

Assignment 1 - Simple Web Server and Client

Description

We prefer Python as it will be easiest for you, but Java or another language is also acceptable. In this assignment, you will write and execute 2 programs, a web server and a web client. For the specific requirements on the HTTP protocol, refer to RFC 2616 (<https://tools.ietf.org/html/rfc2616>). You do not need to support cookies or other forms of state, simply viewing static web pages on the web server. (Updated: <https://tools.ietf.org/html/rfc7230>)

1. Client

Implement a basic socket-based web client by connecting and sending HTTP protocol messages to a web server. There does not need to be any detailed GUI or user interface. The basic call for your program will be:

```
python web_client.py host:port/path [METHOD]
```

The default HTTP request **METHOD** is GET. In addition to GET, please implement HTTP request HEAD.

```
python web_client.py http://www.taylortjohnson.com/test_cs4283.html  
HEAD
```

The output of the program should include the HTTP status, headers and body according to the specifications.

Example:

```
python web_client.py http://www.taylortjohnson.com:80/test_cs4283.html
```

```
HTTP/1.1 200 OK
```

```
Date: Thu, 01 Feb 2018 01:56:47 GMT
```

```
Server: Apache/2.2.34 (Amazon)
```

```
Last-Modified: Tue, 16 Jan 2018 14:07:35 GMT
```

```
ETag: "4019d-13-562e540a6e40d"
```

```
Accept-Ranges: bytes
```

```
Content-Length: 19
```

```
Connection: close
```

```
Content-Type: text/html; charset=UTF-8
```

```
Vanderbilt CS 4283
```

The client program is expected to work with public websites.

2. Server

Implement a basic socket-based web server by listening on a specified port, such as port 80. The basic call for the web server will be:

```
python web_server.py PORT DIRECTORY
```

With the default HTTP port 80 and a directory of files to make available for viewing at the (*nix) directory /www, this is:

```
python web_server.py 80 /www
```

Due to differences in operating systems, programs installed, etc., listening on port 80 may be problematic. For this reason, for testing purposes, start your server on port 8080, or another port that you specify:

```
python web_server.py 8080 /www
```

Note that your web server should continue executing until terminated by the user, i.e., it needs to have an infinite loop where it is continually accepting messages if any data is available. It should make available all files in the specified directory, and return response messages with appropriate HTTP error codes (e.g., 404) if a file does not exist.

Requirements

- The server should support **GET** and **HEAD** methods.
- The server should support 200, 404, and 501 status codes.
 - For 404, return a generic not found body.
- The server should return the following header fields. (minimum)
 - Date
 - Server (use your name)
 - Content-Length (where appropriate)
- The server returns a default file (index.html) when a path is not requested (the root of server is requested).
- The server and client should support large files.

Example

While `python web_server.py 8080 /www` is running

```
python web_client.py localhost:8080/test_cs4283.html
```

returns:

```
HTTP/1.1 200 OK
Date: Thu, 01 Feb 2018 02:32:05 GMT
Server: cs4283-server
Content-Length: 19
Connection: close
```

Content-Type: text/html; charset=UTF-8

Vanderbilt CS 4283

Notes

If you use another programming language other than Python, please provide a Makefile and have the same order of arguments for calls.