Yeseul An
CSS 432
Assignment 2

Simplified HTTP Retriever and Server

The assignment is to implement a retriever and a server program that can practice a simplified version of HTTP. The retriever program and server program work conjunction with any web browser and the server. The retriever program or the browser create GET requests for the objects. The server program deals with GET request and returns 200 OK code if it is successful. Otherwise, the server program should return 404 Not Found code for not found objects. There are three scenarios should be handled from the program.

1) Real web browser accessing your server
2) Your retriever accessing a real server
3) Your retriever accessing a file from your server

Screenshots are attached with the three scenarios above.

Files

432_Assignment2_YeseulAn.zip contains files below:

Server.cpp	Code file of server program
Retriever.cpp	Code file of retriever program
server.sh:	Bash script that runs the server program
retriever_to_browser.sh	Bash script that runs retriever when retrieves a real browser
retriever_to_server.sh	Bash script that runs the retriever when retrieves the server program
README	This file that contains instructions and screenshots
output.txt	output file of the html in text file
index.html	customized html file that is used for retrieval
notfound.html	customized html file for Not Found Page
myjavascript.js	customized java script file that is used for retrieval
cute_dog.jpg	jpg image file that is used for retrieval
screenshot	folder that contains screenshots

How to Compile

How to compile the code:

1. Download and extract the [432_Assignment2_YeseulAn.zip] folder.

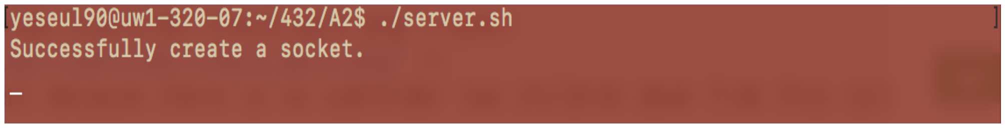
2. SSH to the one of the linux machine in the school lab.
3. Optionally, do “chmod a+x [bashscript.sh]” if needed.

Case 1) Real Web Browser Accessing My Server

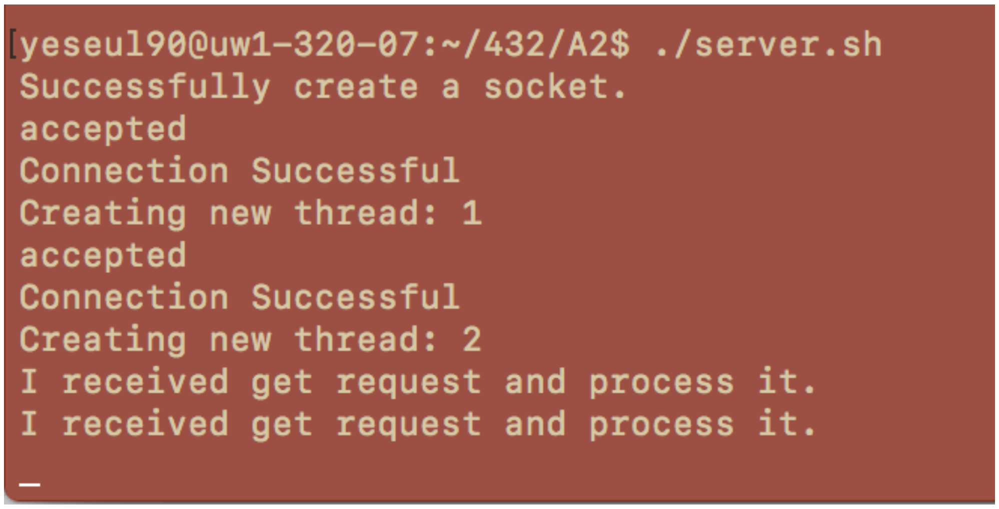
1. Type ifconfig to find out the IP address of the machine. Ex) uw1-320-07 has an IP address of (`172.21.198.87`).
2. Type ./server.sh

Or alternatively,

1. Type `g++ Server.cpp -pthread -o server -std=c++11`
2. Type `./server`
3. Go to the browser, type http://172.21.198.87:2681/index.html

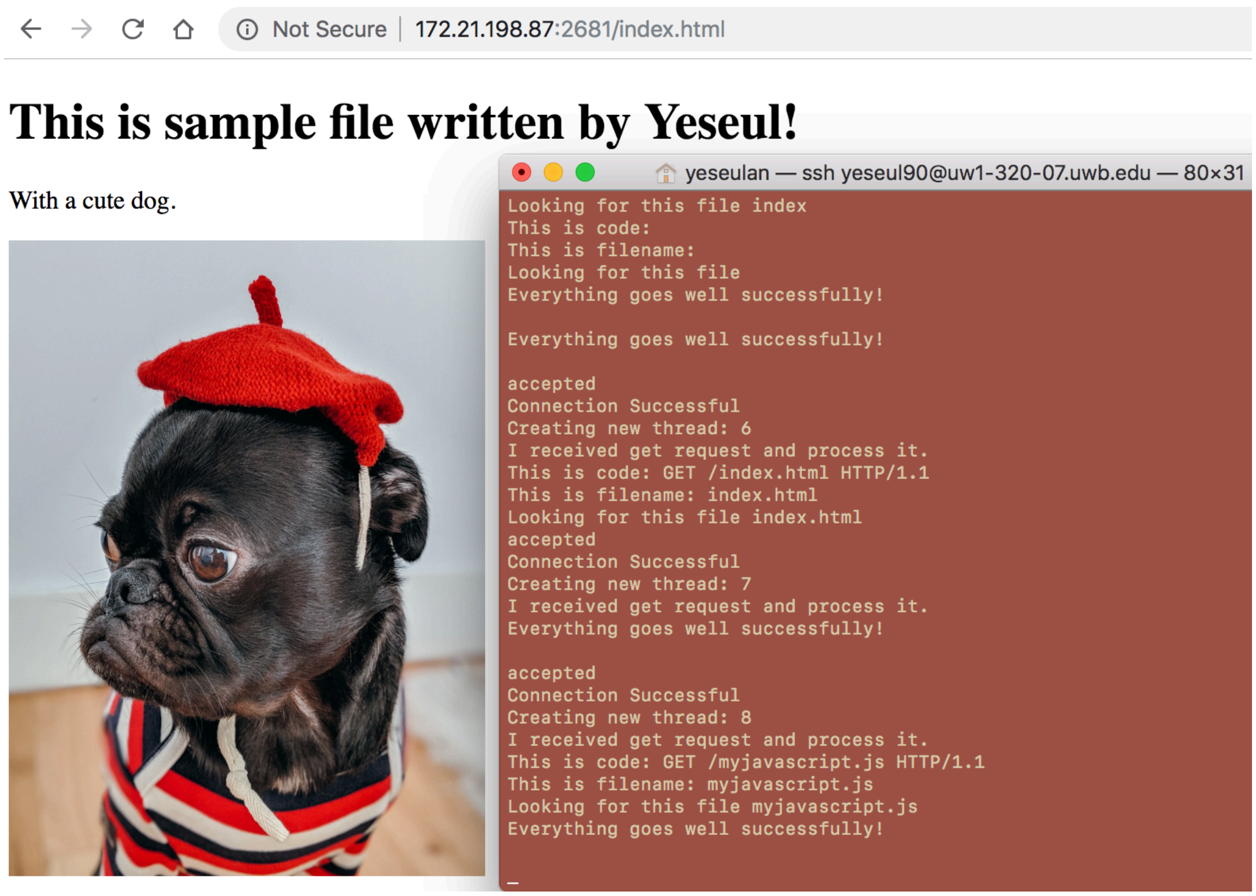


```
[yeseul90@uw1-320-07:~/432/A2$ ./server.sh
 Successfully create a socket.

_
```



```
[yeseul90@uw1-320-07:~/432/A2$ ./server.sh
 Successfully create a socket.
 accepted
 Connection Successful
 Creating new thread: 1
 accepted
 Connection Successful
 Creating new thread: 2
 I received get request and process it.
 I received get request and process it.

 _
```

Above screen shots are shown when the server program run and print the status statements to the console.


Not Secure
172.21.198.87:2681/index.html
yeseulan — ssh yeseul90@uw1-320-07.uwb.edu — 80×31

This is sample file written by Yeseul!

With a cute dog.

```
Looking for this file index
This is code:
This is filename:
Looking for this file
Everything goes well successfully!

Everything goes well successfully!

accepted
Connection Successful
Creating new thread: 6
I received get request and process it.
This is code: GET /index.html HTTP/1.1
This is filename: index.html
Looking for this file index.html
accepted
Connection Successful
Creating new thread: 7
I received get request and process it.
Everything goes well successfully!

accepted
Connection Successful
Creating new thread: 8
I received get request and process it.
This is code: GET /myjavascript.js HTTP/1.1
This is filename: myjavascript.js
Looking for this file myjavascript.js
Everything goes well successfully!

_
```

It will successfully loads the index.html page with the <script> object and <img> object. If it takes long time loading, try re-type and re-hit(http://172.21.198.87:2681/index.html) so that it will load the page.

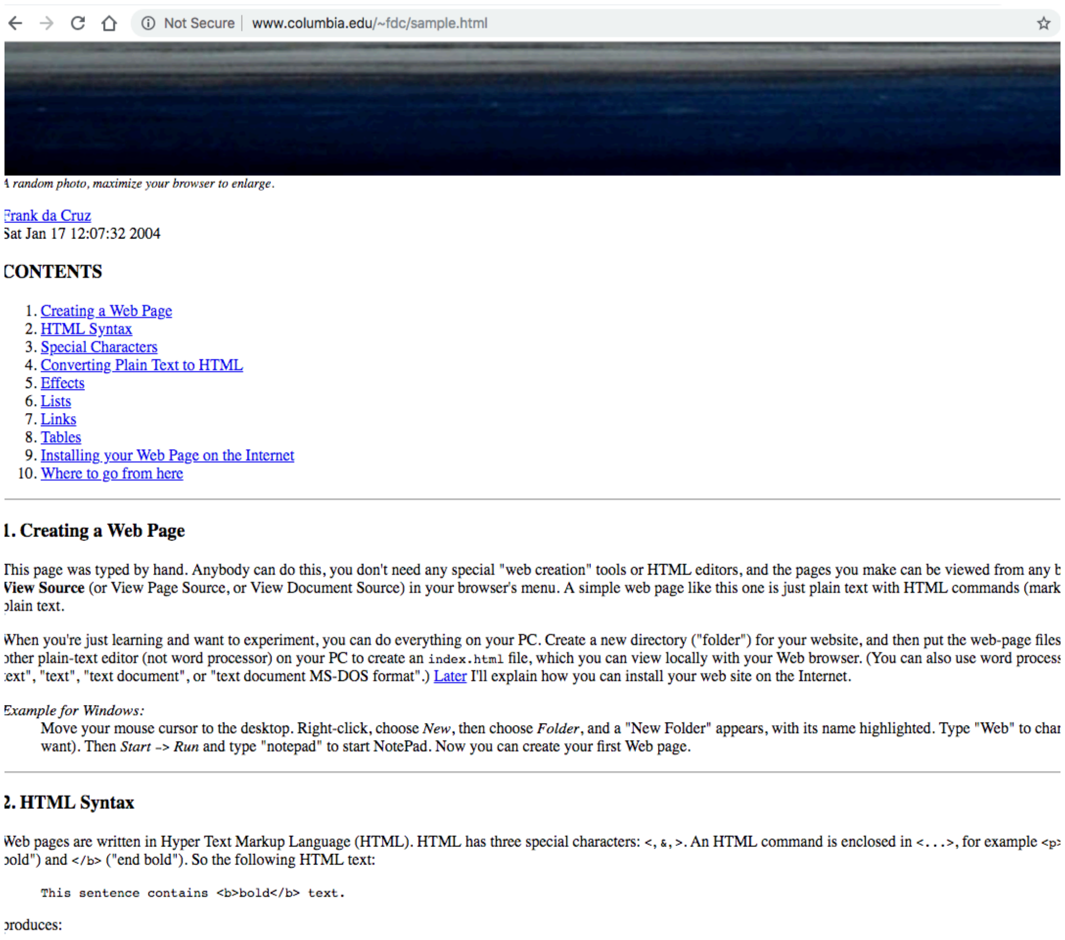
Case 2) Your Retriever Accessing a Real Server

1. Type ./retriever_to_browser.sh

   Or alternatively,

1. Type g++ Retriever.cpp -pthread -o retriever -std=c++11
2. Type ./retriever www.columbia.edu /~fdc/sample.html 80

3. You will see the results similar to below.



Not Secure | www.columbia.edu/~fdc/sample.html

A random photo, maximize your browser to enlarge.

Frank da Cruz
Sat Jan 17 12:07:32 2004

CONTENTS

1. Creating a Web Page
2. HTML Syntax
3. Special Characters
4. Converting Plain Text to HTML
5. Effects
6. Lists
7. Links
8. Tables
9. Installing your Web Page on the Internet
10. Where to go from here

1. Creating a Web Page

This page was typed by hand. Anybody can do this, you don't need any special "web creation" tools or HTML editors, and the pages you make can be viewed from any b
View Source (or View Page Source, or View Document Source) in your browser's menu. A simple web page like this one is just plain text with HTML commands (mark
plain text.

When you're just learning and want to experiment, you can do everything on your PC. Create a new directory ("folder") for your website, and then put the web-page files
other plain-text editor (not word processor) on your PC to create an `index.html` file, which you can view locally with your Web browser. (You can also use word process
text", "text", "text document", or "text document MS-DOS format".) Later I'll explain how you can install your web site on the Internet.

Example for Windows:
Move your mouse cursor to the desktop. Right-click, choose *New*, then choose *Folder*, and a "New Folder" appears, with its name highlighted. Type "Web" to char
want). Then *Start -> Run* and type "notepad" to start NotePad. Now you can create your first Web page.

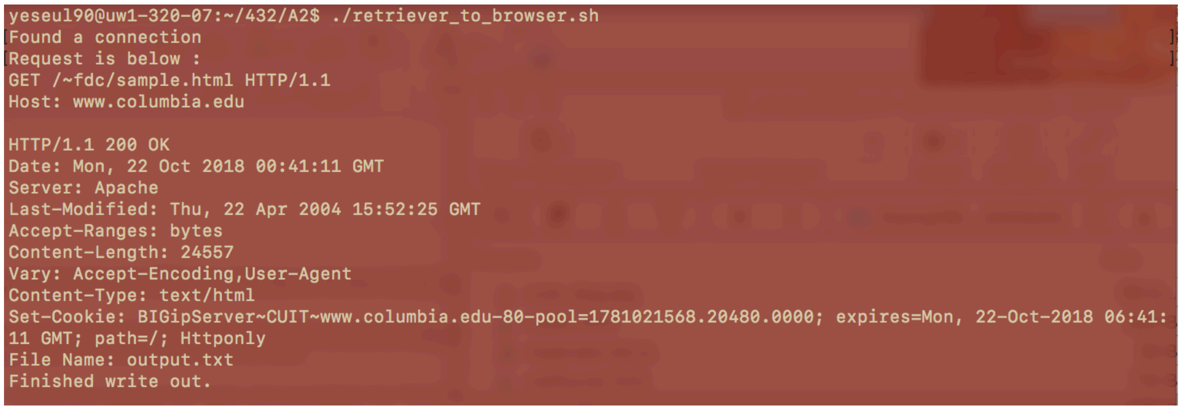
2. HTML Syntax

Web pages are written in Hyper Text Markup Language (HTML). HTML has three special characters: <, &, >. An HTML command is enclosed in <...>, for example <p>
bold") and </b> ("end bold"). So the following HTML text:

```
This sentence contains <b>bold</b> text.
```

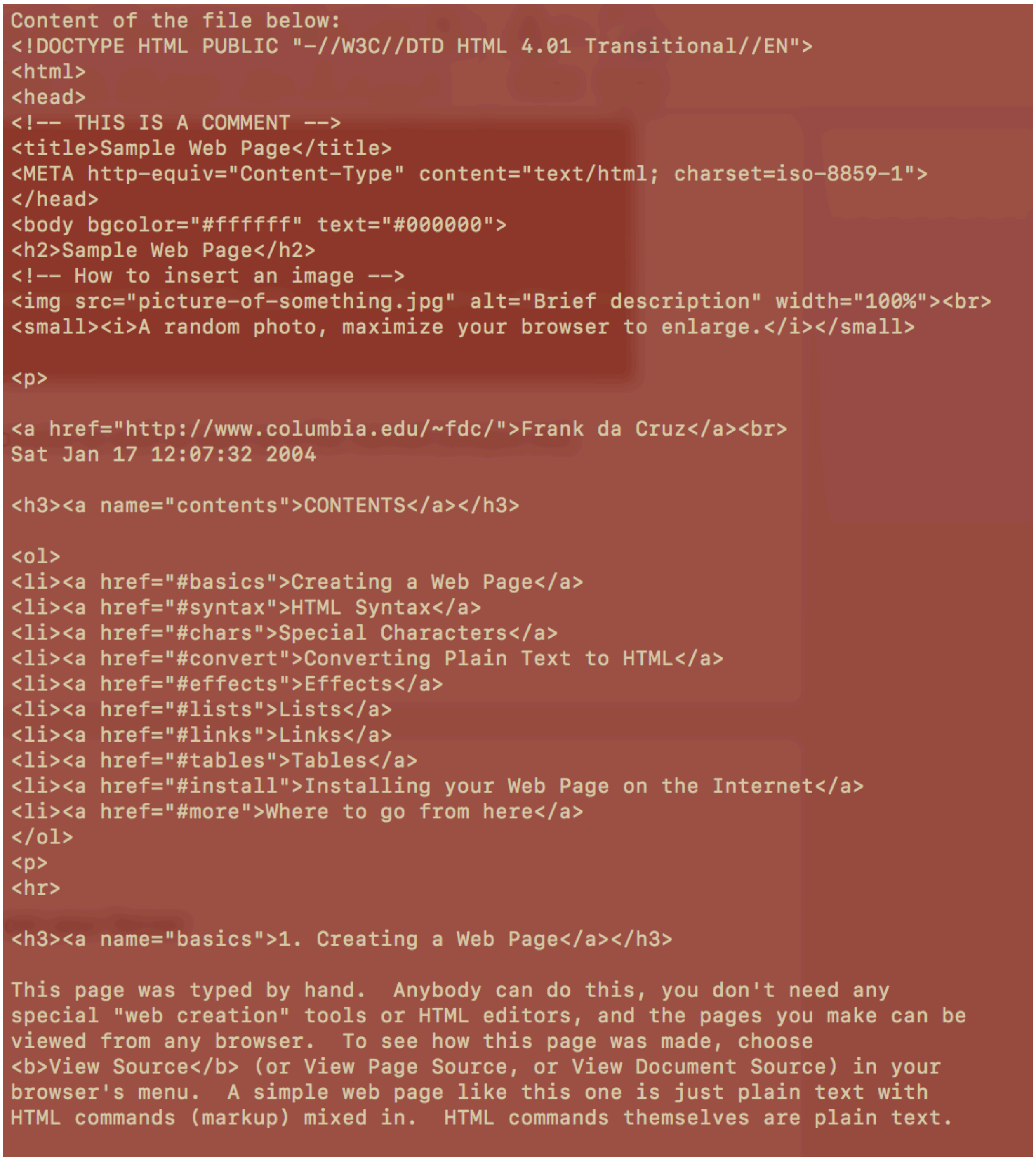
produces:

This is the web browser which my retriever access to the real web server at (www.columbia.edu/~fdc/sample.html).



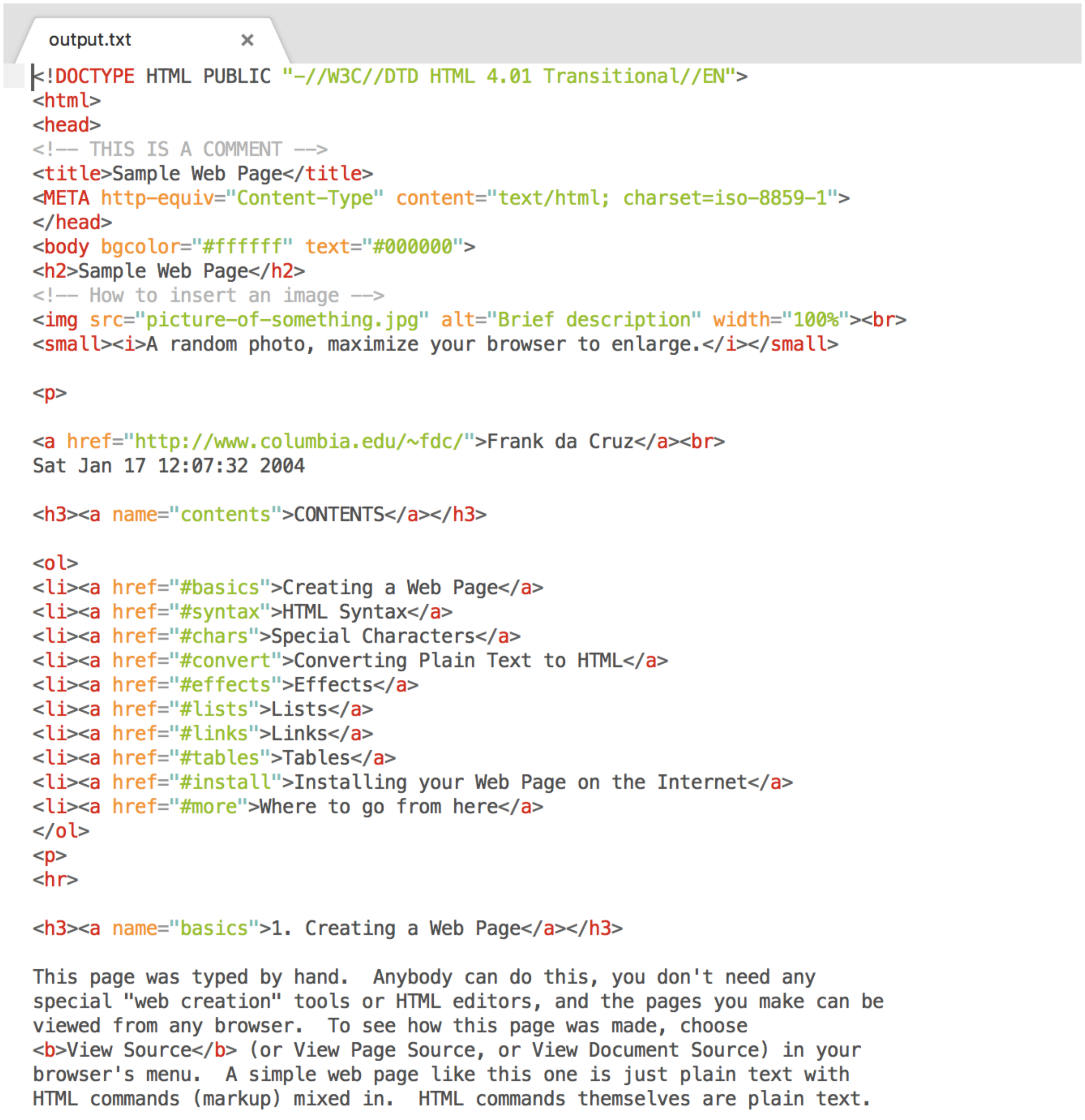
```
yeseul90@uw1-320-07:~/432/A2$ ./retriever_to_browser.sh
Found a connection
Request is below :
GET /~fdc/sample.html HTTP/1.1
Host: www.columbia.edu

HTTP/1.1 200 OK
Date: Mon, 22 Oct 2018 00:41:11 GMT
Server: Apache
Last-Modified: Thu, 22 Apr 2004 15:52:25 GMT
Accept-Ranges: bytes
Content-Length: 24557
Vary: Accept-Encoding,User-Agent
Content-Type: text/html
Set-Cookie: BIGipServer~CUIT~www.columbia.edu-80-pool=1781021568.20480.0000; expires=Mon, 22-Oct-2018 06:41:
11 GMT; path=/; Httponly
File Name: output.txt
Finished write out.
```

The retriever program prints status code information that gained from the browser's server.

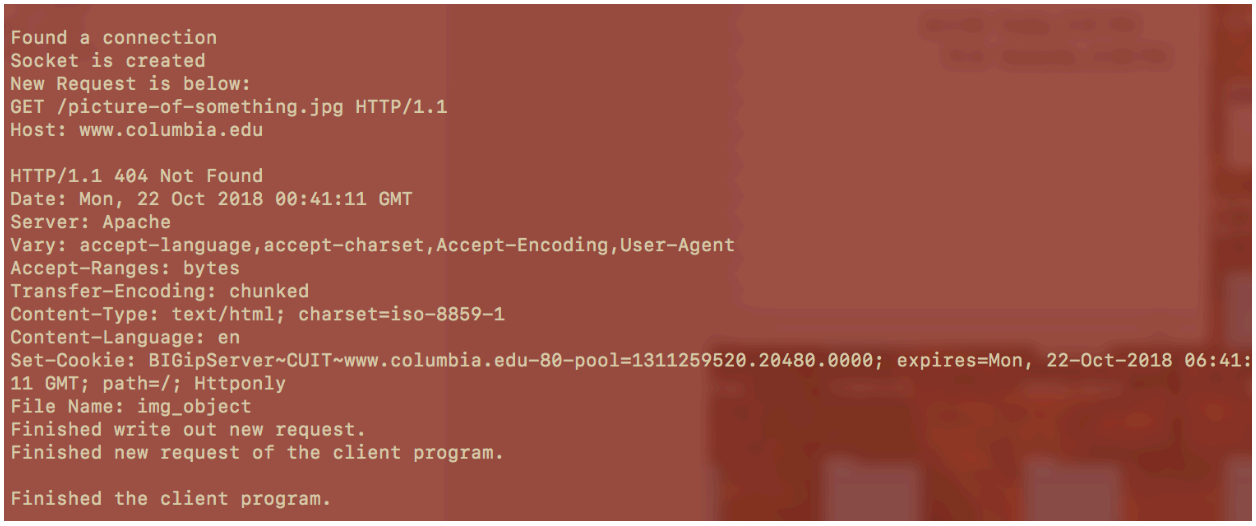


```
Content of the file below:
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<!-- THIS IS A COMMENT -->
<title>Sample Web Page</title>
<META http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body bgcolor="#ffffff" text="#000000">
<h2>Sample Web Page</h2>
<!-- How to insert an image -->
<img src="picture-of-something.jpg" alt="Brief description" width="100%"><br>
<small><i>A random photo, maximize your browser to enlarge.</i></small>

<p>

<a href="http://www.columbia.edu/~fdc/">Frank da Cruz</a><br>
Sat Jan 17 12:07:32 2004

<h3><a name="contents">CONTENTS</a></h3>

<ol>
<li><a href="#basics">Creating a Web Page</a>
<li><a href="#syntax">HTML Syntax</a>
<li><a href="#chars">Special Characters</a>
<li><a href="#convert">Converting Plain Text to HTML</a>
<li><a href="#effects">Effects</a>
<li><a href="#lists">Lists</a>
<li><a href="#links">Links</a>
<li><a href="#tables">Tables</a>
<li><a href="#install">Installing your Web Page on the Internet</a>
<li><a href="#more">Where to go from here</a>
</ol>
<p>
<hr>

<h3><a name="basics">1. Creating a Web Page</a></h3>

This page was typed by hand.  Anybody can do this, you don't need any
special "web creation" tools or HTML editors, and the pages you make can be
viewed from any browser.  To see how this page was made, choose
<b>View Source</b> (or View Page Source, or View Document Source) in your
browser's menu.  A simple web page like this one is just plain text with
HTML commands (markup) mixed in.  HTML commands themselves are plain text.
```

The retriever program prints out the status code and results in the console. Also, prints the web page output to the console as well as save as a file named [output.txt].



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<!-- THIS IS A COMMENT -->
<title>Sample Web Page</title>
<META http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body bgcolor="#ffffff" text="#000000">
<h2>Sample Web Page</h2>
<!-- How to insert an image -->
<img src="picture-of-something.jpg" alt="Brief description" width="100%"><br>
<small><i>A random photo, maximize your browser to enlarge.</i></small>

<p>

<a href="http://www.columbia.edu/~fdc/">Frank da Cruz</a><br>
Sat Jan 17 12:07:32 2004

<h3><a name="contents">CONTENTS</a></h3>

<ol>
<li><a href="#basics">Creating a Web Page</a>
<li><a href="#syntax">HTML Syntax</a>
<li><a href="#chars">Special Characters</a>
<li><a href="#convert">Converting Plain Text to HTML</a>
<li><a href="#effects">Effects</a>
<li><a href="#lists">Lists</a>
<li><a href="#links">Links</a>
<li><a href="#tables">Tables</a>
<li><a href="#install">Installing your Web Page on the Internet</a>
<li><a href="#more">Where to go from here</a>
</ol>
<p>
<hr>

<h3><a name="basics">1. Creating a Web Page</a></h3>

This page was typed by hand.  Anybody can do this, you don't need any
special "web creation" tools or HTML editors, and the pages you make can be
viewed from any browser.  To see how this page was made, choose
<b>View Source</b> (or View Page Source, or View Document Source) in your
browser's menu.  A simple web page like this one is just plain text with
HTML commands (markup) mixed in.  HTML commands themselves are plain text.
```



```
Found a connection
Socket is created
New Request is below:
GET /picture-of-something.jpg HTTP/1.1
Host: www.columbia.edu

HTTP/1.1 404 Not Found
Date: Mon, 22 Oct 2018 00:41:11 GMT
Server: Apache
Vary: accept-language,accept-charset,Accept-Encoding,User-Agent
Accept-Ranges: bytes
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-1
Content-Language: en
Set-Cookie: BIGipServer~CUIT~www.columbia.edu-80-pool=1311259520.20480.0000; expires=Mon, 22-Oct-2018 06:41:
11 GMT; path=/; Httponly
File Name: img_object
Finished write out new request.
Finished new request of the client program.

Finished the client program.
```

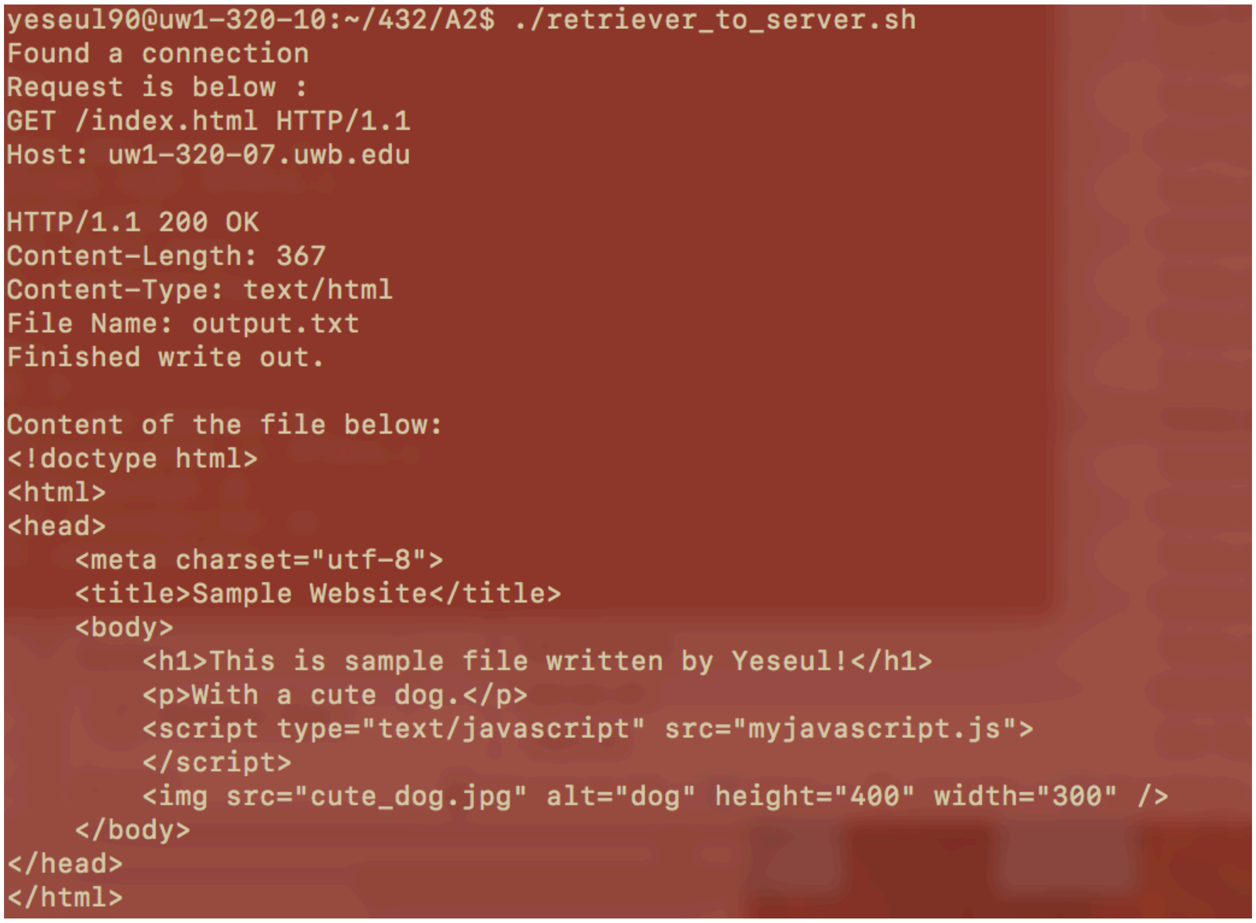
The retriever tries to find <script> and <img> objects and create new GET requests for taking its src and parse the path and file name. If it couldn't find, it returns not found page. If found any, it will save object to the file.

Case 3) Your Retriever Accessing a File From your Server

1. SSH to the one of the linux machine.
2. Type ifconfig to find out the IP address of the machine. Ex) uw1-320-07 has an IP address of (172.21.198.87).
3. Type ./server.sh to connect to the server program first.
4. Open another terminal and SSH to another linux machine.
5. Modify IP address in the ./retriever_to_server.sh. (Default: IP of uw1-320-07).
6. Type ./retriever_to_server.sh

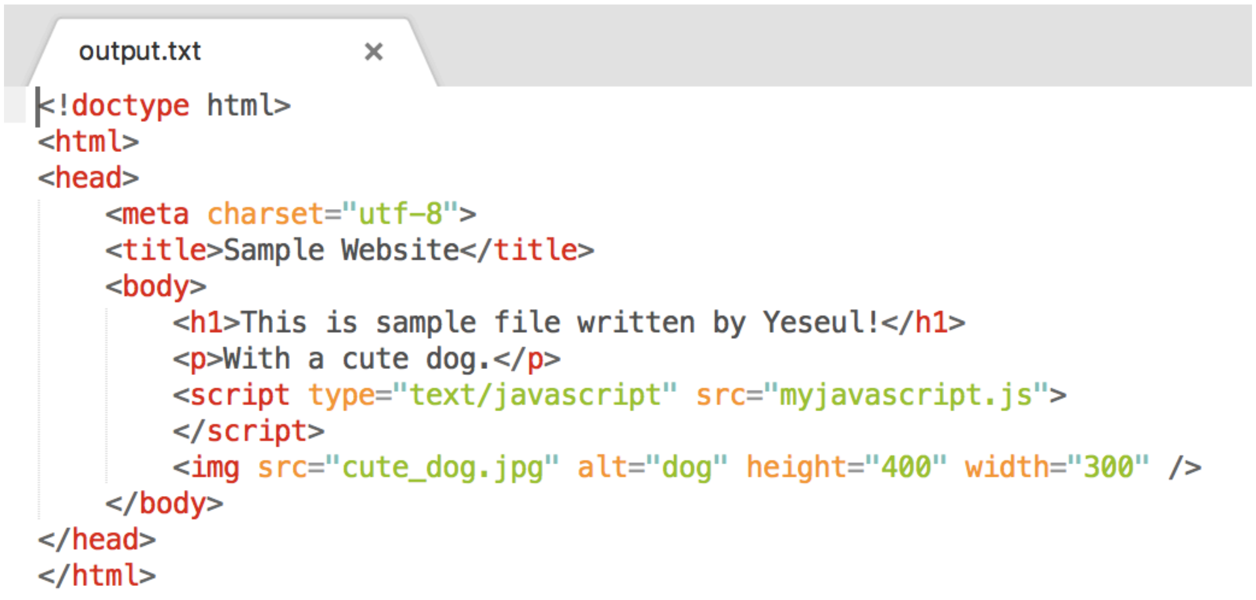
Or alternatively,

5. Type g++ Retriever.cpp -pthread -o retriever -std=c++11
6. Type ./retriever 172.21.198.87 /index.html 2681



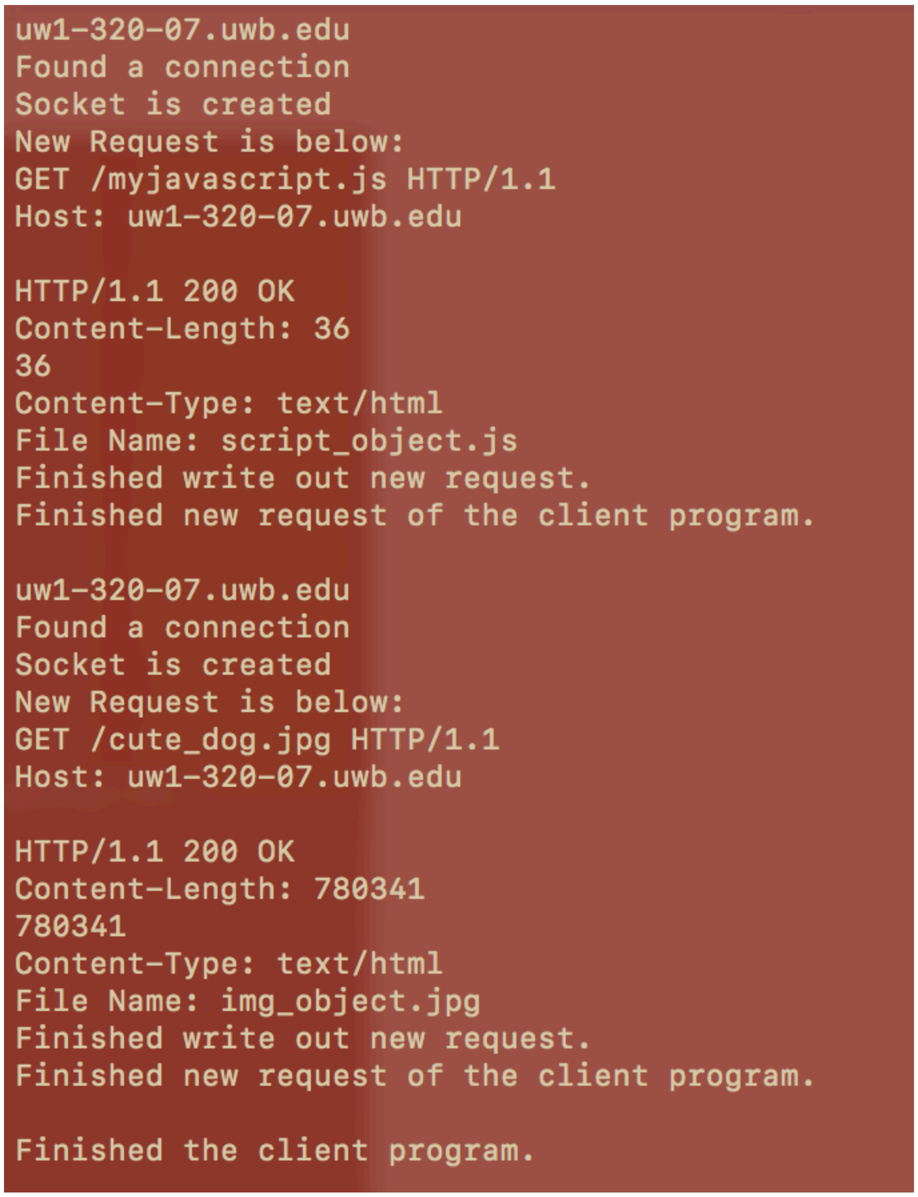
```
yeseul90@uw1-320-10:~/432/A2$ ./retriever_to_server.sh
Found a connection
Request is below :
GET /index.html HTTP/1.1
Host: uw1-320-07.uwb.edu

HTTP/1.1 200 OK
Content-Length: 367
Content-Type: text/html
File Name: output.txt
Finished write out.

Content of the file below:
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>Sample Website</title>
    <body>
        <h1>This is sample file written by Yeseul!</h1>
        <p>With a cute dog.</p>
        <script type="text/javascript" src="myjavascript.js">
        </script>
        <img src="cute_dog.jpg" alt="dog" height="400" width="300" />
    </body>
</head>
</html>
```

The retriever program outputs code, request, file content to the console.



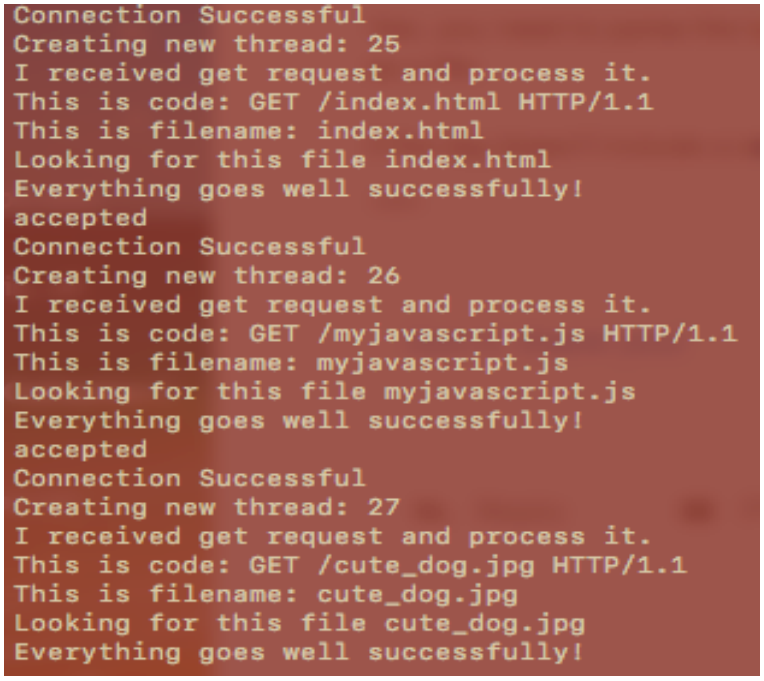
```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>Sample Website</title>
    <body>
        <h1>This is sample file written by Yeseul!</h1>
        <p>With a cute dog.</p>
        <script type="text/javascript" src="myjavascript.js">
        </script>
        <img src="cute_dog.jpg" alt="dog" height="400" width="300" />
    </body>
</head>
</html>
```

Also creates an [output.txt] file that has the content of the data.



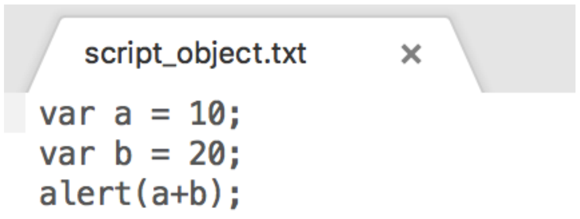
```
uw1-320-07.uwb.edu
Found a connection
Socket is created
New Request is below:
GET /myjavascript.js HTTP/1.1
Host: uw1-320-07.uwb.edu

HTTP/1.1 200 OK
Content-Length: 36
36
Content-Type: text/html
File Name: script_object.js
Finished write out new request.
Finished new request of the client program.

uw1-320-07.uwb.edu
Found a connection
Socket is created
New Request is below:
GET /cute_dog.jpg HTTP/1.1
Host: uw1-320-07.uwb.edu

HTTP/1.1 200 OK
Content-Length: 780341
780341
Content-Type: text/html
File Name: img_object.jpg
Finished write out new request.
Finished new request of the client program.

Finished the client program.
```

For each <script> or <img> object, the retriever program creates a new socket and throws a new GET request and save it as a new file.



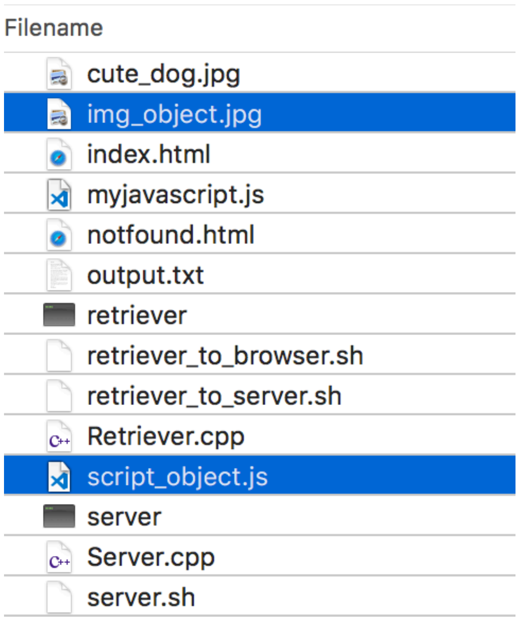
```
Connection Successful
Creating new thread: 25
I received get request and process it.
This is code: GET /index.html HTTP/1.1
This is filename: index.html
Looking for this file index.html
Everything goes well successfully!
accepted
Connection Successful
Creating new thread: 26
I received get request and process it.
This is code: GET /myjavascript.js HTTP/1.1
This is filename: myjavascript.js
Looking for this file myjavascript.js
Everything goes well successfully!
accepted
Connection Successful
Creating new thread: 27
I received get request and process it.
This is code: GET /cute_dog.jpg HTTP/1.1
This is filename: cute_dog.jpg
Looking for this file cute_dog.jpg
Everything goes well successfully!
```

This is corresponding server program side response.



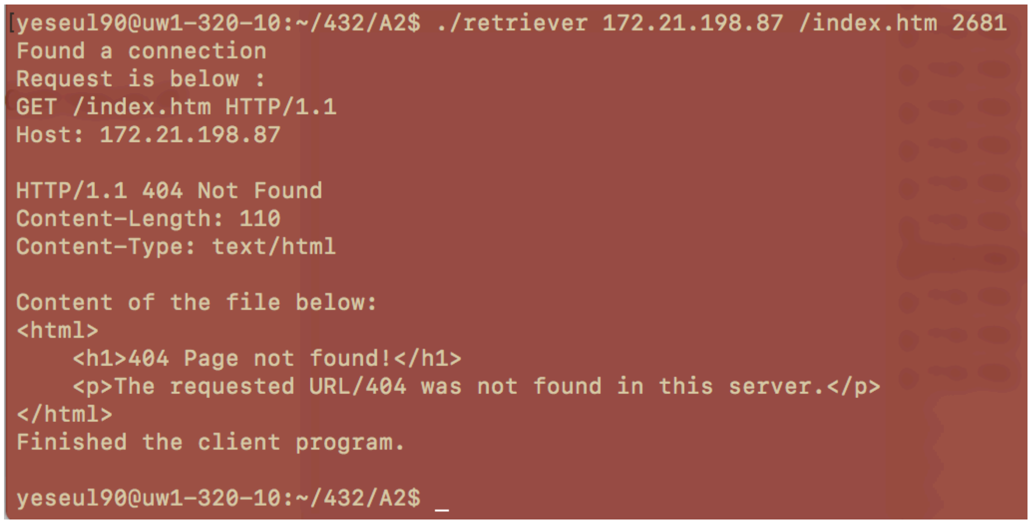
```
var a = 10;
var b = 20;
alert(a+b);
```

Found out that the script text is correctly downloaded as script_object.txt and contains the file that is the same as “myjavascript.js”.



Filename
cute_dog.jpg
img_object.jpg
index.html
myjavascript.js
notfound.html
output.txt
retriever
retriever_to_browser.sh
retriever_to_server.sh
Retriever.cpp
script_object.js
server
Server.cpp
server.sh

[img_object.jpg] and [script_object.js] is objects created and downloaded after running the program.



```
[yeseul90@uw1-320-10:~/432/A2$ ./retriever 172.21.198.87 /index.htm 2681
Found a connection
Request is below :
GET /index.htm HTTP/1.1
Host: 172.21.198.87

HTTP/1.1 404 Not Found
Content-Length: 110
Content-Type: text/html

Content of the file below:
<html>
    <h1>404 Page not found!</h1>
    <p>The requested URL/404 was not found in this server.</p>
</html>
Finished the client program.

yeseul90@uw1-320-10:~/432/A2$ _
```

If retriever tries to retrieve the file that is not existing, the server program sends customized Not Found Page so that it will display on the retriever’s console.