

Re: Most dependable way to run system commands

Re: Most dependable way to run system commands

Source: <http://coding.derkeiler.com/Archive/Perl/perl.beginners/2007-12/msg00315.html>

- *From:* iaccounts@xxxxxxxxxx (Steve Bertrand)
 - *Date:* Thu, 13 Dec 2007 00:23:55 -0500
-

I am heavily modifying (re-writing) some Perl scripts that are bundled within the dialup_admin web interface for FreeRADIUS.

Eventually I'd like to bring the execution of a few system commands (executing mysql binary and importing an SQL file into it) into Perl, however, for now I just need it to work.

Well, that's pragmatic. :-)

LOL, I was hoping to quickly justify my needs to an end :)

But it's generally the best way: Find the answer first, improve upon it second.

Generally, it's always my approach, but I'm sure as any other person here, we've all had to deal with instances where this is but a lucky circumstance. IANWNAPP (I am no where near a professional programmer).

There are few things that are 100% certain.

I was in a hurry when I wrote that, so just like many of the statistics we hear from politicians or some organizations, it came quickly out of the wrong spot.

AFAICT, I've got the backtick option, system or eval.

You don't mean eval. You think you mean exec, but you don't mean that either, probably.

Re: Most dependable way to run system commands

Being honest, I don't know what I meant. I truthfully didn't have time to read through the docs for all the comparisons (although I looked at `eval` in particular: `perldoc -f eval` which didn't seem proper for my specific execution issue (it dealt more with executing Perl code)).

I think you're looking for the program's exit status. Traditionally on Unix and many similar systems, the exit status is an integer, with 0 meaning "normal exit" and anything else meaning that something went wrong. That's the value that's used by programs that run other programs (such as the 'make' system utility) and need to know when a step has failed. In Perl, you can access the exit status with the `$?` variable after running a command via `system` or `backticks`. Be