Ruby trunk - Bug #8066

Inconsistency in ancestors chain

03/10/2013 08:33 PM - prijutme4ty (Ilya Vorontsov)

Status:	Rejected		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	1.9.3p0, 2.0.0p0	**Backport:**	2.3: UNKNOWN, 2.4: UNKNOWN

Description

Method including have some inconsistencies. Let's define module and include(or prepend) and then include it in classes in different order.

module M; end
Class.send :include, M
Module.send :include, M
Class.ancestors

=> [Class, M, Module, M, Object, Kernel, BasicObject]

module M; end
Module.send :include, M
Class.send :include, M
Class.ancestors

=> [Class, Module, M, Object, Kernel, BasicObject]

We see that ancestor chains are different. Is it a spec(i didn't find it in tests) or a bug?

Related issues:

Is duplicate of Ruby trunk - Feature #1586: Including a module already presen...	**Rejected**

History

#1 - 03/11/2013 02:08 AM - Student (Nathan Zook)

This is the behaviour I would expect in all versions of Ruby. The ancestor chain is set at the time that a class is created, and is updated at the time that a module is included in that class. What does not happen is that the chain for a class is recomputed when a module is included in an ancestor of the class.

So:

module N ; end
class C ; include N ; end
module M ; end
N.send :include, M
N.ancestors
=> [N, M]
C.ancestors
=> [C, N, Object, Kernel, BasicObject]

Your example is the natural result of this behaviour. Note, however, that reincluding a module will pick up its current includes:
C.send :include, N
C.ancestors
=> [C, N, M, Object, Kernel, BasicObject]

#2 - 03/11/2013 05:07 PM - prijutme4ty (Ilya Vorontsov)

Isn't it just a conseqency of the fact that module can't be included twice and because of that including a module in parent class prevents including module into a child class.
This behavior disallows for example creating decorators by prepending named modules.

#3 - 11/29/2017 07:14 AM - matz (Yukihiro Matsumoto)

- Status changed from Open to Rejected

The incompatibility that will be caused by the change is intolerable. Any attempt to address the change will make the language far more complex than it currently is.
I have to reject.

Matz.