Introduction to Web Technologies

Internet History

- 1969 – Defense Advanced Research Project Agency (DARPA) creates ARPANET
  - Internet project creates protocols (TCP/IP)
    - Transmission Control Protocol (TCP)
    - Internet Protocol (IP)
    - Allows computers to talk over networks
    - Creates the “Internet” – 4 nodes
  - Links together DARPA supported sites
- 1979 – USENET started
  - Interconnection of Computer Science Departments
    - Duke and UNC-Chapel Hill
  - First emoticon created
    - -) (tongue in cheek)
    - 1982 :-) and :- (get added

Internet History Continued

- 1985 – first domain name – Symbolics.com
- 1986 – NSFNET
  - government support for major Internet backbone
- 1988 – Internet Worm affects 6,000 of 60,000 Internet sites
  - 30 seconds on NBC evening news
  - Son of National Security Agency Chief Scientist

Web History

- 1991 – World Wide Web released by CERN
  - First Web Server
    - WhatYouSeeIsWhatYouGet (WYSYWIG)
  - First Web Site
    - (Stanford Linear Accelerator)
- 1992 – 50 Web servers
- 1993 – first Internet Worms/Spiders/Wanderers/Robots
  - First search engines
  - Mosaic – first graphical browser for Web
  - 341,000% growth rate in WWW

Web History Continued

- Yahoo – Yet Another Hierarchical Officious Oracle
  - 2 Ph.D. Students from Stanford
- Peaks about Dec. 2001 at 40 million web servers
- Keys to success:
  - Client-server architecture
    - Users all over world can run programs on my computer
    - Don’t need to ship software, just publish the url
    - Don’t have to create a GUI – just create a web page
    - Browser runs on client, displays web page
      - Sends messages to server
      - Receives and displays answers

URL- Universal Resource Locator

- Unique address for a resource on the Internet
- Scheme://hostname[:port]/path/filename
  - Scheme:
    - http: HyperText Transfer Protocol
    - ftp: File Transfer Protocol
    - mailto: send email
    - News: newsnet news

HTML – HyperText Markup Language

- Create textual description of appearance of documents
- Document Structure

```
<HTML>
    <HEAD> <TITLE> My Lecture </TITLE>
    </HEAD>
    <BODY> I have to say something.
    </BODY>
</HTML>
```

Meta Document Information

- “meta” - information about
- Meta document - information about the document
- Meta data – information about the data
- Meta tags

```
<META NAME="name"
HTTP-EQUIV="FieldName"
   CONTENT="value" >
```

MetaTags

```
<META HTTP-EQUIV="Refresh"
  CONTENT="2;URL=http:nextdoc.html">
```

- *in 2 sec. redirect browser to nextdoc.html.*

```
<META NAME="keywords"
  CONTENT="Web, HTML, tags">
```

- *Set keywords for the document, used by search engines to index your document*

Basic HTML Tags

```
<b> </b>   - bold
<ul> - unnumbered list
       <li> first thing
       <li> second thing
</ul>
<ol> </ol> - ordered (numbered) list
<p> </p> - paragraph begin and end
<br> - line break
<pre> </pre> - preformatted text
<h1> </h1> - big heading (also h2 .. h6)
<i> </i> - italics
<center> </center>  - center
<!-- comment -->
```

A Minimal HTML Document

```
<html>
    <head>
        <TITLE>A Simple HTML Example</TITLE>
    </head>
    <body>
        <H1>HTML is Easy To Learn</H1>
        <P>Welcome to the world of HTML. This is the first
              paragraph. While short, it is still a paragraph!
        </P>
        <P>And this is the second paragraph.
        </P>
    </body>
</html>
```

Special Characters and Links

- Special Characters
  - unlike rest of HTML, case sensitive

&lt; - less than <
&gt; - greater than >
&nbsp; - non-blocking space

Linking to Documents and Images

- Links to Documents and Images
  - The displayed text is called the “anchor”

```
<A HREF=“Presentation.ppt"> Document in same
                                    directory</A>

<A HREF=“http://www.csce.uark.edu/~sgauch">
                    Full path to a page</A>

<img src="images/sgauch.jpg" WIDTH=190
  HEIGHT= 200>
```

Tables - Syntax

```
<TABLE>
  <TR>
        <TD> Row 1, Cell 1 </TD>
        <TD> Row 1, Cell 2 </TD>
  </TR> <!-- end of first row definition -->
   <TR> <!-- start of last row definition -->
        <TD> Row 2, Cell 1 </TD>
        <TD> Row 2, Cell 2 </TD>
   </TR> <!-- end of last row definition -->
</TABLE> <!-- end of table definition -->
```

Tables – Output

Row 1, Cell 1	Row 1, Cell2
Row 2, Cell 1	Row 2, Cell 2

Creating HTML Files

- Text Editor
  - vi, notepad, emacs, pico, …
  - Don’t need to learn a new editor
  - Have control over output
  - Need to know syntax of HTML
- HTML Authoring Tools
  - Allaire, Netscape, DreamWeaver
  - WYSIWYG
- General Purpose WSYIWYG
  - Word
  - Creates enormous, ugly HTML code
  - Badly formatted, lots of unnecessary tags

CGI – Common Gateway Interface

- An html web page is static (unchanging)
  - Text document sent from server to browser
- CGI program creates dynamic information
  - Program is executed upon demand
  - Generates fresh content for each request

CGI Overview

- Developer creates an HTML page with a <FORM> on it
  - Specifies the name of the program
  - Names some variables that can hold data
- User enters information into the Web page (fills in the variables in the <FORM> and clicks <SUBMIT>
- Browser sends the information to the URL

CGI Overview continued

- Server unpacks the HTTP message
  - Finds the name of the program to call
  - Finds the data
- Server calls the program and passes in the data
- Program generates output and writes it to “standard out” (the screen, usually)
- Server takes the output and passes it along to the browser
- Browser displays the output on the user’s screen

Forms

- Forms are part of regular HTML documents
- There may be more than one form in a document
- Forms may not be nested

<FORM ACTION="url"> … </FORM>

INPUT

- Forms receive input from the user
- Each input area has its own name and type of input it may receive
- Forms may receive input from
  - Text
  - Radio
  - Checkbox
  - Submit
  - Reset
  - Password

Submitting Information via a FORM

- When SUBMIT is pushed, the contents of the form get sent to the server in the form:

programname?var1=value1&var2=value2

- You may send the data via POST or GET
  - You choose this when you write the HTML page with the FORM in it

POST and GET

- POST
  - cgi program reads from stdin (i.e., the keyboard)
  - No limit on the amount of data sent
- GET
  - CGI program reads from an environment variable (QUERY_STRING)
  - Limit on length of data sent (1,000? Characters?)
- Recommend that you use POST

A Simple Perl Program tiny.pl

```
#!/usr/bin/perl
#Prints “Hello Susan”
    use strict;                                      # syntax checking
    use warnings;                                     # runtime checking

    my $username = ‘Susan’;                           # "declare" variable
    print "Hello, $username.\n";                      # print out greeting
```

A Simple CGI Program tinyPL.cgi

```
#!/usr/bin/perl
#Prints static Web page “Hello Susan”

    use strict;                               # syntax checking
    use warnings;                             # runtime checking

    use CGI qw(:standard);                    # use the CGI library
    my $username = ‘Susan’;                   # "declare" the variable

    print header();                           # output http, <HTML>
    start_html(“First CGI program”);         # add a title
    print "Hello, $username.\n";
    print end_html();                        # output </HTML> etc
```

Program Output

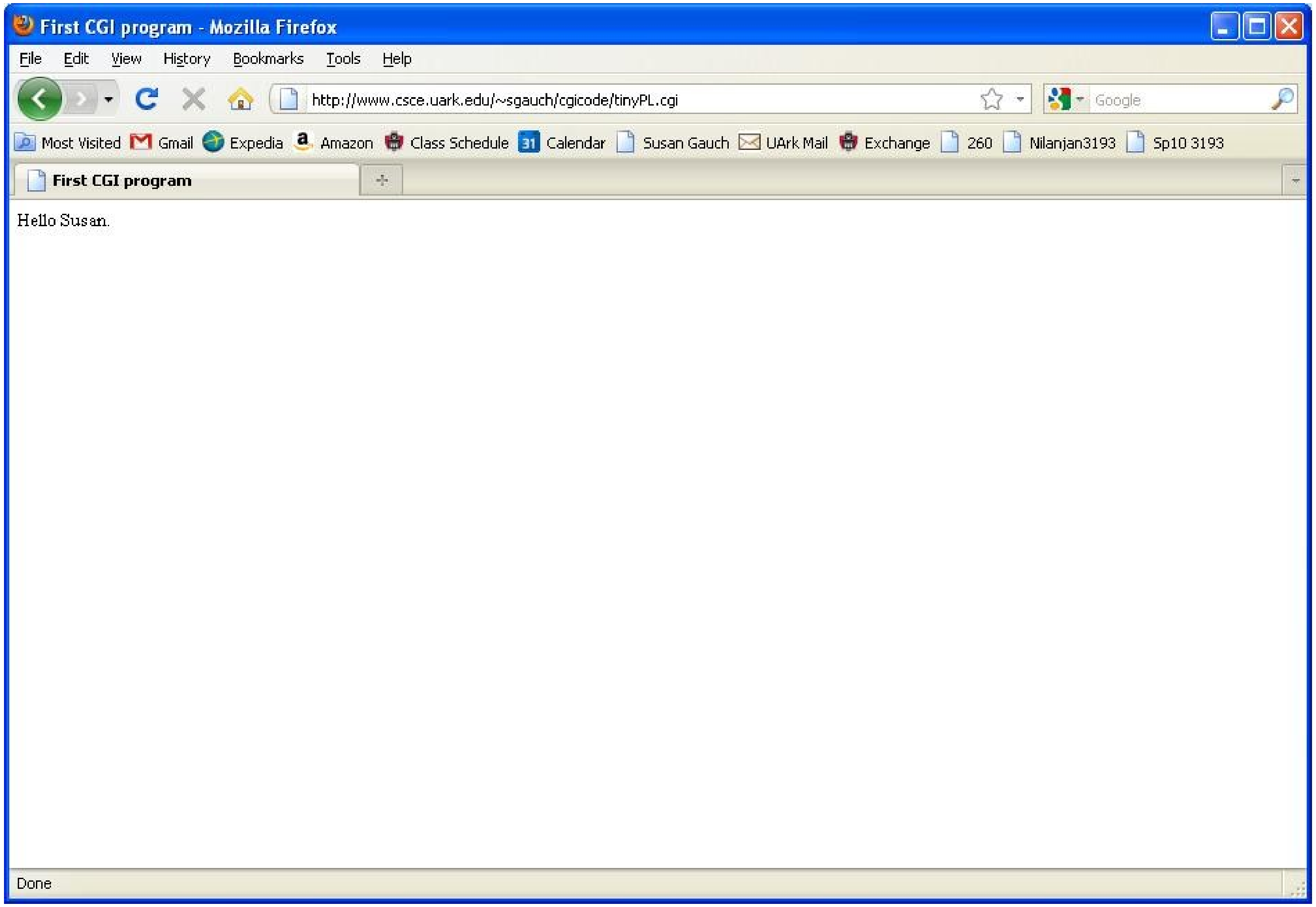
```
Content-Type: text/html; charset=ISO-8859-1
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US"
          xml:lang="en-US">
<head>
<title>First CGI program</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-
          8859-1" />
</head>
<body>
Hello Susan.
</body>
</html>
```

Call the program

In browser bar, type

http://www.csce.uark.edu/~sgauch/cgicode/tinyPL.cgi

What it looks like


First CGI program - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.csce.uark.edu/~sgauch/cgicode/tinyPL.cgi
Google
Most Visited Gmail Expedia Amazon Class Schedule Calendar Susan Gauch UArk Mail Exchange 260 Nilanjan3193 Sp10 3193
First CGI program
Done

```
Hello Susan.
```

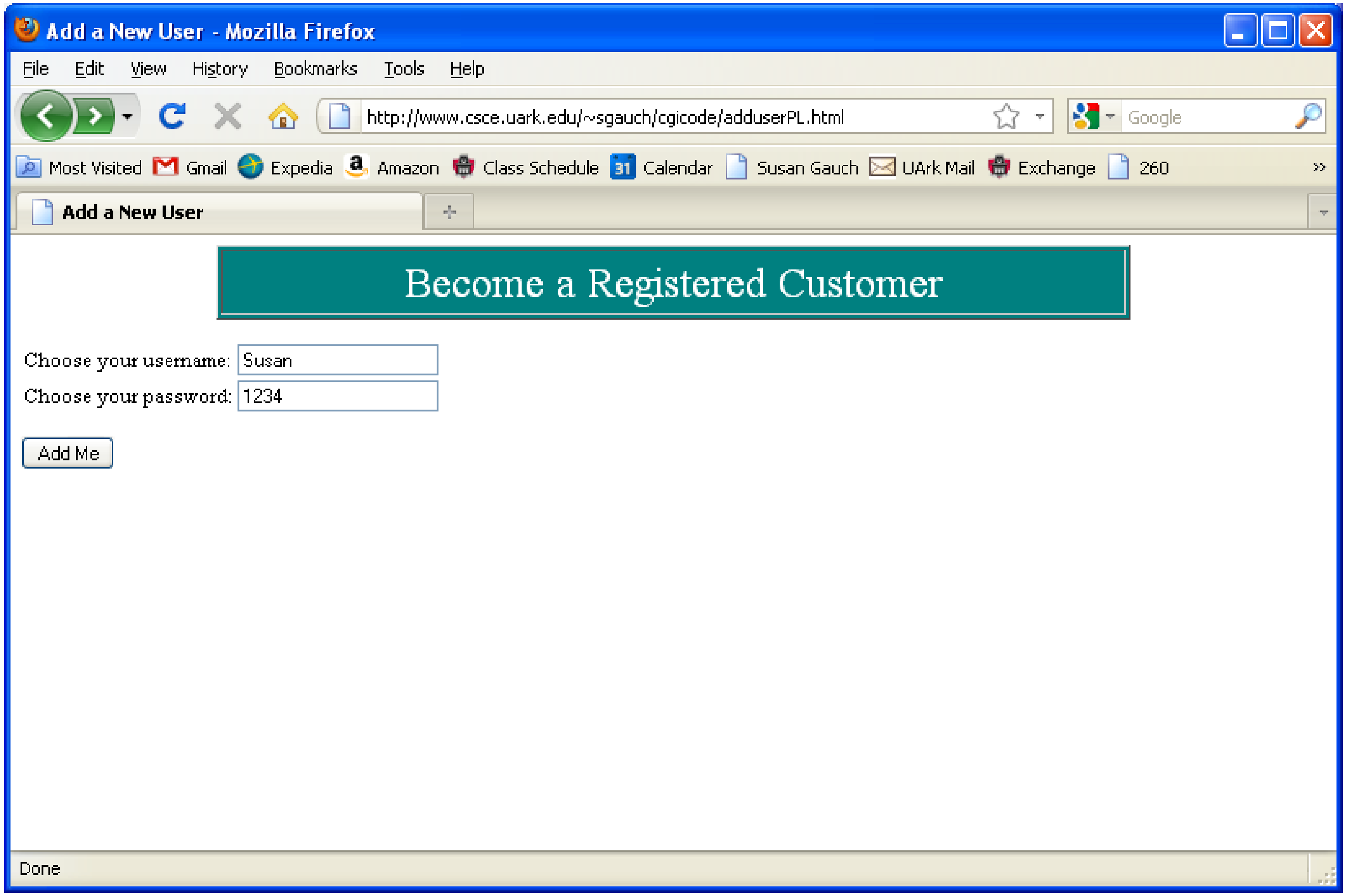
Passing Parameters to a CGI program adduserPL.cgi

```
#!/usr/bin/perl
use strict;
use warnings;
use CGI qw(:standard);                          #use the CGI library

  my $cgiform = new CGI;                        #Create a CGI object

  #Extract username and password from the form-passed parameters
  my $username = $cgiform->param("username");
  my $password = $cgiform->param("password");

 #Output the username and password
  print header();
  print start_html("Add a User");
  print "<h2>Username: $username</h2>\n";
  print "<h2>Password: $password</h2>\n";
  print end_html();
```

Calling it from a Web Page

```
<html><head><title> Add a New User </title></head>
<body bgcolor=white>
<FORM ACTION="adduserPL.cgi“METHOD="POST">
   <center><table width=70% cellpadding=5 border=1 bgcolor=#008080>
         <tr>
               <td align=center> <font size=+3 style=times color=white>
                     Become a Registered Customer</font><br>
               </td>
         </tr>
   </table></center><p>
   <table border="0">
      <tr>
            <td>Choose your username: </td>
            <td><INPUT TYPE="text" SIZE="20" NAME="username"></td>
      </tr>
      <tr>
            <td>Choose your password: </td>
            <td><INPUT TYPE="text" SIZE="20" NAME="password"></td>
      </tr>
   </table>
   <p><input type="submit" value="Add Me">
</form></body></html>
```

What the Web Page Looks Like



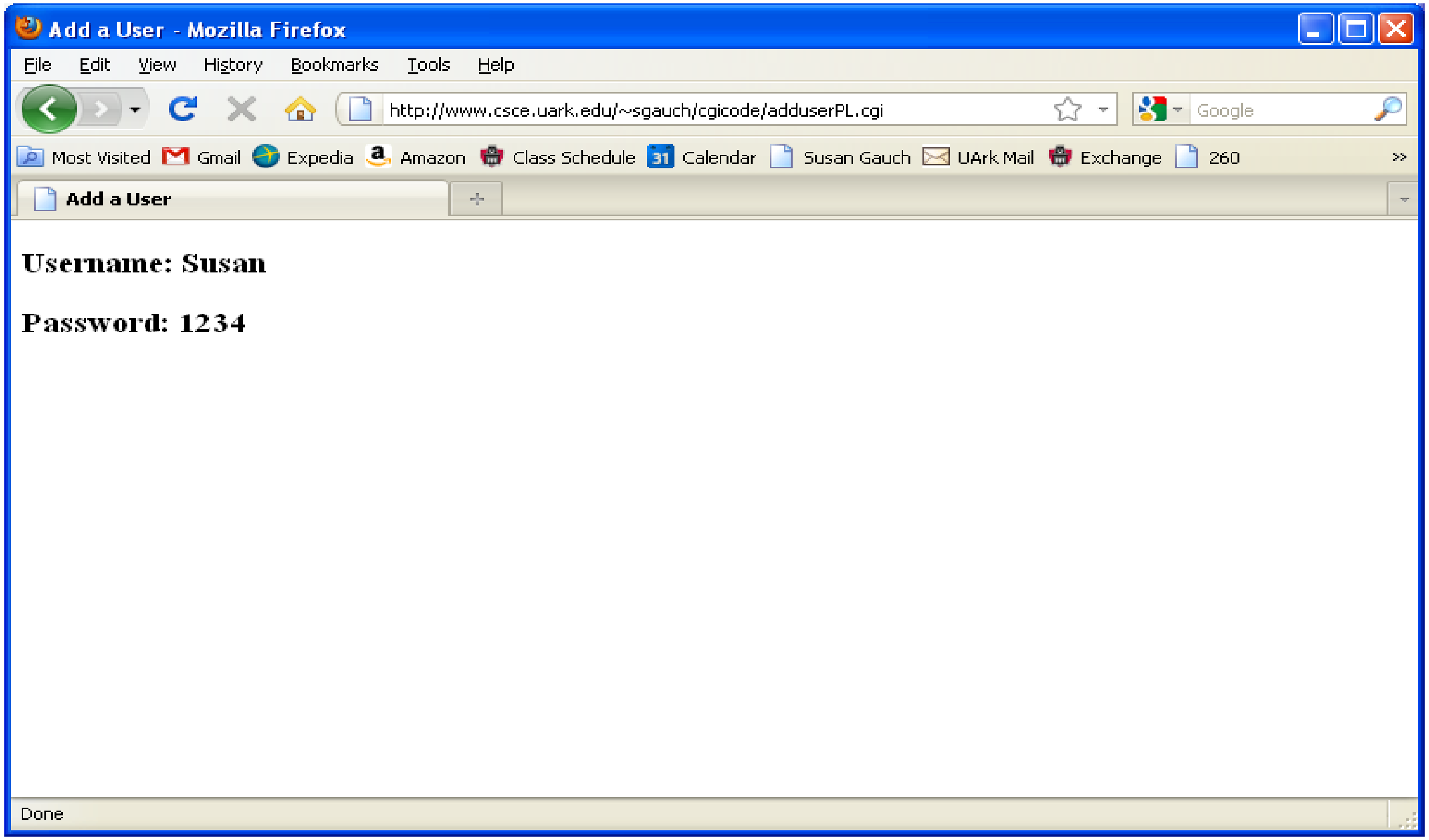
Become a Registered Customer

Choose your username: Susan

Choose your password: 1234

Add a New User - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.csce.uark.edu/~sgauch/cgicode/adduserPL.html
Google
Most Visited Gmail Expedia Amazon Class Schedule Calendar Susan Gauch UArk Mail Exchange 260
Add a New User
Add Me
Done

What the results are


Add a User - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.csce.uark.edu/~sgauch/cgicode/adduserPL.cgi
Google
Most Visited Gmail Expedia Amazon Class Schedule Calendar Susan Gauch UArk Mail Exchange 260
Add a User
Done

Username: Susan

Password: 1234

Creating CGI Programs

First step – Create and view a static html page

- Create your public_html directory
- Set permissions so the world can read & execute it
- Create a simple html page (index.htm)
- Check that you can view it from
- http://www.csce.uark.edu/~yourusername/

Second step – Test existing cgi program

- Copy tiny.html and tinyPL.cgi to your public_html directory
- Test that they work
- Copy adduserPL.html and adduserPL.cgi to your public_html directory
- Test that they work

Creating CGI Programs

Third step – Check that your program produces valid HTML output

- Create a program that runs from the command line
- Put the program in your public_html/ (cgi-bin directory optional)
- Name the program ending in .cgi
- Set the variables in the program
  - e.g., $username = “sgauch”;
- Save the output to a file
  - perl myprog.cgi > output.html
- Remove everything before <html> in nano
- View the file in a browser

Creating CGI programs

- Fourth step – Check that the permissions are set so others besides you can execute the program
  - Have a friend login and run your program from their directory
    - e.g., perl /users/myusername/.public_html/cgi-bin/myprogram.cgi > output
- Fifth step – Create a web page to call a simple program
  - Design a form that calls a dummy cgi
  - That cgi should just print out “hello world”
  - Put the form in your .public_html directory
  - View the form in the browser
  - Click submit
  - Check that you see “hello world”

Creating CGI programs continued

- Sixth step – Create a web page to call a simple version of your final program that confirms parameters are passed
  - In your perl program, comment out all parts of the program
  - Just print the parameters to confirm you're getting them
  - Call this program from the form
- Seventh step – Run the full program
  - Remove comments and test the REAL progrram

Debugging CGI programs

- Permissions problems
  - inadequate permissions
  - Test this by having someone besides yourself execute the code
  - Do and ls –l on the directory structure
    - It should be drwxr-wr-x on all directories
    - Chmod 755 my directory
    - Chmod 644 for your perl program
- Path problems
  - Test this by creating and calling a really simple cgi program
- Invalid HTML produced
  - Call this and save output to a file
  - View file in a browser

Common Problems

#!/usr/bin/perl must be the first line in the file

- Even before any other comments

Remember to call the print header() function

- It must occur before any other print statements

On some systems, the filename must end .cgi not .pl

On some systems, the executables must be in public_html/cgi-bin directory

Useful Links

- www.sergey.com/web_course/
  - JumpStart to the Web Technologies
- http://www.isoc.org/internet/history/
  - History of the Internet and Web
- http://www.w3schools.com/html/
  - A thorough description of HTML tags
- http://www.cgi101.com/book/
  - A good tutorial on CGI programming