Programming in Perl, Fall 97
Project 2
Due : Monday, December 8 at midnight
Weight : 20% of course grade
Instructor : Louis Ziantz

This assignment will allow you to experiment with creating and running a CGI script using Perl. It should be submitted according to the *Submission Guidelines.* Do not forget to comment your program, providing a general description of the Perl script and any subroutines used, along with inline comments for important or complex portions of code. Please get started on the assignment as soon as possible.

1 Motivation

A club whose members are distributed across the world elect a president along with four members of an advisory board every two years. Since many of the members have internet access, they would like to use a simple CGI script to provide the capability of voting online (and thereby reduce mailing costs).

2 Specification

They would like a single script that both generates a voting form and processes a user's votes.

2.1 Form Generation

Candidates for offices are stored in a file called candidates.dat. A line of the file has the following format:

```
<office>:<last-name>,<first-name>:<vote-count><newline>
```

where `<office>` is either `president` or `board-member` and `<vote-count>` is initially 0. For example,

```
president:Abbott,Bud:0
president:Costello,Lou:0
board-member:Marx,Groucho:0
board-member:West,Mae:0
```

You will first need to read and store the names and number of votes in some reasonable data structures (perhaps one hash for presidential candidates and another for board candidates).

The `<TITLE>` of the web page ballot should be `Voting Form`. The page itself should consist of the following:

- A `<H1>` heading that reads `Voting Form`.
- A textfield that allows the user to enter a first name.
- A textfield that allows the user to enter a last name.
- A textfield that allows the user to enter a membership identification number.
- A checkbox that asks whether the user just wants the current results displayed (instead of voting).
- A heading for presidential candidate names, followed by a radio group list of candidates on separate lines. You should only be able to vote for one candidate, and no candidate's box should be checked at the start. The candidates names should be printed in alphabetical order as `<first-name>`, space, `<last-name>`.

- A heading for advisory board candidate names are, followed by a checkbox group list of candidates on separate lines. You should be able to vote for more than one candidate, but no candidate's box should be checked at the start. The candidates names should be printed in alphabetical order as `<first-name>`, space, `<last-name>`.
- A submit button titled `vote` and a reset button titled `clear`.

2.2 Form Processing

Before any processing occurs, your code should verify that the user is a member of the club by making use of a file called membership.dat. A line of this file has the following format:

```
<last-name>,<first-name>&<membership-id-number><newline>
```

For example,

```
Fields,W.C.&2348
```

You should read through the file searching for the membership id number entered by the user. If a match is not found, you should generate a HTML message page that indicates the number was not found in the membership database. If a match is found, you should compare the last name in the file to the last name entered by the user. If they are not the same, you should generate another HTML message page that indicates the number matched, but the name did not. If both the number and the last name entered match an entry from the file, the user is validated and processing should continue; otherwise the script exits. (Yes, technically, all of the Marx brothers could vote with the same id, but that is ok.)

After validation, the candidate file must be read as in the form generation stage.

There are two major options the user can select from the form, based on the single checkbox that inquires about displaying results. If the checkbox is NOT checked, the form should assume that the user is voting. If it is checked, the form should simply display the current state of the election and IGNORE any voting fields set.

2.2.1 If the user is voting...

The script should test to make sure that the user voted for no more than four board members (less than four is fine). It should generate an appropriate HTML message page and exit if this occurs. Otherwise, the votes are recorded in memory and the candidate file is then rewritten with the modified vote count in exactly the same form as you would read in (all candidates should be written out).

An HTML page consisting of a summary of the user's voting should then be generated. It should have the `<TITLE> Voting Summary`. The text from an example page follows:

```
Voting Summary

Thank you Susan Roberts .
Your votes have been recorded as follows:

President: Lou Costello
Board Selections:
Mae West
```

In the above, `Voting Summary` would be a `<H1>` heading and the titles of the offices would be in bold type.

2.2.2 If the user just wants the current results...

An HTML page consisting of a summary of the current state of the election should be generated. It should have the `<TITLE> Current Results`. The text from an example page follows:

```
Current Results

Votes for President:

Bud Abbot: 56
Lou Costello: 84

Votes for Advisory Board:

Groucho Marx: 115
Harpo Marx: 87
Mae West: 123
```

In the above, `Current Results` would be a `<H1>` heading and the headings for the offices would be `<H3>`.

3 Details

- Name the script vote.cgi.
- Include the pragma `use strict;`.
- Do not forget to call the appropriate `start_*` and `end_*` when starting/ending an HTML page or form.
- You may assume the initial data files are error free. I will e-mail the class when sample data files are made available on the course web page (`http://www.cs.rpi.edu/~ziantzl/perl.html`).
- You do not need to write code to prevent the same member from voting more than once (the club trusts its members).
- You will need to lock candidates.dat for concurrent read when you read it for input and for exclusive write when you modify it. You can use the code I gave in class.
- Make sure you display an HTML error message whenever there is an error that you can trap (for example, if a file can't be opened or locked). You can use the `&bail()` subroutine I gave in class or make one of your own.
- Make use of subroutines. The generation and processing of the form should be separate subroutines. The form processing subroutine should call at least two other subroutines — one to display the current results and another to record the votes. Other activities, such as reading the candidate file and generating the different HTML messages are all candidates for separate subroutines.
- Keep the use of global variables to a minimum. Global "constants" such as `$LOCK_EX` are fine. It is acceptable to make the voting data structures global (but you should at least experiment with passing a hash or an array by reference, since it may be on the exam).

4 CGI.pm

The CGI Perl Module is available in the directory `/dept/cs/complang/perl/module`. You should make a soft link to it from any directory in which you are going to execute your CGI script (that is, `ln -s /dept/cs/complang/perl/module/CGI.pm`). If you receive the following message from Perl:

```
Can't locate CGI.pm in @INC at ./vote.cgi line 7.
BEGIN failed--compilation aborted at ./vote.cgi line 7.
```

then you don't have access to it from the directory you are in and need to make the link. For the purposes of this project, the only directory you will need to make a link from is `~/public_html/cgi-bin` (see *Enabling Online CGI Execution* below) unless you want to do offline testing in another directory.

5 Offline Execution

Most of the development of your CGI script should be done in this mode of execution. To get offline execution, you simply run the CGI script as a normal Perl script on, for example, an IBM workstation. When you run the script, you will get the following prompt:

```
% add.cgi
(offline mode: enter name=value pairs on standard input)
```

If you hit Ctrl-D at this point, the script should produce the HTML corresponding to the form that the user will use to enter input. As long as you use the routines `header()`, `start_html()`, `start_form()`, `end_form()`, and `end_html()` with HTML emitting functions placed between `start_html()` and `end_html()` and HTML form widgets only placed between `start_form()` and `end_form()`, the HTML produced should be valid (if not aesthetically pleasing when printed). However, this is a good way to check for run-time warnings and other problems. Continuing the above example, a Ctrl-D would give us

```
Content-type: text/html

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<HTML><HEAD><TITLE>Add Me</TITLE>
</HEAD><BODY><H1>Add Me</H1><HR><FORM METHOD="POST"  ENCTYPE="application/x-www-form-urlencoded">
<P>First Number: <INPUT TYPE="text" NAME="field1" VALUE=""></P>
<P>Second Number: <INPUT TYPE="text" NAME="field2" VALUE=""></P>
<P><INPUT TYPE="submit" NAME="add" VALUE="add"> <INPUT TYPE="reset" VALUE="clear"></P>
</FORM><HR></BODY></HTML>
```

To make your script execute as if it received user input, enter the `-NAME` that you gave each widget followed by an equals sign and a test value you would like it to have. If the value has spaces in it, like *hello there*, it should be enclosed in quotes. When you are done with the list, type Ctrl-D. For add.cgi, we could enter

```
field1=3<newline>
field2=4<newline>
<control>D
```

The script would emit:

```
Content-type: text/html

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
```

```
<HTML><HEAD><TITLE>Add Me</TITLE>
</HEAD><BODY><H1>Add Me</H1><P>3 + 4 = <strong>7</strong>
</P></BODY></HTML>
```

Again, it could also generate Perl warnings or programmer-defined error messages.

What if you have a widget that allows the user to select several values to return to the CGI script (for example, a `checkbox_group`)? Then you simply repeat the widget-name-equals-value combination for each value:

```
operating_systems=AIX<newline>
operating_systems='Sun OS'<newline>
<control>D
```

6 Enabling Online CGI Execution

To run a CGI script on the server, you will need to take the following steps:

1. starting at your home directory, create a directory called **public_html**;
2. `cd` to **public_html** and create a directory called **cgi-bin**;
3. `cd` to **cgi-bin** and type

   ```
   fs sa . system:anyuser rlwk
   ```

   The `rl` (read and lookup) privileges are needed to run any CGI script. The `wk` (write and lock) privileges will be needed because you will be locking and writing to a file from a CGI script for Project 2. **You should set the privileges back to `rl` when you are done testing the project.**
4. make sure the Perl script is contained in a file that ends with the extension `.cgi`;
5. make the CGI script executable (for example, `chmod a+x script.cgi`).

Once this is complete, the script may be run as described in the following section.

7 Online Execution

Bring up your favorite browser, say `netscape`. Open the URL

`http://cgi.cs.rpi.edu/rcs/<your-rcs-id>/<your-script-prefix>.cgi`

For example, if I wanted to run a script called add.cgi, I would open

`http://cgi.cs.rpi.edu/rcs/ziantl/add.cgi`

Your script should now execute, most likely generating a form for the user to enter input. Once the user submits the form, your script will produce another HTML page in response which will be displayed automatically.

8 Server Errors

Suppose you are running in *online mode* and receive the following message or something like it:

```
Internal Server Error

The server encountered an internal error or misconfiguration and was
unable to complete your request.

Please contact the server administrator, www@cs.rpi.edu and inform
them of the time the error occurred, and anything you might have
done that may have caused the error.
```

In this case, you should go back to *offline mode* and test your script again thoroughly. You will probably find that you have an error. If everything seems to run fine offline, but you still get a server error message online, please contact me. Please do not contact `www@cs.rpi.edu`. Take note that this is not an RCS server, so sending mail to anyone at ACS will not be productive.