Ruby master - Bug #18141

Marshal load with proc yield strings before they are fully initialized

09/01/2021 08:17 AM - byroot (Jean Boussier)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td></td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
</tr>
<tr>
<td>ruby -v:</td>
<td></td>
</tr>
<tr>
<td>Backport:</td>
<td>2.6: REQUIRED, 2.7: REQUIRED, 3.0: REQUIRED</td>
</tr>
</tbody>
</table>

**Description**

I assume this is a bug because I can't find any spec or test for this behaviour:

Consider the following script:

```ruby
payload = Marshal.dump("foo")

Marshal.load(payload, -> (obj) {
  if obj.is_a?(String)
    p [obj, obj.encoding]
  end
  obj
})
p [:final, string, string.encoding]
```

outputs:

```
["foo", #<Encoding:ASCII-8BIT>]
[:final, "foo", #<Encoding:UTF-8>]
```

So Marshal call the proc before the string get its encoding assigned, this is because the encoding is stored alongside as a TYPE_IVAR. I think in such cases Marshal should delay calling the proc until the object is fully restored.

A corollary to this behaviour is that the following code:

```ruby
Marshal.load(payload, :freeze.to_proc)
```

raises with can't modify frozen String: "foo" (FrozenError).

Associated revisions

Revision 89242279 - 09/14/2021 11:00 PM - byroot (Jean Boussier)
Marshal.load: do not call the proc until strings have their encoding

Ref: [https://bugs.ruby-lang.org/issues/18141](https://bugs.ruby-lang.org/issues/18141)

History

**#1 - 09/01/2021 05:57 PM - byroot (Jean Boussier)**
I potentially have a fix: [https://github.com/ruby/ruby/pull/4797](https://github.com/ruby/ruby/pull/4797)

**#2 - 09/02/2021 12:40 AM - nobu (Nobuyoshi Nakada)**
Should use ruby_bug "#18141", "..."3.1" instead of ruby_version_is.

**#3 - 09/14/2021 11:05 PM - nobu (Nobuyoshi Nakada)**
- Backport changed from 2.6: UNKNOWN, 2.7: UNKNOWN, 3.0: UNKNOWN to 2.6: REQUIRED, 2.7: REQUIRED, 3.0: REQUIRED
- Status changed from Open to Closed

**#4 - 09/17/2021 02:19 PM - byroot (Jean Boussier)**
So while working on [https://bugs.ruby-lang.org/issues/18148](https://bugs.ruby-lang.org/issues/18148), I discovered that many other types of objects are impacted.
Just a few examples:

```ruby
def round_trip(obj, proc = ->(o) { o.freeze })
  Marshal.load(Marshal.dump(obj), proc)
end

h = {}
h.instance_variable_set(,:foo, 42)  #=>FrozenError: can't modify frozen Hash>
round_trip(h) rescue p $!

a = []
a.instance_variable_set(,:foo, 42)  #=>FrozenError: can't modify frozen Array>
round_trip(a) rescue p $!

Also, probably by design, but since you can replace the object by what the proc returns:

a = {}
a.instance_variable_set(,:foo, 42)
round_trip(a, proc { 24 }) rescue p $!  #=>FrozenError: can't modify frozen Integer>
```

I fixed most cases in [https://github.com/ruby/ruby/pull/4859](https://github.com/ruby/ruby/pull/4859), which is my current attempt at implementing [https://bugs.ruby-lang.org/issues/18148](https://bugs.ruby-lang.org/issues/18148), but since I just noticed this was marked for backport, I might need to split the bug fix from the new feature. No?

#5 - 09/18/2021 07:34 AM - nagachika (Tomoyuki Chikanaga)
Hello byroot,
Thank you for the investigation about the issue. Yes, the patch with only the bug fix is very helpful to maintain stable branches.

#6 - 09/18/2021 01:45 PM - byroot (Jean Boussier)
I made a followup patch: [https://github.com/ruby/ruby/pull/4866](https://github.com/ruby/ruby/pull/4866)

It now handle similar bugs with Array, Hash and other mutable objects. It also handle circular objects.