

**Ubuntu 16.04 LTS Instructions** (As discussed in class, if you're using a different OS and have problems compiling or running the assignment, it is up to you to solve the problem.)

Once you have download the hw1.zip file from OAKS, open a terminal in the folder where the zip file downloaded to. In this example, the folder is Downloads/340, however this can be any folder you choose.

As shown below, use the “unzip” command to extract the files. This will create a new folder named hw1.

```
[munsellb@CSCI-C02RX5D0FVH7 ~/Downloads/340]$ unzip hw1.zip
Archive:  hw1.zip
  creating: hw1/
  inflating: hw1/Makefile
  inflating: hw1/p1.c
  inflating: hw1/server.c
  inflating: hw1/server.h
  inflating: hw1/SimplePost.html
[munsellb@CSCI-C02RX5D0FVH7 ~/Downloads/340]$
```

Next, “cd” into the hw1 folder, and the use the “ls -l” command to list the folder contents. As shown below you should see these files.

```
[munsellb@CSCI-C02RX5D0FVH7 ~/Downloads/340]$ cd hw1
[munsellb@CSCI-C02RX5D0FVH7 ~/Downloads/340/hw1]$ ls -l
total 48
-rw-----@ 1 munsellb  1298827249   247 Sep  4 13:08 Makefile
-rw-----@ 1 munsellb  1298827249   409 Sep  4 13:06 SimplePost.html
-rw-----@ 1 munsellb  1298827249  2548 Sep  4 13:08 p1.c
-rw-----@ 1 munsellb  1298827249  4618 Sep  4 13:08 server.c
-rw-----@ 1 munsellb  1298827249   860 Nov 21  2016 server.h
[munsellb@CSCI-C02RX5D0FVH7 ~/Downloads/340/hw1]$
```

Next, you can make the project executable (p1) by simply typing “make” in the terminal as shown below and then using the “ls -l” command to list the folder contents.

```
[munsellb@CSCI-C02RX5D0FVH7 ~/Downloads/340/hw1]$ make
gcc -c -Wall -g server.c
gcc -c -Wall -g p1.c
gcc server.o p1.o -o p1
[munsellb@CSCI-C02RX5D0FVH7 ~/Downloads/340/hw1]$ ls -l
total 104
-rw-----@ 1 munsellb 1298827249 247 Sep  4 13:08 Makefile
-rw-----@ 1 munsellb 1298827249 409 Sep  4 13:06 SimplePost.html
-rwxr-xr-x  1 munsellb 1298827249 10316 Sep  9 05:24 p1
-rw-----@ 1 munsellb 1298827249 2550 Sep  9 05:24 p1.c
-rw-r--r--  1 munsellb 1298827249 4676 Sep  9 05:24 p1.o
-rw-----@ 1 munsellb 1298827249 4618 Sep  4 13:08 server.c
-rw-----@ 1 munsellb 1298827249 860 Nov 21 2016 server.h
-rw-r--r--  1 munsellb 1298827249 6352 Sep  9 05:24 server.o
[munsellb@CSCI-C02RX5D0FVH7 ~/Downloads/340/hw1]$
```

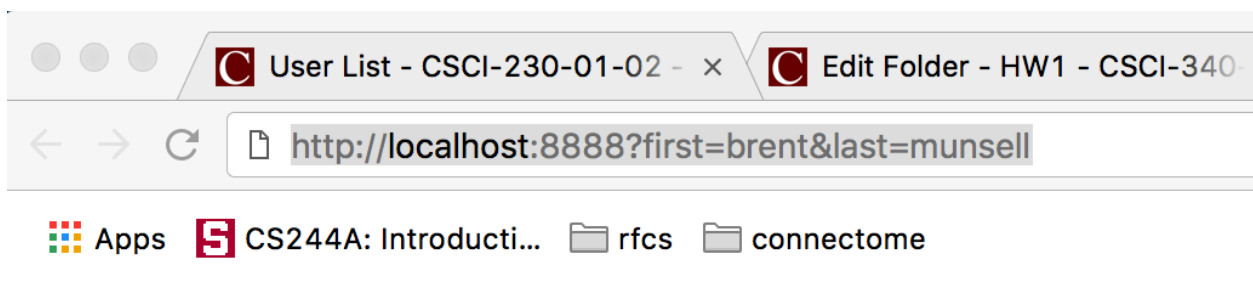
To execute your webserver, simply type “make run”. As shown below this will run a new webserver that is listening for browser connections on port 8888. To stop the server, simply use the Control-C key stroke.

```
[munsellb@CSCI-C02RX5D0FVH7 ~/Downloads/340/hw1]$ make run
./p1 8888
server socket fd = 3
-----
Server socket listening and accepting connections on port 8888

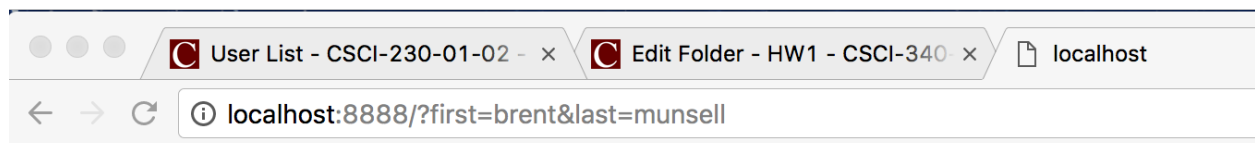
```

## HTTP GET Example

Next, let’s issue a GET message from your browser (Firefox or Chrome is a good choice) that use the first and last key-value pairs shown in the assignment. Note: you can use any key-value pairs, this is just the example ones in the assignment.



After you enter this into the URL portion of the browser, hit the return button. As shown below, you should see an empty key-value table in your browser.



## CSCI 340 (Operating Systems) Project 1

Key	Value
-----	-------

As shown below, if you look at the terminal, you will see the HTTP GET request message sent by the browser to your webserver. Using the HTTP request message format (see Appendix 4.0 in the assignment) the METHOD is set to GET, and the URL is set to “/?first=brent&last=munsell”.

```
GET /?first=brent&last=munsell HTTP/1.1
Host: localhost:8888
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10
Accept: text/html,application/xhtml+xml,application/x
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.8
```

### HTTP POST Example

Next, let's issue a POST message from your browser (Firefox or Chrome is a good choice) that use the SimplePost.html file provided in the hw1.zip file.

First, let's change the IP address of your server to localhost. To do this, open SimplePost.html with a text editor (like vi, vim, or nano) and change “action” to “http://localhost:8888” as shown below, then save the file.

```

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Simple Post</title>
<link rel="shortcut icon" href="http://www.example.com/my_empty_resource" />
</head>
<body>

    <form method="POST" action="http://localhost:8888">

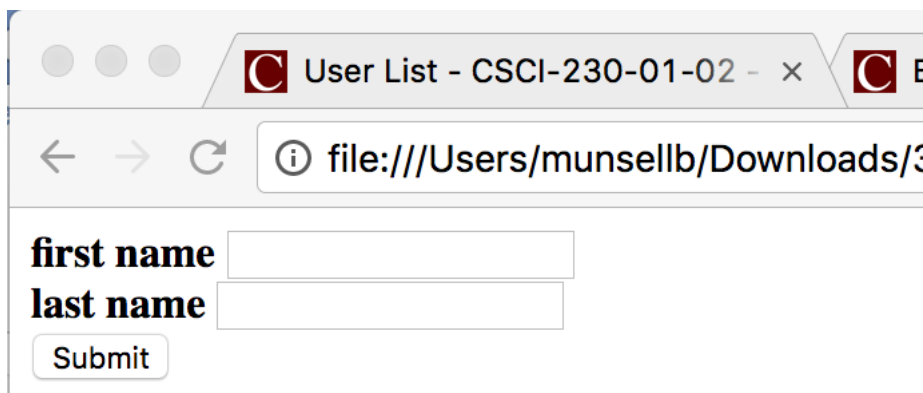
        <b>first name</b>&nbsp;</b><input type="text" name="first"><br>
        <b>last name</b>&nbsp;</b><input type="text" name="last"><br>
        <input type="submit">

    </form>

</body>
</html>

```

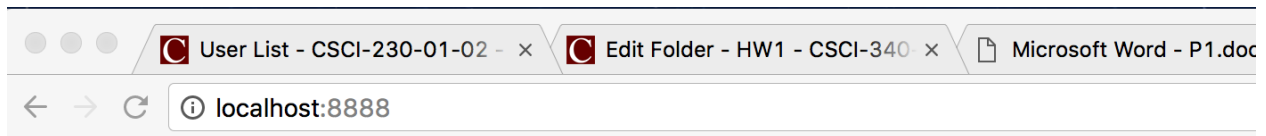
Next, open the SimplePost.html file in your browser. You do this you should see the webpage below.



The screenshot shows a web browser window with the title "User List - CSCI-230-01-02". The address bar displays the file path "file:///Users/munsellb/Downloads/3". The main content area contains a form with the following elements:

- A label "first name" followed by a text input field.
- A label "last name" followed by a text input field.
- A "Submit" button.

Next, put values in the first and last textboxes, then hit the submit button. As shown below, you should see an empty key-value table in your browser.



## CSCI 340 (Operating Systems) Project 1

Key	Value
-----	-------

As shown below, if you look at the terminal, you will see the HTTP POST request message sent by the browser to your webserver. Using the HTTP request message format (see Appendix 4.0 in the assignment) the METHOD is set to POST, and the ENTITY BODY is set to “/?first=brent&last=munsell”.

```
POST / HTTP/1.1
Host: localhost:8888
Connection: keep-alive
Content-Length: 24
Cache-Control: max-age=0
Origin: null
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.109 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.8

first=brent&last=munsell
```

### Some assignment hints

One part of this assignment is to:

- 1) Parse the request message and determine the method type (GET or POST). Once the method type is found, then go to the correct location in the request message (URL or ENTITY BODY) and parse key-value pairs.
- 2) Create a response message using the key-value pairs. Specifically, create a HTML string as shown in Appendix 1.0 or 3.0, place this string in the ENTITY body, then write the response message to the socket.

Note, example HTTP response code is provided in server.c file under the `// THIS IS AN EXAMPLE ENTITY BODY` comment.