컴퓨터 네트워크
과제3
HTTP
201402329
김수현

1. **과제 해결방법**
    a. **해결방법**
        i. server
            1. 소켓을 열고, 클라이언트가 연결될때마다 스레드를 생성한다.
            2. 생성된 스레드에서는 Client로부터 데이터를 받고
            3. 받은 데이터를 parsing하여 원하는 url을 얻고,
            4. url이 있다면 response헤더를 붙여 보내준다
        ii. Client
            1. 일단 서버와 연결이되면 request 를 원하는 Url과 함께 보낸다.
            2. 만약 server에서 response header에 200이 뜨면 parsing을 해서 파일을 저장한다.
            3. 404 error가 뜨면 파일이 없다고 출력한다.
    b. **코드 설명**
        i. server

```
def __init__(self):
    self.ip_address = '127.0.0.1'
    self.port_number = 2345
    self.FILE_OK = 'HTTP/1.1 200 OK\r\n'
    self.FILE_NO = 'HTTP/1.1 404 Not Found\r\n'
```

            1. 생성자
                a. ip주소는 localhost를 나타내는 127.0.0.1,
                b. 포트번호는 예약숫자를 제외한 임의의 번호 2345로 설정
                c. Response Header두개를 설정
            2. CreateSocket(self)

```
def createSocket(self):

        server_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        try:
            server_socket.bind((self.ip_address,self.port_number))
        except:
            print("bind error")
            server_socket.close()
            return

        print("server socekt open...")
        print("listening...")
        server_socket.listen(5)

        print("Connected with client")
        while True:
           client_socket,address = server_socket.accept()
           threading.Thread(target=self.data_send_recv,
args=(client_socket,address)).start()
```

                a. 인자, 반환값 - 없음
                b. 소켓을 열어 bind를 시킨후,

c. socket.listen(5)를 통하여 클라이언트를 5명까지 받도록한다.
d. 클라이언트 소켓을 accept의 한다.
e. accept할때마다 함수 data_send_recv를 실행하도록 하는 스레드를 실행하도록 한다.

3. **data_send_recv**

```
def data_send_recv(self,socket,address):

        data = socket.recv(5000)
        request_url = self.parsing_request_header(data)
        response_header = self.make_response_header(request_url)
        socket.send(response_header.encode())
        time.sleep(5)
```

a. 인자 - clientSocket,address 반환값- 없음
b. 데이터를 클라이언트 소켓으로 받은후,
c. 헤더를 파싱하여 client가 원하는 url만 뻡아낸다.
d. 그 request url 을이용하여 Response를 만든다.
e. 만든것을 clinetsocket에 보내고
f. 보낸후 전부 보내질때까지 5초간 sleep한다.

4. parsing_request_header

```
def parsing_request_header(self,data):
    '''
      요청한 URL 을 파악한다.
       @return:URL
       @param: http header
    '''

    data_str = str(data.decode('utf-8'))
    split_data = data_str.split('\r\n')
    get_url = split_data[0].split()
    print(get_url)
    url = get_url[1]
    url = url.replace('/','')
    if url[0] == '/':
        url = url.replace('/','')
    return url
```

a. 인자 - clientsocket으로 부터 받은 data 반환값 - url
b. 데이터를 먼저 byte파일로 되어 있는것을 utf-8로 디코딩하여 문자열로 만든다
c. Request 메세지는 다음과 같은 형태로 받는다:

```
GET page.html HTTP/1.1
Accept:*/*
Accept-Encoding: gzip,deflate
Accept-Language: utf-8
```

여기서 앞줄만 뽑아내어 GET method 다음 url만뽑아낸다.

1. make_respse_header,response_header_template

```
def response_header_templates(self,data):
     response_headers = {
             "Content-Type":"text/html",
             "Content-Length": len(data),
             "Connection": 'close'
             }
     return response_headers

def make_response_header(self,url):
    """
        @param:request url
       서버가 보유한 파일이라면
        @return:HTTP/1.1 200 OK
       서버가 보유하지 않는 파일이라면
        @return:HTTP/1.1 404 Not Found'
    """

    if os.path.isfile(url):
        file_data = self.get_file(url)
        content_length = str(len(file_data))
        if file_data is not None:
            request_msg = self.FILE_OK+" "+str(self.response_header_templates(file_data))+" \r\n"+file_data
            print(request_msg)
        return request_msg
    else:
        return self.FILE_NO
```

   a. 인자 - url 반환값 request message
   b. 만약 url이 없다면 404메세지만 보내고 있다면 200메세지와, content/type, content-length를 가지고 있는 response header와 내용물을 모두 넣은 reponse 메세지를 반환한다.

ii. client

1. 생성자

```
def __init__(self):
    self.server_IP = '127.0.0.1'
    self.server_port = 2345
    self.request_url = sys.argv[1]
    self.http_method = sys.argv[2]
```

   a. 목적지 server IP와 포트를 설정하고, 원하는 파일인 request_url과 전송 메소드 방식을 인자로 받은것을 생성자로 한다.

2. tcp_connect

```
def tcp_connect(self):
    client_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    client_socket.connect((self.server_IP,self.server_port))

    print("Connet to Sever...")

    http_header = self.make_http_header(self.request_url,self.http_method)
    client_socket.send(http_header.encode())
    print("Send Message to Server : ",http_header)
    recv_data = client_socket.recv(1024)
    print(recv_data)
    self.download_file(client_socket,recv_data)
```

   a. server소켓과 연결하여 requestmessage를 보낸다.
   b. 보낸후 response 데이터를 받아
   c. 만약 404 가 떴따면 file 이 없다고 출력해주고 소켓을 닫아준다
   d. 만약 200이 떴다면 데이터를 받고, 그데이터를 다운로드(복사하여)저장한다.
   e. 인자값 - 없음 반환값 - 없음

3. parsing_Data

```
def parsing_data(self,data):
    split_data = data.split(b'\n')
    get_header = split_data[1].decode('utf-8')
    print(get_header)
    split_data = get_header.split(":")
    data = split_data[2]
    split_data = data.split(',')
    length = split_data[0].replace(' ','')
    return int(length)
```

   a. 인자값 - server로부터 받은 데이터 반환값 - Content 길이
   b. 함수 내용 - 초기 받으려는 데이터보다 원래 데이터가 더 있는지 확인하기 위하여 Content-length를 받는다.

4. download_file

```
def download_file(self,socket,data,length):
    data_piece =data
    file_name_split = self.request_url.split('.')
    file_name = file_name_split[0]+'_download.'+file_name_split[1]
    count = 0;
    with open(file_name,'wb') as f:
        print(length)
        try:
            while True:
                if len(data_piece) >= length:
                    break
                print(data)
                data = socket.recv(1024)
                data_piece += data

            print(8)
            receive_data = data_piece
            data_split = receive_data.split(b' \r\n')
            f.write(data_split[1])
            f.close()
        except:
            print("error while download the ",self.request_url)
```

   a. 인자값 - client Socket, 소켓으로부터 받은 데이터, 전체데이터 길이
   b. 반환값 - 없음
   c. 파일을 열고, 기존에 받았던 데이터 길이와 원래 데이터 길이가 같아질때까지 서버로부터 데이터를 계속 받는다
   d. 모든 데이터를 받으면 헤더부분을 제거하고 원래데이터만 가져가 파일을 생성하여 쓴다.

5. make_http_header

```
def make_http_header(self,url,method):

    http_template = method+' '+url+''' HTTP/1.1\r\nAccept:*/*\r\nAccept-Encoding: gzip,deflate\r\nAccept-Language: utf-8\r\n'''
    return http_template
```

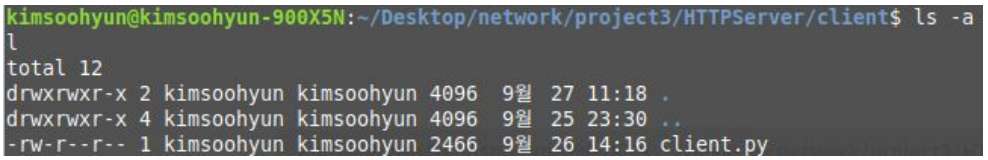
a. 인자값 : 받길원하는 Url, 데이터 전송방식 (GET,POST)를 가지고 데이터 헤더를 보낸다.

```
GET page.html HTTP/1.1
Accept:*/*
Accept-Encoding: gzip,deflate
Accept-Language: utf-8
```

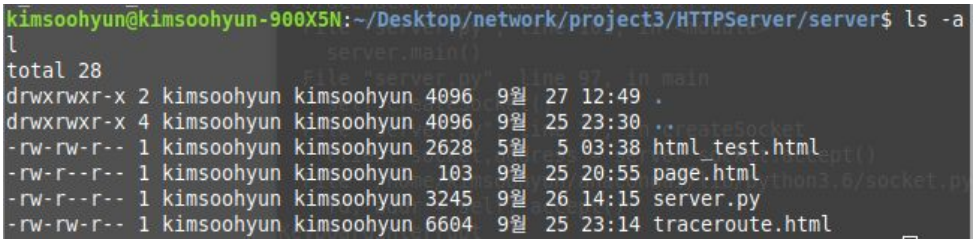
와 같이 보낸다.

c. 결과화면

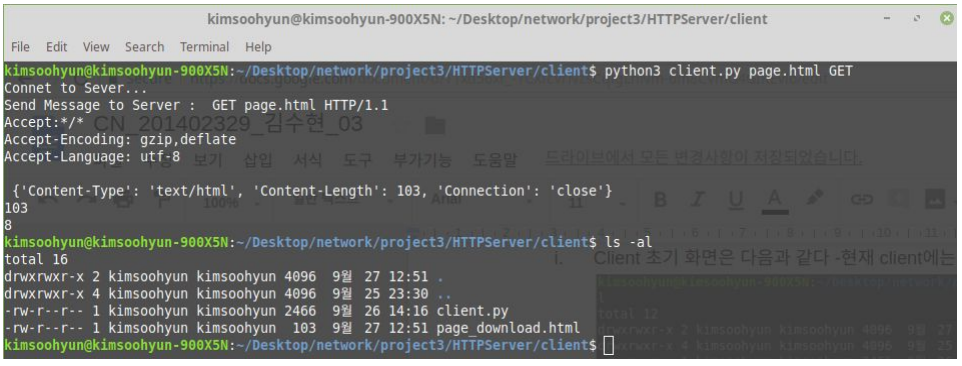
i. Client 초기 화면은 다음과 같다 -현재 client에는 아무것도 없다.



ii. server에 있는 파일과 화면은 다음과 같다 -



iii. server을 켜고 전송하는모습 - client를 보내면 다음과 같이 page가 download된것을 확인 할 수 있다



서버의 모습

```
kimsoohyun@kimsoohyun-900X5N:~/Desktop/network/project3/HTTPServer/server$ python3 server.py
server socekt open...
listening...
Connected with client
['GET', 'page.html', 'HTTP/1.1']
HTTP/1.1 200 OK
 {'Content-Type': 'text/html', 'Content-Length': 103, 'Connection': 'close'}
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

서버에 없는 파일인 경우 클라이언트 :

```
kimsoohyun@kimsoohyun-900X5N:~/Desktop/network/project3/HTTPServer/client$ python3 client.py hello.html GET
Connet to Sever...
Send Message to Server :  GET hello.html HTTP/1.1
Accept:*/*
Accept-Encoding: gzip,deflate
Accept-Language: utf-8

404 Not Found  ERROR : Not Found File: hello.html
```

서버에 없는 파일인 경우

web에서 화면 모습