Server & Client, Python

A Python socket-programmed web-server and client.

Overview

This project implements a web server and client in Python, with the express purpose of sending static HTML files and receiving connections from incoming users.

The lab utilizes lower level socket libraries for the main purposes of creating TCP connections, and sending and receiving messages through said TCP connections. Despite this, the major amount of work is done in the application layer, through parsing and constructing HTTP request packets and response packets.

How To Run

All source code is contained within the `src` file. To run either the server or client, first open a command line into the main directory, then run either:

```
# Server
python src/server.py
# Client
python src/client.py
```

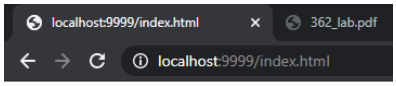
Server Functionality

The server is created using the socket library, just as the client is. Unlike the client, however, the server runs constantly, accepting requests of maximum 1024 bytes and responding with two conditions.

If the request is a proper HTTP GET request to the file `index.html`, the server will respond by loading and sending the `index.html` file to the connection. An example of this functionality (from the console view) is shown below:

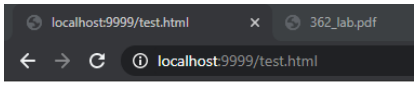
```
PS F:\Code\school_projects\SnC-Python> python src/server.py
Server is online and awaiting requests.
Connected to ('127.0.0.1', 55404)
GET /index.html HTTP/1.1
Host: localhost:9999
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:81.0) Gecko/20100101 Firefox/81.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: username-localhost-8888="2|1:0|10:1601404512|23:username-localhost-8888|44:Y2Y5NDlkNGZmNmYxNDczZThmN2Q3ZGRiYThlZTFhOWY=|7445bdb7bbbdc2bf8cd030268f28a6da44f3344b474402b64b5dea036170b34b"; _xsrf=2|21573589|82485a3609dccdbe81b39cbd707b0acd|1601404512
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0


Client request: /index.html
```

As shown, upon a request, the HTTP request is printed to the console, as well as the specific requested filepath below. The HTML file is also properly sent (in this case, it is sent to the browser as shown in the following picture.)



This is the index page.

Upon requesting any other page (for instance, `/` or `/test.html`) an error will be returned (specifically, the `404` error, denoting a missing or nonexistant file on the server.) The logging for these events is the same on the console client; However, the browser shows the proper page load.



Error 404: File Not Found

In this case, the `filenotfound.html` file is sent to the client with an HTML error code in response.

It is important to note that in order for the server to work across computers and across the internet, I had to perform several settings and variations on the provided code. Firstly, I got an error when trying to connect to port 8080 - this is because 8080 is a reserved port for other applications and development servers on my computer, and I had to open that in my firewall.

In order to allow access from other computers, I had to forward the port to my machine while my server was running in my router to get other computers to connect using my public domain as well.

Client Functionality

The client application is a simple console application that only runs once and terminates execution immediately after. It asks the user for a connection host, port, and filepath to load. After doing so, it creates a connection, requests information from that connection, and prints out the response (including status code and body.)

Two examples of the client interacting with my server are shown below:

```
PS F:\Code\school_projects\SnC-Python> python src/server.py
Server is online and awaiting requests.
Connected to ('127.0.0.1', 54934)
GET /index.html HTTP/1.1
Host: localhost:9999


Client request: /index.html
```

```
PS F:\Code\school_projects\SnC-Python> python src/client.py
Enter the host URI: localhost
Please enter the port: 9999
Please enter the file you want to request from the server: index.html
HTTP/1.1 200 OK
Content-Type: text/html

<html>
    <body>
        <h1>This is the index page.</h1>
    </body>
</html>
```

This is the client interacting with the server and requesting the index HTML file.

```
PS F:\Code\school_projects\SnC-Python> python src/server.py
Server is online and awaiting requests.
Connected to ('127.0.0.1', 54934)
GET /index.html HTTP/1.1
Host: localhost:9999


Client request: /index.html
```

```
PS F:\Code\school_projects\SnC-Python> python src/client.py
Enter the host URI: localhost
Please enter the port: 9999
Please enter the file you want to request from the server: index.html
HTTP/1.1 200 OK
Content-Type: text/html

<html>
    <body>
        <h1>This is the index page.</h1>
    </body>
</html>
```

This is the client interacting with the server and requesting some other HTML file.

In the above pictures, the client is shown interacting with the server through a locally hosted application. However, the client is capable of connecting to and communicating with online resources as well. An example of the client connecting to and requesting the main index page of `www.google.com` on the HTTP port is shown below.

```
PS F:\Code\school_projects\SnC-Python> python src/client.py
Enter the host URI: www.google.com
Please enter the port: 80
Please enter the file you want to request from the server: /
HTTP/1.1 200 OK
Date: Mon, 19 Oct 2020 21:04:40 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2020-10-19-21; expires=Wed, 18-Nov-2020 21:04:40 GMT; path=/; domain=.google.com; Secure
Set-Cookie: NID=204=i-YgnZhEDAyj3BOI_Sb0kFxCthLYsLbWeyFl2sTqSjV0o6J2nxID87hs5Njnp51zcN1CtWYZPNkjM9oHJVDWMyCVEM5FK3dlHWHj
R2j5X121TuZWN8QgktnuHZ90OPeh-ovu4UMSc6ncwzhlDObtOwGx2XvqRbsD80lpSr_3cP8; expires=Tue, 20-Apr-2021 21:04:40 GMT; path=/;
domain=.google.com; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding
Transfer-Encoding: chunked

49e7
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en"><head><meta content="Search the world's
 information, including webpages, images, videos and more. Google has many special features to help you find exactly wha
t you're looking for." name="description"><met
```

As shown, the entire HTTP response is printed out (including the `200 OK` status at the top, notifying our client that we have successfully made a proper request.) The body is shown at the bottom, separated by some spacing; This is an HTML response, which when loaded in, contains Google's main metadata for their site. Since the response from the server is limited to 1024 bytes, this body is truncated.