Week 2

HTTP
Connection Oriented vs Connectionless Protocols
Stateless vs Statefull Protocols
Web Servers - Basic Functions
HTML and CGI
Creating Forms (pages 52 - 61)
Introduction to Perl (pages 187 - 242)

HTTP
• HyperText Transfer Protocol: it is the protocol that client and servers use on the WWW
• It runs on top of TCP/IP
• A simple, but fast protocol by which requests are made by clients and responded by servers:
  – client connects to the Web Server on the host specified in URL
  – a document is requested
  – a server program that listens for these requests sends the document
  – the connection is closed
HTTP

- HTTP is a connection-less protocol (as opposed to a connection-oriented protocols). For each HTTP request:
  - the client connects to the server
  - it makes a request
  - it gets the response
  - it disconnects

- In connection-oriented protocols, the connection is maintained after the first response is satisfied

Connection-oriented protocols

- Maintain the connection after a request is serviced, to be ready to service another request.
  Clients:
  - connect to the server
  - make a request
  - get the response
  - maintain the connection until it is cut off (typically by the client)
- FTP (File Transfer Protocol): the connection is maintained after a file is downloaded
Connection-oriented protocols

- With these type of protocols, maintaining the connections consume system resources, since there could be many ‘alive’ ones
- Servers can get bogged down
- They are somewhat more efficient, since after the first connection there is no need to establish further connections
- It is geared to a low, controlled number of transactions

Connectionless protocols

- The connection is not maintained, so system resources are not used after the transaction is completed
- They can serve a high number of transactions with very low overhead
- However, when the same client requests additional data, a new connection must be established
- They are indicated when a high, unknown number of transactions is expected
Stateless vs. stateful protocols

- Stateful protocols keep state information (values of variables, processes running) after a transaction has been processed
- With stateless protocols, no state information is maintained
- Stateless protocols are simpler, lighter, have less overhead
- The problem is when you need to maintain state information

Web Servers - Basic Functions

Constantly listening to HTTP requests at either the default port 80, or any other user assigned port.

Retrieving the requested documents and the embedded elements, which may include HTML, audio and image files.
Web Servers - Basic Functions
cont’d

Processing special directives embedded in a requested HTML (usually with extension.shtml) known as Server Side Includes (SSI).

Invoking an executable code or script such as CGI script, Active server page, Java servlet or remote objects (RMI or CORBA)

HTML and CGI

• The HTML protocol also supports a way of communicating between a client and a server by executing programs: the Common Gateway Interface

• When a HTTP server receives a request invoking a program, it runs the program (usually called a script) and sends the request back to the requesting browser
HTML and CGI

• Since HTML is fixed, there is a need for a mechanism by which we can send information to the server to be processed by the script: this is called a form.
• A form is a HTML document that points to a CGI script. When the script is invoked (by pressing a button on the form, for example), the data included into the form is submitted to the server.

HTML and CGI

• The server then runs the CGI script and passes the data to it.
• Typically the script analyses the data, operates with it and prints something that is returned to the browser for the client to see.
• There are two methods to pass information from a form to a script:
  – GET method
  – POST method
GET method

- With this method, the data is sent and stored into environment variables:
  - REQUEST_METHOD stores the method (GET or POST)
  - QUERY_STRING stores the data
- This approach is size-limited
- It is used mostly when data is appended to the URL:
  http://.. /pizza.pl?favorite=australia

POST method

- The data sent is not stored in an environment variable, but in standard input
- There is no limit to the length of the data sent
- The length is stored in the variable CONTENT_LENGTH
- CONTENT_LENGTH is used to find out how much to read from standard input, but this is done through Perl standard functions
Creating a Form

A form has 3 important parts:

• the FORM tag, which includes the URL of the CGI script that will process the form; the form
• the form elements
• the submit button

Sample Form using GET

```html
<HTML>
<HEAD><TITLE>Pizza Question</TITLE> </HEAD>
<BODY>
<H1>A Pizza Question</H1>
<form action="http://www.cs.rmit.edu.au/cgi-bin/pizza.pl" method="GET">
  What is your favorite pizza?
  <input type="text" name="favorite">
  
  <input type="submit">
</form>
</BODY>
</HTML>
```
Sample Form using Post

```html
<HTML>
  <HEAD>
    <TITLE>Pizza Question</TITLE>
  </HEAD>
  <BODY>
  <H1>A Pizza Question</H1>
  <FORM
    ACTION="http://www.cs.rmit.edu.au/cgi-bin/pizza.pl METHOD="POST">
    What is your favorite pizza?
    <INPUT TYPE="text" NAME="favorite">
  <P>
    <INPUT TYPE="submit">
  </FORM>
  </BODY>
</HTML>
```

Creating Text Boxes

- Text boxes can contain one line of free form text and are typically used for names.
- It starts with `<INPUT TYPE =`
- The TYPE must be set to “text”
- To identify the input data to the server a name must be used as in NAME = “name”
- If an initial value is desired specify VALUE = “ “
- The size of the box and the maximum number of characters can be specified using SIZE = n and MAXLENGTH = n.
Creating Larger Text Areas

• For larger text areas use the tag
  `<TEXTAREA ... and end with
  </TEXTAREA>`

• The height of the text area in rows can be specified with `Rows = n`,

• The width of the text area in characters can be specified with `Cols = n`,

• Visitors can enter up to 32,700 characters.

Creating Other Elements

Password boxes are similar to text boxes, but when the visitor types in it, the letters are hidden by asterisks. Specify:

`<Input TYPE = “password”>`

For creating radio buttons, check boxes and menus please refer to the text book pages 52 - 56.

`<INPUT TYPE = “radio”>`

`<INPUT TYPE = “checkbox”>`

`<SELECT NAME =`
<HTML>
<HEAD>      <TITLE>Online Pizza Store</TITLE>   </HEAD>   <BODY>      <H1>Online Pizza Store</H1>      <H2>Order A Pizza</H2>      <FORM ACTION="placeorder.cgi" METHOD="POST">         <B>Name:</B>         <INPUT TYPE="TEXT" NAME="name" VALUE="" SIZE="20" MAXLENGTH="40">         <P>
         <B>Beverage:</B>
         <INPUT TYPE="RADIO" NAME="drink" VALUE="cola"> Coca Cola
         <INPUT TYPE="RADIO" NAME="drink" VALUE="sprite"> Sprite
         <INPUT TYPE="RADIO" NAME="drink" VALUE="fanta"> Fanta
         <INPUT TYPE="RADIO" NAME="drink" VALUE="pepsi"> Pepsi
         <INPUT TYPE="RADIO" NAME="drink" VALUE="7up"> 7up
         <P>
         <B>Pizza Toppings:</B>
         <INPUT TYPE="CHECKBOX" NAME="topping" VALUE="cheese"> Cheese
         <INPUT TYPE="CHECKBOX" NAME="topping" VALUE="capicum"> Capicum
         <INPUT TYPE="CHECKBOX" NAME="topping" VALUE="anchovies"> Anchovies
         <INPUT TYPE="CHECKBOX" NAME="topping" VALUE="mushrooms"> Mushrooms
         <INPUT TYPE="CHECKBOX" NAME="topping" VALUE="prawn"> Prawns
         <P>
         <INPUT TYPE="SUBMIT" NAME="order" VALUE="Send Order">
      </FORM>   </BODY></HTML>
Running a CGI Program

Examples below points to a cgi file (written in Perl) which will print a personalised Hello World Greeting depending on your input.

You may either point your browser directly at the CGI script that you want to run passing it your name or set up a form that

Perl

• Why Perl?
  – Practical Extraction and Report Language
  – It is high-level
  – It is portable
  – It is easy to quickly write very powerful programs
  – Programmers don’t need to learn a lot of Perl to start coding
  – It is brilliant for processing text
  – It is very flexible, no need for complex syntax declarations
Perl

- Disadvantages:
  - It is sometimes cryptic
  - It is a bit slow
  - It could lead to unreadable code and un-maintainable systems

Storing Values in Perl

- Scalar Variable ($varname)
  - used for for containing strings, integers and floating point numbers.
- Indexed array (@arrayname)
  - uses an integer called index to reference individual array elements
- Hash (%hashname)
  - A hash uses keys that are strings to reference individual elements
Perl Variables

• Uninitialized variables are created automatically with a null value: "", or 0
• Depending on the context (where they are used), variables will be interpreted as strings, numbers, booleans. For example:

```perl
$dalmatians = "100";
print $dalmatians + 1;
```

will print 101

A sample program

```perl
$num2 = $num1 + 5;
print "*** Testing scalar ***\n";
print "Adding num1 5 yields: $num2.\n";
$s1 = "Today";
$s2 = " is Wednesday";
$s1 = $s1 . $s2;
print "\n$s1 is now: $s1\n";

print "\n*** Testing array ***\n";
@group = ("Ali","Bill","Mark");
print "The array group contains: @array 1\n";
print "The 1st elements of array contains: $group[0] \n";
# range operator can be used to store all values in a range
@gROUP2 = (A..J);
print "The array group2 contains: @group2 \n";

print "\n*** Testing hash ***\n";
%formdata = ("name", "Mark", "Phone", "95601567", "age", "21");
# use a key (string) to access individual elements
print "\n$formdata{\"name\"} \n";
```
Output

*** Testing scalar ***
Adding num1 5 yields: 5.
$s1 is now: Today is Wednesday

*** Testing array ***
The array group contains: Ali Bill Mark
The 1st element of group contains: Ali
The array group2 contains: A B C D E F G H I J

*** Testing hash ***
Mark

Quotation Marks

The paired backquotes (` `) tell Perl to perform the system action inside the quotation marks.

The paired double quotation marks (" " ) tell Perl to look for special characters and to interpret them inside the string.

The paired single quotation marks (‘ ‘) tell Perl not to look for or process any special characters in the print string.
Control Structures

Perl supports many control structures similar to C and Java.

For repeating statements until a condition is true use either while or do-while loops as in:

```perl
while (condition) {
    statements;
}
```

```perl
do {
    statements;
} while (condition);
```
Quiz - Are the output same?

```perl
$count = 10;
$sum = 0;
while ( $count < 8 )
    { $sum = $sum + $count;
      $count = $count + 2;
    }
print "sum is now $sum\n";
```

```perl
$count = 10;
$sum = 0;
do {
    $sum = $sum + $count;
    $count = $count + 2;
} while ( $count < 8 );
print "sum is now $sum\n";
```

```perl
foreach construct
To go through each line of an array or hash Perl uses the foreach structure.
This has the form

```perl
# retrieves each element from the array
foreach $student (@class) {
    print "$person\n";  # Print the item
}
```

To retrieve each name value pair from the hash %formdata first use the keys function to retrieve the keys of the hash.

```perl
foreach $key (keys %formdata) {
    print "name = $key value = $formdata($key)";
}
```
An example Comparing Strings

The string comparison operators are
- eq (equal)
- ne (not equal)
- gt (greater than)
- lt (less than)
- ge (greater than or equal to)
- le (for less than or equal to)

```
print "Password ? ";
$pass = <STDIN>;
chop $pass;  # Remove the newline at the end
while ($pass ne "secret")
{
    print "Please repeat again ? ";
    $pass = <STDIN>;
    chop $pass;
}
```

Conditionals

Conditionals can be created with Perl constructs if, else and elsif.

```
if (!$a)  # The ! is the not operator
{
    print "The string is empty\n";
}
elsif (length($a) == 1) {
    print "The string has one character\n";
}
else {
    print "The string has two or more
characters\n";
}
```
Logical operators
As in C and Java we have the same logical operators && and || and !.

```perl
print "Password ? ";
$pass = <STDIN>;
chop $pass; # Remove the newline at the end
while ($pass ne "secret" && $count < 3) {  
  print "Please repeat again ? ";
  $pass = <STDIN>;
  chop $pass;
  $count++;
}  
if ($count == 3)  
  print "nPlease see System Administrator";  
else  
  print "nYou may access now";
```

Tutorials
Assume that you are required to create a simple auctioning system whereby one could sell and buy items.

You are required to create a simple HTML form that allows the user to enter the following information before sending it to "http://www.cs.rmit.edu.au/~charles/cgi-bin/auctionsell.cgi in GET mode. This page will give acknowledge receipt of your submission. (Next week you will begin to write a more useful cgi file which will store this information in a file)

<table>
<thead>
<tr>
<th>ID</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of product</td>
<td>Book/CD/Electronic-item/Other</td>
</tr>
<tr>
<td>Description</td>
<td>Maximum 30 characters</td>
</tr>
<tr>
<td>City</td>
<td>Melbourne/Sydney/Pert/Brisbane</td>
</tr>
<tr>
<td>Viewing Time</td>
<td>office-hours/evenings/weekends</td>
</tr>
<tr>
<td>Closing date</td>
<td>dd/mm/yy (no checking at this stage)</td>
</tr>
<tr>
<td>Minimum Price</td>
<td></td>
</tr>
</tbody>
</table>

Discuss what will be the appropriate user interface elements given the requirements.

Discuss how your design could be coded in html.

Discuss what kind of file (structure) could be used in the server side to keep track of all the items submitted for sale. What additional file could be used if we also want to keep track of all the bids for the items offered for sale.

Discuss how a more visual representation of the product can be given to the potential buyer.
Lab Exercise

Write and test your html file containing the form discussed in the tutorials.

Write a perl program (test.pl) that will read in a person’s name and age. Display the message “young” if the person’s age is under 30 years and “not young” otherwise.

(To interpret the program use: perl test.pl)