

NAME

`cpan` - easily interact with CPAN from the command line

SYNOPSIS

```
# with arguments and no switches, installs specified modules
cpan module_name [ module_name ... ]

# with switches, installs modules with extra behavior
cpan [-cfgimtTw] module_name [ module_name ... ]

# with just the dot, install from the distribution in the
# current directory
cpan .

# without arguments, starts CPAN.pm shell
cpan

# dump the configuration
cpan -J

# load a different configuration to install Module::Foo
cpan -j some/other/file Module::Foo

# without arguments, but some switches
cpan [-ahrvACDlLO]
```

DESCRIPTION

This script provides a command interface (not a shell) to CPAN. At the moment it uses `CPAN.pm` to do the work, but it is not a one-shot command runner for `CPAN.pm`.

Options

- a Creates a `CPAN.pm` autobundle with `CPAN::Shell->autobundle`.
- A module [module ...]
Shows the primary maintainers for the specified modules.
- c module
Runs a 'make clean' in the specified module's directories.
- C module [module ...]
Show the *Changes* files for the specified modules
- D module [module ...]
Show the module details.
- f Force the specified action, when it normally would have failed. Use this to install a module even if its tests fail. When you use this option, `-i` is not optional for installing a module when you need to force it:

```
% cpan -f -i Module::Foo
```
- F Turn off `CPAN.pm`'s attempts to lock anything. You should be careful with this since you might end up with multiple scripts trying to muck in the same directory. This isn't so much of a concern if you're loading a special config with `-j`, and that config sets up its own work directories.
- g module [module ...]
Downloads to the current directory the latest distribution of the module.
- G module [module ...]
UNIMPLEMENTED

Download to the current directory the latest distribution of the modules, unpack each distribution, and

create a git repository for each distribution.

If you want this feature, check out Yanick Champoux's `Git::CPAN::Patch` distribution.

- h Print a help message and exit. When you specify `-h`, it ignores all of the other options and arguments.
- i Install the specified modules.
- I Load `local::lib` (think like `-I` for loading lib paths).
- j `Config.pm`
Load the file that has the CPAN configuration data. This should have the same format as the standard `CPAN/Config.pm` file, which defines `$CPAN::Config` as an anonymous hash.
- J Dump the configuration in the same format that `CPAN.pm` uses. This is useful for checking the configuration as well as using the dump as a starting point for a new, custom configuration.
- l List all installed modules with their versions
- L author [author ...]
List the modules by the specified authors.
- m Make the specified modules.
- O Show the out-of-date modules.
- p Ping the configured mirrors
- P Find the best mirrors you could be using (but doesn't configure them just yet)
- r Recompiles dynamically loaded modules with `CPAN::Shell->recompile`.
- t Run a 'make test' on the specified modules.
- T Do not test modules. Simply install them.
- u Upgrade all installed modules. Blindly doing this can really break things, so keep a backup.
- v Print the script version and `CPAN.pm` version then exit.
- V Print detailed information about the cpan client.
- w UNIMPLEMENTED
Turn on cpan warnings. This checks various things, like directory permissions, and tells you about problems you might have.

Examples

```
# print a help message
cpan -h

# print the version numbers
cpan -v

# create an autobundle
cpan -a

# recompile modules
cpan -r

# upgrade all installed modules
cpan -u

# install modules ( sole -i is optional )
cpan -i Netscape::Bookmarks Business::ISBN

# force install modules ( must use -i )
```

```
cpan -fi CGI::Minimal URI
```

ENVIRONMENT VARIABLES

CPAN_OPTS

cpan splits this variable on whitespace and prepends that list to @ARGV before it processes the command-line arguments. For instance, if you always want to use `local:lib`, you can set `CPAN_OPTS` to `-I`.

EXIT VALUES

The script exits with zero if it thinks that everything worked, or a positive number if it thinks that something failed. Note, however, that in some cases it has to divine a failure by the output of things it does not control. For now, the exit codes are vague:

- 1 An unknown error
- 2 There was an external problem
- 4 There was an internal problem with the script
- 8 A module failed to install

TO DO

* one shot configuration values from the command line

BUGS

* none noted

SEE ALSO

Most behaviour, including environment variables and configuration, comes directly from `CPAN.pm`.

SOURCE AVAILABILITY

This code is in Github:

```
git://github.com/briandfoy/cpan_script.git
```

CREDITS

Japheth Cleaver added the bits to allow a forced install (`-f`).

Jim Brandt suggest and provided the initial implementation for the up-to-date and Changes features.

Adam Kennedy pointed out that `exit()` causes problems on Windows where this script ends up with a `.bat` extension

AUTHOR

brian d foy, <bdfoy@cpan.org>

COPYRIGHT

Copyright (c) 2001-2013, brian d foy, All Rights Reserved.

You may redistribute this under the same terms as Perl itself.