

NAME

Class::ISA -- report the search path for a class's ISA tree

SYNOPSIS

```
# Suppose you go: use Food::Fishstick, and that uses and
# inherits from other things, which in turn use and inherit
# from other things. And suppose, for sake of brevity of
# example, that their ISA tree is the same as:

@Food::Fishstick::ISA = qw(Food::Fish Life::Fungus Chemicals);
@Food::Fish::ISA = qw(Food);
@Food::ISA = qw(Matter);
@Life::Fungus::ISA = qw(Life);
@Chemicals::ISA = qw(Matter);
@Life::ISA = qw(Matter);
@Matter::ISA = qw();

use Class::ISA;
print "Food::Fishstick path is:\n ",
      join(", ", Class::ISA::super_path('Food::Fishstick')),
      "\n";
```

That prints:

```
Food::Fishstick path is:
Food::Fish, Food, Matter, Life::Fungus, Life, Chemicals
```

DESCRIPTION

Suppose you have a class (like `Food::Fish::Fishstick`) that is derived, via its `@ISA`, from one or more superclasses (as `Food::Fish::Fishstick` is from `Food::Fish`, `Life::Fungus`, and `Chemicals`), and some of those superclasses may themselves each be derived, via its `@ISA`, from one or more superclasses (as above).

When, then, you call a method in that class (`$fishstick->calories`), Perl first searches there for that method, but if it's not there, it goes searching in its superclasses, and so on, in a depth-first (or maybe "height-first" is the word) search. In the above example, it'd first look in `Food::Fish`, then `Food`, then `Matter`, then `Life::Fungus`, then `Life`, then `Chemicals`.

This library, `Class::ISA`, provides functions that return that list -- the list (in order) of names of classes Perl would search to find a method, with no duplicates.

FUNCTIONS

the function `Class::ISA::super_path($CLASS)`

This returns the ordered list of names of classes that Perl would search thru in order to find a method, with no duplicates in the list. `$CLASS` is not included in the list. `UNIVERSAL` is not included -- if you need to consider it, add it to the end.

the function `Class::ISA::self_and_super_path($CLASS)`

Just like `super_path`, except that `$CLASS` is included as the first element.

the function `Class::ISA::self_and_super_versions($CLASS)`

This returns a hash whose keys are `$CLASS` and its (super-)superclasses, and whose values are the contents of each class's `$VERSION` (or `undef`, for classes with no `$VERSION`).

The code for `self_and_super_versions` is meant to serve as an example for precisely the kind

of tasks I anticipate that `self_and_super_path` and `super_path` will be used for. You are strongly advised to read the source for `self_and_super_versions`, and the comments there.

CAUTIONARY NOTES

- * `Class::ISA` doesn't export anything. You have to address the functions with a "`Class::ISA::`" on the front.
- * Contrary to its name, `Class::ISA` isn't a class; it's just a package. Strange, isn't it?
- * Say you have a loop in the ISA tree of the class you're calling one of the `Class::ISA` functions on: say that `Food` inherits from `Matter`, but `Matter` inherits from `Food` (for sake of argument). If Perl, while searching for a method, actually discovers this cyclicity, it will throw a fatal error. The functions in `Class::ISA` effectively ignore this cyclicity; the `Class::ISA` algorithm is "never go down the same path twice", and cyclicities are just a special case of that.
- * The `Class::ISA` functions just look at @ISAs. But theoretically, I suppose, AUTOLOADs could bypass Perl's ISA-based search mechanism and do whatever they please. That would be bad behavior, tho; and I try not to think about that.
- * If Perl can't find a method anywhere in the ISA tree, it then looks in the magical class `UNIVERSAL`. This is rarely relevant to the tasks that I expect `Class::ISA` functions to be put to, but if it matters to you, then instead of this:

```
@supers = Class::Tree::super_path($class);
```

do this:

```
@supers = (Class::Tree::super_path($class), 'UNIVERSAL');
```

And don't say no-one ever told ya!

- * When you call them, the `Class::ISA` functions look at @ISAs anew -- that is, there is no memoization, and so if ISAs change during runtime, you get the current ISA tree's path, not anything memoized. However, changing ISAs at runtime is probably a sign that you're out of your mind!

COPYRIGHT

Copyright (c) 1999, 2000 Sean M. Burke. All rights reserved.

This library is free software; you can redistribute it and/or modify it under the same terms as Perl itself.

AUTHOR

Sean M. Burke sburke@cpan.org