NAME

inc::latest - use modules bundled in inc/ if they are newer than installed ones

SYNOPSIS

```
# in Build.PL
use inc::latest 'Module::Build';
```

DESCRIPTION

The inc::latest module helps bootstrap configure-time dependencies for CPAN distributions. These dependencies get bundled into the inc directory within a distribution and are used by Build.PL (or Makefile.PL).

Arguments to inc::latest are module names that are checked against both the current @INC array and against specially-named directories in inc. If the bundled version is newer than the installed one (or the module isn't installed, then, the bundled directory is added to the start of <@INC> and the module is loaded from there.

There are actually two variations of inc::latest — one for authors and one for theinc directory. For distribution authors, the inc::latest installed in the system will record modules loaded via inc::latest and can be used to create the bundled files in inc, including writing the second variation as inc/latest.pm.

This second inc::latest is the one that is loaded in a distribution being installed (e.g. from Build.PL). This bundled inc::latest is the one that determines which module to load.

Special notes on bundling

The inc::latest module creates bundled directories based on the packlist file of an installed distribution. Even though inc::latest takes module name arguments, it is better to think of it as bundling and making available entire *distributions*. When a module is loaded through inc::latest it looks in all bundled distributions in inc/ for a newer module than can be found in the existing @INC array.

Thus, the module-name provided should usually be the "top-level" module name of a distribution, though this is not strictly required. For example, Module::Build has a number of heuristics to map module names to packlists, allowing users to do things like this:

```
use inc::latest 'Devel::AssertOS::Unix';
```

even though Devel::AssertOS::Unix is contained within the Devel-CheckOS distribution.

At the current time, packlists are required. Thus, bundling dual-core modules may require a 'forced install' over versions in the latest version of perl in order to create the necessary packlist for bundling.

USAGE

When calling use, the bundled inc::latest takes a single module name and optional arguments to pass to that module's own import method.

```
use 'inc::latest' 'Foo::Bar' qw/foo bar baz/;
```

Author-mode

You are in author-mode inc::latest if any of the Author-mode methods are available. For example:

```
if ( inc::latest->can('write') ) {
inc::latest->write('inc');
}
```

loaded_modules()

```
my @list = inc::latest->loaded_modules;
```

This takes no arguments and always returns a list of module names requested for loading via "use inc::latest successful or not.

write()

```
inc::latest->write( 'inc' );
```

This writes the bundled version of inc::latest to the directory name given as an argument. It almost all cases, it should be '`inc`'.

bundle_module()

```
for my $mod ( inc::latest->loaded_modules ) {
inc::latest->bundle_module($mod, $dir);
}
```

If `$mod` corresponds to a packlist, then this function creates a specially-named directory in `$dir` and copies all .pm files from the modlist to the new directory (which almost always should just be 'inc'). For example, if Foo::Bar is the name of the module, and `$dir` is 'inc', then the directory would be 'inc/inc_Foo-Bar' and contain files like this:

```
inc/inc_Foo-Bar/Foo/Bar.pm
```

Currently, `$mod` **must** have a packlist. If this is not the case (e.g. for a dual-core module), then the bundling will fail. You may be able to create a packlist by forced installing the module on top of the version that came with core Perl.

As bundled in inc/

All methods are private. Only the `import` method is public.

AUTHOR

Eric Wilhelm <ewilhelm@cpan.org>, David Golden <dagolden@cpan.org>

COPYRIGHT

SEE ALSO

Module::Build