

# Troubleshooting CGI Scripts

## Generic CGI Troubleshooting

The web designer's guide at <http://www.bignosebird.com> offers the following guide to the first steps to take when considering a CGI that doesn't work:

The most important thing you can do when faced with a script failure problem is take about ten deep breaths before dealing with it. Then, follow these simple steps to trace the cause of the problem.

- **Check your E-mail.**

Go read your mail. Maybe there is a note from the ServInt NOC notifying you that something has been changed on the system that might affect your site. If this is the case, then the letter probably holds the answer. If the letter does not make sense to you, the reply to the letter and ask for clarification.

- **Check the script's modification date**

Look at the directory listing for the script. If the date seems more recent than the last time you remember working on it, then either you or somebody else modified it. If you have a back-up copy of the script, compare the two and look for differences. This also applies to any related configuration files.

- **Check your website's physical path.**

Whether using FTP or telnet, issue the `pwd` command to find out the full path to the directory where your script resides. If it looks like it no longer agrees with what paths are in your script(s), edit your scripts with the new path. If you were not notified, write your sysadmin a polite letter thanking them for not notifying you of a change that affected the operation of your site. Of course, if you find an E-mail from three days ago notifying you of an upcoming change...

Beyond those general problems, some of the most common sources of difficulty are:

- **Filename extensions**

The program's name does not end with the `.cgi` extension, and so cannot be executed by Apache. If this is the case, you can use the `mv` command (see *mv*, page 81) to rename the script with a command like:

```
server: %mv yourscript yourscript.cgi
```

- **Permissions problems**

The program is not able to be executed because its permissions (see *permissions in Unix*, page 84) are not set properly.

File permissions allow read, write, and execute access to users based on their user identification (also known as **uid**), and their membership to certain groups. You can use the command **chmod** (see *chmod*, page 39) to change a file's permissions. Here is an example:

```
server: % ll form.cgi
1 -rwx----- 1 shishir 974 Oct 31 22:15 form.cgi*
```

This has a permission of `0700` (octal), which means that no one (besides the owner) can read to, write from, and execute this file. Let's use the `chmod` command to change the permissions:

```
server: % chmod 755 form.cgi
server: % ll form.cgi
1 -rwxr-xr-x 1 shishir 974 Oct 31 22:15 form.cgi*
```

This changes the permissions so that users in the same group as `shishir`, as well as all other users have the permission to read from, and execute this file.

Make sure that the program is *world-executable*. Usually, the permissions need to be similar to `-rwxr-xr-x` (full read-

write-execute for the individual owner, read and execute permissions for the group and the world) when you do a long listing for the program file. Usually, this can be fixed with a command (see *chmod*, page 39) like:

```
server: % chmod a+x yourscript.cgi
```

## • ***Path Problems***

Make sure the paths included in the program code itself are correct. Nearly all client home directories start with the `/usr/home/username/` tree, and nearly all web sites start in the `/usr/home/username/html/` or `/usr/home/username/public_html/` directory.

## • ***Compatibility Problems***

If you obtained the program from an outside source (as many of our clients do), check with any readme files or similar instructions in the program to make sure that the program is compatible with BSD-style Unix (specifically **BSDI/OS 2.1**), and the **Apache** web server (version 1.1.3 or later). Also, check those guidelines for information on file ownership for the program, as the program may need to be owned by the `web` user for it to operate properly in conjunction with the web server.

# Perl-specific Troubleshooting

There are literally thousands of things that could stop a Perl script from executing properly. The Official Perl Website (<http://www.perl.com>) gives the following as a list of questions to ask yourself about your script *before* you ask anyone else:

### ***Q: Who owns the script?***

A: I do.

### ***Q: What are the script's permissions?***

A: They were `0600`, but I just realized that's not executable. I'd better make it `0755` instead.

### ***Q: Under what user ID does the server run its CGI programs?***

A: `web` (Oops, it can't write my files or directories.)

### ***Q: Can the server's user ID write any files you're trying to write?***

A: Nope – I own the files, but it doesn't *run* as me and the permissions are `0600` instead of `0666`. I guess that's why it can't open my own files.

### ***Q: What happens when you run it interactively?***

A: I didn't know you could run these interactively because I never bothered to read the documentation for the `CGI.pm` library.

### ***Q: What's in my site's error log?***

A: I never thought to look. Oh, there it is in `/usr/home/username/www/logs` – never mind.

### ***Q: What's the Perl version and OS version?***

A: Perl version 5.004 (as of this writing), BSDI/OS 2.1 (Try using `perl -v` and `uname -a` to find out)

### ***Q: What's the library version?***

A: `grep -i version` in the library, or for `CGI.pm`, do this:

```
server: % perl -le 'use CGI; print $CGI::VERSION'
2.21
```

### ***Q: What's the path to your perl executable specified in your script?***

A: /usr/bin/perl or /usr/bin/perl5 (oops, of course it can't find it)

**Q: What happens when you add Perl's -w flag?**

A: It tells me about my silly mistakes that are listed in detail in the `perldiag` manpage where I diligently looked them up.

**Q: What happens when you add `use strict`?**

A: It makes me declare my variables and quote my strings and finds all these silly errors which I've carefully corrected using my `()` declarations, `use vars`, and quotes.

**Q: Did you remember to output the MIME type before any other non-header output (other headers might be `Location:` or `Set-Cookie:`)**

A: Oh, right. It has to be a valid header and *then* a valid body. I guess I need to say this stuff earlier than I was doing, and make sure I actually put *two* newlines out, not just one:

```
print "Set-cookie: GroversDelight\n";
print "Content-Type: text/html\n\n"; # <- two newlines
print "<HEAD><TITLE>Sample Title</TITLE></HEAD>\n";
```

**Q: Did you remember to flush `STDOUT` at the top of your script so the MIME type gets out before any errors?**

A: Nope – gosh, no wonder it doesn't get out before it bombs! I guess I better add this to the top:  
`$| = 1`

**Q: What happens when you check return values of each and every one of your syscalls?**

A: That seems like way too much work, but sure enough, just as soon as I added something like

```
open(FILE, ">some_file")
|| die("can't write some_file: $!");
```

the error log showed that `#!` contains `Permission denied` or `No such file or directory`, and everything became clear.

**Q: Did you use the standard `CGI.pm` module to do this for you instead of parsing by hand (which would be a really idiotic idea) or use the more limited `cgi-lib.pl` library?**

A: Gee, you mean somebody else has actually done this kind of thing before? I didn't know I didn't have to do it all myself, and that I could get the latest version of the neat CGI module from [http://www.perl.com/cgi-bin/cgi\\_mod?modules=CGI](http://www.perl.com/cgi-bin/cgi_mod?modules=CGI).

**Q: Did you remember to post to `comp.infosystems.www.authoring.misc`, instead of blasting the `comp.lang.perl.*` groups with questions that aren't related to Perl in the least?**

A: Nope – is that why I don't get any useful answers but just a bunch of flames instead?

**Q: Are you getting the "Server: Error 500" message?**

A: You can get a server error for the following reasons:

- If the script does not contain the `#!/usr/bin/perl` or `#!/usr/bin/perl5` header line that points to the Perl interpreter, or if the path to the interpreter (and/or a library file) is invalid.
- If the first line output from the script is not a valid HTTP header (i.e. "Content-type: text/html"), or if there *isn't* a blank line after the header data.
- If there is a syntax error in your script. To be safe, always run it from the command-line *before* you set it up to run automatically and depend on it to work.