

**NAME**

`ExtUtils::Embed` – Utilities for embedding Perl in C/C++ applications

**SYNOPSIS**

```
perl -MExtUtils::Embed -e xsinit
perl -MExtUtils::Embed -e ccopts
perl -MExtUtils::Embed -e ldopts
```

**DESCRIPTION**

`ExtUtils::Embed` provides utility functions for embedding a Perl interpreter and extensions in your C/C++ applications. Typically, an application *Makefile* will invoke `ExtUtils::Embed` functions while building your application.

**@EXPORT**

`ExtUtils::Embed` exports the following functions:

`xsinit()`, `ldopts()`, `ccopts()`, `perl_inc()`, `ccflags()`, `ccdlflags()`, `xsi_header()`, `xsi_protos()`, `xsi_body()`

**FUNCTIONS****`xsinit()`**

Generate C/C++ code for the XS initializer function.

When invoked as `perl -MExtUtils::Embed -e xsinit --` the following options are recognized:

**-o <output filename>** (Defaults to **perlxi.c**)

**-o STDOUT** will print to STDOUT.

**-std** (Write code for extensions that are linked with the current Perl.)

Any additional arguments are expected to be names of modules to generate code for.

When invoked with parameters the following are accepted and optional:

`xsinit($filename, $std, [@modules])`

Where,

**\$filename** is equivalent to the **-o** option.

**\$std** is boolean, equivalent to the **-std** option.

**[@modules]** is an array ref, same as additional arguments mentioned above.

Examples

```
perl -MExtUtils::Embed -e xsinit -- -o xsinit.c Socket
```

This will generate code with an `xs_init` function that glues the perl `Socket::bootstrap` function to the C `boot_Socket` function and writes it to a file named `xsinit.c`.

Note that DynaLoader is a special case where it must call `boot_DynaLoader` directly.

```
perl -MExtUtils::Embed -e xsinit
```

This will generate code for linking with DynaLoader and each static extension found in `$Config{static_ext}`. The code is written to the default file name `perlxi.c`.

```
perl -MExtUtils::Embed -e xsinit -- -o xsinit.c \
    -std DBI DBD::Oracle
```

Here, code is written for all the currently linked extensions along with code for DBI and DBD::Oracle.

If you have a working DynaLoader then there is rarely any need to statically link in any other extensions.

**`ldopts()`**

Output arguments for linking the Perl library and extensions to your application.

When invoked as `perl -MExtUtils::Embed -e ldopts --` the following options are recognized:



**-std**

Output arguments for linking the Perl library and any extensions linked with the current Perl.

**-I <path1:path2>**

Search path for ModuleName.a archives. Default path is @INC. Library archives are expected to be found as `/some/path/auto/ModuleName/ModuleName.a`. For example, when looking for `Socket.a` relative to a search path, we should find `auto/Socket/Socket.a`.

When looking for `DBD::Oracle` relative to a search path, we should find `auto/DBD/Oracle/Oracle.a`

Keep in mind that you can always supply `/my/own/path/ModuleName.a` as an additional linker argument.

`-- <list of linker args>`

Additional linker arguments to be considered.

Any additional arguments found before the `--` token are expected to be names of modules to generate code for.

When invoked with parameters the following are accepted and optional:

`ldopts($std,[@modules],[@link_args],$path)`

Where:

**\$std** is boolean, equivalent to the `-std` option.

**[@modules]** is equivalent to additional arguments found before the `--` token.

**[@link\_args]** is equivalent to arguments found after the `--` token.

**\$path** is equivalent to the `-I` option.

In addition, when `ldopts` is called with parameters, it will return the argument string rather than print it to STDOUT.

Examples

```
perl -MExtUtils::Embed -e ldopts
```

This will print arguments for linking with libperl and extensions found in `$Config{static_ext}`. This includes libraries found in `$Config{libs}` and the first `ModuleName.a` library for each extension that is found by searching @INC or the path specified by the `-I` option. In addition, when `ModuleName.a` is found, additional linker arguments are picked up from the `extralibs.ld` file in the same directory.

```
perl -MExtUtils::Embed -e ldopts -- -std Socket
```

This will do the same as the above example, along with printing additional arguments for linking with the `Socket` extension.

```
perl -MExtUtils::Embed -e ldopts -- -std Msql -- \
-L/usr/msql/lib -lmysql
```

Any arguments after the second '`--`' token are additional linker arguments that will be examined for potential conflict. If there is no conflict, the additional arguments will be part of the output.

`perl_inc()`

For including perl header files this function simply prints:

```
-I$Config{archlibexp}/CORE
```

So, rather than having to say:

```
perl -MConfig -e 'print "-I$Config{archlibexp}/CORE"'
```

Just say:

```
perl -MExtUtils::Embed -e perl_inc
```



***ccflags()*, *ccdlflags()***

These functions simply print \$Config{ccflags} and \$Config{ccdlflags}

***ccopts()***

This function combines perl\_inc(), ccflags() and ccdlflags() into one.

***xsi\_header()***

This function simply returns a string defining the same EXTERN\_C macro as *perlmain.c* along with #including *perl.h* and *EXTERN.h*.

***xsi\_protos(@modules)***

This function returns a string of boot\_{\$ModuleName} prototypes for each @modules.

***xsi\_body(@modules)***

This function returns a string of calls to newXS() that glue the module *bootstrap* function to *boot\_ModuleName* for each @modules.

*xsinit()* uses the *xsi\_\** functions to generate most of its code.

**EXAMPLES**

For examples on how to use ExtUtils::Embed for building C/C++ applications with embedded perl, see perlembd.

**SEE ALSO**

perlembd

**AUTHOR**

Doug MacEachern <dougm AT osf DOT org>

Based on ideas from Tim Bunce <Tim DOT Bunce AT ig DOT co DOT uk> and *minimod.pl* by Andreas Koenig <k AT anna DOT in-berlin.de> and Tim Bunce.

