Ruby master - Bug #17180

Ractor and constant referencing

09/19/2020 09:24 PM - kirs (Kir Shatrov)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 3.0.0dev (2020-09-19T08:47:40Z master 73a626c078) [x86_64-darwin19]	**Backport:**	2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN

Description

Hey there.

From a high level, this bug report is describing my experience building something simple with Ractor without any shared state. I hope this is helpful as a feedback for the stable Ractor implementation in Ruby 3.0.

I've been iterating on a simple web server that utilizes Ractor. One of the things that a web server does is parsing HTTP headers. Thankfully, there's code in Ruby's standard library that can do that, so I decided to leverage WEBrick::HTTPRequest.

Here's my example:

```
require 'webrick'
require 'stringio'

parser = Ractor.new do
  s = StringIO.new("GET / HTTP/1.1\n"\
    "Host: example.com\n"\
    "User-Agent: curl/7.64.1\n"\
    "Accept: */*")

  req = WEBrick::HTTPRequest.new(WEBrick::Config::HTTP)
  req.parse(s)
end

Ractor.select(parser)
```

This fails with can not access non-sharable objects in constant WEBrick::Config::HTTP by non-main Ractor. (NameError).

Let's try to copy that constant and pass it to the Ractor:

```
require 'webrick'
require 'stringio'

parser = Ractor.new(WEBrick::Config::HTTP) do |config|
  s = StringIO.new("GET / HTTP/1.1\n"\
    "Host: example.com\n"\
    "User-Agent: curl/7.64.1\n"\
    "Accept: */*")

  req = WEBrick::HTTPRequest.new(config)
  req.parse(s)
end

Ractor.select(parser)
```

This failed with can't dump hash with default proc. I've guessed that maybe WEBrick::Config::HTTP has a default proc? Let's try to work around:

```
require 'webrick'
require 'stringio'

http_c = WEBrick::Config::HTTP
```

```
http_c.default = nil

parser = Ractor.new(http_c) do |config|
  s = StringIO.new("GET / HTTP/1.1\n"\
    "Host: example.com\n"\
    "User-Agent: curl/7.64.1\n"\
    "Accept: */*")

  req.parse(s)
end

Ractor.select(parser)
```

It failed with:

```
/Users/kir/src/github.com/ruby/ruby/lib/webrick/httprequest.rb:570:in `read_line': can not access 
non-sharable objects in constant WEBrick::LF by non-main ractor. (NameError)
```

I went ahead and made WEBrick freeze strings:

```
diff --git a/lib/webrick/httputils.rb b/lib/webrick/httputils.rb
index 76d4bd0dc7..273c596368 100644
--- a/lib/webrick/httputils.rb
+++ b/lib/webrick/httputils.rb
@@ -13,9 +13,9 @@
 require 'tempfile'

 module WEBrick
-  CR   = "\x0d"     # :nodoc:
-  LF   = "\x0a"     # :nodoc:
-  CRLF = "\x0d\x0a" # :nodoc:
+  CR   = "\x0d".freeze     # :nodoc:
+  LF   = "\x0a".freeze     # :nodoc:
+  CRLF = "\x0d\x0a".freeze # :nodoc:
```

Now it's failing with:

```
/Users/kir/src/github.com/ruby/ruby/lib/singleton.rb:124:in `instance': can not access instance va
riables of classes/modules from non-main Ractors (RuntimeError)
    from /Users/kir/src/github.com/ruby/ruby/lib/webrick/utils.rb:132:in `register'
    from /Users/kir/src/github.com/ruby/ruby/lib/webrick/utils.rb:256:in `timeout'
    from /Users/kir/src/github.com/ruby/ruby/lib/webrick/httprequest.rb:559:in `_read_data'
    from /Users/kir/src/github.com/ruby/ruby/lib/webrick/httprequest.rb:570:in `read_line'
    from /Users/kir/src/github.com/ruby/ruby/lib/webrick/httprequest.rb:447:in `read_request_line'
    from /Users/kir/src/github.com/ruby/ruby/lib/webrick/httprequest.rb:201:in `parse'
    from ./test.rb:16:in `block in <main>'
```

because WEBrick is manipulating with a class variable.

I've commented out some of that code in WEBrick for the sake of experiment and it started to fail with:

```
/Users/kir/src/github.com/ruby/ruby/lib/uri/common.rb:171:in `parse': can not access non-sharable 
objects in constant URI::RFC3986_PARSER by non-main ractor. (NameError)
    from /Users/kir/src/github.com/ruby/ruby/lib/webrick/httprequest.rb:484:in `parse_uri'
    from /Users/kir/src/github.com/ruby/ruby/lib/webrick/httprequest.rb:217:in `parse'
```

This can be easily reproduced with a smaller piece of code:

```
parser = Ractor.new do
  URI::parse("http://example.com")
end

Ractor.select(parser)
```

IMO, something as tiny and safe as URI::parse should be allowrd to be called from a Ractor. We're either being too strict to the developer in terms of what constants they can reference from inside a Ractor, or we're missing a bunch of freeze in the standard library. It's likely the latter, but in that case we'll need to push for freeze as much as we can to make Ractor friendly to developers.

I'm very excited to see this happening in Ruby 3.0 and I'd like to help as much as I can to make it a great feature of Ruby.

History

#1 - 09/19/2020 09:33 PM - kirs (Kir Shatrov)

Another example:

```
Person = Struct.new(:name)
CREATOR = Person.new("Matz").freeze

parser = Ractor.new { CREATOR }
Ractor.select(parser)
```

```
./test.rb:12:in `block in <main>': can not access non-sharable objects in constant Object::CREATOR by non-main
 Ractor. (NameError)
make: *** [runruby] Error 1
```

Should something as plain as a frozen struct be shareable?

I am aware that you can make it copy the constant (Ractor.new(CREATOR) { |creator| creator } but it will become really lengthy to copy every constant that will ever be referenced from a Ractor. I don't think that will make developers happy.

#2 - 09/20/2020 05:43 PM - Eregon (Benoit Daloze)

For the Struct example above, probably it needs the "Matz" String to be frozen too, or #deep_freeze (#17145) to mark it as a shareable value.