Ruby master - Feature #14123

Kernel#pp by default

11/21/2017 08:02 AM - mame (Yusuke Endoh)

Status:	Closed
Priority:	Normal
Assignee:	matz (Yukihiro Matsumoto)
Target version:	2.5

Description

Matz, may I commit this? I really want this.

```
diff --git a/prelude.rb b/prelude.rb
index 7b98e28285..87f49ac9fb 100644
--- a/prelude.rb
+++ b/prelude.rb
@@ -141,3 +141,11 @@ def irb
     irb
   end
 end
+
+module Kernel
+  def pp(*objs)
+    undef :pp
+    require 'pp'
+    pp(*objs)
+  end
+end
```

Associated revisions

Revision 23c1fccf - 11/30/2017 01:31 AM - mame (Yusuke Endoh)

prelude.rb: Add Kernel#pp, a trigger for lib/pp.rb

[Feature #14123]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@60944 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 60944 - 11/30/2017 01:31 AM - mame (Yusuke Endoh)

prelude.rb: Add Kernel#pp, a trigger for lib/pp.rb

[Feature #14123]

Revision 60944 - 11/30/2017 01:31 AM - mame (Yusuke Endoh)

prelude.rb: Add Kernel#pp, a trigger for lib/pp.rb

[Feature #14123]

Revision 60944 - 11/30/2017 01:31 AM - mame (Yusuke Endoh)

prelude.rb: Add Kernel#pp, a trigger for lib/pp.rb

[Feature #14123]

History

#1 - 11/21/2017 08:11 AM - mame (Yusuke Endoh)

Brief background: I talked about pp with some people at RubyConf, and all people are tired of writing require "pp". And then, we reached this simple solution. Note that we already have a similar trick, as Binding#irb, though I'm unsure if it is officially accepted or not.

#2 - 11/21/2017 08:14 AM - hsbt (Hiroshi SHIBATA)

+1

I sometimes get undefined method `pp' for main:Object (NoMethodError) error when I did debug my script. It's useful for me.

#3 - 11/21/2017 03:05 PM - jeremyevans0 (Jeremy Evans)

I'm not opposed to this feature, but the proposed implementation is not thread-safe, and also breaks when running ruby chrooted. If Kernel#pp is implemented, I request the implementation be thread-safe and that it fall back to Kernel#p if requiring pp fails.

#4 - 11/21/2017 03:28 PM - Eregon (Benoit Daloze)

jeremyevans0 (Jeremy Evans) wrote:

> I'm not opposed to this feature, but the proposed implementation is not thread-safe, and also breaks when running ruby chrooted. If Kernel#pp is implemented, I request the implementation be thread-safe and that it fall back to Kernel#p if requiring pp fails.

The thread safety problem is due to undef, right?
Requiring the same file is thread-safe.
So maybe we just need to hide the redefinition warning (e.g. by setting $VERBOSE) during the require call?

Regarding visibility, #pp should be a module_function of Kernel like #p.

#5 - 11/21/2017 04:15 PM - k0kubun (Takashi Kokubun)

> also breaks when running ruby chrooted

Regardless of proposed implementation, require 'pp' wouldn't work if chrooted. I couldn't understand why it's problematic.

> Regarding visibility, #pp should be a module_function of Kernel like #p.

It's already module_function, isn't it?
https://github.com/ruby/ruby/blob/246c986eac342ae7bbf05f3be97c92fc71f9d21d/lib/pp.rb#L26

(edit) For undef, probably we need the same approach as irb:
https://github.com/ruby/ruby/blob/246c986eac342ae7bbf05f3be97c92fc71f9d21d/lib/irb.rb#L706

#6 - 11/21/2017 05:11 PM - jeremyevans0 (Jeremy Evans)

Eregon (Benoit Daloze) wrote:

> The thread safety problem is due to undef, right?

Correct. The pp library defines Kernel#pp, so in verbose mode if you don't undef or remove the method first, this causes a warning in verbose mode. I think a verbose mode warning is preferable to thread-unsafe code. Here's a possible alternative implementation:

```
module Kernel
  def pp(*a)
    require 'pp'
  rescue LoadError
    p(*a)
  else
    pp(*a)
  end
end
```

To avoid the verbose mode warning, we could have pp include a different module in Object instead of modifying Kernel directly, and then use Kernel.send(:remove_method, :pp) to remove the method, but we'd need to rescue NameError raised by the remove_method call to ensure thread safety in that case.

Note that in all of these cases, Kernel#pp will break if Kernel is frozen.

#7 - 11/21/2017 05:37 PM - shevegen (Robert A. Heiler)

I agree with Yusuke Endoh.

I also wanted to suggest this many times before but shied away because
I was not sure if this would have any chance, perhaps due to speed
reason or something.

I love pp. It's my favourite way to "debug". :D

Just as tenderlove wrote on his blog that he is a "puts debugger",
I am a "pp debugger". :D

And the additional require is annoying. I mean, it's not a huge deal, mind you, just one extra line, but if it were possible to do away with it, I'd be all in favour of it. \o/

#8 - 11/29/2017 05:19 AM - matz (Yukihiro Matsumoto)

Sounds good. Need to discuss implementation detail.

Matz.

#9 - 11/30/2017 01:05 AM - mame (Yusuke Endoh)

We discussed this issue at today's Ruby committer's meeting, and matz accepted the feature itself.

jeremyevans0 (Jeremy Evans) wrote:

> I request the implementation be thread-safe and that it fall back to Kernel#p if requiring pp fails.

Thank you for your opinion.

Okay, I move undef pp from prelude.rb to lib/pp.rb (See the last patch in detail). The fatal race condition is fixed.
"Redefinition warnings" issue remains, but I think it is not significant because pp itself is "not so" thread-safe. Simultaneous multiple calls to pp will interleave the outputs as below. In principle, a user have to do exclusive control appropriately when calling pp (if s/he really cares).

```
$ ruby -rpp -e 'Thread.new { pp [1] * 10000 }; pp [2]*10000' | uniq
[1,
 1,
 1,[2,
 2,

 1,
 1, 2,
 2,

 1,
 1]
 2,
 2]
```

And, many in the meeting were against fallback to p. This is just a useful shortcut for pp, so it should fail when require "pp" fails.

A revised patch:

```
diff --git a/lib/pp.rb b/lib/pp.rb
index 7d1c502817..0e737d23f6 100644
--- a/lib/pp.rb
+++ b/lib/pp.rb
@@ -17,6 +17,7 @@ def pretty_inspect
   # prints arguments in pretty form.
   #
   # pp returns argument(s).
+  undef pp if method_defined?(:pp)
   def pp(*objs)
     objs.each {|obj|
       PP.pp(obj)
diff --git a/prelude.rb b/prelude.rb
index 7b98e28285..3069fdbaf0 100644
--- a/prelude.rb
+++ b/prelude.rb
@@ -141,3 +141,10 @@ def irb
     irb
   end
 end
+
+module Kernel
+  def pp(*objs)
+    require 'pp'
+    pp(objs)
+  end
+end
```

I'll commit it in a few days.

#10 - 11/30/2017 01:27 AM - mame (Yusuke Endoh)

mame (Yusuke Endoh) wrote:

> "Redefinition warnings" issue remains

I misunderstood. Ko1 said to me that the current require does exclusive control, so the redefinition warning issue should not occur (if it occurs, it is a bug). I'll commit it soon.

#11 - 11/30/2017 01:31 AM - mame (Yusuke Endoh)

- Status changed from Open to Closed

Applied in changeset trunk|r60944.

---

prelude.rb: Add Kernel#pp, a trigger for lib/pp.rb

[Feature #14123]

#12 - 11/30/2017 09:18 PM - Eregon (Benoit Daloze)

mame (Yusuke Endoh) wrote:

> I misunderstood. Ko1 said to me that the current require does exclusive control, so the redefinition warning issue should not occur (if it occurs, it is a bug). I'll commit it soon.

Right, multiple calls to require "pp" will wait for one another and only one will actually load the file.
This didn't look thread-safe at first look but it seems fine with require's guarantees (assuming people only do require "pp", no #load but that sounds unlikely).

#13 - 11/30/2017 09:24 PM - Eregon (Benoit Daloze)

$ ruby -ryaml -e 'pp YAML.load_file ".travis.yml"'

is probably a nice example for this feature :)

#14 - 11/30/2017 09:41 PM - jeremyevans0 (Jeremy Evans)

Eregon (Benoit Daloze) wrote:

> Right, multiple calls to require "pp" will wait for one another and only one will actually load the file.
> This didn't look thread-safe at first look but it seems fine with require's guarantees (assuming people only do require "pp", no #load but that sounds unlikely).

What happens in the following circumstance:

- Thread 1 calls Kernel#pp, defined in prelude, which requires 'pp'.
- During requiring of pp:

```
undef pp if method_defined?(:pp)
# Thread switch here
def pp(*objs)
```

- Thread 2 calls Kernel#pp during thread switch

Unless the thread switch is completely blocked during the execution of required code, it seems like this is still not thread safe.

#15 - 12/01/2017 12:42 AM - mame (Yusuke Endoh)

Okay, my third try: https://bugs.ruby-lang.org/projects/ruby-trunk/repository/revisions/60948

Thank you!

#16 - 12/01/2017 10:26 AM - Eregon (Benoit Daloze)

jeremyevans0 (Jeremy Evans) wrote:

> Unless the thread switch is completely blocked during the execution of required code, it seems like this is still not thread safe.

Nice catch!

mame (Yusuke Endoh) I think the new code is worth a comment that this is done for thread-safety in lib/pp.rb

I think suppressing warnings around the definition of pp would be nicer here.
A seemingly useless alias feels like a workaround.
Unfortunately the save/restore dance for changing $VERBOSE is not very nice.

Maybe we should have:

```
Warning.ignore do
  def pp
    ...
  end
end
```

If you think that's a good idea I'm happy to do those changes and open an issue for Warning.ignore.

#17 - 12/01/2017 10:49 AM - akr (Akira Tanaka)

mame (Yusuke Endoh) wrote:

> Okay, my third try: https://bugs.ruby-lang.org/projects/ruby-trunk/repository/revisions/60948

It is still not thread safe.

Consider a thread context switch at just after defining Kernel#pp
before defining PP class.

```
Index: lib/pp.rb
===================================================================
--- lib/pp.rb   (revision 60960)
+++ lib/pp.rb   (working copy)
@@ -26,6 +26,7 @@ module Kernel
   end
   undef __pp_backup__ if method_defined?(:__pp_backup__)
   module_function :pp
+  sleep 1 # thread context switch
 end

 ##
```

With the above patch, "uninitialized constant Kernel::PP" can
happen as as follows.

```
% ./ruby -w -Ilib -e '
t1 = Thread.new {
  Thread.current.report_on_exception = true
  pp :foo1
}
t2 = Thread.new {
  Thread.current.report_on_exception = true
  sleep 0.5
  pp :foo2
}
t1.join rescue nil
t2.join rescue nil
'
#<Thread:0x000055dbf926eaa0@-e:6 run> terminated with exception:
Traceback (most recent call last):
    3: from -e:9:in `block in <main>'
    2: from /home/ruby/tst2/ruby/lib/pp.rb:22:in `pp'
    1: from /home/ruby/tst2/ruby/lib/pp.rb:22:in `each'
/home/ruby/tst2/ruby/lib/pp.rb:23:in `block in pp': uninitialized constant Kernel::PP (NameError)
:foo1
```

Fix is simple: replacing Kernel#pp at last as r60961.

#18 - 12/01/2017 11:55 AM - mame (Yusuke Endoh)

akr-san, thank you!

Eregon, I think $VERBOSE dance is also thread-unsafe. So I personally like Warning.ignore if it is thread-local. I don't know how difficult it is to implement, though.