NAME

base - Establish an ISA relationship with base classes at compile time

SYNOPSIS

```
package Baz;
use base qw(Foo Bar);
```

DESCRIPTION

Unless you are using the `fields` pragma, consider this module discouraged in favor of the lighter-weight `parent`.

Allows you to both load one or more modules, while setting up inheritance from those modules at the same time. Roughly similar in effect to

```
package Baz;
BEGIN {
require Foo;
require Bar;
push @ISA, qw(Foo Bar);
}
```

When `base` tries to `require` a module, it will not die if it cannot find the module's file, but will die on any other error. After all this, should your base class be empty, containing no symbols, `base` will die. This is useful for inheriting from classes in the same file as yourself but where the filename does not match the base module name, like so:

```
# in Bar.pm
package Foo;
sub exclaim { "I can have such a thing?!" }

package Bar;
use base "Foo";
```

There is no *Foo.pm*, but because `Foo` defines a symbol (the `exclaim` subroutine), `base` will not die when the `require` fails to load *Foo.pm*.

`base` will also initialize the fields if one of the base classes has it. Multiple inheritance of fields is **NOT** supported, if two or more base classes each have inheritable fields the 'base' pragma will croak. See fields for a description of this feature.

The base class' `import` method is **not** called.

DIAGNOSTICS

Base class package "%s" is empty.
: base.pm was unable to require the base package, because it was not found in your path.

Class 'Foo' tried to inherit from itself
: Attempting to inherit from yourself generates a warning.

```
package Foo;
use base 'Foo';
```

HISTORY

This module was introduced with Perl 5.004_04.

CAVEATS

Due to the limitations of the implementation, you must use base *before* you declare any of your own fields.

SEE ALSO

fields