# Chapter 1

# Practical assignment 1

## 1.1 Background

When the web was invented, web pages were encoded as static HTML pages. In addition, web pages could be generated by CGI programs. As time progressed a number of mechanisms were developed to shift (some) execution from the web server to the browser. Examples include Java applets, Javascript and a variety of other mechanisms to enable web servers to serve 'active' content. However, HTML remains the major notation used to encode content (and active content is typically incorporated into the HTML encoded pages). However, no browser-side active content should be used in any of the assignments of this module. Note that the use of CGI programs will be required in this assignment (and you are looowed to use CGI programs in nay of the other assignments, when meaningful. (Most assignments will not be web-based, so CGI would only be an option in a few of the assignments.)

HTML has been revised several times and HTML 5 is the current standard. You are expected to use (correct) HTML 5 for all assignments where HTML is used.

Web pages may be served by general purpose servers, or servers designed for a specific application. Apache is (and has for a long time been) the most popular general purpose web server. Chapter B describes the manner in which a virtual computer could be instantiated for practical assignments in this module. After your virtual computer has been built it should be simple to install Apache on it (if not already present after initial installation of the selected Linux distribution). Any search engine will point one to further instructions — if required.

In order to use the HTTP server one typically simply has to copy the HTML files to the appropriate 'home' directory and the server will start serving those. To enable dynamic content the early web servers provided a mechanism that was (and

still is) called CGI (*Common Gateway Interface*). A CGI program is a program that 'generates' output in the form of HTML; this HTML will then be sent to the browser, were it would be rendered.

A simple Hello World CGI program may look as follows (using some imaginary programming language):

```
print "<!DOCTYPE HTML>"
print "<HEAD>"
print "<TITLE>Example</TITLE>"
print "</HEAD>"
print "<BODY>"
print "<P>Hello world</P>"
print "</BODY>"
print </hHTML>"
end
```

If you run this program on a console it will simply print out the marked up "Hello world" lines. However, when installed in the CGI directory of a web server, it will 'print' its output to the pipe that connects the web server to the browser, and the page will be rendered by the web browser. For it to work, the program should be executable code and the web server must be authorised to execute it. Since the CGI and 'home' directories are not always treated equally, you may need a file called `index.htm`, `index.html`, or similar suitable 'start' file that will be displayed when you open the home directory of the server. This (static) file may include an `<a href="..."` link that points to the CGI program. When one clicks on the link, the CGI program will be executed.

Note that a CGI program may also 'print' `<a href="..."` lines — just as if they were embedded in a static file.

Of course, writing such a program that simply produces static content is pretty useless; it is intended to be used in a context where what it outputs will change depending on current conditions. As an example, the CGI program may be connected to a database that operates as a back-end and that, for example, contains the types and numbers of items a merchant has in stock. Running this program will then not display static data, but the current stock levels of the merchant.

More generally, the CGI program may need some inputs so that one may, for example, query the stock level of some specific item. However, in this assignment we will not assume that one can provide the CGI program with input. Another simplifying assumption for this assignment is that the back-end will consist program (without any need for a database). Our intention is that you should demonstrate that

- You are able to install and use the web server;

- You can install one or more simple static pages in the appropriate server directory;
- You can install one or more executable CGI programs in the appropriate directory; and
- Get it all to work via a browser.

You may use any language to write your CGI programs. Run them from the console / command line to test them. As an example, if your CGI program is called `test`, run `./test` from the command line; you should see the output that should look like something encoded in HTML. Better still, execute `./test > test.htm` and the output will be redirected to `test.htm`; open `test.htm` in your favourite browser and the expected output should be rendered properly. Also consider validating `test.htm` using

`https://validator.w3.org/`

to be sure that your program generates valid HTTP. (It is a good idea to also validate any static HTML files you create for this — and other — assignments.)

Recall that the 'back-end' of your system will consist of a simple data file in an appropriate directory and with an appropriate initial value. Remember to create it once you have read the remainder of the assignment.

The assignment focusses on so-called server-side execution, so you are not allowed to execute any code in the browser — hence no JavaScript or other client-side software is allowed.

Package the files for this assignment (static HTML and executable CGI programs) in an archive that will install the files into the correct directory irrespective of the current directory from which the archive is unpacked. This requires you to use `tar`, `tgz` and/or `zip` to package the files using absolute paths.[1] The correct absolute paths may be changed in the Apache configuration file(s), but you are expected to use the default locations from the Linux distribution you use for this assignment. (If your CGI programs are compiled, let the archive unpack your source code to your home directory — or a suitable subdirectory in your home directory.)

Note that the intention is not to show your prowess to build sites. It is intended to show that you are able to achieve the goals set out above A tiny site will thus suffice.

[1] Many students have in the past struggled to figure out how to use absolute paths. Figure this out early, because it is really simple once you are aware of the concept!

## 1.2 Your assignment

Write one or more CGI programs and/or static HTML files that display two random numbers, along with the instruction that the user should click on the larger of the two numbers, Each of the numbers is displayed as a hyperlink (that is, between `<a ...>` and `</a>` tags). The larger of the two numbers will then be linked to, say, a file called `right.htm`, while the other may be linked to a file called `wrong.htm`. The file `right.htm` contains an appropriate congratulatory message, while `wrong.htm` contains whatever you want to display to anyone unable to select the correct answer. Both these files contain a hyperlink that allows the user to try again.

You do not have to use these naming conventions or set of files — a single CGI script may suffice.

Obviously the suggested solution contains a major flaw: If the user hovers above a number, it may be obvious that it is a link to `right.htm` or `wrong.htm`. This may mean that the user does not really think about the challenge that needs to be solved. For this assignment this flaw is acceptable — unless your personality does not allow you to live with such a flawed solution.

Note that you should consider the possibility that the browser may cache a web page, and that your newly generated numbers are not displayed; the browser just sticks to one set of numbers. Insert an appropriate meta-tag to ensure that this does not happen.

In case you missed it: The system is dynamic since the CGI program generates a new pair of numbers for each round, and it should be unpredictable whether the first or second number in any round would be the larger of the two.

Is it best to write a program such as this to execute in an infinite loop, or should there be some option to 'exit'?

## 1.3 Assessment

This assignment — like most of those that will follow — will count out of 10. If you manage to execute your assignment perfectly, you will be awarded 10. (This will not be true for subsequent assignments). However, marks will be deducted for aspects that do not work correctly; examples include files that are installed in incorrect directories from your archive, a web server that does not serve the pages correctly, or output that is clearly wrong.