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 [−cfFimtTw] module_name [ module_name ... ]

# use local::lib
cpan −I module_name [ module_name ... ]

# one time mirror override for faster mirrors
cpan −p ...

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

# without arguments, starts CPAN.pm shell
cpan

# without arguments, but some switches
cpan [−ahpruvACDLOPX]
```

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. This prints one line for each out-of-date module (meaning, modules locally installed but have newer versions on CPAN). Each line has three columns: module name, local version, and CPAN version.

−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 module [module ...]
: Install the specified modules. With no other switches, this switch is implied.

−I
: Load `local::lib` (think like `−I` for loading lib paths). Too bad `−l` was already taken.

−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.

−M mirror1,mirror2,...
: A comma-separated list of mirrors to use for just this run. The `−P` option can find them for you automatically.

−n
: Do a dry run, but don't actually install anything. (unimplemented)

−O
: Show the out-of-date modules.

−p
: Ping the configured mirrors and print a report

−P
: Find the best mirrors you could be using and use them for the current session.

−r
: Recompiles dynamically loaded modules with CPAN::Shell−>recompile.

−s
: Drop in the CPAN.pm shell. This command does this automatically if you don't specify any arguments.

−t module [module ...]
: 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.

−x module [module ...]
: Find close matches to the named modules that you think you might have mistyped. This requires the optional installation of Text::Levenshtein or Text::Levenshtein::Damerau.

−X
: Dump all the namespaces to standard output.

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::Booksmarks Business::ISBN

# force install modules ( must use −i )
cpan −fi CGI::Minimal URI

# install modules but without testing them
cpan −Ti CGI::Minimal URI
```

Environment variables

There are several components in CPAN.pm that use environment variables. The build tools, ExtUtils::MakeMaker and Module::Build use some, while others matter to the levels above them. Some of these are specified by the Perl Toolchain Gang:

Lancaster Concensus: <https://github.com/Perl−Toolchain−Gang/toolchain−site/blob/master/lancaster−consensus.md>

Oslo Concensus: <https://github.com/Perl−Toolchain−Gang/toolchain−site/blob/master/oslo−consensus.md>

NONINTERACTIVE_TESTING
: Assume no one is paying attention and skips prompts for distributions that do that correctly. `cpan(1)` sets this to `1` unless it already has a value (even if that value is false).

PERL_MM_USE_DEFAULT
: Use the default answer for a prompted questions. `cpan(1)` sets this to `1` unless it already has a value (even if that value is false).

CPAN_OPTS
: As with `PERL5OPT`, a string of additional `cpan(1)` options to add to those you specify on the command line.

CPANSCRIPT_LOGLEVEL
: The log level to use, with either the embedded, minimal logger or Log::Log4perl if it is installed. Possible values are the same as the `Log::Log4perl` levels: `TRACE`, `DEBUG`, `INFO`, `WARN`, `ERROR`, and `FATAL`. The default is `INFO`.

GIT_COMMAND
: The path to the `git` binary to use for the Git features. The default is `/usr/local/bin/git`.

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       The 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 in the CPAN.pm repository:

```
        https://github.com/andk/cpanpm
```

The source used to be tracked separately in another GitHub repo, but the canonical source is now in the above repo.

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