Lab Project 3: **Web Server**

- **Number of members per group: 1 or 2**
- **Use C language**

In this project you develop a web server that is accessible by clients. Once you type the IP address (and port number) of the web server (for example in the Chrome browser), it returns one of the following to the client's browser:

- **1: IP Configuration Report**
- **2: Regular HTTP server (returns index.html)**

With Option 1:
The result of the `ifconfig` command is returned to the client. You need to use `Content-type: text/plain` in the response message.

- Note: `ifconfig` returns network interfaces and their configuration.

With Option 2:
Returns the `index.html` file on the web server. You need to create this file manually. You need to use "`Content-type: text/html`" in the response message.
If the file is not found on the server, an error message is returned to the client.

Note: Code skeleton is provided.

Note: Do not use port 80. Your operating system might have this port already reserved.

The result of choosing Option 1 looks like this:

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.143  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fd55:f497:b3a0:0:ad2e:cf9d:69fc:a690  prefixlen 64  scopeid 0x0<global>
        inet6 fd55:f497:b3a0::7d9  prefixlen 128  scopeid 0x0<global>
        inet6 fded:661d:362e:0:8c2f:e606:9a72:c8ad  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::88f7:83a6:58df:7c60  prefixlen 64  scopeid 0x20<link>
        inet6 2601:646:8d00:8a04:dae4:3d3a:f2bc:f91b  prefixlen 64  scopeid 0x0<global>
        inet6 fdee:73ff:7e71:0:37c6:f0f9:554e:5651  prefixlen 64  scopeid 0x0<global>
        ether b8:27:eb:cc:42:74  txqueuelen 1000  (Ethernet)
        RX packets 346942  bytes 48032323 (45.8 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 110285  bytes 27342920 (26.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 9584  bytes 964021 (941.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9584  bytes 964021 (941.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether b8:27:eb:99:17:21  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The result of choosing Option 2 looks like this:

HTML file returned by my server!

Please note that since your web server only returns index.html, you need to find another solution to load the image file in the web page.

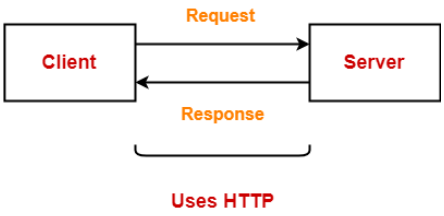
Additional Notes:

- Register a function to handle SIGINT and ask if the user wants to close the program
- Use `setsockopt` to avoid the "address already in use" error

More information

The server starts first and receives one argument, which is the port number.

- Initially HTTP Client (i.e., web browser) sends a HTTP request to the HTTP Server.
- Server processes the request received and sends HTTP response to the HTTP client.

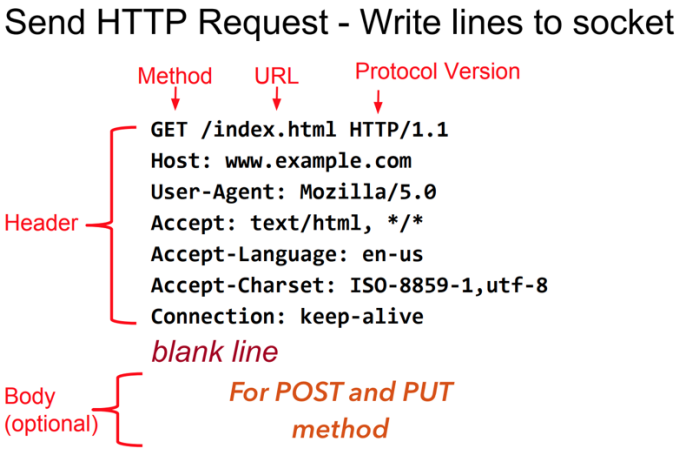


To connect the client to the server, we use the URL of the website in the browser.

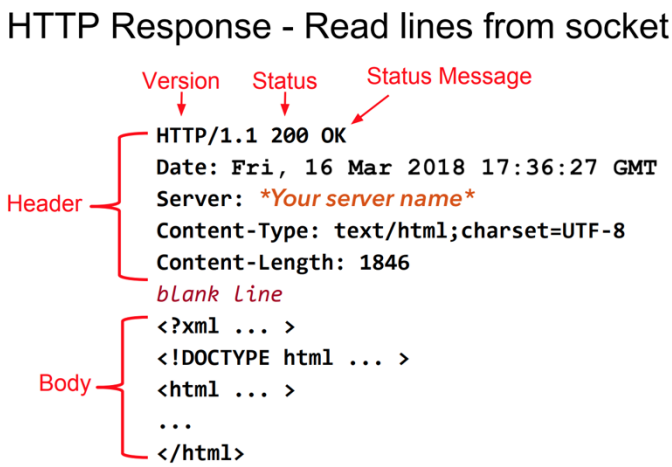
Since the default http port it 80, once you type http://www.scu.edu in the browser, the browser uses the URL/address http://www.scu.edu:80

The webpage returned depends on sever configuration. Some servers have public.html and some will have index.html. In this example, we consider index.html as default page.

Once you type `localhost:8080/index.html`, your web browser actually sends the following message to the web server to request for a specific content:



In response to the request, the client expects a response in the following format:



To prepare the response, first we need to construct the Header. Then insert a blank line, then we can send our message/data.

Assume your reply message is "I am a network professional!". We create the HTTP Header as follows:

```
char *http_header =
    "HTTP/1.1 200 OK\n
     Content-Type: text/plain\n
     Content-Length: 26\n\n
     I am a network professional!";
```

- **HTTP/1.1 200 OK**: HTTP version we are using, Status code, and Status message.
- **Content-Type**: text/plain: We are sending a plain text. There are many Content-Types.
- **Content-Length**: 12: How many bytes the server is sending to the client. The web-browser only reads how much we mention here.