

Ruby master - Bug #11189

alias prepended module

05/27/2015 07:58 PM - ko1 (Koichi Sasada)

Status: Open	
Priority: Normal	
Assignee: matz (Yukihiro Matsumoto)	
Target version:	
ruby -v:	Backport: 2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN

Description

```
module P
  def m1
    p :P_m1
    super
  end

  def m2
    p :P_m2
    super
  end
end

class C0
  def m1
    p :C0_m1
  end
end

class C1 < C0
  def m1
    p :C1_m1
    super
  end
  prepend P
  alias m2 m1
end

C1.new.m2
```

#####

```
:P_m2
:P_m1
:C0_m1
```

#####super ##### 2 #####super
#####2#####Ruby #####

#####

- alias C1 #####C1#m2 P#m1 #####

#####

- m2 #####
- P#m2 #####
- P#m2 super #####
- C1#m2 #####P#m1 ##### alias #####
- P#m1 #####super #####
- ##### C1#m1 #####
- C0#m1 #####

```
P#m2 -> P#m1
```

```
C1#foo
```

```
2.1 C1#m1
```

```
* ruby 2.1.5p312 (2015-03-10 revision 49912) [i386-mswin32_110]
:P_m2
:P_m1
:C1_m1
:C0_m1
```

```
alias + prepend
[Bug #7842]
```

```
:P_m1
:C1_m1
:C0_m1
```

Related issues:

Related to Ruby master - Bug #7842: An alias of a "prepend"ed method skips th...

Closed

02/13/2013

History

#1 - 05/27/2015 08:03 PM - ko1 (Koichi Sasada)

- C1 (m_tbl)
- T_ICLASS[m_tbl] -> P
- T_ICLASS[C1] m_tbl -> C1
- C0
- Object...

```
alias C1 m_tbl C1 C1
```

- alias C1::m_tbl
- T_ICLASS[C1] m_tbl

#2 - 05/28/2015 12:33 AM - usa (Usaku NAKAMURA)

- Related to Bug #7842: An alias of a "prepend"ed method skips the original method when calling super added

#3 - 08/12/2019 10:59 PM - jeremyevans0 (Jeremy Evans)

ko1 (Koichi Sasada) wrote:

```
:P_m1
:C1_m1
:C0_m1
```

This output would be against my expectation. If C1 prepends P, then methods in P must be considered before methods in C1. Consider C1.ancestors:

```
[P, C1, C0, Object, Kernel, BasicObject]
```

P#m2 must be called before C1#m2, and alias m2 m1 in C1 only modifies the method table in C1, not in P.

All Ruby versions I tested have the same behavior as Ruby 2.1:

```
:P_m2
```

```
:P_m1
:C1_m1
:C0_m1
```

There does appear to be a related bug in `super_method`, though:

```
C1.new.method(:m2)
# => #<Method: C1(P)#m2>
```

```
C1.new.method(:m2).super_method
# => #<Method: C1(P)#m2(m1)>
```

```
C1.new.method(:m2).super_method.super_method
# => nil
# Should be #<Method: C1#m1>
```

Note that because of the way alias works with `prepend`, you can get an infinite loop in method lookup with aliases and only calling `super`:

```
module P
  def m1
    super
  end
```

```
  def m2
    super
  end
end
```

```
class C
  prepend P
  alias m2 m1
  alias m1 m2
end
```

```
C.new.m2
# SystemStackError
```