Computer Network Project 2 **Simple Webserver**

CSI4106-01

Fall, 2017

(Difficulty ★★★☆☆)

Prelim.

Before you do this project, you must be fully aware of **"Project Policy Notice"**

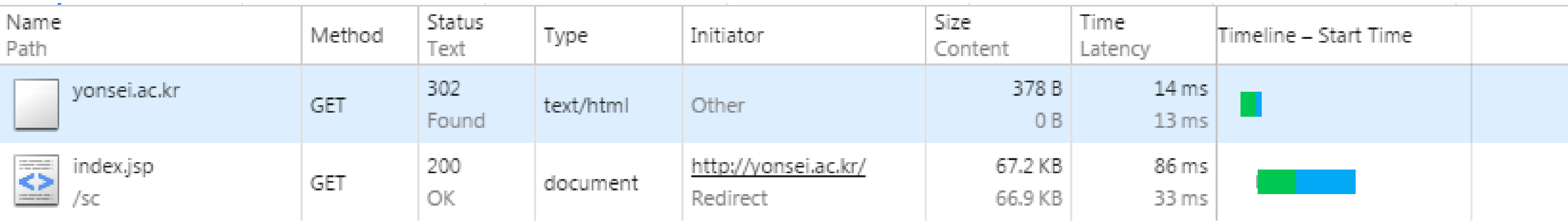
A typical HTTP Communication

- When you make a request: www.yonsei.ac.kr

1. The client asks IP address to DNS server
   - (e.g.) 168.126.63.1 (kns.kornet.net)
2. The client gets 165.132.13.38 as the IP address
3. **The client asks a webpage to 165.132.13.38:80**
4. **The server returns a HTML file as response**
5. **The client asks additional files for rendering the HTML file.**

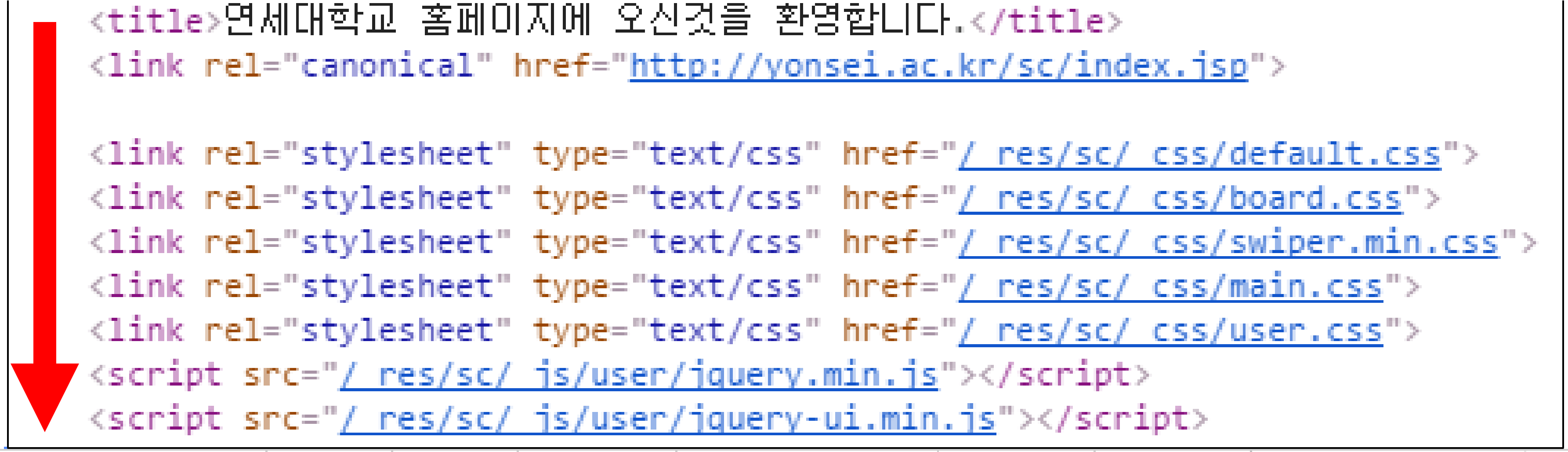
How CLIENT works (example)

- Chrome ➔ Developer Tools (F12) ➔ Network Tab
  - It shows how Chrome fetches yonsei.ac.kr webpage



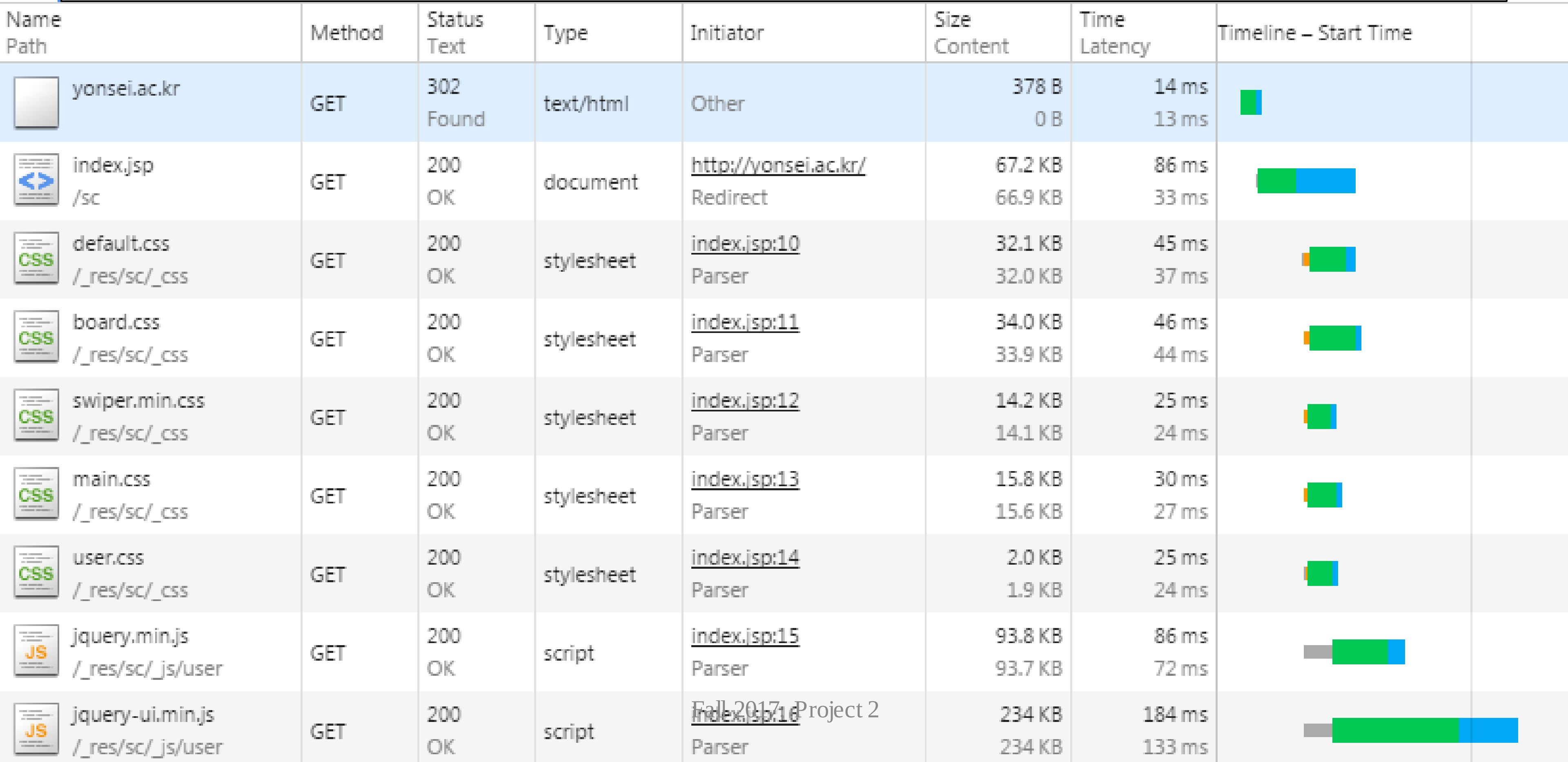
Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline – Start Time
yonsei.ac.kr	GET	302 Found	text/html	Other	378 B 0 B	14 ms 13 ms	
index.jsp /sc	GET	200 OK	document	http://yonsei.ac.kr/ Redirect	67.2 KB 66.9 KB	86 ms 33 ms	

1. Send a request to “yonsei.ac.kr”
2. Got **HTTP/1.1 302 Found** (Temporarily Moved)
3. Make a redirect to “yonsei.ac.kr/sc/index.jsp”
4. Fetch the HTML page
5. **Render the page with requests of CSS/JS/JPG…**



```
<title>연세대학교 홈페이지에 오신것을 환영합니다.</title>
<link rel="canonical" href="http://yonsei.ac.kr/sc/index.jsp">

<link rel="stylesheet" type="text/css" href="/_res/sc/_css/default.css">
<link rel="stylesheet" type="text/css" href="/_res/sc/_css/board.css">
<link rel="stylesheet" type="text/css" href="/_res/sc/_css/swiper.min.css">
<link rel="stylesheet" type="text/css" href="/_res/sc/_css/main.css">
<link rel="stylesheet" type="text/css" href="/_res/sc/_css/user.css">
<script src="/_res/sc/_js/user/jquery.min.js"></script>
<script src="/_res/sc/_js/user/jquery-ui.min.js"></script>
```



Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline – Start Time
yonsei.ac.kr	GET	302 Found	text/html	Other	378 B 0 B	14 ms 13 ms	
index.jsp /sc	GET	200 OK	document	http://yonsei.ac.kr/ Redirect	67.2 KB 66.9 KB	86 ms 33 ms	
default.css /_res/sc/_css	GET	200 OK	stylesheet	index.jsp:10 Parser	32.1 KB 32.0 KB	45 ms 37 ms	
board.css /_res/sc/_css	GET	200 OK	stylesheet	index.jsp:11 Parser	34.0 KB 33.9 KB	46 ms 44 ms	
swiper.min.css /_res/sc/_css	GET	200 OK	stylesheet	index.jsp:12 Parser	14.2 KB 14.1 KB	25 ms 24 ms	
main.css /_res/sc/_css	GET	200 OK	stylesheet	index.jsp:13 Parser	15.8 KB 15.6 KB	30 ms 27 ms	
user.css /_res/sc/_css	GET	200 OK	stylesheet	index.jsp:14 Parser	2.0 KB 1.9 KB	25 ms 24 ms	
jquery.min.js /_res/sc/_js/user	GET	200 OK	script	index.jsp:15 Parser	93.8 KB 93.7 KB	86 ms 72 ms	
jquery-ui.min.js /_res/sc/_js/user	GET	200 OK	script	index.jsp:16 Parser	234 KB 234 KB	184 ms 133 ms	

How WEBSERVER works

1. Listen to port 80 via TCP (or a specific port)
2. Read and parse a HTTP request
   - (e.g.) `/index.html` with Mobile User-agent
3. Write a packet of HTTP response
   - Its body has data of `/index.html`

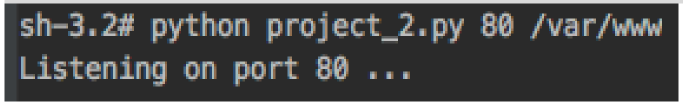
- *You may know about…*
  - *socket(), listen(), connect(), bind(), accept()…*
  - *TCP Socket, Stream…*
  - *Socket Initialize, Open and Close…*

Mandatory Assignment (100pts)

- Write a code of simple http webserver **to serve** webpages including html/css/js/jpg/png files
- Implement Required Functions (1)~(3)

- **Follow the usage format below**

```
(Format)   ./run.sh port rootDir
(Example) ./run.sh 8080 /var/www
http://my_ip_address:port/ ➔ http://10.0.0.1:8080/
** It serves /var/www/index.html as default.
We provide sample homepage files for test.
>> Your webserver should work with those files.
```



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

Required Functions (1) Generating Response

- When you generate a HTTP response
- In the response header, you should include
  - Content-Length
  - Content-Type (follow only 5 types below)

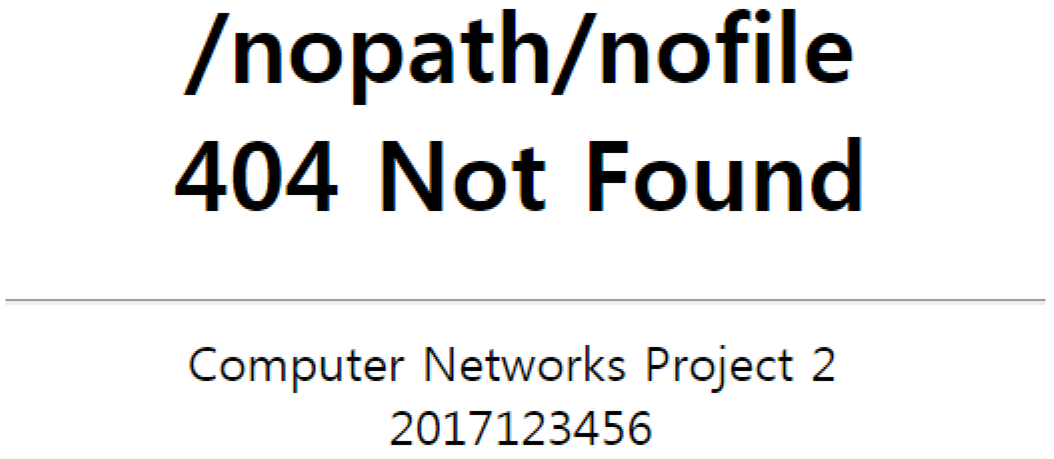
MIME-type	description	extension
text/html	HTML file	html
text/css	Cascading style sheet	css
text/javascript	Javascript source file	js
image/jpeg	JPEG image file	jpg
image/png	png image file	png

Required Functions
(2) Path Translation

- Format of your Webserver Address must be **http://IP:port/** (e.g. http://165.132.0.1:8080)
- Set physical root directory: /var/www
- Set the default index page: index.html
  - http://IP:port/ → GET / → GET /index.html
- Basic path processing
  - http://IP:port/css/abc.css → GET /css/abc.css → It actually serves /var/www/css/abc.css

Required Functions (3) Error Exception

- Hint: **HTTP/1.1 404 Not Found**
- (e.g.) GET **/nopath/nofile**
- **Follow the below format (including your ID!!!)**

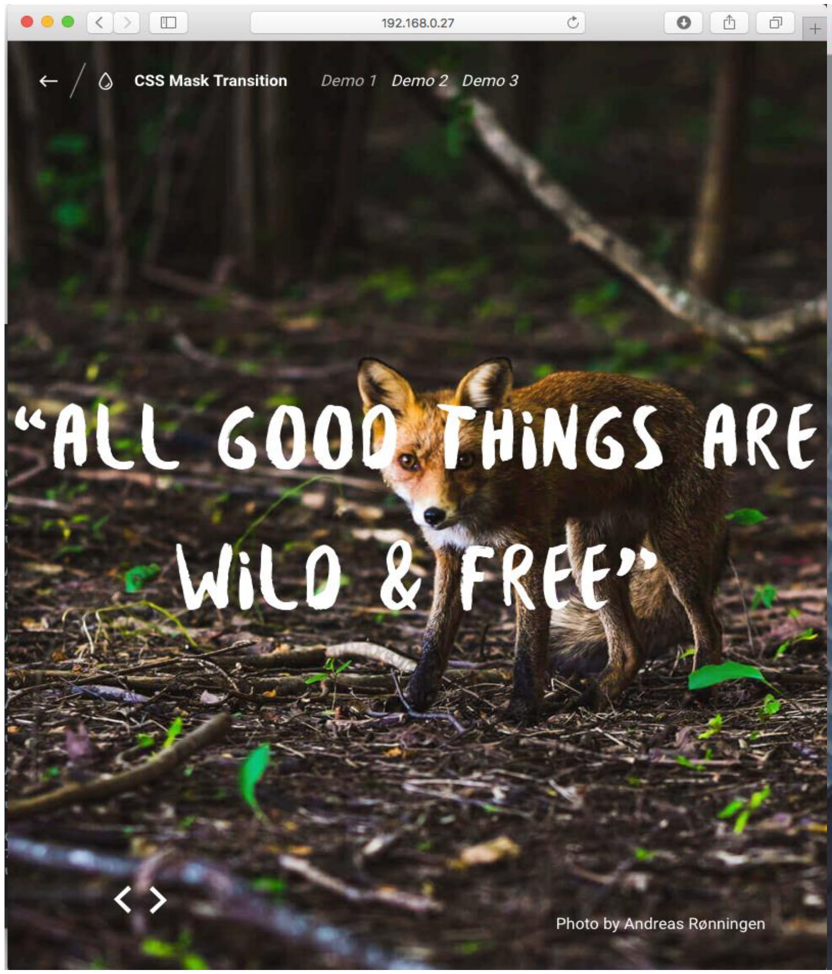


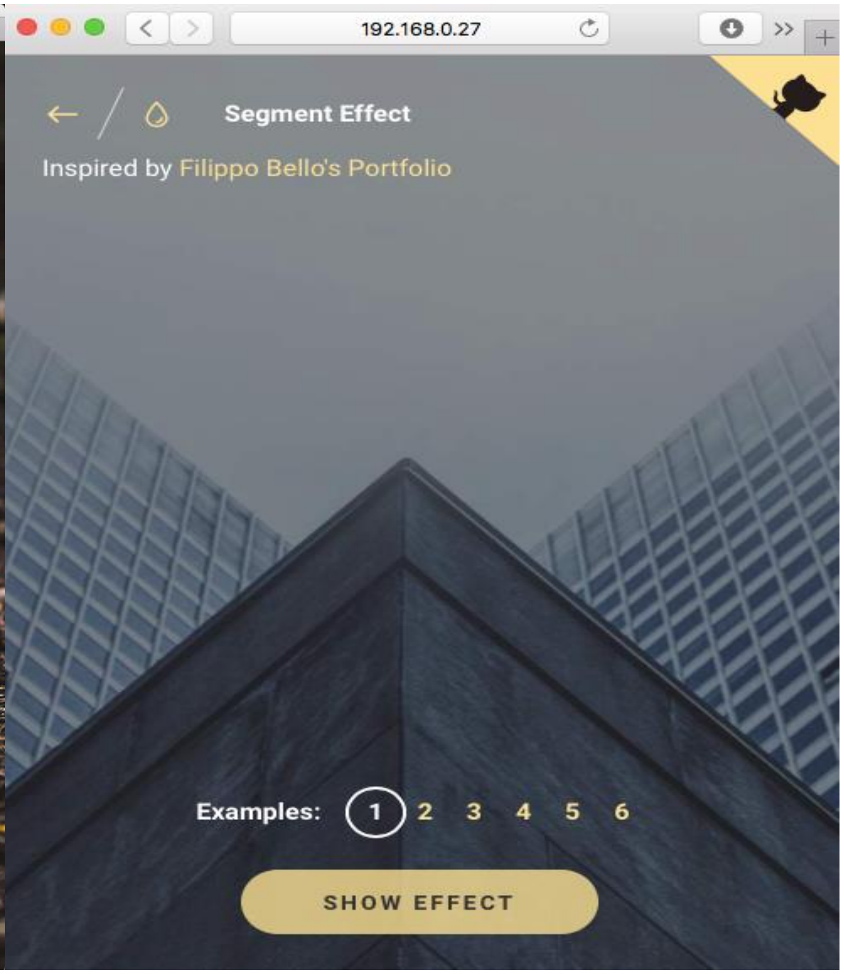
/nopath/nofile

404 Not Found

Computer Networks Project 2
2017123456

Additional Assignment (1) +15pts





Desktop User Fall 2017, Project 2 **Mobile User**

Additional Assignment (1) with **`website.zip`** (we provide)

- **Goal: Request Parsing of User-agent**
- For desktop users
  - Your homepage is located in “root” folder
- For mobile users
  - Your homepage is located in “mobile” folder.

>>> This is not just a simple Redirection!!

- When a mobile user accesses your server
  - **(X) GET / ➔ GET /mobile/index.html**
  - **(O) GET / ➔ GET /index.html**
  - **Fetch files from mobile folder**

How to use **`website.zip`**

- You may extract files onto server’s root folder for your Additional Assignment 1
- The **`website.zip`** file includes HTML, CSS, JS, JPG image files.
  - / → Desktop version files
  - /mobile → Mobile version files

- TA will test your webserver with these files!

Additional Assignment (1)

- Hint: **`User-agent`**
- Desktop
  - Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36
- Nexus 4
  - Mozilla/5.0 (Linux; Android 4.4.2; Nexus 4 Build/KOT49H) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1847.114 Mobile Safari/537.36
- iPhone 6
  - Mozilla/5.0 (iPhone; CPU iPhone OS 6_0 like Mac OS X) AppleWebKit/536.26 (KHTML, like Gecko) Version/6.0 Mobile/10A5376e Safari/8536.25

Additional Assignment (2) **+20pts**

- Goal: Mimic CGI Processing! (Naïve)
- Target Language : Python (*.py)
- Example for "**`GET /plus.py?5,7`**"
  - (X) Show the whole codes as text file
  - (O) Show the code results → 12
- It actually executes **`py plus.py 5 7`**

```
...
Def F(a,b)
    print (a+b)
...
```

Additional Assignment (2)

- If you can do C language...?
- Target Language : C (*.c ➔ *.o)
- Example for "**`GET /plus.c?5,7`**"
  - (X) Show the whole codes as text file
  - (O) Show the code results ➔ 12
- It actually compiles **`plus.c`** to **`plus.o`** and executes **`./plus.o 5 7`**

```
...
void function F(int a, int b){
   printf("%d", a+b);
}
...
```

Additional Assignment (2)

- You should build the codes for **either C or Python** language for this AA.
- Hints
  - CGI on typical webservers
  - Command Line Argument / Execution of Shell Command
  - Standard Streams (stderr, stdin, stdout)
- TAs does not provide any codes to test your webserver for this AA ➔ Make your own.
- TAs may test different CGI programs for the evaluation

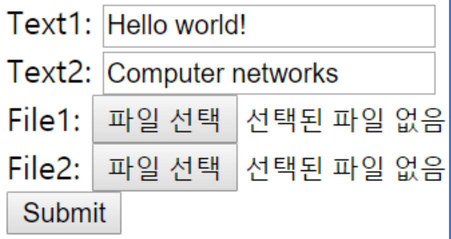
Additional Assignment (3) **+35pts**

- Goal: Implement POST Text and File Upload
- Hints
  - Analyze how POST upload works
  - ~~*application/x-www-form-urlencoded*~~
  - **multipart/form-data**
  - ***aa3-result.html*** **is not an actual HTML file**
- Directions
  - Uploaded files should be stored in root folder
  - The files should be retrieved via any web-browser.
  - Build your own two pages: "aa3" and "aa3-result"

Additional Assignment (3)

Your webserver should have the page of http://my-ip/aa3.html
Note that TAs may modify this page for the evaluation

```
…
<form action="http://my-ip/aa3-result.html" method="post"
enctype="multipart/form-data">
  Text1: <input type="text" name="text1" value="Hello World!"><br>
  Text2: <input type="text" name="text2" value="Computer Networks!"><br>
  File1: <input type="file" name="file1"><br>
  File2: <input type="file" name="file2"><br>
  <button type="submit">Submit</button>
</form>
…
```


Text1: Hello world!
Text2: Computer networks
File1: 파일 선택 선택된 파일 없음
File2: 파일 선택 선택된 파일 없음

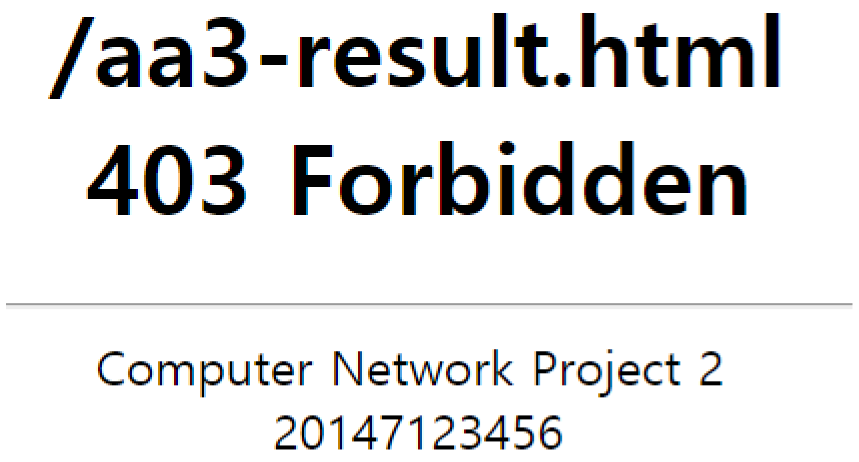
Also please implement /aa3-result.html as the result page of /aa3.html
Please print the results like below. The link should work (download)

[text] %name% : %value%
[file] %name% : %filename.ext% (%size in KB%) [Download: %download link%]

```
…
text1 : Hello World!
text2 : Computer Networks!
File1 : abc.zip (1529KB) [Download: http://my-ip/abc.zip]
File2 : image.jpg (32KB) [Download: http://my-ip/image.jpg]
…
```

Additional Assignment (3)

- If someone accesses **/aa3-result.html** directly (i.e., its referrer is not **/aa3.html**), your server should present **403 Forbidden Error**.


/aa3-result.html

403 Forbidden

Computer Network Project 2
20147123456

Directions

- **This is an individual project**
- Language: **C or Python**
  - **C: gcc (>=4.8.5)**
  - **Python: Python 2 (>=2.7.5) or Python 3 (>=3.5.2)**
- OS: **CentOS 7 or Ubuntu 14.04** or higher
- You must use only ***internal*** libraries.
  - for Python : SimpleHTTPServer, BaseHTTPServer, SocketServer → ***NOT ALLOWED***
  - ***Any 3rd party framework: NOT ALLOWED***

Deliverables:
OS[c|u]_YourID_ProjectNo.zip
(e.g., `u_2017147123_2.zip`)

>>>>> Do not include any folders in the zip file

TA tests with `./setup.sh && ./run.sh`

- **project.py** or **project.c**
  - Your code with detail comments
- **run.sh**
  - This runs your webserver in **background**.
- **setup.sh**
  - This should install dependencies or compile your code
- **report.pdf**
  - Your comprehensive comments of this project

Helpful Keywords

- TCP (socket programming)
- HTTP Packet (request and response)
- Chrome – Developer Tools (F12 key)
- Basic idea of HTML/CSS/JS
- Python Basics / Command Line Arguments
- MIME types (Content-types)
- Error Codes (403, 404, 500, 502, 503)
- User-agent switcher for chrome extension

Implementation Scope

- **The goal is not to write a complete and perfect webserver.**
- STATIC files only (no CGI/PHP script)
  - ***except for AA2 and AA3***
- HTTP 1.1 only (not HTTPS)
- Do not care Consistent Connection
  - You just need to use Content-length only
- Do not care DNS / Domain-relevant issues
- Do not care Performance issues

- **DUE DATE**

  31/Oct/2017 23:55:00 KST

  No exception for exceeding deadline

- **Delay Policy**

  -33%pts for ~1/Nov 23:55:00

  -66%pts for ~2/Nov 23:55:00

  -100%pts for 2/Nov 23:55:00~

You agree with the following statement by submitting your assignment on YSCEC

Ctrl+C and Ctrl+V is not Code Referencing,

Plagiarizing = 0pts = Fail

No exception for any kinds of cheating and copying

Score Policy: *Max. 100+70 pts*

1	Not submitted / not working / missing files	**0 pts**
2	Overdue ➔ Delay	**-33% pts/day**
3	Your webserver malfunctions	-20 pts/function
4	Additional assignment is implemented	+15/+20/+35 pts
5	**The rules or directions are not followed**	-10 pts/rule
6	**Any 3rd party framework is used**	**0 pts**
7	**Plagiarizing / Over-implementation (Any kinds of Suspicion of Code-copy)**	**0 pts**
8	**Impolite Report / Lack of Comments**	**0 pts / -50% pts**

Questions are welcome on YSCEC but,

“Try Google first”

“Look up others’ questions”