NAME

Sub::Defer – Defer generation of subroutines until they are first called

SYNOPSIS

```
use Sub::Defer;

my $deferred = defer_sub 'Logger::time_since_first_log' => sub {
   my $t = time;
   sub { time - $t };
};

 Logger->time_since_first_log; # returns 0 and replaces itself
 Logger->time_since_first_log; # returns time - $t
```

DESCRIPTION

These subroutines provide the user with a convenient way to defer creation of subroutines and methods until they are first called.

SUBROUTINES

defer_sub

```
my $coderef = defer_sub $name => sub { ... }, \%options;
```

This subroutine returns a coderef that encapsulates the provided sub – when it is first called, the provided sub is called and is –itself– expected to return a subroutine which will be goto'ed to on subsequent calls.

If a name is provided, this also installs the sub as that name – and when the subroutine is undeferred will re-install the final version for speed.

Exported by default.

Options

A hashref of options can optionally be specified.

package
: The package to generate the sub in. Will be overridden by a fully qualified `$name` option. If not specified, will default to the caller's package.

attributes
: The "Subroutine Attributes" in perlsub to apply to the sub generated. Should be specified as an array reference.

undefer_sub

```
my $coderef = undefer_sub \&Foo::name;
```

If the passed coderef has been deferred this will "undefer" it. If the passed coderef has not been deferred, this will just return it.

If this is confusing, take a look at the example in the "SYNOPSIS".

Exported by default.

defer_info

```
my $data = defer_info $sub;
my ($name, $generator, $options, $undeferred_sub) = @$data;
```

Returns original arguments to defer_sub, plus the undeferred version if this sub has already been undeferred.

Note that `$sub` can be either the original deferred version or the undeferred version for convenience.

Not exported by default.

undefer_all

```
undefer_all();
```

This will undefer all deferred subs in one go. This can be very useful in a forking environment where child processes would each have to undefer the same subs. By calling this just before you start forking children you can undefer all currently deferred subs in the parent so that the children do not have to do it. Note this may bake the behavior of some subs that were intended to calculate their behavior later, so it shouldn't be used midway through a module load or class definition.

Exported by default.

undefer_package

```
undefer_package($package);
```

This undefers all deferred subs in a package.

Not exported by default.

SUPPORT

See Sub::Quote for support and contact information.

AUTHORS

See Sub::Quote for authors.

COPYRIGHT AND LICENSE

See Sub::Quote for the copyright and license.