Marshal.dump(closed_io) raises IOError instead of TypeError

08/16/2021 11:33 AM - larskanis (Lars Kanis)

Status: Closed
Priority: Normal
Assignee: 
Target version:
ruby -v: ruby 3.1.0dev (2021-08-16T08:00:19Z master a8714b83c4) [x86_64-linux]
Backport: 2.6: UNKNOWN, 2.7: UNKNOWN, 3.0: UNKNOWN

Description
Marshal.dump is expected to raise a TypeError for unmarshallable objects. But closed streams raise an IOError:

```ruby
$ ruby -e "io=IO.pipe.first; io.close; Marshal.dump(io)"
-e:1:in `internal_encoding': closed stream (IOError)
  from -e:1:in `dump'
  from -e:1:in `<main>'
```

This issue is present in all current ruby versions.

Associated revisions
Revision 6594623f - 08/22/2021 01:33 AM - Lars Kanis
Fix Marshal.dump(closed_io) to raise TypeError and allow encoding on closed IO

Marshalling a closed IO object raised "closed stream (IOError)" before instead of TypeError. This changes IO#(in|ex)ternal_encoding to still return the encoding even if the underlying FD is closed.

Fixes bug #18077

Revision d574b841 - 08/22/2021 03:11 AM - nobu (Nobuyoshi Nakada)
Fix failures on non-UTF-8 environment [Bug #18077]

Call IOSpecs.io_fixture with the default encoding explicitly. IOSpecs.closed_io calls the method without optional mode which is set to UTF-8 by default, while the default external encoding depends on the locale environment variables.

History
#1 - 08/16/2021 12:40 PM - nobu (Nobuyoshi Nakada)
One concern for that PR, probably may not be a matter, is closed IOs will no longer raise IOError on other Encoding operations too.

```ruby
$ ./ruby -v -e 'p Encoding.compatible?(Encoding::US_ASCII, File.open(IO::NULL).tap(&:close))'
ruby 3.1.0dev (2021-08-16T08:00:19Z master a8714b83c4) [x86_64-darwin19]
-e:1:in `internal_encoding': closed stream (IOError)
  from -e:1:in `compatible?'
  from -e:1:in `<main>'
```

#2 - 08/18/2021 03:14 AM - larskanis (Lars Kanis)
https://github.com/ruby/ruby/pull/4749 is another fix without the above side effect. Is it OK?

#3 - 08/18/2021 04:51 PM - Eregon (Benoit Daloze)
Why does IO#internal_encoding (and external_encoding) raise if the IO is closed? They could just return the encoding, it's still stored in in the IO instance, and we are not trying to access the fd, right?

JRuby 9.2.17.0 does not raise, and I think it is the better behavior here.
I don't have any strong opinion but Eregon (Benoit Daloze)’s approach looks a bit better.

Eregon (Benoit Daloze) This was my first thought as well, but the current behavior is already defined in ruby-spec.

I would prefer the behavioral change of IO#(in|ex)ternal_encoding being usable on closed IOs and prepared a PR: https://github.com/ruby/ruby/pull/4758

Since this is a bug report, I think that the old example in rubyspec should be removed.

I removed the old example from https://github.com/ruby/ruby/pull/4758.

Eregon (Benoit Daloze) This was my first thought as well, but the current behavior is already defined in ruby-spec.

We can change it in ruby/spec (adding version guards and adding the new expectation). And in fact I would be glad if the adapted specs would test the more sensible behavior. ruby/spec often tests whatever the behavior of CRuby is (for best compatibility with other Rubies), it might not always make sense.

I should have read the PR first. Yes, removing the old example and adding the new one with a version guard is perfect here :) That allows other implementations & older versions to potentially use the new behavior already, and there is probably very little to no code relying on the previous exception.

Fix Marshal.dump(closed_io) to raise TypeError and allow encoding on closed IO

Mashalling a closed IO object raised “closed stream (IOError)” before instead of TypeError. This changes IO#(in|ex)ternal_encoding to still return the encoding even if the underlying FD is closed.

Fixes bug #18077