ECE 435 – Network Engineering
Lecture 3

Vince Weaver

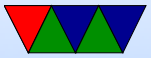
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

4 February 2021

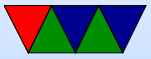
Announcements

- Homework #1 was posted/due Friday now (extension)
- Last class cancelled due to snow. I guess they're going to be cancelling remote/hyrbid classes.
- Sorry if that last class was a bit overwhelming with the C
- Yes, HW#1 and #2 lots of coding. Know non-computer engineers not like this. Amount of coding drops off a lot after HW#2.



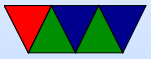
HW #1 notes

- `strace` can be useful when tracking down issues.
- Finding the "struct sockaddr" can be difficult. even if you find in under /usr/include it's tricky as it's a struct that is multiplexed via casting (to handle all possible socket types). Horrible thing about C.
- How read() and write() work.
  - Some people are having issues where they are writing 256 bytes (write will write as many bytes as you said, even if they are trailing zeros), but only reading 255.



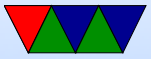
This means the next read is going to get the last 0 rather than the following write.

- When reading, read(fd,buffer,size); What happens if you read 10 bytes but other side only has 4? Only read 4 (result).
- What happens read 10 bytes and other size has 12? You read 10 (result) but to get the rest you need to read again, otherwise it's there the next time you read. You can do a while loop.
- Also note that when you write, you should specify how many bytes you are writing or your whole buffer gets



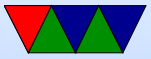
sent even if empty.

- You haven't learned this, but incoming port from the client isn't going to be same as the listening port on the server. The outgoing client port is a random, higher value.
- Why not just malloc() each buffer to the exact size as needed? You can, it's just horribly inefficient. But on modern GHz machines with GB of RAM maybe that doesn't matter.



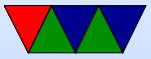
Internet Applications

- Often Client/Server
- Server "daemon"
- Listens on port
  - IANA (Internet Assigned Name Authority) "well-known" ports 0-1023
  - Registered ports: 1024-49151
  - Dynamic/Private 49152-65535
- Start at boot time? Old days inetd, these days systemd



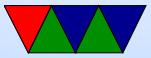
Server Types

- Concurrent – handle multiple connections at time (forks or threads)
- Iterative – handles one connection at a time, rest wait on queue
- Iterative Connectionless – common+trivial, short lived
- Iterative Connection – high latency
- Concurrent Connectionless – when need fast turnaround, low latency DNS, NFS
- Concurrent Connection – widely used. WWW.



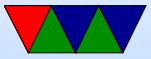
Protocols

- What type of protocol should talk?
- Fixed-length binary?
- Free-form ASCII text?
- 7-bit ASCII vs Unicode?

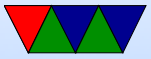


The World Wide Web (history)

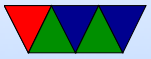
- Before: getting files via cd-rom or ftp (or e-mail/ftp gateways!), search with archie (archive w/o the V, not comic related)
- gopher: university of Minnesota, 1991. search with jughead/veronica
  Why fail? UMN tried to charge license fee, much more restricted file format than html.
- World-Wide-Web: Tim-Berners Lee, CERN, Initial Proposal 1989, first text-based prototype 1991



- Marc Anderson UIUC worked on graphical browser, Mosaic, 1993
- Anderson went on to form Netscape Communications 1994. Webserver software, made Navigator ("mozilla") relatively cheap/free to drive uptake of web servers.
- Microsoft Internet Explorer. Licensed version of Mosaic. 1995 (as add-on to Win95). MS paid percentage royalties to Spyglass Mosaic, so what happened when they gave it away for free?
- Browser wars.
- Netscape bought by AOL in 1998



- By 2000, IE had over 80% due to bundling with windows, famous lawsuit
- Gap between IE6 and IE7 of 5 years (2001 to 2006)
- Netscape released firefox as open source in 2004
- Safari/Webkit browsers based off of KDE browser
- Google Chrome took over the lead around 2012 or so
- Standards fight. ACID test.

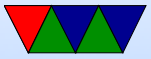


Top Browsers

1996	Mosaic 1.2%	Netscape 77.3%	IE 19%				
2003	IE 94%	Firefox 2%	Safari —	Opera 1%	Navigator 1%		
2010	IE 42%	Firefox 29%	Chrome 11%	Safari 6%			
2017	Chrome 46.5%	Safari 21.5%	IE 10.1%	Firefox 6.3%	Edge 1.9%	Opera 1.3%	Android 1.2%

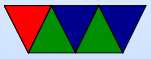
Stats from Wikipedia. EWS for 1996. TheCounter.com for 2003, wikimedia 2010,2017

Other browsers: midori, lynx, links, w3m

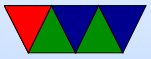


HTML

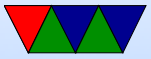
- HTML – hyper text markup language
- Based on SGML (Standard Generalized Markup Language)
- Hypertext (documents that can link to each other) actually proposed by Vannevar Bush in 1945
- Simplest form, just a text file with some extra commands specified in angle brackets, and usually a closing tag with a / in it. Case insensitive (though supposed to use lowercase these days).



- Standards
  - Internet Engineering Task Force (IETF) HTML 2.0 in 1994
    RFC 1866, 1867, 1942, 1980, 2070
  - Since 1996 by the World Wide Web Consortium (W3C)
  - 2000 HTML (ISO/IEC 15445:2000)
  - HTML 4.01 in 1999
  - HTML5 by Web Hypertext Application Technology Working Group in 2014
  - Javascript (ECMAscript) ECMA-262 and ISO/IEC 16262

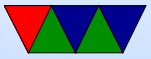


- HTML4 vs HTML5 vs XHTML
- XML extensible markup language, can do things like add new tags on fly



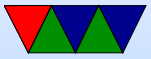
Sample ancient HTML

```
<!DOCTYPE html>
<html>
<head><title>ECE435 Test</title></head>
<body>
<center><h1>ECE435 Test</h1></center>

<hr>

This is a test.
<img src="image.png" alt="alternate text">
```

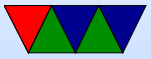


```
<br>Line Break
<!-- Comment -->
<p> Paragraph
<b>Bold</b> <i>Italic</i>

<a href="other.html">A link to another page</a>

</body>
</html>
```

- Tables also easy to do.

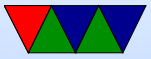


- Early on vendors went crazy with custom tags: Marquee tag, Blink Tag. Frames.
- “view source”
- Originally idea was no formatting, web browser should automatically display simple text in a way to best be displayed on your local machine
  Publishers/graphics designers got a hold of it and that’s where all the pixel perfect positioning stuff came in
- CSS (cascading style sheets), Javascript
- Submitting back to the website, HTML forms

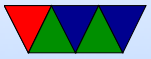


Dynamic Content

- Server Side
  - cgi-bin: Write a program that takes input as environment vars, output as standard out sent to the requesting browser.
    Can write in any program. Typically was things like perl, I often did this in C or even Fortran
  - Dynamic content – SSI (server side includes)
  - Server extensions (such as PHP, modperl, ASP, .NET) more commonly used (with security issues)

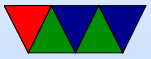


- Client Side
  - Javascript horror. Client side, code runs on your computer rather than on the server.



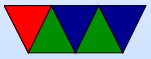
www on the client side

- A URL (uniform resource locator) specifies the document you want
  `http://web.eece.maine.edu/~vweaver/`
- Browser parses URL
- It looks up the address of web.eece.maine.edu via DNS
- DNS reports 192.168.8.99
- Browser makes TCP connection to port 80 (default) of 192.168.8.99
  how do you specify other port? web.eece.maine.edu:8080



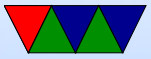
why would you want to?
security

- The client requests the file vweaver/index.html (index.html is the default). Often there's a server root, and special handling for user dirs (with the tilde) that are often in user home dirs, as in this case
- The server returns this file
- The TCP connection is closed
- The browser displays the text
- The browser might have to fetch any images linked to by the document



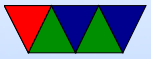
Non-HTML

- You can serve up any kind of binary file. Often have associated MIME-type like with e-mail
- Browser also often has built in support
- GIF (trouble due to patents), PNG. SVG?
- MP3 music? Movies?
- Plugins. Flash? Java? PDF?



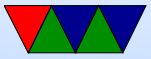
Web-servers

- famously netcraft had a list (meme netcraft reports BSD is dying)
- NCSA was first popular one
- Apache ("a patchy" version of NCSA) took over
- Microsoft IIS
- Other companies like Sun/Netscape/SGI
- nginx ("engine-x")
  Designed to be faster than apache (Apache has lots of RAM overhead)



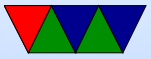
Solve c10k problem (having 10k active socket connections at once)

- lighthttpd ("lightly")

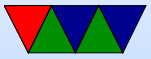


simple web server

- Listen on port 80
- Accept a TCP connection
- Get name of file requested
- Read file from disk
- Return to client
- Release TCP connection
- How do we make this faster?
  - Cache things so not limited by disk
    (also cache in browser so not limited by network)



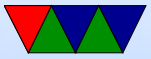
- Make server multithreaded



URLs

- URI (uniform resource identifier)
- URL (uniform resource locator) subset of URI, includes info on how to find the resource (protocol and server)

- URN (uniform resource name) asks for a document but from anywhere. I.e. give it something like an ISBN and returns the book
- `scheme:[//[user:password@]host[:port]][/]path[?query][#fragment]`
  `: / ? # [ ] @ reserved, must encode if use %3f`
  `query key1=value1&key2=value2 or key1=value1;key2=value2`
- protocol: http, ftp, file, news, gopher, mailto, telnet



Continued Next Lecture