




COMP2720: Automating Tools for New Media

Manipulating the network



Networks: Two or more computers communicating

- Networks are formed when distinct computers communicate via some mechanism.
 - Rarely does the communication take the place of 0/1 voltages over a wire.
 - Too hard to make work over distances.
 - More common is the use of frequencies (maybe in the sound range, but maybe not).
 - For example, a *modem* (modulator-demodulator) takes your computer's 0's and 1's and translates them into sound frequencies that can pass over the sound wire and be decoded on the other side.



Networks, networks everywhere

- If you're driving a newer car, you probably have a network in there.
 - There are lots of computers in your car (controlling air flow, gas flow; making the air bag work) and they communicate.
- You can have a network in your own home, or even on an airplane.
 - Can use radio signals for communication (wireless).
 - Or can string a cable between two computers.

Networks have layers

- Networks have several *layers* to them.
 - At the bottom level is the physical substrate.
 - What are the signals being passed on?
 - Levels higher determine how data is encoded.
 - Do we use sound frequencies to represent 0's and 1's, or radio waves?
 - Do we send a bit at a time? A byte at a time? Or in packets larger than that?
 - Levels even higher determine the protocol of communication.
 - How do I address a particular computer I want to talk to? Or many computers?
 - How do I tell a computer that I want to talk to it? That I'm starting to send it data? What it's supposed to do with it? When we're done?



Internet: A collection of networks

- The *Internet* is a network of networks.
- If you put a device in your home so that your computers can talk to one another, you have a network.
 - A wireless base station, or an Ethernet router, perhaps.
 - You can probably reach printers on your network, or copy files between computers.
- If you now connect your network (through an Internet Service Provider (ISP)) to the global Internet, your network becomes yet another part of the whole Internet.



Internet is based on agreements on encodings

- The Internet is built on a set of agreements about:
 - How computers will be addressed.
 - A set of four numbers (each one byte now, soon to grow) separated by periods, e.g., 10.1.0.5, or 150.203.99.8 (www.anu.edu.au)
 - A way of associating domain names with these numbers, like www.anu.edu.au (which really is a name that resolves to a set of four numbers), using domain name servers.
 - How computers will communicate.
 - That data will be put into packets with various pieces in them.
 - That computers will format their data and talk to one another using TCP/IP (*a protocol*).
 - How packets are routed around the network to find their destination.



The Internet is not new

- The Internet agreements date back 40 years.
- It was originally set up for military applications.
 - One of the features of the Internet is that packets find their destination even if part of the Internet is destroyed, damaged, or subject to censorship.
- The Internet originally had only a handful of computers (nodes) on it, but it has grown dramatically in recent years.

Protocols on the Internet

- But all that just lets us pass data back and forth.
 - What does the data say?
 - What does the data do?
- One of the first applications placed on top of the Internet was *electronic mail*.
 - The mail protocols have evolved over time to their standard forms today.
- The *file transfer protocol* (FTP) allows computers to move files between each other.
 - It defines what one side says to the other when copying a file over (e.g., “STO filename”) and how the file will be encoded.
 - Nowadays: Replaced by SSH/SCP/SFTP (secure shell, copy)



Then there's the Web

- The Web dates only back to the 1990's (for a little history see: <http://www.w3.org/History.html>)
- The Web is (again) a set of agreements
 - On how to refer to everything on the Internet: The URL (Uniform Resource Locator)
 - On how to create documents that refer to things all over the Internet: HTTP (HyperText Transfer Protocol)
 - On how those documents will be formatted: Using HTML (HyperText Markup Language)



HyperText: Non-linear text

- Hypertext is a term invented by Ted Nelson in the 1960's.
 - It refers to text that is non-linear, which the computer makes possible.
 - You're familiar with this on the Web:
 - Read a little on a page,
 - click,
 - continue reading on some other page anywhere on the Internet,
 - go back,
 - etc.



The point of the Web is Hypertext

- *Tim Berners-Lee* (one of the Web's designers) wanted a way to create readable documents that could reference material anywhere on the Internet in a hypertext format.
- There are technical flaws in what he did:
 - For example, the phenomena of “dead links” couldn’t happen in other hypertext systems before the Web.
- But it worked and has become a worldwide standard.



HyperText Transfer Protocol (HTTP)

- HTTP defines a *very* simple protocol for how to exchange information between computers.
- It defines the pieces of the communication.
 - What resource do you want?
 - Where is it?
 - Okay, here's the type of thing it is (JPEG, HTML, whatever), and here it is.
- And the words that the computers say to one another:
 - Complex things like “GET”, “PUT” and “OK”

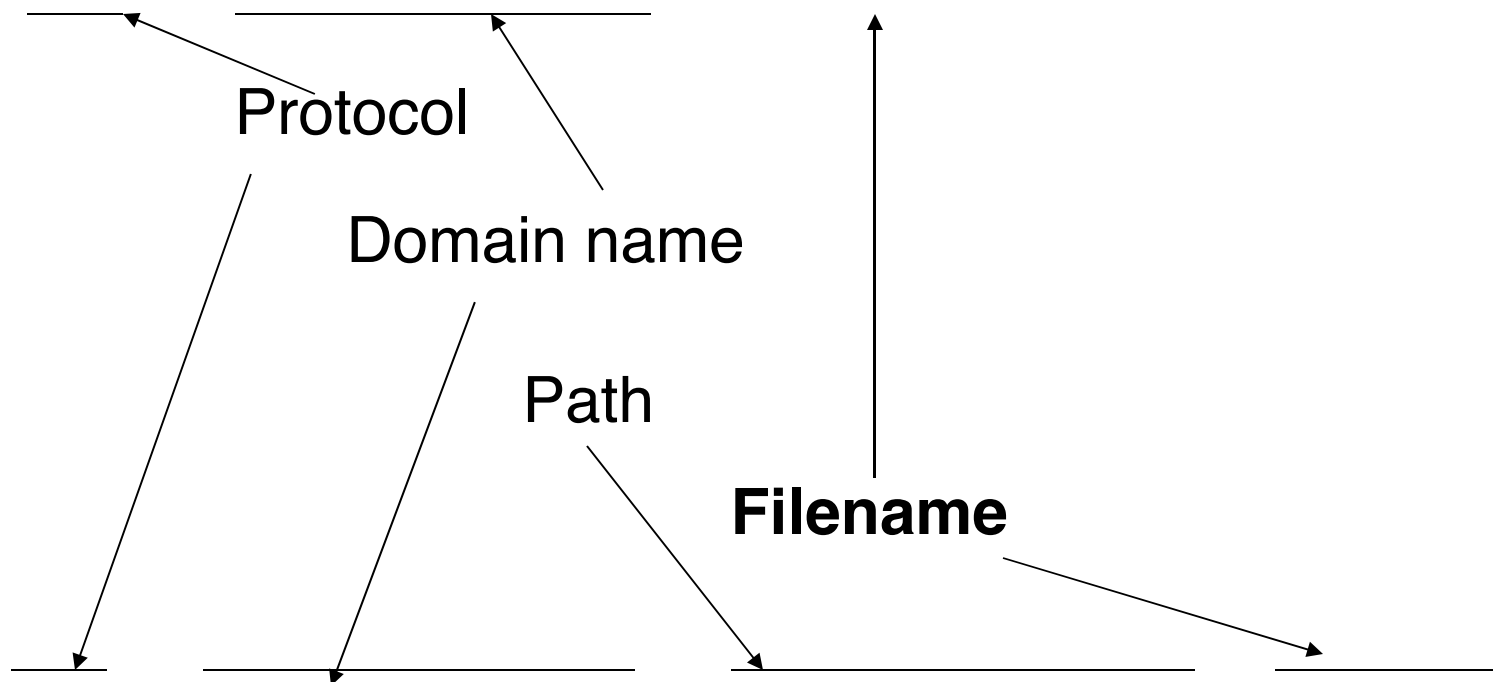


Uniform Resource Locators (URL)

- URLs allow us to reference any material anywhere on the Internet.
 - Strictly speaking, any computer providing a protocol accessible via URL.
 - Just putting your computer on the Internet does not mean that all of your files are accessible to everyone on the Internet.
- URLs have four parts:
 - The *protocol* to use to reach this resource,
 - the *domain* name of the computer where the resource is,
 - the *path* on the computer to the resource,
 - and the *name* of the resource.

Example URLs

http://www.cc.gatech.edu/index.html



ftp://cleon.cc.gatech.edu/pub/guzdial/papers/sigcse2003.pdf

What if there is no path?

- *Web servers* (programs that understand the HTTP protocol) typically have a special directory that they serve from.
 - Files in that special directory are directly referable without specifying a path.
- Sub-directories within the server directory can be accessed in terms of a path.
 - But always starting from the server directory, so not everything on your computer is always accessible.

A browser is a client

- Your Web browser is called a *client* accessing a Web *server*.
- Programs like **Mozilla, Firefox, Safari, Opera** or **IE** understand a lot about Internet protocols.
 - They know how to interpret HTML and display it graphically.
 - If the HTML references other resources, like JPEG pictures, the client fetches them and displays them where appropriate.
 - Your client knows the details of the HTTP (and maybe FTP, mailto, gopher...) protocols so that it can request the resources you request.

You don't need a browser to use the Internet

- Your mail program also understands some Internet protocols.
- Python (and other languages) have modules that allow you to use these protocols.
 - In Python, we can read any URL as if it was a file.

```
>>> import urllib
>>> connection =
    urllib.urlopen("http://www.bom.gov.au/weather/act")
>>> weather = connection.read()
>>> connection.close()
```



Storing a file is different

- It is possible to send information to a Web server.
 - That's how search functions, forms, etc. work.
- But it's more complicated than just reading, and it requires an accepting program on the Web server.
- It isn't hard to send information to an FTP server, though.
- But first, let's make our temperature-finding function useful by directly reading the Weather page...

Getting the temperature live

```
import urllib

def findTemperatureLive():
    # Get the weather page

    connection=urllib.urlopen("http://www.bom.gov.au/products/IDN10035.shtml")
    weather = connection.read()
    connection.close()
    #weatherFile =
    getMediaPath("bom_act_forecast.htm")
    #file = open(weatherFile,"rt")
    #weather = file.read()
    #file.close()

    # Find the Temperature
    curloc = weather.find("Max ")
    if curloc <> -1:
        # Now, find new line "\n" following the temp
        tempstart = curloc+4 # Position after "Max "
        tempend = weather.find("\n", tempstart)
        print "Maximum
        temperature:",weather[tempstart+1:tempend]

    if curloc == -1:
        print "They must have changed the page
        format -- can't find the temp"
```



Running it

```
>>> findTemperatureLive()
```

```
Maximum temperature: 18
```



The Interactive Web

- The first use of HTTP was just to send around static pages and images (and sounds and...)
- Later extensions allowed for users providing input to the server (such as for doing searches).
 - Originally, this was just “CGI” (*Common Gateway Interface*) scripts.
 - Later, servlets and applets and PHP and AJAX and so on ...



Interactive Web requires programs to generate HTML

- Typically, a Web server will have some directory specified “special.”
 - Files referenced there aren’t just returned to the client.
 - Instead, the files are executed and the result is returned to the input.
 - There’s even a mechanism where the client can provide input to the executed files, e.g., a search string.
- Those special files would generate HTML.
 - The generated HTML might be based on up-the-minute information like stock quotes and temperature sensors and database queries.
- Thus, to have an interactive Web, we need to write programs that write HTML.



Python is a popular program for writing CGI scripts

- Python is often used for writing these kinds of scripts.
 - Python includes libraries for handling input via Web forms
 - As we can see, it already has libraries for handling networking.



HTML is just text in a file

- We enter our text and our tags in just a plain old ordinary text file.
- Use an extension of “.html” (“.htm” if your computer only allows three characters) to indicate HTML.
- JES works just fine for editing and saving HTML files.
 - Just don't try to load them!

The Simplest Web Page

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transition//EN"  
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
```

```
<head>
```

```
<title>The Simplest Possible Web Page</title>
```

```
</head>
```

```
<body>
```

```
<h1>A Simple Heading</h1>
```

```
<p>This is a paragraph in the simplest  
possible Web page.</p>
```

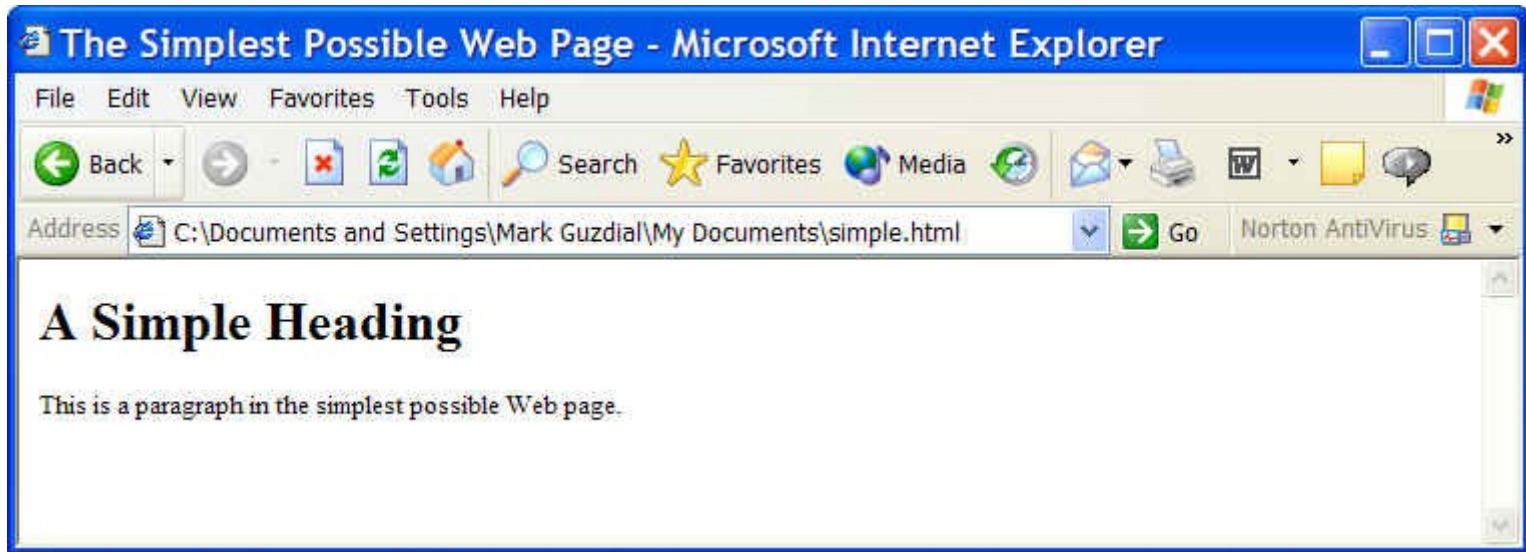
```
</body>
```

```
</html>
```

Yes, that whole
thing is the
DOCTYPE

No, it doesn't matter
where you put
returns, or extra
spaces

What it looks like in a browser





Is this a Web page?

- Of course, it is!
- The only difference between this page and one on the Web is that the one on the Web (a) has been uploaded to a Web server and (b) placed in a directory that the Web server can access.
- We (you) will do this with the portfolio!



Best way to learn HTML: Look at pages!

- View source all the time, especially when there's something new and cool that you've never seen before.
- There are lots of good on-line tutorials, especially by David Raggett (a leading developer at W3C), see <http://www.w3.org/MarkUp/Guide>
- There's even a few good books.
- Also, chapter 11 in textbook!



HTML is *not* a programming language

- Using HTML is called “coding” and it is about getting your codes right.
- But it’s not about coding programs.
- HTML is **not** programming language; it has no
 - Loops
 - ifs (conditionals)
 - Variables (for carrying values)
 - Data types (CL can do without, but very useful)
 - Ability to read and write files
- These features are necessary for carrying out computation
- Bottom line: HTML does not communicate process!

We can use programs to *generate* HTML

```
def makePage():
    file=open("generated.html","wt")
    file.write("""<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
    Transition//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>The Simplest Possible Web Page</title>
</head>
<body>
<h1>A Simple Heading</h1>
<p>Some simple text.</p>
</body>
</html>""")
    file.close()
```

A Generated Page

```
<!DOCTYPE HTML PUBLIC "-//  
  //W3C//DTD HTML 4.01  
  Transition//EN"  
  "http://www.w3.org/TR/html4/loose  
  .dtd">  
<html>  
<head>  
<title>The Simplest Possible Web  
  Page</title>  
</head>  
<body>  
<h1>A Simple Heading</h1>  
<p>Some simple text.</p>  
</body>  
</html>
```





Tailoring the output

- That works, but that's boring.
- Why would you want to just put in a file what you can put in via a text editor?
 - Why you write a program: Replicability, communicating process... and tailorability!
- Let's make a homepage creator!
 - A home page should have your name, and at least one of your interests.

A homepage editor

```
def makeHomePage(name, interest):
    file=open("homepage.html","wt")
    file.write("""<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
    Transition//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>"""+name+""""s Home Page</title>
</head>
<body>
<h1>Welcome to """+name+""""s Home Page</h1>
<p>Hi! I am """+name+"""". This is my home page!
I am interested in """+interest+""""</p>
</body>
</html>""")
    file.close()
```

makeHomePage("Alex" , "Quantum computers")

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
 4.01 Transition//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Alex's Home Page</title>
</head>
<body>
<h1>Welcome to Alex's Home Page</h1>
<p>Hi! I am Alex. This is my home page!
I am interested in Quantum computers</p>
</body>
</html>
```



Works...but painful

- Try to change the home page code.
 - Maybe insert a picture, or another line about interests, or a favorite URL.
- It's hard, isn't it?
 - It's hard to track down all those quotes, insert the +'s and variables in the right place, and it's one loooooong string.
- Can we make it easier to work with?
 - Sure! Let's use more functions!

New Homepage Program

```
def makeHomePage(name, interest):
    file=open("homepage.html","wt")
    file.write(doctype())
    file.write(title(name+"s Home Page"))
    file.write(body("""
<h1>Welcome to """+name+""""s Home Page</h1>
<p>Hi! I am """+name+"""". This is my home page!
I am interested in """+interest+""""</p>"""))
    file.close()
```

Up here on top is where we deal with the parts that we might likely change.

```
def doctype():
    return '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
    Transition//EN" "http://www.w3.org/TR/html4/loose.dtd">'
```

Bury the yucky doctype here — may we never deal with it again!

```
def title(titlestring):
    return "<html><head><title>"+titlestring+"</title></head>"
```

```
def body(bodystring):
    return "<body>"+bodystring+"</body></html>"
```

Here are more details we don't really want to deal with.



What to do now

- Read section 10.5, 11.1 and 11.2 in text book
- Next week:
 - Two VPython lectures in week 8 (first week after break)
 - Lab 4 is in week 8!
- Module on Visual Python (3D animation): one lecture will be on VPython itself, and one (by Hugh Fisher) on `py3d` module.
- Work on portfolio over the semester break!
- Start looking into Assignment 2!
- Catch up on lab work!