Ruby master - Bug #18170
Exception#inspect should not include newlines
09/16/2021 02:34 AM - mame (Yusuke Endoh)

Status: Assigned
Priority: Normal
Assignee: mame (Yusuke Endoh)
Target version: ruby -v:
Backport: 2.6: UNKNOWN, 2.7: UNKNOWN, 3.0: UNKNOWN

Description
Is this intentional?

```ruby
p StandardError.new("foo\nbar")
 #=>
 # #<StandardError: foo
 # bar>
```

I expect #inspect returns a one-line string. How about returning #<StandardError: "foo\nbar"> or something?

Recently, multi-line error messages have been increasing by the introduction of did_you_mean and error_highlight. Printing an object that contains such an exception leads to a tricky output:

```ruby
class Foo
  def initialize
    @exception = begin; exampl; rescue Exception; $!; end
  end

  def example
  end
end

p Foo.new
 #=>
 # #<Foo:0x00007f15aeb4ba48 @exception=#<NameError: undefined local variable or method `exampl' for
 #<Foo:0x00007f15aeb4ba48 ...> #
 # @exception = begin; exampl; rescue Exception; $!; end
 # ^^^
 # Did you mean? example>>
```

This issue was originally found by ioquatix (Samuel Williams)

Related issues:
Related to Ruby master - Feature #18296: Custom exception formatting should o...
Open

History
#1 - 09/16/2021 08:42 AM - mame (Yusuke Endoh)
- Assignee set to mame (Yusuke Endoh)
- Status changed from Open to Assigned

Matz said "give it a try". I'll create a PR

#2 - 09/16/2021 10:16 AM - ioquatix (Samuel Williams)
Thanks mame (Yusuke Endoh).

#3 - 09/17/2021 08:04 AM - mame (Yusuke Endoh)
Created a PR: https://github.com/ruby/ruby/pull/4857

As expected, I had to modify some tests. Is this acceptable?

03/21/2022
#4 - 09/24/2021 06:56 PM - Eregon (Benoit Daloze)
The quotes around the message feel redundant to me and likely to cause much more incompatibility. The message is the main attribute of an exception, I don't think we need to quote it.
I'm neutral regarding escaping \n and non-printable characters.

#5 - 09/25/2021 04:58 PM - mame (Yusuke Endoh)
Eregon (Benoit Daloze) wrote in #note-4:
The quotes around the message feel redundant to me and likely to cause much more incompatibility.

Because we don't inspect an exception object so often, I think the redundancy is not much of a problem. I somewhat agree with the concern about incompatibility. Though it is not very admirable to depend on the return value of #inspect, some changes of tests are actually needed for my PR.
I'm neutral regarding escaping \n and non-printable characters.

I think of three options.
1. No change
2. Apply String#inspect to the message (as my PR does)
3. Apply String#inspect to the message, and drop the first and last quotes from the returned string

(3) is very ad-hoc but maybe will work well in many cases.

#6 - 09/26/2021 12:07 PM - Eregon (Benoit Daloze)
mame (Yusuke Endoh) wrote in #note-5:
Because we don't inspect an exception object so often, I think the redundancy is not much of a problem. I somewhat agree with the concern about incompatibility. Though it is not very admirable to depend on the return value of #inspect, some changes of tests are actually needed for my PR.

Right, the #inspect value should generally not be relied upon, but I feel something like rescue => e; p e or log e.inspect might not be so rare, especially when there was no Exception#full_message (or users might not know about it).
Or even as a simple way to include the cause like rescue => e; raise FooException, "some error (#{e.inspect}).
Exception#to_s is not useful as it only returns the message.
Yet another case is in IRB:

> e
=> #<RuntimeError: my message>
is nice and I think what people expect.

> e
=> #<RuntimeError: "my message">
not so much.
So I think 1 or 3 is better.

Of course there is no guarantee general inspect returns a single-line, so it seems best if usages of Exception#inspect or Exception#message handle multiple lines correctly.

#7 - 09/26/2021 08:17 PM - ioquatix (Samuel Williams)
I am strongly against (3), because I don't think we should be introducing ad-hoc behaviour.
I am strongly in favour of (2) and we should try to be as consistent with the rest of the implementations of #inspect.

The quotes around the message feel redundant to me and likely to cause much more incompatibility.

Evidence?

Because we don't inspect an exception object so often, I think the redundancy is not much of a problem. I somewhat agree with the concern about incompatibility. Though it is not very admirable to depend on the return value of #inspect, some changes of tests are actually needed for my PR.

100% agree.
so it seems best if usages of Exception#inspect or Exception#message handle multiple lines correctly.

In my experience, this would break a ton of logging systems, e.g.

```ruby
#!/usr/bin/env ruby
require 'logger'
exception = StandardError.new("foo
bar")
logger = Logger.new($stderr)
logger.debug(exception)
```

We should investigate in more detail. mame (Yusuke Endoh) how do you expect this to work with the new multi-line error messages?

#9 - 09/27/2021 09:59 AM - Eregon (Benoit Daloze)
ioquatix (Samuel Williams) wrote in #note-8:

so it seems best if usages of Exception#inspect or Exception#message handle multiple lines correctly.

In my experience, this would break a ton of logging systems, e.g.

There will always be objects with multiple-lines #inspect. Those system are broken if they think #inspect only ever returns 1 line, there was never such a guarantee.

#10 - 09/27/2021 10:02 AM - Eregon (Benoit Daloze)
ioquatix (Samuel Williams) wrote in #note-7:

Evidence?

See the PR: https://github.com/ruby/ruby/pull/4857/files
For example minitest and irb tests are affected. Likely other test frameworks too.

#11 - 09/29/2021 09:47 AM - ioquatix (Samuel Williams)
I asked Jon Rowe from RSpec to check this. He monkey patched (3) into rspec and:

I monkey patched an equivalent fix into my local code, and nothing failed, so I’m 80% confident we don’t even match on inspect outputs in our specs

Followed by:

rspec isn’t broken by this but please don’t escape \n in message

Which is okay since we aren't proposing any changes to Exception#message, only #inspect.

I can also follow up with minitest authors if that's helpful.

#12 - 10/21/2021 05:08 AM - matz (Yukihiro Matsumoto)
I agree with the new behavior that wraps messages with newlines.

Matz.

#13 - 10/25/2021 08:59 AM - mame (Yusuke Endoh)
We discussed this ticket at the dev-meeting, and matz basically accepted as above.

The approved spec is very hacky. Only when the message includes new lines, Exception#inspect applies inspect to the message. Otherwise, it embeds the message as is.

```ruby
p StandardError.new("foo\nbar") #=> #<StandardError:foo\nbar>
p StandardError.new("foo bar")  #=> #<StandardError:foo bar>
```
p StandardError.new("foo\bar") #=> #<StandardError: foo\bar>

Note that there is no space between the colon and the double quote in the first line. This is a mark showing whether inspect is used.

p StandardError.new("foo\nbar") #=> #<StandardError: foo\nbar>
p StandardError.new("foo\nbar!") #=> #<StandardError: foo\nbar>

Parsing the result of #inspect is never recommended, though. I'll update my PR later to follow the approved spec.

#14 - 10/25/2021 10:20 AM - ioquatix (Samuel Williams)

Why is that? Why is it necessary?

The approved spec is very hacky.

I agree it's ad-hoc and I dislike that we could not just call inspect on every message which would be consistent and straightforward to implement. The proposed solution seems both inefficient (need to scan string before printing) and inconsistent (now we still have two different formatting rules).

matz (Yukihiro Matsumoto) what problem are we trying to prevent by not just applying #inspect to every message?

#15 - 10/25/2021 03:07 PM - mame (Yusuke Endoh)
ioquatix (Samuel Williams) wrote in #note-14:

matz (Yukihiro Matsumoto) what problem are we trying to prevent by not just applying #inspect to every message?

Incompatibility. In general, users should not depend on the return value of #inspect. That being said, we don't want to introduce incompatibility unnecessarily. According to my PR, the simple approach actually required some modification to tests, so the incompatibility issue is not imaginary but real. We'd avoid the hassle here.

Considering this issue from a practical point of view, the problem occurs only when newlines are included. We don't have to escape other letters. So matz (Yukihiro Matsumoto) chose to avoid an incompatibility as far as reasonably practicable.

akr (Akira Tanaka) suggested the "no-space" hack to indicate whether #inspect is applied or not, and matz (Yukihiro Matsumoto) left to me whether the hack is introduced or not. I have no strong opinion about that, but just for the case, I'd like to make them distinguishable.

#16 - 10/26/2021 08:59 AM - ioquatix (Samuel Williams)
mame (Yusuke Endoh) thanks for all your work on this issue.

Incompatibility. In general, users should not depend on the return value of #inspect. That being said, we don't want to introduce incompatibility unnecessarily. According to my PR, the simple approach actually required some modification to tests, so the incompatibility issue is not imaginary but real. We'd avoid the hassle here.

We tested the change with the entire RSpec code base and there was no failure. Because RSpec has specific error matching predicates which match the error, not the output of inspect. I imagine the same is true for Minitest and similar systems. I can't imagine there would be any systems that will match directly the output of Exception#inspect in actual real world tests, and even if there are, they are easily fixed.

Regarding the Ruby test failures, maybe the problem is that the test of the exact format is incorrect. Since we don't expect users to depend on exact format, why do our tests depend on the exact format? We can rework these tests to be more flexible.

we don't want to introduce incompatibility unnecessarily

I'm willing to break compatibility if it means a simpler and more consistent implementation, especially longer term. To this end, the current proposed change seems okay to me, but I wouldn't introduce any changes to spacing to determine the difference. As you said, we don't want users to parse the result of inspect, but by introducing spacing changes, you are indicating some change. Such spacing is also inconsistent with (all?) other implementations of #inspect.

#17 - 11/10/2021 11:03 AM - Eregon (Benoit Daloze)
I've been thinking about this again, and I'm not sure if

```
<Foo:0x00007ff15aeb4ba48 @exception=>#<NameError: undefined local variable or method `exampl' for #<Foo:0x00007ff15aeb4ba48 ...>
  @exception = begin; exampl; rescue Exception; $!; end
```

03/21/2022
Did you mean? example>

or

```ruby
#<Foo:0x00007f15aeb4ba48 @exception=#<NameError: "undefined local variable or method 'exampl' for #<Foo:0x00007f15aeb4ba48 ...
    @exception = begin; exampl; rescue Exception; $!; end

Did you mean? exampl>}
```

is better.

The object's inspect is unreadable in both cases anyway.

For readability of the error, I believe the first is better.
And if the error was shown alone (common case, e.g. if an exception is shown in IRB or with `p exception`), the first is clearly much better:

```ruby
#<NameError: undefined local variable or method 'exampl' for #<Foo:0x00007f15aeb4ba48 ...
    @exception = begin; exampl; rescue Exception; $!; end

Did you mean? exampl>
```

vs

```ruby
#<NameError: "undefined local variable or method 'exampl' for #<Foo:0x00007f15aeb4ba48 ...
    @exception = begin; exampl; rescue Exception; $!; end

(that's completely unreadable)
```

I think needing to read the inspect of an object containing an exception is pretty rare, and not worth making Exception#inspect less readable (which can be very useful for debugging).

And there is an easy workaround, that object storing an exception can have a custom #inspect and something like:

```ruby
#<ExceptionWrapper:0x00007f15aeb4ba48 for NameError with message:
    @exception = begin; exampl; rescue Exception; $!; end

Did you mean? exampl>
```

That's readable.

There might also be the need to disable did_you_mean and error_highlight for a given exception, I think that's a reasonable request, but also a different feature.

Maybe having something like Exception#original_message seems a simple way to do that (the message, unaffected by did_you_mean/error_highlight/etc).

Or have Exception#message assemble different parts of the exception, and did_you_mean/error_highlight would use a proper hook to add extra data.

Then Exception#message could accept some argument whether you want the extra data or not, or just a single line, etc.

#18 - 11/10/2021 11:10 AM - loquatix (Samuel Williams)

Here is what I want:

- Exception.new(message).message is always message.
- Top level unhandled exception handler, which defaults to some well defined interface for printing exceptions.
- Well defined hooks for augmenting the exception printing out process.
- Exception#inspect should be exactly the default implementation provided by Object#inspect.

In addition to this, in order to support printing exceptions to a format agonistic output, we need to make sure that meta-data attached to the exception is sufficiently well defined so as not to require implementation specific interfaces like RubyVM::AST.of.

Anyway, I know that we can't have some/all of this, but I don't think it's unreasonable position to take.

#19 - 11/10/2021 11:10 AM - Eregon (Benoit Daloze)

BTW it seems irb disables error_highlight somehow:

```bash
$ ruby -ve 'self.instance_off'
ruby 3.1.0dev (2021-11-08T13:15:21Z master bd2674ad33) [x86_64-linux]
-e:1:in `<main>': undefined method `instance_off' for main:Object (NoMethodError)

self.instance_off

Did you mean? instance_off?
```

vs

```bash
$ irb
irb(main):001:0> self.instance_off
```

03/21/2022
ioquatix (Samuel Williams) wrote in #note-18:

Here is what I want:

- Exception.new(message).message is always message.

That's hard, e.g. for Errno#message.
I think having a way to get the original message or the message unaffected by did_you_mean/error_highlight is good though.

- Top level unhandled exception handler, which defaults to some well defined interface for printing exceptions.

Isn't that already the case? The top level handler is basically $stderr.puts exc.full_message.

- Well defined hooks for augmenting the exception printing out process.

Agreed, did_you_mean/error_highlight should use a better hook than overriding message unilaterally (or using set_message).

- Exception#inspect should be exactly the default implementation provided by Object#inspect.

Can't be, message is not an instance variable so it wouldn't be shown at all.

In addition to this, in order to support printing exceptions to a format agonistic output, we need to make sure that meta-data attached to the exception is sufficiently well defined so as not to require implementation specific interfaces like RubyVM::AST.of.

Anyway, I know that we can't have some/all of this, but I don't think it's unreasonable position to take.

I think it's a reasonable position and some of these have a decent chance to be accepted. I don't see how changing/breaking Exception#inspect improves anything, so I suggest we don't change Exception#inspect.

BTW it seems irb disables error_highlight somehow:

Because irb uses Kernel#eval to execute the input code, error_highlight cannot get the source code. Currently, there is a secret API, RubyVM.keep_script_lines = true, which enables error_highlight in irb.

```ruby
$ irb
irb(main):001:0> RubyVM.keep_script_lines = true
=> true
irb(main):002:0> l.time
(1:time:2:in`<main>': undefined method 'time' for 1:Integer (NoMethodError)
```

#20 - 11/10/2021 11:50 AM - Eregon (Benoit Daloze)

Eregon (Benoit Daloze) wrote in #note-19:

BTW it seems irb disables error_highlight somehow:
I have updated my PR to follow the approved semantics. [https://github.com/ruby/ruby/pull/4857](https://github.com/ruby/ruby/pull/4857)