NAME

FCGI - Fast CGI module

SYNOPSIS

```
use FCGI;

my $count = 0;
my $request = FCGI::Request();

while($request->Accept() >= 0) {
print("Content-type: text/html\r\n\r\n", ++$count);
}
```

DESCRIPTION

Functions:

FCGI::Request

Creates a request handle. It has the following optional parameters:

input perl file handle (default: *STDIN)
output perl file handle (default: *STDOUT)
error perl file handle (default: *STDERR)
These filehandles will be setup to act as input/output/error on successful Accept.

environment hash reference (default: %ENV)
The hash will be populated with the environment.

socket (default: 0)
Socket to communicate with the server. Can be the result of the OpenSocket function. For the moment, it's the file descriptor of the socket that should be passed. This may change in the future.

You should only use your own socket if your program is not started by a process manager such as mod_fastcgi (except for the FastCgiExternalServer case) or cgi-fcgi. If you use the option, you have to let your FastCGI server know which port (and possibly server) your program is listening on. See remote.pl for an example.

flags (default: 0)
Possible values:

FCGI::FAIL_ACCEPT_ON_INTR
If set, Accept will fail if interrupted. It not set, it will just keep on waiting.

Example usage: my `$req` = FCGI::Request;

or: my `%env`; my `$in` = new IO::Handle; my `$out` = new IO::Handle; my `$err` = new IO::Handle; my `$req` = FCGI::Request($in, `$out`, `$err`, %env);

FCGI::OpenSocket(path, backlog)

Creates a socket suitable to use as an argument to Request.

path Pathname of socket or colon followed by local tcp port. Note that some systems take file permissions into account on Unix domain sockets, so you'll have to make sure that the server can write to the created file, by changing the umask before the call and/or changing permissions and/or group of the file afterwards.

backlog Maximum length of the queue of pending connections. If a connection request arrives with the queue full the client may receive an error with an indication of ECONNREFUSED.

FCGI::CloseSocket(socket)
Close a socket opened with OpenSocket.

`$req`->*Accept()*
Accepts a connection on `$req`, attaching the filehandles and populating the environment hash. Returns 0 on success. If a connection has been accepted before, the old one will be finished first.

Note that unlike with the old interface, no die and warn handlers are installed by default. This means that if you are not running an sfio enabled perl, any warn or die message will not end up in the server's log by default. It is advised you set up die and warn handlers yourself. FCGI.pm contains an example of die and warn handlers.

`$req`->*Finish()*
Finishes accepted connection. Also detaches filehandles.

`$req`->*Flush()*
Flushes accepted connection.

`$req`->*Detach()*
Temporarily detaches filehandles on an accepted connection.

`$req`->*Attach()*
Re-attaches filehandles on an accepted connection.

`$req`->*LastCall()*
Tells the library not to accept any more requests on this handle. It should be safe to call this method from signal handlers.

Note that this method is still experimental and everything about it, including its name, is subject to change.

`$env` = `$req`->*GetEnvironment()*
Returns the environment parameter passed to FCGI::Request.

($in, `$out`, `$err`) = `$req`->*GetHandles()*
Returns the file handle parameters passed to FCGI::Request.

`$isfcgi` = `$req`->*IsFastCGI()*
Returns whether or not the program was run as a FastCGI.

LIMITATIONS

FCGI.pm isn't Unicode aware, only characters within the range 0x00-0xFF are supported. Attempts to output strings containing characters above 0xFF results in a exception: (F) `Wide character in %s`.

Users who wants the previous (FCGI.pm <= 0.68) incorrect behavior can disable the exception by using the `bytes` pragma.

```
{
use bytes;
print "\x{263A}";
}
```

AUTHOR

Sven Verdoolaege <skimo@kotnet.org>