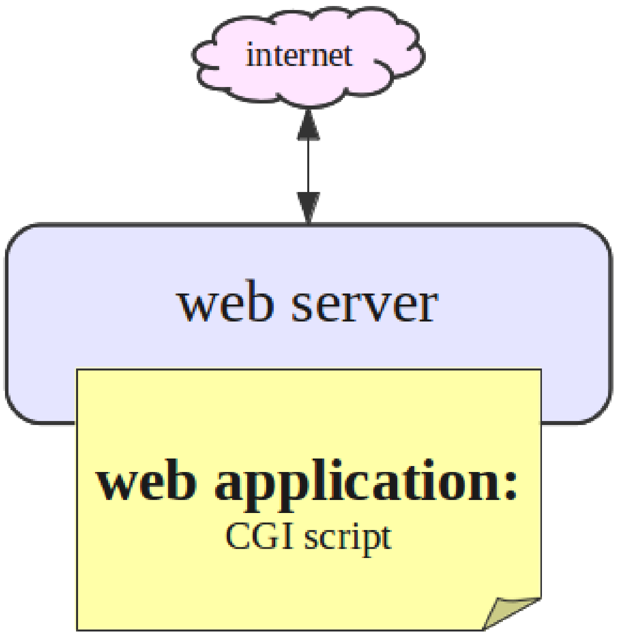
COMP 2021

Unix and Script Programming

CGI Programming in Perl

Common Gateway Interface (CGI)

- CGI is a protocol that defines how a Web server program interacts with application programs.
- HTML works for static web pages. CGI programs help to design dynamic web pages.
- A *CGI program* allows the user to interact with a web page by generating HTML code that depends on the user input.
- Since the Web mainly contains text, Perl is a popular language for CGI programming because it is good at text manipulation.



CGI (cont.)

```
Web                       Web        CGI           Appl.
Browser     Internet      Server     Protocol      Program

            HTTP                     Env. Var.
            request                  stdin
Send ----------------> Convert -----------> Process
                                                 |
            HTTP                                 |
            response                 stdout      V
Receive <------------- Convert <----------- Process
```

- Web server provides most of the input information to application programs through environment variables.
- Data send in the HTTP request with the GET method is converted to a special environment variable, QUERY_STRING.
- Data send in the HTTP request with the POST method is converted to the standard input (stdin) channel.
- Data printed to the standard output (stdout) channel is converted to the HTTP response.

CGI Programming Environment

- Cs System doesn't allow CGI
- We will work under ITSC ihome
- Your webpage http://ihome.ust.hk/~username
  - Activate your ihome service http://www.ust.hk/itsc/webguide/home/enable.html
  - Create an index.html file under your home directory
  - Use FTP client to upload/download files to your homepage, e.g. FileZilla, WS-FTP http://itsc.ust.hk/services/general-it-services/communication-collaboration/ihome/transfer-files/
  - Place your CGI programs in a directory called cgi-bin in your home directory, and set appropriate permissions
    http://itsc.ust.hk/services/general-it-services/communication-collaboration/ihome/running-cgi-programs/

1st CGI Program: Hello World

```
#!/usr/local/bin/perl5 -w
# helloworld.cgi: first CGI program

print "Content-type:text/html\n\n";
print '<html>';
print '<head>';
print '<title>Hello World - First CGI Program</title>';
print '</head>';
print '<body>';
print '<h2>Hello World! This is my first CGI
program</h2>';
print '</body>';
print '</html>';
```

Hello World Details

- The `Content-type` line identifies the type of output we are generating (`text/html`).
- It is immediately followed by a blank line, which must contain no spaces or tabs. This line separates the CGI header from the HTML code.
- After the blank line comes the HTML, which is sent to be formatted and displayed on the user's browser.

Hello World with Here Document

```
#!/usr/local/bin/perl5 -w
# helloworld_here.cgi
# Perl here document
  print <<END_of_HTML;
  Content-type: text/html

  <HTML>
        <HEAD>
               <TITLE> Hello World with Perl here
  document</TITLE>
        </HEAD>

        <BODY>
          <H1>Hello World</H1>
          <P> Hello everybody. This is the first CGI I wrote with
  Perl here document.</P>
        </BODY>

  </HTML>
END_of_HTML
```

Perl Here Documents

- Here document allows to quote multiline strings without worrying about the quotes and escapes.
- It starts with the << and a word called the *end token*
- The string begins on the next line and continues up to a line containing the end token at the start of the line.
- Here documents are very useful for generating HTML

Here Document Example

```
#!/usr/local/bin/perl5 -w
$heredoc = <<HEREEND;
Everything after
the start of the here-doc
is part of the string until
 we get to the
HEREEND
print $heredoc;
```

2nd CGI Program: Time-Date

```
#!/usr/local/bin/perl5 -w
# datetime.cgi
@months = qw(Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec);
@weekDays = qw(Sun Mon Tue Wed Thu Fri Sat Sun);
($second, $minute, $hour, $dayOfMonth, $month, $yearOffset, $dayOfWeek, $dayOfYear,
$daylightSavings) = localtime();
$year = 1900 + $yearOffset;
$theTime = "$weekDays[$dayOfWeek] $months[$month] $dayOfMonth, $year";

print "Content-type: text/html\n\n";
print << END_of_HTML;
<html>
          <head>
          <title>Time-Date: second CGI Program</title>
          </head>
          <body>
          <h1>Time-Date: second CGI Program</h1>
          <p> Unprocessed time: localtime().</p>
          <p>The time now is $theTime.</p>
          </body>
</html>
END_of_HTML
```

3rd CGI Example: CGI Environment Variable

```
#!/usr/local/bin/perl5 -w
# listCGIvar.cgi: list out all the CGI variables

print "Content-type: text/html\n\n";
print "<html>";
print "<head>";
print "<title>List all CGI variables: 3rd CGI Program</title>";
print "</head>";
print "<body>";
print "<font size=+1>Environment</font>\n";
foreach (sort keys %ENV)
{
  print "<b>$_</b>: $ENV{$_}<br>\n";
}
print "</body>";
print "</html>";
```

The CGI.pm Module

- Using here documents in Perl is still a painful way to generate HTML.
- Perl has a CGI module to make it easier.
- To use the CGI module in your program, include the following line near the top of your program:

```
use CGI qw(:standard);
```

- The `use` statement is like `#include` in C++; it brings in predefined functions from another file at compile time.
- More script examples of CGI.pm
  - http://www.wiley.com/legacy/compbooks/stein/source.html
  - And a lot more from Internet

Hello World using CGI.pm

- Below is the "Hello World" program using the CGI module:

```
#!/usr/local/bin/perl5 -w
# hello world CGI program using CGI module
# helloworld_pm.cgi

use CGI qw(:standard);
print header();
print start_html("Hello World with CGI.pm module");
print h1("Hello World");
print p("Hello everybody. This is a hello world with
  CGI.pm module.");
print end_html();
```

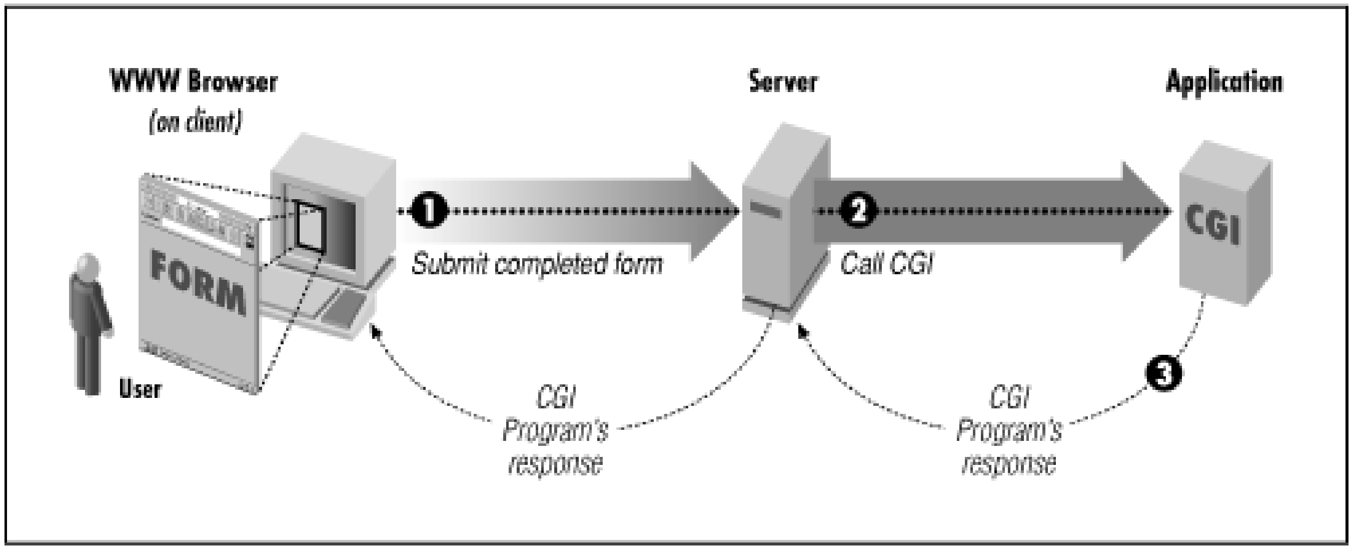
- CGI module functions return strings, which we can then send to print.

CGI.pm Details

- In the previous program,
  - `header()` returns a string containing the `Content-type` line with a following blank line
  - `start_html(`*`string`*`)` returns `string` as an HTML title
  - `h1(`*`string`*`)` returns `string` as a first-level HTML heading, and
  - `p(`*`string`*`)` would return `string` as a new HTML paragraph
  - `end_html()` returns </html>

CGI.pm Forms

- CGI.pm provides various *widgets* for accepting user input in forms.
- You can easily have text field, checkbox, radio button, menu, scrolled list, multiline text, buttons, and more



CGI.pm Form Example: COMP2021 Student Survey

```
#!/usr/local/bin/perl5 -w
# a CGI form example
# CGIform.cgi

use CGI qw(:standard);

print header();
print start_html(-title => 'COMP2021 Student Background: CGI form
example'),
    h1('COMP2021 Student Background Survey'),
    start_form,
    "Your name? ",textfield(-name=>'name', -defaults=>'Chan Tai Man'),
    p,
    "Your major?",
    p,
    radio_group(-name=>'major',
                -values=>['COMP','CPEG','ECE','ENGG', 'Others'],
                -defaults=>['COMP']),
    p,
```

Multi-line `print` statement

`p` puts a new paragraph/newline

```
 "Year of study? ",
    popup_menu(-name=>'year',
               -values=>['1','2','3','4']),
    p,
    "Why you choose COMP2021?",
     popup_menu(-name=>'reason', -values=>['required','interested']),
        p,
        "What's your feeling of COMP2021?",
        checkbox_group(-name=>'feeling',
                    -values=>['Fun','Boring','Difficult','Just right', 'A
piece of cake'],
                    -defaults=>['Fun']),
        p,
    submit('send'), reset('clear'),
    end_form,
    hr;
```

"Submit Query" used as default submit button name

```
if (param()) { # if the form has already been filled out
        my $who = param('name');
        my $dept = param('major');
        my $why = param('reason');
        my $feedback = param('feeling');
    print
        "Your name is ",em(param('name')),
        p,
        "Your are ",em(param('major'))," year ", em(param('year')), "
student";
        if($feedback eq "Fun" ){ print p("I'm glad you enjoyed the
course.");       }
        if ($feedback eq "Boring"){ print p("Oops, why?"); }
        print hr;
        if($why eq "required" ){ print p("$who in $dept, try hard to get
good grade!");
        }else{ print p("$who in $dept, hope you have fun!");}
        print hr;
}
print end_html;
```

em generates <em> HTML tag (~italics)

More on COMP2021 Student Survey

- You need `start_form()` before you add your form items.
- Form items are often called inside a `p()` function.
- The first argument is usually the name of the form item
- Items can also have default value

- More CGI.pm form example available
  - http://perlmeme.org/tutorials/cgi_form.html

A Better Approach

- To help you design your programs into nice readable web forms, we suggest the following architecture (pseudo code). Refer to `CGIformv2.cgi.`

```
1. print http header
2. print html_header_method # To make your pages
   look the same
3. a. if there are no parameters output the form
4. b. else if there is a key parameter # You may
   include a handle the results of the form # hidden
   'mode' field # to make this easier
5. print end_html_method # May include a standard
   footer
```