Ruby - Bug #19374

Issue with Ractor.make_shareable with curried procs

01/24/2023 12:40 PM - luke-gru (Luke Gruber)

Status:	Closed		
Priority:	Normal		
Assignee:	ko1 (Koichi Sasada)		
Target version:			
ruby -v:		**Backport:**	2.7: UNKNOWN, 3.0: UNKNOWN, 3.1: UNKNOWN, 3.2: UNKNOWN

Description

This works, but shouldn't:

```
class Worker
  def start(&blk)
    blk = blk.curry # bug in ruby allows sharing of non-shareable proc
    Ractor.make_shareable(blk)
    @ractor = Ractor.new(blk) do |b|
      main = b.call
      p "from ractor: #{main}"
    end
  end

  def work
    @ractor.take
  end
end

worker = Worker.new
a = self # unshareable main object
p "from main: #{a}"
worker.start { a }

worker.work
```

The curried proc has a reference to the original proc and it's not checked for shareability.

Associated revisions

Revision d80f3a287c5c8d0404b6cb837db360cab320cde1 - 03/26/2025 11:05 PM - luke-gru (Luke Gruber)

Ractor.make_shareable(proc_obj) makes inner structure shareable

Proc objects are now traversed like other objects when making them
shareable.

Fixes [Bug #19372]
Fixes [Bug #19374]

Revision d80f3a287c5c8d0404b6cb837db360cab320cde1 - 03/26/2025 11:05 PM - luke-gru (Luke Gruber)

Ractor.make_shareable(proc_obj) makes inner structure shareable

Proc objects are now traversed like other objects when making them
shareable.

Fixes [Bug #19372]
Fixes [Bug #19374]

Revision d80f3a28 - 03/26/2025 11:05 PM - luke-gru (Luke Gruber)

Ractor.make_shareable(proc_obj) makes inner structure shareable

Proc objects are now traversed like other objects when making them
shareable.

Fixes [Bug #19372]
Fixes [Bug #19374]

History

#1 - 01/25/2023 01:25 PM - luke-gru (Luke Gruber)

This issue is fixed by https://github.com/ruby/ruby/pull/7182. I will add a test to that PR for this.

#2 - 01/27/2023 03:48 AM - hsbt (Hiroshi SHIBATA)

- Status changed from Open to Assigned

- Assignee set to ko1 (Koichi Sasada)

#3 - 08/18/2024 01:22 PM - reesericci (Reese Armstrong)

Hey y'all -

I'm commenting here to say that this bug seems to be the only way I was able to pass a Proc to a Ractor even though it doesn't reference self - so wondering what the path is for that instead of exploiting this bug. For context, here's my code:

```
class Transaction
  def initialize(&block)
    @block = Ractor.make_shareable(block.curry)
    @original_state = nil
  end
```

```
  def apply(obj)
    begin
      @original_state = obj.dup
      @new_obj = obj.deep_transform_values { |value| value = value.dup }
      @block.call(@new_obj)
      obj.replace(@new_obj)
    rescue => e
      puts e
      rollback(obj)
      raise e
    end
  end
end
```

Which then this entire object gets passed into a Ractor and that's where .apply() is called.

Thanks,

--reese

#4 - 01/14/2025 03:03 AM - luke-gru (Luke Gruber)

There's a new feature request that should remedy this: https://bugs.ruby-lang.org/issues/21033

#5 - 03/26/2025 11:05 PM - luke-gru (Luke Gruber)

- Status changed from Assigned to Closed

Applied in changeset git|d80f3a287c5c8d0404b6cb837db360cab320cde1.

Ractor.make_shareable(proc_obj) makes inner structure shareable

Proc objects are now traversed like other objects when making them
shareable.

Fixes [Bug #19372]
Fixes [Bug #19374]