Ruby - Bug #5236

Including a module in a superclass after it has been included in a subclass leads to infinite recursion if the module uses `super`

08/27/2011 08:39 AM - myronmarston (Myron Marston)

Status:	Closed		
Priority:	Normal		
Assignee:	nobu (Nobuyoshi Nakada)		
Target version:	2.0.0		
ruby -v:	1.9.x	**Backport:**	

Description

Under these particular circumstances, you get infinite recursion and a system stack error on 1.9 but not 1.8:

- Create a module that has a method that uses super
- Include that module in a class after it has already been included in one of its subclasses
- Instantiate an object of said subclass and call the method

On 1.8, the super works as expected. On 1.9 you get a SystemStackError. Here's example code that demonstrates the issue:

example.rb

```
module MyModule
def some_method; super; end
end

class MyBaseClass; end

class MySubClass < MyBaseClass;
include MyModule
end

# To trigger this bug, we must include the module in the base class after

# the module has already been included in the subclass. If we move this line

# above the subclass declaration, this bug will not occur.

MyBaseClass.send(:include, MyModule)

MySubClass.new.some_method
```

The output

```
 ruby --version
ruby 1.8.7 (2011-02-18 patchlevel 334) [i686-darwin10.6.0]
 ruby example.rb
example.rb:2:in some_method': super: no superclass method some_method' for #MySubClass:0x1001bc2d0 (NoMethodError)
from example.rb:2:in `some_method'
from example.rb:13

 ruby --version
ruby 1.9.2p180 (2011-02-18 revision 30909) [x86_64-darwin10.6.0]
 ruby example.rb
example.rb:2: stack level too deep (SystemStackError)
```

I've tried it on 1.9.3.preview-1 and I get the SystemStackError there, too. I would expect 1.9 to act like 1.8 here, and not infinitely recurse on itself.

Related issues:	
Related to Ruby - Bug #3351: stack overflow on super	**Closed**
Related to Ruby - Feature #1586: Including a module already present in ancest...	**Rejected**

History

#1 - 10/31/2011 09:31 AM - pda (Paul Annesley)

This bug causes a lot of trouble for RSpec developers and end users alike:
https://github.com/rspec/rspec-rails/issues/371#issuecomment-1430232

It'd be great to see it fixed on Ruby's side.

Cheers!
Paul

#2 - 03/11/2012 03:24 PM - ko1 (Koichi Sasada)

- Assignee set to ko1 (Koichi Sasada)

- Target version set to 2.0.0

#3 - 03/18/2012 06:46 PM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Assigned

#4 - 11/26/2012 09:23 AM - ko1 (Koichi Sasada)

- Assignee changed from ko1 (Koichi Sasada) to nobu (Nobuyoshi Nakada)

nobu, could you check it?

#5 - 01/23/2013 04:45 PM - nobu (Nobuyoshi Nakada)

- Status changed from Assigned to Closed

Already fixed probably at r36595 etc.