# COMP284 Practical 3
# Perl (3)

## Introduction

- This practical contains further exercises that are intended to familiarise you with Perl Programming. While you work through the tasks below compare your results with those of your fellow students and ask for help and comments if required.
- This document can be found at

  http://cgi.csc.liv.ac.uk/~ullrich/COMP284/notes/practical03.pdf

  and you might proceed more quickly if you cut-and-paste code from that PDF file. Note that a cut-and-paste operation may introduce extra spaces into your code. It is important that those are removed and that your code exactly matches that shown in this worksheet.
- The exercises and instructions in this worksheet assume that you use the Department's Linux systems to experiment with Perl.

  If you want to use the Department's Windows systems and our Perl installation on Windows instead, then you can do so.
- To keep things simple, we will just use a text editor, a terminal and a web browser. You can use whatever text editor and web browser you are most familiar or comfortable with.
- If you do not manage to get th
  complete them in you



## Exercises

1. Let us start by creating a first CGI script using Perl.

   a. Create a directory

      $HOME/public_html/cgi-bin/

      and make sure that `$HOME`, `public_html` and `cgi-bin` are readable and executable by everyone using the command

      ```
      chmod a+rx $HOME $HOME/public_html $HOME/public_html/cgi-bin
      ```

      in a terminal.

   b. Open a text editor and enter the following Perl code:

      ```
      #!/usr/bin/perl
      # Author: <your name>
      # CGI Perl script: Environment variables [perl03A]
      use CGI qw(-utf8 :all);
      binmode(STDOUT, ":encoding(utf-8)");

      print header(-charset=>'utf-8');
      foreach $key (keys %ENV) {
        print "The value of $key is $ENV{$key}", br(), "\n";
      }
      ```

      Replace *<your name>* with your own name.

c. Save the code to a file named `perl03A` in `$HOME/public_html/cgi-bin/`.

d. In a terminal, go to the directory in which the file has been stored, make the file `perl03A` **executable** for everyone, but only readable and writable by yourself, using

```
chmod a+x,og-rw perl03A
```

and execute the Perl script using the command

```
./perl03A
```

Check that there are no syntax errors and that the script produces some output.

This should become standard practice for you. CGI scripts that contain syntax errors produce no meaningful output or error messages when accessed using a web browser (as we will do in Exercise 1e). So, always assure yourself first that there are no syntax errors before accessing one of your scripts via the web.

e. Now open a web browser and access the URL

`http://cgi.csc.liv.ac.uk/cgi-bin/cgiwrap/<user>/perl03A`

where *<user>* is your user name.

f. Make sure that both Exercise 1d and 1e produce output. Compare the output. Why do Exercise 1d and 1e produce different output?

2. Let us move on to a slightly more complicated CGI script.

a. Open a text editor and enter the following Perl code.



```
#!/usr/bin/perl
# Author: <y
# CGI Perl script: U

use CGI qw(-utf-8 :all *table);
use LWP::Simple qw(get);
binmode(STDOUT, ":encoding(utf-8)");

print header(-charset=>'utf-8'), "\n",
      start_html({-title=>'URL retrieval',
                  -author=>'<your_email_address>'}), "\n";

if (param('URL')) {
   print h1("Environment variables"), br(), "\n";
   foreach $key (keys %ENV) {
     print "The value of $key is $ENV{$key}", br(), "\n";
   }
   print h1("Parameters"), br(), "\n";
   foreach $key (param()) {
     print "The value of $key is ",param($key), br(), "\n";
   }
   print h1("Content of ".param('URL')), "\n";
#  Retrieve URL here and assign it to $text
   print $text, br(), "\n";
}

print h1("URL retrieval"), "\n";
```

```
print start_form({-method=>"POST"
   -action=>"http://cgi.csc.liv.ac.uk/cgi-bin/cgiwrap/<user>/perl03B"});
print label("URL: ");
print textfield({-name=>'URL',
                 -size=>200}), "\n";
print br(), "\n";
print submit({-name=>'submit',
              -value=>'Process'}), "\n";
print end_form, end_html;
```

Replace *<your name>*, *<your_email_address>* and *<user>* with your own name, e-mail address. and user name, respectively.

b. This script uses the `LWP::Simple` module. Have a look at the description of this module at

   http://search.cpan.org/~gaas/libwww-perl-6.05/lib/LWP/Simple.pm

   Try to find a function that can be used to fetch a document identified by a given URL. At the point indicated in the code above, use that function to retrieve the document identified by the URL `param('URL')` and store it in `$text`.

c. Save the code to a file named `perl03B` in `$HOME/public_html/cgi-bin/`.

d. In a terminal, go to the directory in which the file has been stored, make the file **executable** for everyone, but only readable and writable by yourself, and execute the Perl script using t

   ```
   ./perl03
   ```

   Check that there a p as output.



e. Now open a web browser and access the URL

   ```
   http://cgi.csc.liv.ac.uk/cgi                    /perl03B
   ```

   where *<user>* is your user name.

   You should see an HTML form that allows you to enter a URL.

f. Enter the URL below into the textfield of the form and press the submit button (labelled 'Process').

   ```
   http://cgi.csc.liv.ac.uk/~ullrich/COMP284/tests/a1test1.txt
   ```

   You should now be shown a different page that displays the values of environment variables and parameters as well as the contents of the document available at the URL above, and the same HTML form at the bottom of the page.

   How does this work?

g. The text stored in `$text` obviously contains HTML markup. Modify the line

   ```
   print $text, br(), "\n";
   ```

   in `perl03B` so that the content of `$text` is shown verbatim, including the HTML markup. Hint: CGI.pm has a function `escapeHTML` that can help with that. But it does not preserve line breaks and whitespace. Some extra work is required for that.

h. Modify `perl03B` with code that extracts the names from `$text` and stores them in an array `@strings`.

i. Add the following code to `perl03B`, right after the code you have created in Exercise 2h.

```
%count = ();
foreach $string (@strings) {
  $count{$string}++;
}
foreach $key (keys %count) {
  print $key, ": ", $count{$key}, br(), "\n";
}
```

j. Make sure that Exercises 2h and 2i lead to the correct result for the URL in 2f.

k. Modify the code in 2i so that the output produced by the second foreach-loop is a two-dimensional HTML table of the form below. Ideally the names are sorted according the number of occurrences. Use CGI.pm HTML shortcuts.

Name count

| Name | No of occurrences |
|---|---|
| Andreas Schoknecht | 4 |
| Torsten Ullrich | 3 |

⋮

Hint 1: You either have to import additional HTML shortcuts `start_table` and `end_table` that allow you to produce an opening table tag and a closing table tag, respectively, or you have to first co n use the HTML shortc

Hint 2: Use Perl's nt according the values stored in `%count`.



3. The Perl CGI script in Exercise 2 used a form with only a single tex we work on a Perl CGI script with a more complex form.

a. Open a text editor and enter the following Perl code:

```
#!/usr/bin/perl
# Author: <your name>
# CGI Perl script: Form elements and Processing inputs [perl03C]

use CGI qw(-utf8 :all);
binmode(STDOUT, ":encoding(utf-8)");

print header(-charset=>'utf-8'), "\n",
      start_html({-title=>'My HTML Form',
                  -author=>'<your_email_address>'}), "\n";

if (param('submit')) {
   print h1("Environment variables"), "\n";
   foreach $key (keys %ENV) {
     print "The value of $key is $ENV{$key}", br(), "\n";
   }
   print h1("Parameters"), "\n";
   foreach $key (param()) {
     print "The value of $key is ",param($key), br(), "\n";
```

```
   }
}

print h1("Form"), "\n";

print start_form({-method=>"GET",
    -action=>"http://cgi.csc.liv.ac.uk/cgi-bin/cgiwrap/<user>/perl03C"});
print label("User name: ");
print textfield({-name=>'username',
                 -value=>'dave',
                 -size=>100}), "\n";
print br(), "\n";

print submit({-name=>'submit',
              -value=>'Click for response'}), "\n";
print end_form, end_html;
```

Replace *<your_name>*, *<your_email_address>* and *<user>* with your own name, e-mail address and user name, respectively.

Note that CGI.pm produces HTML4 markup. According to the HTML4 standard, forms are required to have an action attribute. For HTML5 markup, this requirement has been dropped and most browsers submit form data to the page containing the form, if no action is specified. Most browsers also do that for HTML4 markup although that means the mark

b. Save the code to a file n /cgi-bin/.

c. In a terminal, go to the di
**executable** for everyone, but only readable and wri ute the Perl script using the command

```
./perl03C
```

Check that there are no syntax errors and that the script pr
output.

d. Now open a web browser and access the URL

`http://cgi.csc.liv.ac.uk/cgi-bin/cgiwrap/<user>/perl03C`

where *<user>* is your user name.

You should see an HTML form that allows you to enter a user name.

e. Enter you own user name and press the submit button. Just as with the Perl CGI script in Exercise 2 you are transferred to a different page that displays the value of environment variables and parameters as well as the same HTML form at the bottom of the page.

In contrast to the script in Exercise 2 the current script uses the GET request method instead of the POST request method. What is the difference?

f. The line

```
    print "The value of $key is ",param($key), br(), "\n";
```

makes the script susceptible to *cross-site scripting*: Printing out `param($key)` incorporates whatever the user has entered into the text field of the form. This may include dangerous JavaScript code. We should either validate that this is not the case before printing out `param($key)` or be more careful with the way we print the user's input.

Check that the problem exists by entering

```
<script>alert("Hacked");</script>
```

as the username into the form and submitting it.

To solve the problem, adopt the approach taken in Exercise 2g. Check that this solution indeed works.

In your own time, read

http://www.perl.com/pub/2002/02/20/css.html

for a more detailed discussion of the problem and possible solutions.

g. Modify `perl03C` so that in addition to a user name the form also allows you to enter your first name(s) and your surname. Do it in such a way that the associated parameters will be '`firstname`' and '`surname`'.

Furthermore, add a CGI.pm HTML shortcut that displays the user name, first name, and surname in the page that is produced by processing the form input.

Test your solution with various names. In particular, try surnames like `O'Donnell` and `van Eijk` and observe how these are presented in the environment variables versus parameters.

h. Modify `perl03C` so that the string associated with the '`surname`' is converted to uppercase.

Add another CGI.pm HTML shortcut that displays the converted surname in the page that is produced by processing the form input.



i. Modify `perl03C` so that users of Microsoft Internet Explorer will receive the message 'Change to a bette



Hint: Take a look at

http://msdn.microsoft.com/en-us/l v=vs.85).aspx



to see what values the environment variable ight take for various versions of Microsoft Internet Explorer.

j. Modify `perl03C` so that only requests that come from a PC within the department (or the university) are answered. All other requests should result in an empty HTML page or an error message.

k. Modify `perl03C` so that the user can use the HTML form to indicate which degree programme he/she is studying (among the programmes G400, G401, G402, G403, G490, G491, G500, G501, G502, G503, G610, G611, G700, G701, N300). Use radio buttons to do so.

The page produced by processing the HTML form should indicate which programme the user is studying.

l. The radio buttons in Exercise 3k take up quite a lot of space, a popup menu would be more 'space efficient'. Modify `perl03C` so that instead of radio buttons a popup menu is used for the selection of a degree programme.

m. Modify `perl03C` so that a count is kept that is increased with each consecutive 'call' of the page and displayed on the results page.

Hint: A hidden field might be the easiest way of doing this.

4. Write a Perl script `perl03D` that takes an arbitrary sequence of numbers as input, plus an option `--op=<op>` where `<op>` is one of '*' or '+', computes as result the product or sum of the number sequence, and displays that result.

   Examples:
   `./perl03D --op='+' 1 2 3 4` returns 10
   `./perl03D --op='*' 2 3 4` returns 24