

Implementing Custom CGI scripts

Our UNIX web servers have the capability to run CGI scripts in your very own "cgi-local" directory. Scripts may be written in Perl, Unix SH, KSH, CSH, and C (NOT C++) languages.

Perl is our language of choice as it is a world standard and is well suited to CGI. In addition, Perl code does not require manual compilations whereas C code must be compiled* on our web servers prior to use. NT hosting customers have a "cgi-bin" directory. Customers may write scripts in PERL, C, C++.

Here are some helpful tips to follow when installing scripts:

Shell Scripts (UNIX only)

1. Upload to your cgi-local directory to ensure proper file permission settings
2. Upload in ASCII transfer mode (and NOT BINARY mode)
3. The first line of each script must read: `#!/bin/sh` , `#!/bin/csh` or `#!/bin/ksh` based on whichever shell scripts you prefer using.
4. Reference the script using `/cgi-local` (and NOT `/cgi-bin`)
5. Always remember to include `echo "Content-type: text/html\n\n"`

Perl Scripts

1. Upload to your cgi-local directory to ensure proper file permission settings
2. Upload in ASCII transfer mode (and NOT BINARY mode)
3. The first line of each script must read: `#!/usr/local/bin/perl`
4. Use the Perl script checker in your Control Panel to check syntax.
5. Reference the script using `/cgi-local` (and NOT `/cgi-bin`)
6. Always remember to include `print "Content-type: text/html\n\n";` or alternatively using the Perl module `CGI.pm` (If you do not, your scripts will not run and you will get an Internal Server Error message).

```
use CGI qw(:cgi-lib :standard);  
print header();
```

If a script calls another file within your account, but the script does NOT require a URL, you need to use the system path. Instead of using the absolute path to your home directory ("`/www26/web/someid`"), you should instead use the **DOCUMENT_ROOT** environment variable (`$ENV{DOCUMENT_ROOT}` in Perl) to determine the path of your files or programs within a script.

Change this: `/www23/web/yourid/data/fact.html`
To this: `$ENV{DOCUMENT_ROOT}/data/fact.html`

- The system path to the **sendmail** program on our UNIX server is `/usr/lib/sendmail`

- The system path to the **date** command on our UNIX server is **/sbin/date**

Automated Tasks (cron.sh)

A cron job can be set up to automatically run a script(s) within your cgi-local directory (if available). We can schedule one cron job per account to run between the hours of 2:00AM and 4:00AM ET on a daily basis. To request a cron job, you will first need to create a shell script which includes the system path to the script(s) you wish to have executed.

Example:

```
#!/bin/sh
```

```
/u/web/userid/cgi-local/script1.pl  
/u/web/userid/cgi-local/script2.pl
```

In the example above, two scripts located within the account's cgi-local directory will be run. You can add or subtract lines depending on how many scripts you wish to have executed. Be sure to substitute the userid of the site in question. URLs or references to server numbers (e.g. /www23/testco/... will NOT work.)

The newly created shell script must be named "cron.sh" and be uploaded to your hosting account's cgi-local directory in ASCII transfer mode. When complete, send your request to Technical Support. If desired, you can specify a time between the hours of 2:00 AM and 4:00 AM EST that you would like the cron job to be scheduled. If no time is specified, the cron job will be scheduled to run sometime during that period of time.

Once your cron job is scheduled, you can update your cron.sh as needed without contacting Technical Support unless you wish to change the scheduled run time.

C Compilations

We can run only two C compilations for you. Therefore, we suggest Perl as an alternative scripting language. C++ is NOT supported.

If you would like to implement code that is written in C, you will first need to create and upload, along with the source code, a makefile to your hosting account's "cgi-local" directory. Once the makefile and code have been uploaded, you will need to contact your Affiliate and request that we compile the code. Realize that YOU must create the makefile; we cannot compile the code without a makefile.

Shown below is the code for a simple makefile. Note that we are unable to assist in the creation or editing of the makefile. If you need further information on creating makefiles, please see our **Makefiles Links** section.

The following is an example of a simple makefile:

```
CC=cc  
AR=ar
```

```
test: test.cgi test.cgi:  
test.h test.c ${CC} -o test.cgi test.c
```

Makefiles Links

[Make: A Tutorial](#): An overview of Makefiles that includes step-by-step information on how to set them up. A general knowledge of UNIX, although helpful, is not required in order to understand this tutorial.

[GNU Make](#): A comprehensive technical manual providing details of Make, the program called by Makefile. A general understanding of UNIX is required in order to best utilize this manual.

***** Note *****

As we make only C-based makefiles, when using either of the above-mentioned Makefile documents, disregard any Makefile types that are not C.