id outputed by inspect and to_s output does not allow to find actual object_id and vice-versa

Hello, here is my first ruby issue sorry in advance if it is incorrectly filled.

The value returned by #object_id is not aligned anymore with displayed info in #inspect and #to_s methods.

with ruby < 2.7

Object.new.tap { |o| p "#to_s=#{o.to_s}, #inspect=#{o.inspect}, __id__=#{o.__id__}, shifted_id=#{(o.__id__ << 1).to_s(16)}" }
    "#to_s=#{Object:0x0000000000d202a8}, #inspect=#{Object:0x0000000000d202a8}, __id__=6881620, shifted_id=d202a8"

with ruby >= 2.7

    Object.new.tap { |o| p "#to_s=#{o.to_s}, #inspect=#{o.inspect}, __id__=#{o.__id__}, shifted_id=#{(o.__id__ << 1).to_s(16)}" }
    "#to_s=#{Object:0x0000555dc8640b88}, #inspect=#{Object:0x0000555dc8640b88}, __id__=220, shifted_id=1b8"

Consequences

It makes harder:

- to implement a clean override of the #inspect method. i.e. How to keep the same output without ability to compute to the same "object_id" value.
- to debug the object using the inspect output. i.e. ObjectSpace._id2ref(id_from_inspect >> 1) used to work, now it doesn't (RangeError: <xXx> is not id value).

Suggestion

IMHO either:

- the #to_s and #inspect documentation are obsolete The default [...] [shows|prints] [...] an encoding of the object id and the change could have been a bit more advertised
- they should use the result of #object_id instead of displaying the object pointer address

Another solution could be to provide a method to get access to the address, but I'm not sure you want that.

P.S. While debugging my problem I found this ruby-forum thread where people dived a bit more than me into ruby's code.

History

#1 - 10/20/2020 03:55 PM - chrisseaton (Chris Seaton)

Additional context is that #object_id used to use the object's address, and is now a simple incrementing number, and #inspect still and has always used the object's address.

I believe the documentation is correct though?

https://ruby-doc.org/core-2.7.2/Object.html#method-i-inspect

Note that an additional complicating issue is that Ruby objects may now move, so addresses are not stable, and so the result of #inspect is not stable.
Two calls to #inspect on the same object can return different values non-deterministically. Is that what we want?

I think that #inspect should be changed to use #object_id.

#2 - 10/20/2020 06:31 PM - jorel (Joel Johnson)

Similarly, the object_id and object addresses are potentially problematic when making use of ObjectSpace.trace_object_allocations_start and dumping the json data periodically in trying to match up the object over time if GC.compact has happened.

I think this highlights having object_id being a nice higher level concept for these use cases instead of relying on the address that is effectively an implementation detail.

#3 - 10/20/2020 08:06 PM - Eregon (Benoit Daloze)

Agreed on showing the #object_id in #inspect seems better (and more useful for identifying a specific object).

Annih (Baptiste Courtois) wrote:

It makes harder:

- to implement a clean override of the #inspect method. i.e. How to keep the same output without ability to compute to the same "object_id" value.

A simple way is "#{super[0...-1]} extra info>". Trying to manually format the object_id is often not a good idea, notably because object_id can be negative (especially on 32-bit platforms).

#4 - 10/20/2020 08:15 PM - nateberkopec (Nate Berkoepec)

This looks like an oversight to me? I agree with Benoit - showing object_id everywhere we used to show address seems the best way forward.

#5 - 10/20/2020 09:32 PM - tenderlovemaking (Aaron Patterson)

- File 0001-Use-the-object-id-in-the-default-implementation-of-i.patch added

I think adding the object id to inspect is a good idea, though I think it should just have the object id, not object_id >> 1 (or any other permutation). I don't think adding the object id to heap dumps is a good idea though since we would have to generate an object id for every object in the heap.

I've attached a patch with the object id in inspect

#6 - 10/20/2020 09:35 PM - tenderlovemaking (Aaron Patterson)

Another solution could be to provide a method to get access to the address, but I'm not sure you want that.

btw you can get the address of an object like this:

```
irb(main):001:0> require "fiddle"
=> true
irb(main):002:0> x = Object.new
=> #<Object:0x00007fee918e0148>
irb(main):003:0> Fiddle.dlwrap(x).to_s(16)
=> "7fee918e0148"
```

Files

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001-Use-the-object-id-in-the-default-implementation-of-i.patch</td>
<td>1.16 KB</td>
<td>10/20/2020</td>
<td>tenderlovemaking (Aaron Patterson)</td>
</tr>
</tbody>
</table>