we write

Send 403 if the file is forbidden

404 if the file is not Found

200 if the file exists and readable.

PUT



Server

```
bliu62@bliu62-VirtualBox:~/Desktop$
bliu62@bliu62-VirtualBox:~/Desktop$ ./server
[+] server is waiting...
Process  Request
Constructing Response
Response Sent
[+] server is waiting...
```

Client

```
bliu62@bliu62-VirtualBox:~/Desktop/client$
bliu62@bliu62-VirtualBox:~/Desktop/client$
bliu62@bliu62-VirtualBox:~/Desktop/client$
bliu62@bliu62-VirtualBox:~/Desktop/client$
bliu62@bliu62-VirtualBox:~/Desktop/client$
bliu62@bliu62-VirtualBox:~/Desktop/client$ curl -s -T text.txt http
://localhost:8080/test1.txt
bliu62@bliu62-VirtualBox:~/Desktop/client$
```

Send the PUT message to the server; the server responds to the message according to the status of file and stores the content of file sended by client in the given filename. Using diff to test whether there is a difference between the client's file and file we created.

Send

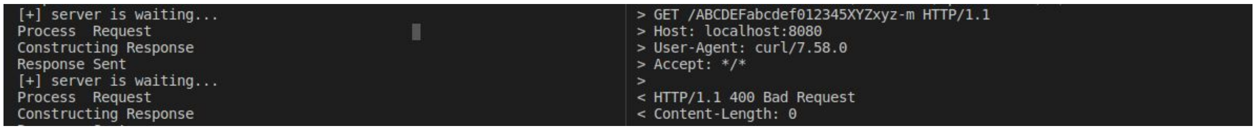
500 if Internal server error occurs

200 if the file was received and created successfully.

Error Handling

400 Bad Request-client side

If the filename is not in correct format, the server will send a bad request to the client.

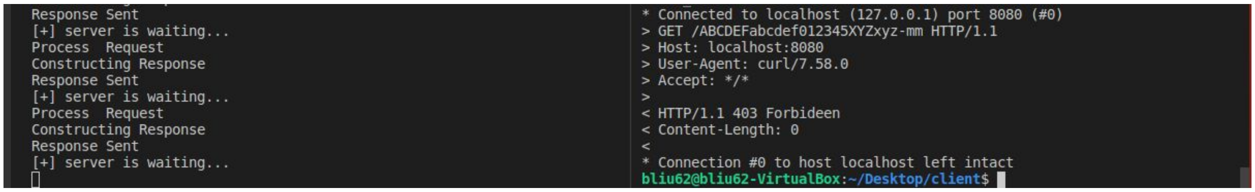


```
[+] server is waiting...
Process  Request
Constructing Response
Response Sent
[+] server is waiting...
Process  Request
Constructing Response
```

```
> GET /ABCDEFabcdef012345XYZxyz-m HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 400 Bad Request
< Content-Length: 0
```

The length of the filename is not 27 characters or contains invalid characters , the server sends a Bad Request.

403 Forbidden

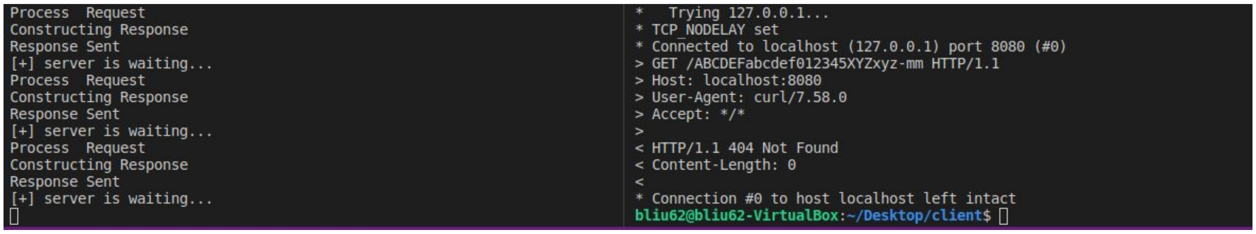


```
Response Sent
[+] server is waiting...
Process  Request
Constructing Response
Response Sent
[+] server is waiting...
Process  Request
Constructing Response
Response Sent
[+] server is waiting...
```

```
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /ABCDEFabcdef012345XYZxyz-mm HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 403 Forbideen
< Content-Length: 0
<
* Connection #0 to host localhost left intact
bliu62@bliu62-VirtualBox:~/Desktop/client$
```

If the client does not have access to the file, the server will send a 403 Forbidden to the client.

404 Not found



```
Process  Request
Constructing Response
Response Sent
[+] server is waiting...
Process  Request
Constructing Response
Response Sent
[+] server is waiting...
Process  Request
Constructing Response
Response Sent
[+] server is waiting...
```

```
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /ABCDEFabcdef012345XYZxyz-mm HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 404 Not Found
< Content-Length: 0
<
* Connection #0 to host localhost left intact
bliu62@bliu62-VirtualBox:~/Desktop/client$
```

If the file asked by the client does not exist, the server will send 404 Not found to the client.

Question

1.The portion of code to handle errors properly is about 50%, half of my time is spent on how the server responds accordingly with different errors. 80 percent of time we devote to debugging my code to ensure the server behaves properly to the different errors.

2. We must handle the 400 Bad REQUEST, 403 FORBIDDEN, 404 NOT FOUND, AND 500 INTERNAL SERVER ERROR

3. In my program, I respond with 500 INTERNAL SERVER ERROR if the connection is closed or ending the communication early. If the connection is closed, the server will try to read chunks of code. After checking if the data read from the socket does not match the data written to the file,, there must be loss of connection.

4. Yes the endianness is important to the HTTP protocol since the order of byte is crucial when sending and receiving on the socket. The endianness decides where the most significant and least significant bits are. The order of message will definitely influence the outcome of communication between server and client.