Ruby - Bug #17986

Ractor is stdlib Socket unfriendly

06/15/2021 01:37 AM - kvokka (Mike Beliakov)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 3.0.1p64 (2021-04-05 revision 0fb782ee38) [x86_64-darwin19]	**Backport:**	2.6: UNKNOWN, 2.7: UNKNOWN, 3.0: UNKNOWN

Description

Description

In the process of playing with Ractors was found, that there is no way to use stdlib Sockets.

My intent was to implement bidirectional connection with Socket using Ractors.

This small console snippet is self explanatory:

```
[1] pry(main)> s=Ractor.make_shareable(TCPSocket.open('localhost', 9100))
=> #<TCPSocket:fd 14, AF_INET6, ::1, 52292>
[2] pry(main)> s.print 'foo'
FrozenError: can't modify frozen TCPSocket: #<TCPSocket:fd 14, AF_INET6, ::1, 52292>
from (pry):5:in `write'
[3] pry(main)> s=Ractor.make_shareable(TCPSocket.open('localhost', 9100), copy: true)
=> #<TCPSocket:fd 16, AF_INET6, ::1, 52295>
[4] pry(main)> s.print 'foo'
FrozenError: can't modify frozen TCPSocket: #<TCPSocket:fd 16, AF_INET6, ::1, 52295>
```

The only option is to move socket in Ractor, but in this case there is no way to share
the socket between 2 Ractors, so I can not put a listener loop on the socket.

History

#1 - 06/15/2021 04:13 AM - jeremyevans0 (Jeremy Evans)

The code you are showing (make_shareable with a socket) is broken because in order to be shareable, an object must be immutable, and a socket cannot really be immutable and usable.

You can work around it using for_fd (this should show that both ractors are processing the socket):

```
require 'socket'

Thread.new do
  client = TCPServer.new(9100).accept
  10.times do |i|
    sleep 1
    client.write i.to_s
  end
end
sleep 1

socket = TCPSocket.open('localhost', 9100)

sock_ractor = lambda do |i, socket|
  Ractor.new(i, socket.to_i) do |i,  fd|
    sock = TCPSocket.for_fd(fd)
    while socks = IO.select([sock]).first
      socks.each do |s|
        $stderr.syswrite("#{i}:#{s.read(1)}\n")
      end
    end
  end
end

r1 = sock_ractor[1, socket]
```

```
r2 = sock_ractor[2, socket]

sleep 11
```

I don't think the behavior you are showing is a bug, so I think this should be closed. However, I'll wait for a while and see if another core team member feels differently.

#2 - 06/15/2021 10:04 PM - kvokka (Mike Beliakov)

> I don't think the behavior you are showing is a bug, so I think this should be closed. However, I'll wait for a while and see if another core team member feels differently.

Thank you @jeremyevans0 (Jeremy Evans) for the idea, you gave me TIL!

This hack works, so my only question for now is that it should be either better documented (can make a PR to Ractor docs with this snippet if you don't mind) or it should be possible to use stdlib parts more transparently. Which way is better is up to core team.

#3 - 06/16/2021 02:30 AM - kvokka (Mike Beliakov)

This hack has a limitation which i faced with specs.

For socket testing purposes i have i small class (It should be adjusted for using with Ractors, share it just to share the idea of the problem) which replay everything from file to socket (to avoid real communication), like VHS gem.
But because of Ractor limitations i can not use it inside Ractor (instance variables are prohibited) and technically I re-instantiate the socket in Ractor.

The option might be to monkey patch TCPSocket.for_fd, but the trick is, that because it should be invoked in Ractor I can not share any spec information without another hack(s).

#4 - 06/28/2021 07:37 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Closed