CS 361 – Lab 10
HTTP Headers & Concurrent Servers

MONDAY, APRIL 15TH 2019.

HTTP Headers

- Basic structure of HTTP Transaction,
  - Open a Connection.
  - Send a **REQUEST.**
  - Receive a **RESPONSE.**
  - Close the Connection.

```
unix> telnet www.aol.com 80          Client: open connection to server
Trying 205.188.146.23...             Telnet prints 3 lines to the terminal
Connected to aol.com.
Escape character is '^]'.
GET / HTTP/1.1                       Client: request line
host: www.aol.com                    Client: required HTTP/1.1 HOST header
                                     Client: empty line terminates headers .
HTTP/1.0 200 OK                      Server: response line
MIME-Version: 1.0                    Server: followed by five response headers
Date: Mon, 08 Jan 2001 04:59:42 GMT
Server: NaviServer/2.0 AOLserver/2.3.3
Content-Type: text/html              Server: expect HTML in the response body
Content-Length: 42092                Server: expect 42,092 bytes in the resp body
```

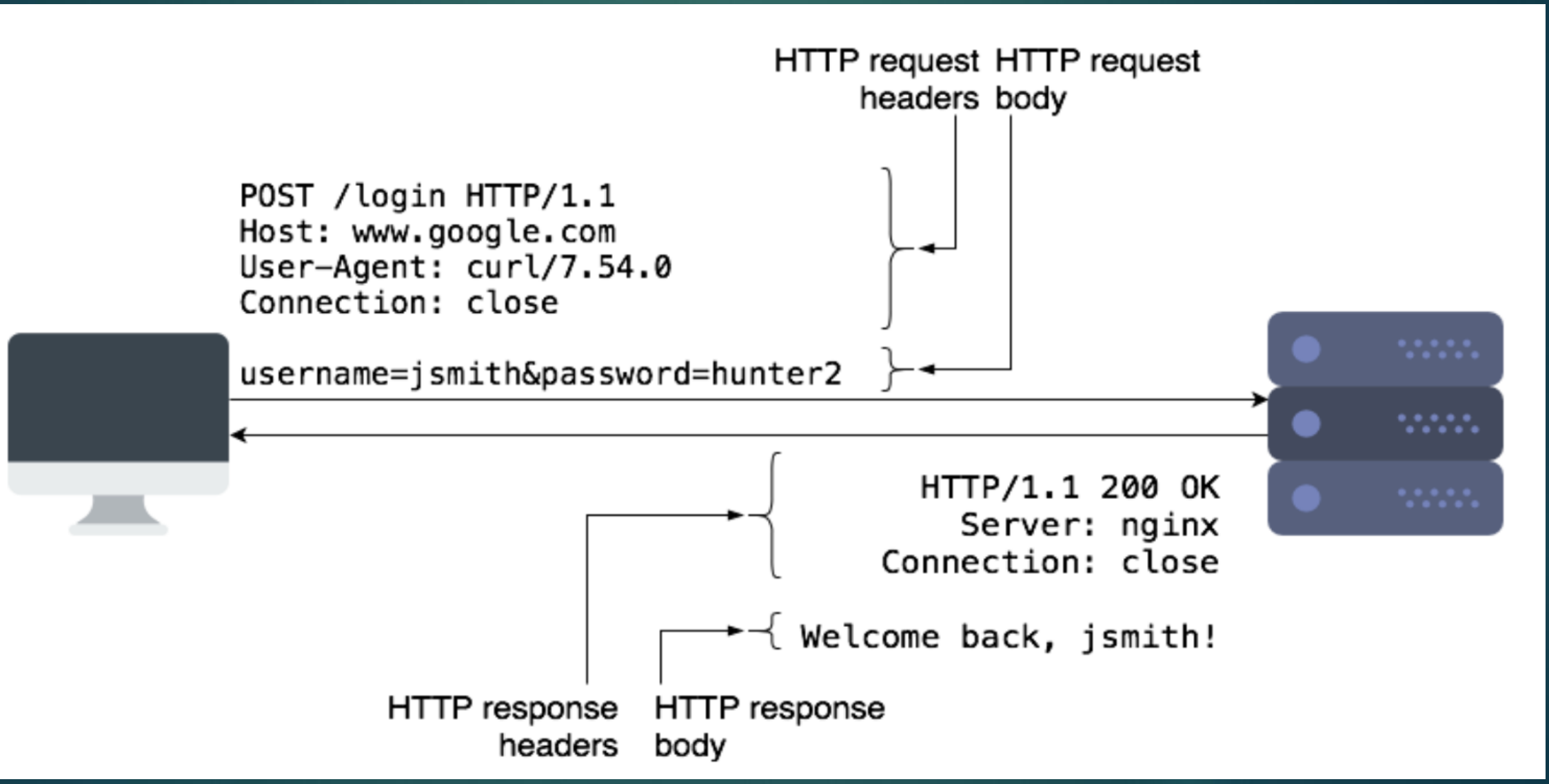
HTTP Requests

- HTTP request is a *request line*, followed by zero or more *request header*
- Request line: `<method> <uri> <version>`
  - `<version>` is HTTP version of request (`HTTP/1.0` or `HTTP/1.1`)
  - `<uri>` is typically URL for proxies, URL suffix for servers.
  - `<method>` is either `GET`, `POST`, `OPTIONS`, `HEAD`, `PUT`, `DELETE`, or `TRACE`.
  - `GET`: Retrieve static or dynamic content
    - Arguments for dynamic content are in URI
    - Workhorse method (99% of requests)
  - `POST`: Retrieve dynamic content
    - Arguments for dynamic content are in the request body
- Request headers: `<header name>: <header data>`
  - Provide additional information to the server.

HTTP Responses

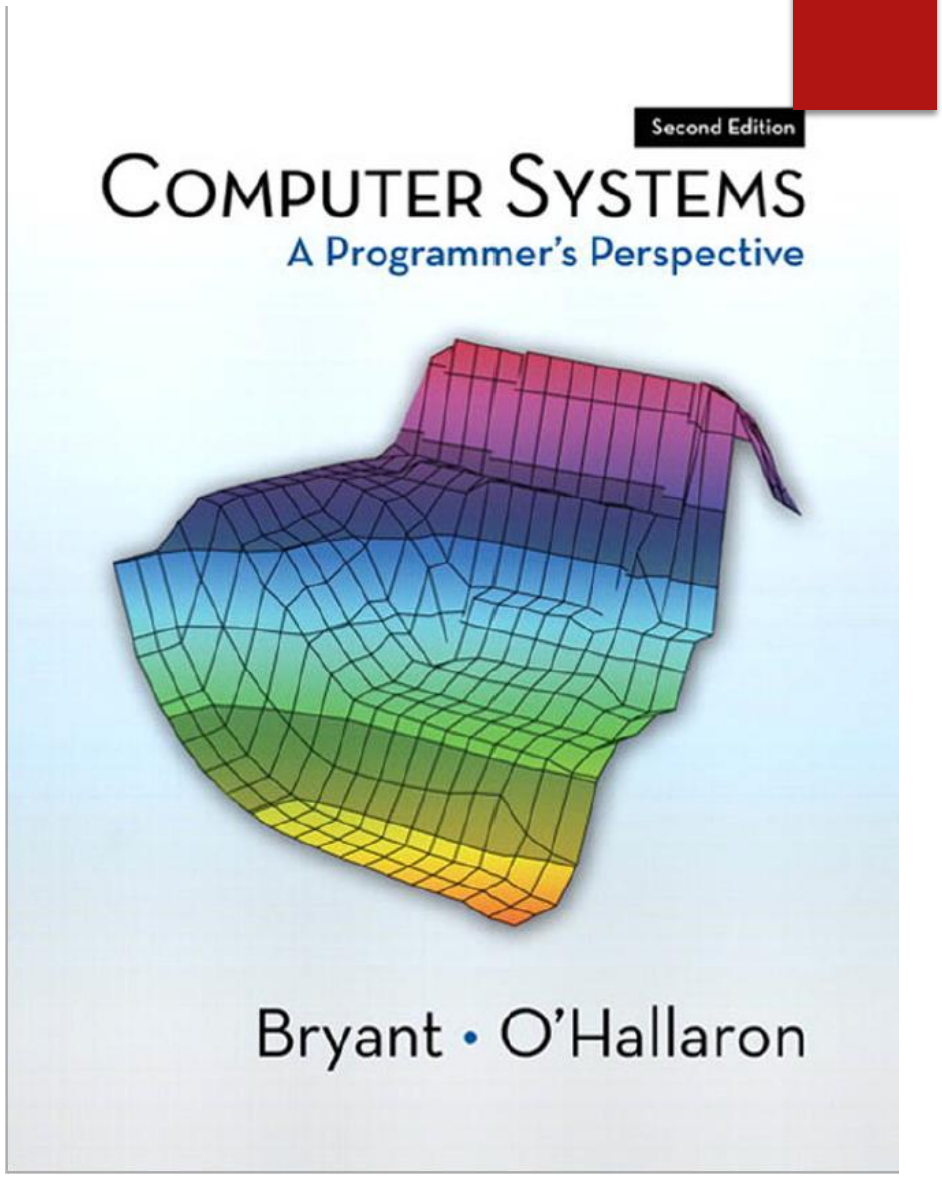
- HTTP response is a *response line* followed by zero or more *response headers*.
- Response line:
- `<version> <status code> <status msg>`
  - <version> is HTTP version of the response.
  - <status code> is numeric status.
  - <status msg> is corresponding English text.
    - 200 OK Request was handled without error
    - 403 Forbidden Server lacks permission to access file
    - 404 Not found Server couldn't find the file.
- Response headers: `<header name>: <header data>`
  - Provide additional information about response
  - `Content-Type:` MIME type of content in response body.
  - `Content-Length:` Length of content in response body.

A transaction



Further Reading

- 11.5 & 11.6



Concurrent Programs

- Three main approaches to write concurrent programs
- **Process** –
  - With this approach, each logical control flow is a process that is scheduled and maintained by the kernel. Since processes have separate virtual address spaces, flows that want to communicate with each other must use some kind of explicit inter-process communication (IPC) mechanism.
- **I/O Multiplexing** –
  - This is a form of concurrent programming where applications explicitly schedule their own logical flows in the context of a single process. Logical flows are modeled as state machines that the main program explicitly transitions from state to state as a result of data arriving on file descriptors. Since the program is a single process, all flows share the same address space.
- **Threads** –
  - Threads are logical flows that run in the context of a single process and are scheduled by the kernel. You can think of threads as a hybrid of the other two approaches, scheduled by the kernel like process flows, and sharing the same virtual address space like I/O multiplexing flows.

Process

- Most simplest from of concurrency
- Key methods
  - Fork()
  - Exec()
  - WaitPid()
- We have been implementing this approach since the beginning
- A few Pointers
  - This approach has clean model for sharing information between parent and children
    - For example file table descriptors between child and parent process
  - Both the processes have separate address space, which might be advantageous or disadvantageous
    - It is impossible for one process to accidentally overwrite memory of other process (Virtual Memory)
    - At the same time, it is difficult to share information between processes as there is considerable overhead in IPC.

Curl

- Command Line tool and a library
- Used to receive and send data between a client and a server or any two machines connected over the internet.
- It supports a wide range of protocols like HTTP, FTP, IMAP, LDAP, POP3, SMTP and many more.
- “man curl” in terminal for more

```
~ $ curl -Li http://www.facebook.com/
HTTP/1.1 302 Found
Location: https://www.facebook.com/
Content-Type: text/html; charset="utf-8"
X-FB-Debug: eWOW1Dpquwgm/zj4YT2Cht4rrhEGSm1OydUHjczvaxr1g6atZQU4RxUwoI0jh8sjLhog
h51o8txIDadyNqkYMA==
Date: Sat, 15 Dec 2018 11:31:30 GMT
Content-Length: 0
Connection: keep-alive

HTTP/2 200
cache-control: private, no-cache, no-store, must-revalidate
pragma: no-cache
strict-transport-security: max-age=15552000; preload
vary: Accept-Encoding
x-content-type-options: nosniff
x-frame-options: DENY
x-xss-protection: 0
expires: Sat, 01 Jan 2000 00:00:00 GMT
set-cookie: fr=1HG7G888ZSheZu24U..BcFOYT.zh.AAA.0.0.BcFOYT.AWUsNnOa; expires=Fri
, 15-Mar-2019 11:31:31 GMT; Max-Age=7776000; path=/; domain=.facebook.com; secur
e; httponly
set-cookie: sb=E-YUXKQlJqPp-J4U8bn7eWsG; expires=Mon, 14-Dec-2020 11:31:31 GMT;
Max-Age=63072000; path=/; domain=.facebook.com; secure; httponly
content-type: text/html; charset="utf-8"
x-fb-debug: 0YE5YXuBd7SZ6xnwcS8l3lw7tVcpV0vGMvROvnlESbLEiFHMWIQuJ7P3FTkQtLirLpbn
l2bGRdSAhQdqZi3l9Q==
date: Sat, 15 Dec 2018 11:31:31 GMT

<!DOCTYPE html>
<html lang="kn" id="facebook" class="no_js">
<head><meta charset="utf-8" /><meta name="referrer" content="default" id="meta_r
```

Lets begin simple

- Modify the server code in from the last lab to,
  - Accept more than 1 connection
  - Serve more than one connection simultaneously
- Read the manual page for curl and deduce the command to call the server from the terminal and display the output to the same.