

Homework 1

1. Single connection mode (serial) and multiple connection mode (parallel with multiple threads)

In our code, ten threads have been initialized.

```
1 import BaseHTTPServer
2 import CGIHTTPServer ## This line enables CGI error reporting
3 from SocketServer import ThreadingMixIn
4 import threading
5 import time
6 import socket
7
8
9 if __name__ == '__main__':
10     addr = ('', 5000)
11     handler = CGIHTTPServer.CGIHTTPRequestHandler
12     print("Server started, listen to port 5000!")
13     server = BaseHTTPServer.HTTPServer(addr, handler)
14     for i in range(10): # create 10 threads
15         server_thread = threading.Thread(target=server.serve_forever)
16         server_thread.daemon = True
17         server_thread.start()
18     time.sleep(9e9)
```

2. HTTP GET requests with query and header parsing

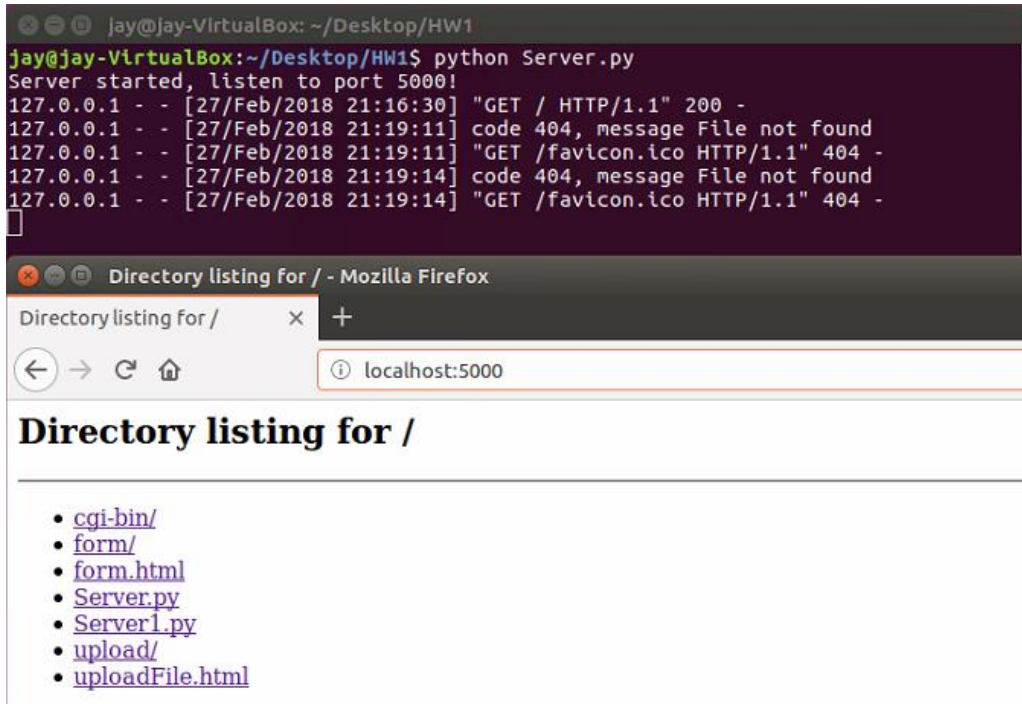
When the server is running, use the curl command in Linux, the result of HTTP GET requests are showed below. Also, use GET command in Linux, you can get the content of webpage.

```
jay@jay-VirtualBox: ~
jay@jay-VirtualBox:~$ curl -I -L http://localhost:5000
HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/2.7.13
Date: Wed, 28 Feb 2018 02:34:59 GMT
Content-type: text/html; charset=UTF-8
Content-Length: 444

jay@jay-VirtualBox:~$ GET http://localhost:5000
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><html>
<title>Directory listing for /</title>
<body>
<h2>Directory listing for /</h2>
<hr>
<ul>
<li><a href="cgi-bin/">cgi-bin</a>
<li><a href="form/">form</a>
<li><a href="form.html">form.html</a>
<li><a href="Server.py">Server.py</a>
<li><a href="Server1.py">Server1.py</a>
<li><a href="upload/">upload</a>
<li><a href="uploadFile.html">uploadFile.html</a>
</ul>
<hr>
</body>
</html>
```

3. Automatic directory listing

Run the Server: `python Server.py`, the directory listing is showed. Need to note, the `cgi-bin/` directory cannot be accessed, because the CGI scripts are in this directory, which may not be read or modified by clients.



The screenshot shows a terminal window and a web browser window. The terminal window, titled 'jay@Jay-VirtualBox: ~/Desktop/HW1', shows the command `python Server.py` being executed. The output indicates the server started on port 5000 and received several requests: a successful GET request for '/' (200 OK) and three failed GET requests for '/favicon.ico' (404 Not Found). The web browser window, titled 'Directory listing for / - Mozilla Firefox', shows the directory listing for the root directory. The listing includes links to `cgi-bin/`, `form/`, `form.html`, `Server.py`, `Server1.py`, `upload/`, and `uploadFile.html`.

```
jay@Jay-VirtualBox: ~/Desktop/HW1
jay@Jay-VirtualBox:~/Desktop/HW1$ python Server.py
Server started, listen to port 5000!
127.0.0.1 - - [27/Feb/2018 21:16:30] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/Feb/2018 21:19:11] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [27/Feb/2018 21:19:11] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [27/Feb/2018 21:19:14] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [27/Feb/2018 21:19:14] "GET /favicon.ico HTTP/1.1" 404 -
```

Directory listing for / - Mozilla Firefox

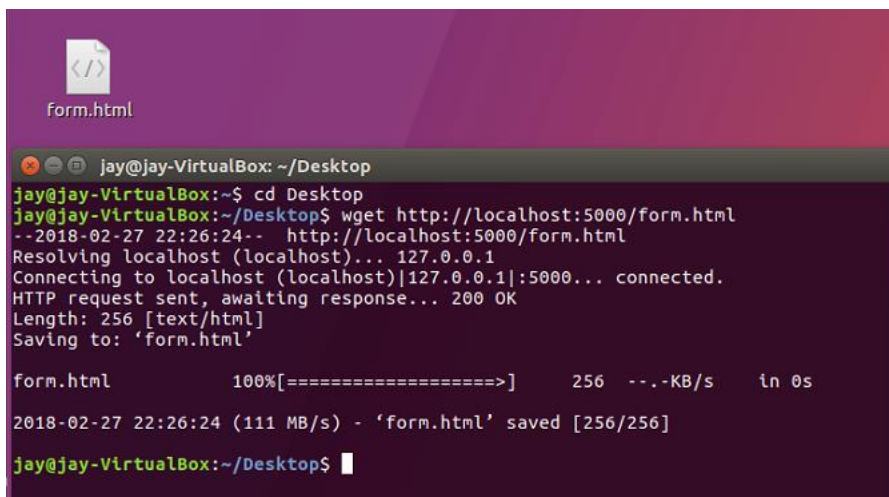
localhost:5000

Directory listing for /

- [cgi-bin/](#)
- [form/](#)
- [form.html](#)
- [Server.py](#)
- [Server1.py](#)
- [upload/](#)
- [uploadFile.html](#)

4. Static file transport

For downing files from the server, we use the `wget` command:
`wget http://localhost:5000/form.html`

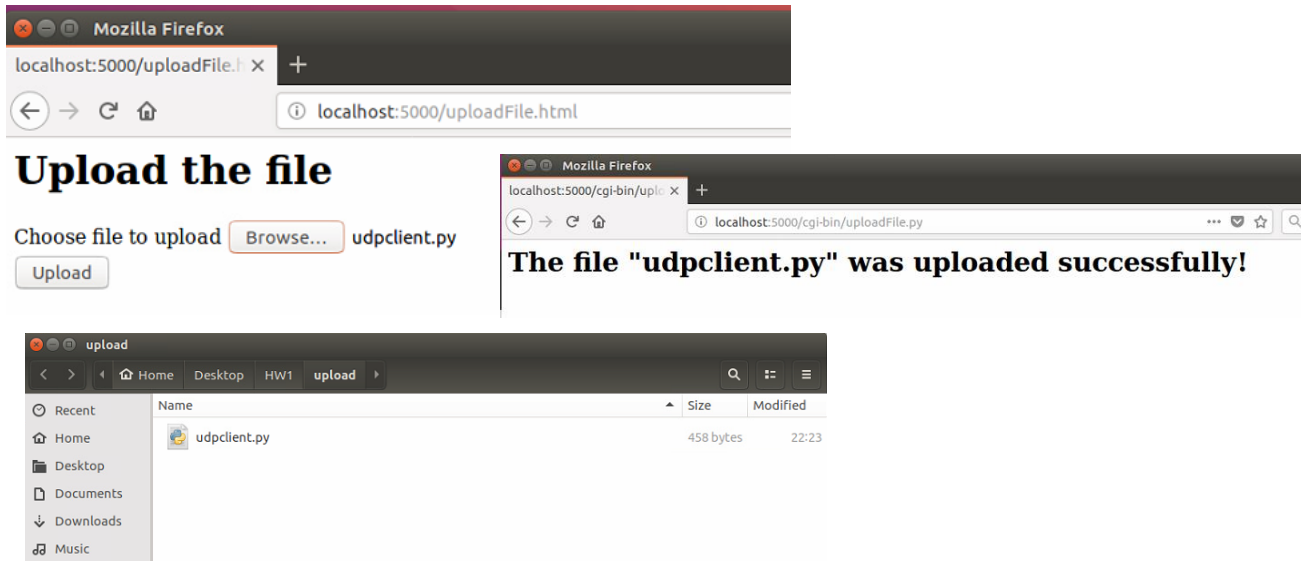


The screenshot shows a terminal window titled 'jay@Jay-VirtualBox: ~/Desktop'. The user runs the command `wget http://localhost:5000/form.html`. The output shows the file being downloaded from the server. The file size is 256 bytes, and it is saved as 'form.html'. The download progress is shown as 100% completed.

```
jay@Jay-VirtualBox: ~/Desktop
jay@Jay-VirtualBox:~$ cd Desktop
jay@Jay-VirtualBox:~/Desktop$ wget http://localhost:5000/form.html
--2018-02-27 22:26:24-- http://localhost:5000/form.html
Resolving localhost (localhost)... 127.0.0.1
Connecting to localhost (localhost)|127.0.0.1|:5000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 256 [text/html]
Saving to: 'form.html'

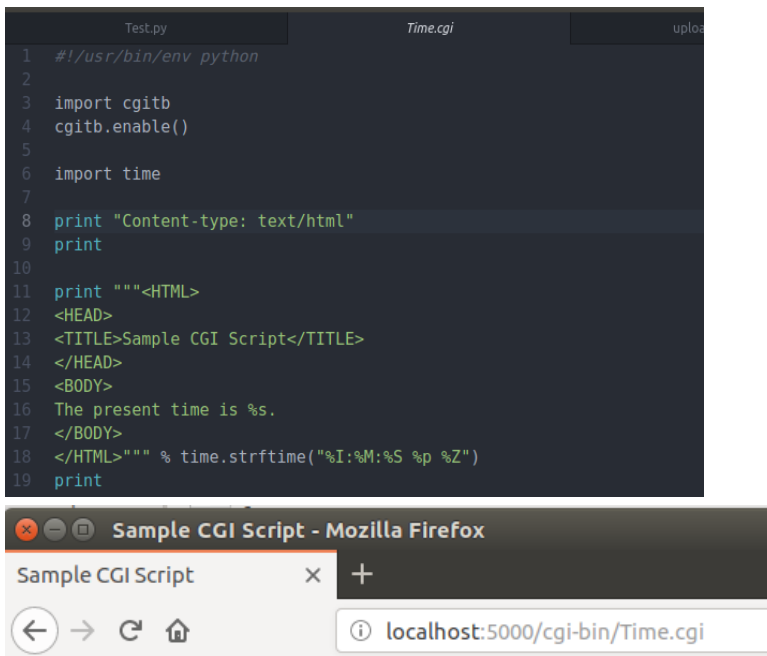
form.html      100%[=====]      256  --.-KB/s   in 0s
2018-02-27 22:26:24 (111 MB/s) - 'form.html' saved [256/256]
jay@Jay-VirtualBox:~/Desktop$
```

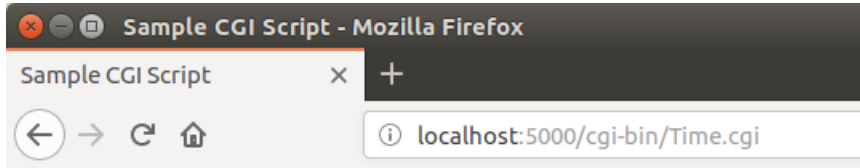
For uploading the files to the server, open the uploadFile.html, then simply choose the file to upload from our local function.



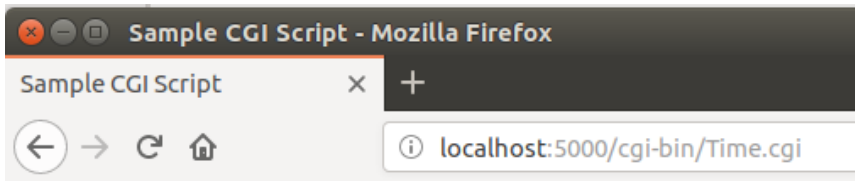
5. Basic CGI support by running a sample CGI script on the server side

We created a simple test CGI script: Time.cgi. When we browser to the `http://localhost:5000/cgi-bin/Time.cgi`, the CGI script will be executed by the server, the current time will be displayed in the page. Note: To run the CGI script, you need to make the code executable, like `chmod +x Time.cgi`





The present time is 03:50:11 PM CST.



The present time is 03:51:22 PM CST.

Extra credit: implement basic user interaction on submitting a form.

We first created the form.html, then when the form is submitted, the formSave.py CGI script will be executed, the form data will be saved to the form directory named with formData.txt.

