COMP[29]041 16s2 (http://www.cse.unsw.edu.au/~cs2041/16s2/)

Yes you can write A CGI script in Shell

Software Construction (http://www.cse.unsw.edu.au/~cs2(

Aims

This week you meet the bare metal of the CGI interface.

Assessment

Submission: `give cs2041 lab09 browser.cgi login.cgi`
Deadline: either during the lab, or Monday 3 October 11:59pm (midnight)
Assessment: Make sure that you are familiar with the lab assessment criteria (lab/assessment.html).

Storing CGI scripts on gitlab.cse.unsw.edu.au

At CSE CGI scripts must be placed in your `public_html` directory.

This means we need to use a separate repository for your CGI scripts. on `gitlab.cse.unsw.edu.au` for your scripts. One has already been set up for you.

```
$ cd
$ chmod 751 .
$ mkdir public_html
$ chmod 755 public_html
$ cd public_html
$ git init
$ chmod 700 .git
$ # Make sure you replace the zID below with your own
$ git remote add origin gitlab@gitlab.cse.unsw.EDU.AU:z5555555/16s2-comp2041-cgi.git
$ git pull origin master
```

Exercise: A CGI script in Shell

Write a shell CGI script `browser.cgi` which prints the details of the web prowser accessing. It should print the IP address & the hostname of the machine browser is running on, and it should print the browser's *user agent* string.

You'll find the repository already contains this starting point code.

It prints some information about the web server running the CGI script.

```
#!/bin/sh
echo Content-type: text/html
echo

host_address=`host $REMOTE_ADDR 2>&1|grep Name|sed 's/.*: *//'`

cat <<eof
<!DOCTYPE html>
<html lang="en">
<head>
<title>Webserver IP, Host and Software</title>
</head>
<body>
This web server is running on at IP address: <b>$SERVER_ADDR</b>
<p>
This web server is running on hostname: <b>$SERVER_NAME</b>
<p>
This web server is <b>$SERVER_SOFTWARE</b>
</body>
</html>
eof
```

server.cgi (lab/cgi/intro/server.cgi)

This web server is running on at IP address: **129.94.242.40**

This web server is running on hostname: **cgi.cse.unsw.edu.au**

This web server is **Apache/1.3.34 Ben-SSL/1.55 (Debian) mod_ssl/2.8.25 OpenSSL/0.9.8c**

```
<html>
<head>
<title>Webserver IP, Host and Software</title>
</head>
<body>
This web server is running on at IP address: <b>129.94.242.40</b>
<p>
This web server is running on hostname: <b>cgi.cse.unsw.edu.au</b>
</p><p>
This web server is <b>Apache/1.3.34 Ben-SSL/1.55 (Debian) mod_ssl/2



</p>
```

You can try it the starting point code like this.

```
$ cd
$ chmod 751 .
$ cd public_html/lab09
$ chmod 755 . ..
$ chmod 755 browser.cgi
$ firefox http://cgi.cse.unsw.edu.au/~z5555555/lab09/browser.cgi &
```

If you are not working in a CSE lab run firefox (or another web browser) on your local machine.

For example if you are working at home on a laptop and using ssh to connect to CSE, run the web browser on your lapop and supply the URL `http://cgi.cse.unsw.edu.au/~z5555555/lab09/browser.cgi` .

Change browser.cgi to make it behave exactly like this example implementation:

browser.cgi (lab/cgi/intro/browser.cgi)

Your browser is running at IP address: **129.94.8.189**

Your browser is running on hostname: **uniwide-pat-pool-129-94-8-189.gw.unsw.edu.au**

Your browser identifies as: **Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_5) AppleWebKit/603.2.4 (KHTML, like Gecko) Version/10.1.1 Safari/603.2.4**

```
<html>
<head>
<title>IBrowser IP, Host and User Agent</title>
</head>
<body>
Your browser is running at IP address: <b>129.94.8.189</b>
<p>
Your browser is running on hostname: <b>uniwide-pat-pool-129-94-8-1
</p><p>
Your browser identifies as: <b>Mozilla/5.0 (Macintosh; Intel Mac OS X 1



</p>
```

Hints

You need to produce identical output to pass the autotest for this exercise.

Note in the CGI examples (lab/cgi/code/cgi/cgi_examples.html) from lectures you been shown a CGI script which prints the environmental variables:

```
#!/bin/sh
echo Content-type: text/html
echo

cat <<eof
<!DOCTYPE html>
<html lang="en">
<head>
<title>Environment Variables</title>
</head>
<body>

Here are the environment variables the web server has passed to this CGI script:
<pre>
eof

env

cat <<eof
</pre>
</body>
</pre>
</html>
eof
```

env.cgi (lab/cgi/intro/env.cgi)

Here are the environment variables the web server has passed to this CGI script:

```
SERVER_SIGNATURE=
Apache/1.3.34 Ben-SSL/1.55 Server at cgi.cse.unsw.edu.au Port 443


UNIQUE_ID=WUtho4Fe8igAAHebcBs
REDIRECT_SCRIPT_URL=/~cs2041cgi/16s2/lab/cgi/intro/env.cgi
HTTP_USER_AGENT=Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_5) AppleWeb
SERVER_PORT=443
REDIRECT_SCRIPT_URI=https://cgi.cse.unsw.edu.au/~cs2041cgi/16s2/lab/cgi/
HTTP_HOST=cgi.cse.unsw.edu.au
DOCUMENT_ROOT=/var/apache
SCRIPT_FILENAME=/web/cs2041cgi/16s2/lab/cgi/intro/env.cgi
HTTPS=on
REQUEST_URI=/~cs2041cgi/16s2/lab/cgi/intro/env.cgi
SCRIPT_NAME=/~cs2041cgi/16s2/lab/cgi/intro/env.cgi
SCRIPT_URI=https://cgi.cse.unsw.edu.au/~cs2041cgi/16s2/lab/cgi/intro/env
```

```
<html>
<head>
<title>Environment Variables</title>
</head>
<body>

Here are the environment variables the web server has passed to this CG
<pre>SERVER_SIGNATURE=<address>Apache/1.3.34 Ben-SSL/1.55 Se

UNIQUE_ID=WUtho4Fe8igAAHebcBs
REDIRECT_SCRIPT_URL=/~cs2041cgi/16s2/lab/cgi/intro/env.cgi
HTTP_USER_AGENT=Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_5) A
SERVER_PORT=443
```

The command "host" will given an IP address print the corressponding hostname (if there is one). For example:

```
$ host 129.94.242.20
20.242.94.129.in-addr.arpa domain name pointer williams.orchestra.cse.unsw.EDU.AU.
```

If your script is producing a 500 error from the webserver you can obtain some debugging info by creating a `.htaccess` file with these contents:

```
<Files "login.cgi">
SetHandler application/x-setuid-cgid
</Files>
```

See here (http://taggi.cse.unsw.edu.au/FAQ/CGI_scripts/) for more info.

Sample solution for browser.cgi

```
#!/bin/sh

# print content-type lines ASAP to make debugging easier if there are errors
echo Content-type: text/html
echo

# translate IP address to a hostname

host_address=`host $REMOTE_ADDR 2>&1|sed 's/[ .]*$//;s/.* //'`

cat <<eof
<!DOCTYPE html>
<html lang="en">
<head>
<title>IBrowser IP, Host and User Agent</title>
</head>
<body>
Your browser is running at IP address: <b>$REMOTE_ADDR</b>
<p>
Your browser is running on hostname: <b>$host_address</b>
<p>
Your browser identifies as: <b>$HTTP_USER_AGENT</b>
</body>
</html>
eof
```

Exercise: Simple Authentication in A CGI Script

In the tute you saw a Perl program which read a username and password and then checked that the password matches against one stored for the user in `accounts/` *username*`/password`

Write a Perl CGI script `login.cgi` which does the same thing.

You'll find the repository already contains this starting point code.

```
#!/usr/bin/perl -w

use CGI qw/:all/;
use CGI::Carp qw/fatalsToBrowser warningsToBrowser/;

print header, start_html('Login');
warningsToBrowser(1);

$username = param('username') || '';
$password = param('password') || '';

if ($username && $password) {
    print "$username authenticated.\n";
} else {
    print start_form, "\n";
    print "Username:\n", textfield('username'), "\n";
    print "Password:\n", textfield('password'), "\n";
    print submit(value => Login), "\n";
    print end_form, "\n";
}
print end_html;
exit(0);
```

Match EXACTLY the behaviour of this example implementation:

login.cgi (lab/cgi/authenticate/login.cgi)

Username: Password: Login

```
<html>
<head>
<title>Login Page</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-885
</head>
<body>
<form method="post" action="/~cs2041cgi/16s2/lab/cgi/authenticate/lo

Username:
<input type="text" name="username">
Password:
<input type="text" name="password">
<input type="submit" name="value" value="Login">
```

Note if the user supplies only their username, then the example implementation requests their password - you must match this behaviour.

Note if the user supplies only their password, then the example implementation requests their username - you must match this behaviour.

You must also match the example implementation behaviour for an unknown username.

Hints

Get a Perl program like the one in your tute working first then cut-and-paste the relevant parts of the debugged code into your CGI script.

Use a hidden variable to store the username if no password is supplied. Similarly use a hidden variable to store the password if no username is supplied.

```
$ cd
$ cd public_html/lab09
$ chmod 755 login.cgi
$ vi login.cgi
$ firefox http://cgi.cse.unsw.edu.au/~z5555555/lab09/login.cgi &
```

You can also browse the users files here (lab/cgi/authenticate/accounts)

Sample solution for login.cgi

```
#!/usr/bin/perl -w

use CGI qw/:all/;
use CGI::Carp qw/fatalsToBrowser warningsToBrowser/;

print header, start_html('Login Page');
warningsToBrowser(1);

$username = param('username') || '';
$password = param('password') || '';

# limit username to 256 word characters (
$username = substr $username, 0, 256;
$username =~ s/\W//g;

if ($username && $password) {
        $password_file = "accounts/$username/password";
        if (!open F, '<', $password_file) {
        print "Unknown username!\n";
        } else {
        $correct_password = <F>;
        chomp $correct_password;
        if ($password eq $correct_password) {
                print "$username authenticated.\n";
        } else {
                print "Incorrect password!\n";
        }
    }
} else {
        print start_form, "\n";
        if ($username) {
        print hidden('username'),"\n";
        } else {
        print "Username:\n", textfield('username'), "\n";
        }
        if ($password) {
        print hidden('password');
        } else {
        print "Password:\n", textfield('password'), "\n";
        }
        print submit(value => Login), "\n";
        print end_form;
}
print end_html;
exit(0);
```

Challenge Exercise: Combining an Application & CGI script

Combine the code for `login.pl` & `login.cgi` to produce a Perl program `login.pl.cgi` which can be run both from the command line and as a CGI scrip Note you can run a CGI script direct from the command line supplying encoded parameters on STDIN - just like a web server does - this can be useful for debugging but is not what is wanted here.

Instead make your program detect that it is being run directly from the command line and if so behave like a normal application. For

```
$ login.pl.cgi
username: andrewt
password: correct horse battery staple
You are authenticated.
$ firefox http://www.cse.unsw.edu.au/~abcd123/login.pl.cgi &
...
```

Its not hard - and it is useful trick to add a simple debugging framework to a CGI script.

Sample solution

```
#!/usr/bin/perl -w

use CGI qw/:all/;
use CGI::Carp qw/fatalsToBrowser warningsToBrowser/;

$running_as_cgi = defined $ENV{'REQUEST_URI'};

if ($running_as_cgi) {
        print header, start_html('Login Page');
        warningsToBrowser(1);

        $username = param('username') || '';
        $password = param('password') || '';

        # limit username to 256 word characters (
        $username = substr $username, 0, 256;
        $username =~ s/\W//g;
} else {
    print "username: ";
    $username = <STDIN>;
    chomp $username;
    print "password: ";
    $password = <STDIN>;
    chomp $password;
}

if (($running_as_cgi && $username && $password) || !$running_as_cgi) {
        $password_file = "accounts/$username/password";
        if (!open F, '<', $password_file) {
        print "Unknown username!\n";
        } else {
        $correct_password = <F>;
        chomp $correct_password;
        if ($password eq $correct_password) {
                print "$username authenticated.\n";
        } else {
                print "Incorrect password!\n";
        }
    }
} else {
        print start_form, "\n";
        if ($username) {
        print hidden('username'),"\n";
        } else {
        print "Username:\n", textfield('username'), "\n";
        }
        if ($password) {
        print hidden('password');
        } else {
        print "Password:\n", textfield('password'), "\n";
        }
        print submit(value => Login), "\n";
        print end_form;
}

if ($running_as_cgi) {
        print end_html;
}
```

Finalising

You must show your solutions to your tutor and be able to explain how they work. Once your tutor has discussed your answers with you, you should subm them using:

```
$ give cs2041 lab09 browser.cgi login.cgi
```

Whether you discuss your solutions with your tutor this week or next week, you must submit them before the above deadline.