Computer Network Project 2

Due date

7/Nov/2016 23:55:00
(delay - -15pts per day)

Goal

- understanding basic knowledge how the webserver works (socket, tcp server and http...)
- implementing a simple webserver handling http requests

OS & Language

- Linux Ubuntu 14.04.4 LTS or CentOS7(>=7.2)
- C (gcc 4.8.5) or Python2,3 (>=2.7.5 or >=3.5.2)

Restraints

- Only internal modules(or libraries) except for below are allowed
  - **[python] SimpleHTTPServer, BaseHTTPServer and ScoketServer modules are not allowed**
- It's allowed to refer to any examples on internet but you must write your own codes
  - The plagiarism detection program will be used to grade

Basic Requirement (100pts)

1. Building WebServer

- Bind the socket so that your program listen on a specific port and configure a root directory, in which html files are located
- Port number and directory should be set when you run the program like below
  - ./run.sh [port number] [directory path]
  - ex)

```
sh-3.2# python project_2.py 80 /var/www
Listening on port 80 ...
```

2. Handling HTTP Requests

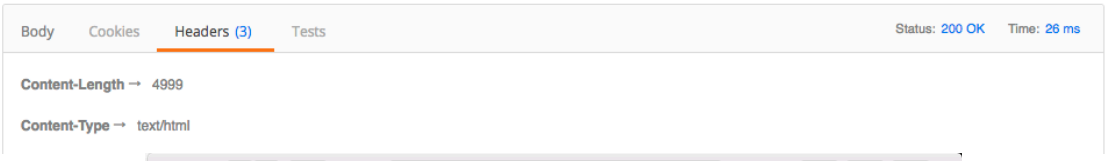
- Read request data whenever clients make requests to http://[ip]:[port number]
- Parse request data so that you can serve a web page like below
  (Assume that /var/www/ is root directory)
  - Files corresponding to request path should be served
    - Ex) GET /img/building-1.jpg -> /var/www/img/building-1.jpg
  - Request path below should be differently translated
    - GET / -> /var/www/index.html

3. Generating Responses

- Respond with status code (200) and file corresponding to request path
    - Header should contain the below
        - Content-Type
            - Application/javascript
            - text/html
            - text/css
            - image/jpeg
            - image/png
        - Content-Length

    - Web pages will not be shown properly if you do not build header corresponding a file and you can check response header in Postman (Chrome extension)

ex)



- Respond with status code (404) and 404 script when a client requests inappropriate path
    - ex) GET /nodir/nofile.html-> 404 status and 404 script
    - 404 script
        - [Request Path] should be path the client requests
        - <html><head><title>404 Not Found</title></head><body>
          <h1>404 Not Found</h1>
          The requested URL [Request Path] was not found on this server.
          </body></html>

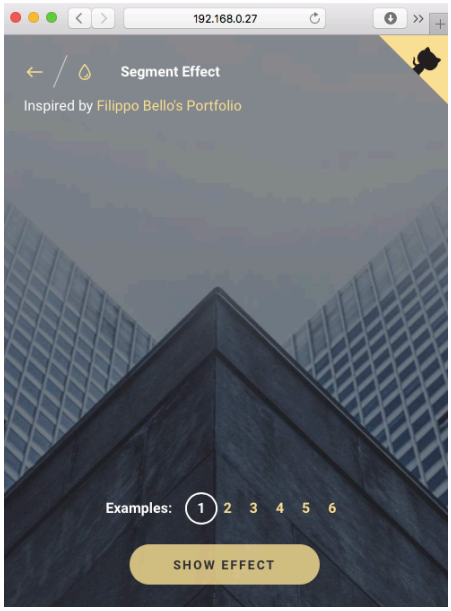
Additional assignment (50pts)

1. Redirection (5pts)

- /go/[name] should be redirected to http://www.[name].com
    - ex) GET /go/facebook -> http://www.facebook.com

2. User-Agent (10pts)

- For mobile user and pc user, different pages should be served
    - Files in /mobile folder should be served for mobile user
        - Ex) GET / -> /mobile/index.html



(Chrome/Safari browser supports mobile User-Agent)

3. Cookie (35pts)

- Files in /secret directory should be served only for authenticated clients
    - POST /login request with ?id=yonsei&pw=network
        - Generate a custom cookie

            - Cookie should be your student number
            - Ex) Cookie = 2009147006
        - Redirect to /secret/index.html
        - If id and pw are not correct, forbidden page should be served
    - The client should be able to access files in /secret directory with the cookie
    - Status code (403) and 403 script should be served when the client access to the files in /secret directory without the cookie or try to login with incorrect id and pw
        - [Request Path] should be path the client requests
        - <html><head><title>Access Forbidden</title></head><body>
          <h1>403 Forbidden</h1>
          You don't have permission to access the requested URL /nopath/nofile. There is either no index document or the directory is read-protected.
          </body></html>

Submission

- [c|u]_studentNumber_2.zip
    - Specify OS you tested
        - c = CentOS
        - u = Ubuntu
    - readme.txt
    - project_2.[py|c]
    - run.sh
    - setup.sh
    - report.pdf
        - Introduction
        - Specify what you have implemented
        - Code commentary block by block
        - How your code works
        - Screenshot with your explanation
        - What you have learned by this project
        - Reference