NAME

CPANPLUS::inc

DESCRIPTION

OBSOLETE

NAME

CPANPLUS::inc - runtime inclusion of privately bundled modules

SYNOPSIS

```
### set up CPANPLUS::inc to do it's thing ###
BEGIN { use CPANPLUS::inc };

### enable debugging ###
use CPANPLUS::inc qw[DEBUG];
```

DESCRIPTION

This module enables the use of the bundled modules in the `CPANPLUS/inc` directory of this package. These modules are bundled to make sure `CPANPLUS` is able to bootstrap itself. It will do the following things:

Put a coderef at the beginning of `@INC`

This allows us to decide which module to load, and where to find it. For details on what we do, see the `INTERESTING MODULES` section below. Also see the `CAVEATS` section.

Add the full path to the `CPANPLUS/inc` directory to `$ENV{PERL5LIB}`.

This allows us to find our bundled modules even if we spawn off a new process. Although it's not able to do the selective loading as the coderef in `@INC` could, it's a good fallback.

METHODS

CPANPLUS::inc->inc_path()

Returns the full path to the `CPANPLUS/inc` directory.

CPANPLUS::inc->my_path()

Returns the full path to be added to `@INC` to load `CPANPLUS::inc` from.

CPANPLUS::inc->installer_path()

Returns the full path to the `CPANPLUS/inc/installers` directory.

CPANPLUS::inc->original_perl5lib

Returns the value of $ENV{PERL5LIB} the way it was when `CPANPLUS::inc` got loaded.

CPANPLUS::inc->original_perl5opt

Returns the value of $ENV{PERL5OPT} the way it was when `CPANPLUS::inc` got loaded.

CPANPLUS::inc->original_inc

Returns the value of @INC the way it was when `CPANPLUS::inc` got loaded.

CPANPLUS::inc->limited_perl5opt(@modules);

Returns a string you can assign to `$ENV{PERL5OPT}` to have a limited include facility from `CPANPLUS::inc`. It will roughly look like:

```
-I/path/to/cpanplus/inc -MCPANPLUS::inc=module1,module2
```

CPANPLUS::inc->interesting_modules()

Returns a hashref with modules we're interested in, and the minimum version we need to find.

It would looks something like this:

```
{   File::Fetch             => 0.06,
    IPC::Cmd                => 0.22,
    ....
}
```

INTERESTING MODULES

CPANPLUS::inc doesn't even bother to try find and find a module it's not interested in. A list of *interesting modules* can be obtained using the `interesting_modules` method described above.

Note that all subclassed modules of an `interesting module` will also be attempted to be loaded, but a version will not be checked.

When it however does encounter a module it is interested in, it will do the following things:

Loop over your @INC

> And for every directory it finds there (skipping all non directories -- see the CAVEATS section), see if the module requested can be found there.

Check the version on every suitable module found in @INC

> After a list of modules has been gathered, the version of each of them is checked to find the one with the highest version, and return that as the module to `use`.
>
> This enables us to use a recent enough version from our own bundled modules, but also to use a *newer* module found in your path instead, if it is present. Thus having access to bugfixed versions as they are released.
>
> If for some reason no satisfactory version could be found, a warning will be emitted. See the DEBUG section for more details on how to find out exactly what CPANPLUS::inc is doing.

DEBUG

Since this module does `Clever Things` to your search path, it might be nice sometimes to figure out what it's doing, if things don't work as expected. You can enable a debug trace by calling the module like this:

```
use CPANPLUS::inc 'DEBUG';
```

This will show you what CPANPLUS::inc is doing, which might look something like this:

```
CPANPLUS::inc: Found match for 'Params::Check' in
'/opt/lib/perl5/site_perl/5.8.3' with version '0.07'
CPANPLUS::inc: Found match for 'Params::Check' in
'/my/private/lib/CPANPLUS/inc' with version '0.21'
CPANPLUS::inc: Best match for 'Params::Check' is found in
'/my/private/lib/CPANPLUS/inc' with version '0.21'
```

CAVEATS

This module has 2 major caveats, that could lead to unexpected behaviour. But currently I don't know how to fix them, Suggestions are much welcomed.

On multiple `use lib` calls, our coderef may not be the first in @INC

> If this happens, although unlikely in most situations and not happening when calling the shell directly, this could mean that a lower (too low) versioned module is loaded, which might cause

failures in the application.

Non-directories in @INC

Non-directories are right now skipped by CPANPLUS::inc. They could of course lead us to newer versions of a module, but it's too tricky to verify if they would. Therefor they are skipped. In the worst case scenario we'll find the sufficing version bundled with CPANPLUS.