It is surprising that #object_id returns signed value. Let me explain show two examples. Working with 32b Ruby (ruby 2.4.0p0 (2016-12-24 revision 57164) [i386-linux]) to make this issue more apparent.

```ruby
GC.disable
3_000_000.times { p Object.new.inspect }
EOR
"#<Object:0x57d49a5c>
"#<Object:0x57d499a8>
"#<Object:0x57d49930>
"#<Object:0x57d498b8>

... snip ...
"#<Object:0x828bf164>
"#<Object:0x828bf0ec>
"#<Object:0x828bf074>
"#<Object:0x828beffc>
"#<Object:0x828bef84>
^C-:2:in `p': Interrupt
from -:2:in `block in <main>'
from -:2:in `times'
from -:2:in `<main>'
"#<Object:0x8290b1f4>
```

In this example, the "object_id", which is part of the inspect object is unsigned, since it is printed using C sprintf with %p format. There are other libraries, which tries to mimic the output [1]. The implementation is approximately following:

```ruby
$ ruby << \EOR
GC.disable
3_000_000.times { p A.new.inspect }
EOR
"#<A:0x58585428>
"#<A:0x585852d4>
"#<A:0x585851bc>
"#<A:0x5858507c>
"#<A:0x58584ec4>
"#<A:0x58584d5c>
"#<A:0x58584c1c>

... snip ...
```

And the output is quite surprising to me. Why the object_id should be signed value? It doesn't make any sense to me. Is this implementation wrong or is Ruby wrong?

History

#1 - 04/03/2017 02:24 PM - nobu (Nobuyoshi Nakada)
It's not to make object_ids Bignum as possible.

#2 - 04/03/2017 08:10 PM - vo.x (Vit Ondruch)
Ok, you want to prevent Bignums, but what is the suggested solution here? Do some pack("I").unpack("L") to get the expected value? Or you can provide object_hexid [1]. Or provide some %p equivalent formatter?

Because I am afraid that the "inspect" misuse is widespread. Here are some examples, even in Ruby codebase:

https://github.com/ruby/ruby/blob/trunk/lib/rubygems/dependency_list.rb#L107
https://github.com/ruby/ruby/blob/trunk/lib/rubygems/platform.rb#L115
https://github.com/ruby/ruby/blob/trunk/lib/rdoc/attr.rb#L90

#3 - 04/21/2017 09:04 AM - vo.x (Vit Ondruch)
Last CI build of concurrent-ruby in Fedora failed again:

https://kojipkgs.fedoraproject.org/work/tasks/6296/19116296/build.log

Is there a chance to find some generic reliable solution to this? Or is it just feature and I should persuade all the project to fix their Regexp [1] or implementation of #object_id?

#4 - 04/21/2017 01:11 PM - nobu (Nobuyoshi Nakada)
vo.x (Vit Ondruch) wrote:

Is there a chance to find some generic reliable solution to this? Or is it just feature and I should persuade all the project to fix their Regexp [1] or implementation of #object_id?

They seem to have the solution already.

The default Kernel#to_s returns a string with the class name and the object ID.

**#5 - 04/27/2017 01:55 PM - vo.x (Vit Ondruch)**

I created PR fixing the projects listed above.

But still, this seems so generic and unexpected issue, possibly influencing libraries which are part of StdLib [3, 4]. I'd like to see this improved. Couldn't the #format accept "object_id" formatter for example?


**#6 - 04/28/2017 07:46 AM - nobu (Nobuyoshi Nakada)**

Such feature would be nice, but can't help libraries which support older versions.

commit 1f7154a4ae0e480774dbfe79905b21d26d5b4cbc
Author: Nobuyoshi Nakada <nobu@ruby-lang.org>
Date: Fri Apr 28 16:35:48 2017

```
`%I` for object ID
```

diff --git a/sprintf.c b/sprintf.c
index f2d51f1c7e...759e32fed4 100644
--- a/sprintf.c
+++ b/sprintf.c
@@ -25,6 +25,7 @@
 static char *fmt_setup(char*, size_t, int, int, int, int);
 static char *ruby_ultoa(unsigned long val, char *endp, int base, int octzero);
+static char *ruby_ptoa(VALUE val, char *endp, int base, int octzero);
 static char
 sign_bits(int base, const char *p)
@@ -781,6 +782,18 @@
 rb_str_format(int argc, const VALUE *argv, VALUE fmt)
 }
 break;
+ case 'I':
+ { 
+ VALUE arg = GETARG();
+ prec = (sizeof_VOIDP * CHAR_BIT + 3) / 4;
+ CHECK(prec + 2);
+ PUSH_("0x", 2);
+ t = ruby_ptoa(arg, &buf[blen + prec], 16, 0);
+ if (t > &buf[blen]) memset(&buf[blen], '0', t - &buf[blen]);
+ blen += prec;
+ }
+ break;
+ case 'd':
+ case 'i':
+ case 'o':
@@ -1255,6 +1268,18 @@
 ruby_ultoa(unsigned long val, char *endp, int base, int flags)
 return BSD__ultoa(val, endp, base, octzero, xdigs);
 }
+static char *
+ruby_ptoa(VALUE val, char *endp, int base, int flags)
+{
+ const char *xdigs = lower_hexdigits;
+ int octzero = flags & FSHARP;
+ #ifdef HAVE_SANE_QUAD_
+ return BSD__uqtoa(val, endp, base, octzero, xdigs);
+ #else
+ return BSD__ultoa(val, endp, base, octzero, xdigs);
+ #endif
+ }
+ int
 ruby_vsnprintf(char *str, size_t n, const char *fmt, va_list ap)
{

diff --git a/test/ruby/test_sprintf.rb b/test/ruby/test_sprintf.rb
#7 - 04/28/2017 01:44 PM - vo.x (Vit Ondruch)
The patch is missing documentation, otherwise I love it!

#8 - 06/27/2017 01:29 PM - vo.x (Vit Ondruch)
Yet another instance of this issue:

https://github.com/socketry/timers/pull/63

#9 - 12/13/2018 01:09 AM - headius (Charles Nutter)
You would be well-advised to avoid object_id. It does not do what you think it does. By returning a pointer reference into the garbage-collected heap, it's possible for the same object_id to refer to different objects over time.

I have proposed deprecating and eventually removing both object_id and _id2ref, since implementing them safely would largely make them useless:
https://bugs.ruby-lang.org/issues/15408

#10 - 06/01/2020 09:57 PM - jeremyevans0 (Jeremy Evans)
- Status changed from Open to Closed

With the change to use an incrementing object_id in Ruby 2.7, I don't think this is an issue anymore. The object_id counter is stored as a Ruby value, so at some point it probably goes from Fixnum to Bignum, but it should not go negative.