Ruby trunk - Feature #7517

Fixnum::MIN,MAX

12/05/2012 10:31 PM - matz (Yukihiro Matsumoto)

Status:	Assigned
Priority:	Normal
Assignee:	matz (Yukihiro Matsumoto)
Target version:	

Description

Maximum (or minimum) number of fixnum can not be get in portable fashion, so I propose Fixnum::MAX and Fixnum::MIN just like Float::MAX and Float::MIN.

Matz

Related issues:

Related to Ruby trunk - Feature #10728: Warning for Fixnum#size to use RbConf...	**Open**	**01/10/2015**

History

#1 - 12/05/2012 10:49 PM - Eregon (Benoit Daloze)

I agree, I was thinking about it the other day remembering all the custom test code to find these and comparing with Float constants.

#2 - 12/05/2012 11:10 PM - mrkn (Kenta Murata)

+1

#3 - 12/06/2012 03:24 AM - shyouhei (Shyouhei Urabe)

I'm not against these constant. Just wanted to note that (for instance) max Fixnum can portally be obtained via:

1 << (1.size * 8 - 2) - 1

#4 - 12/06/2012 03:53 AM - headius (Charles Nutter)

On Wed, Dec 5, 2012 at 12:24 PM, shyouhei (Shyouhei Urabe)
shyouhei@ruby-lang.org wrote:

> I'm not against these constant. Just wanted to note that (for instance) max Fixnum can portally be obtained via:
>
> 1 << (1.size * 8 - 2) - 1

Not on JRuby. JRuby's Fixnum is always a full signed 64-bit integer.

```
system ~/projects/jruby $ jirb
irb(main):001:0> 1.size
=> 8
irb(main):002:0> big = java.lang.Long::MAX_VALUE
=> 9223372036854775807
irb(main):003:0> big.to_s(2)
=> "111111111111111111111111111111111111111111111111111111111111111"
irb(main):004:0> big.class
=> Fixnum
irb(main):005:0> small = java.lang.Long::MIN_VALUE
=> -9223372036854775808
irb(main):006:0> small.to_s(2)
=> "-1000000000000000000000000000000000000000000000000000000000000000"
irb(main):007:0> small.class
=> Fixnum
```

I support adding MIN and MAX to Fixnum.

#5 - 12/06/2012 07:23 AM - shyouhei (Shyouhei Urabe)

On 12/05/2012 10:38 AM, Charles Oliver Nutter wrote:

> On Wed, Dec 5, 2012 at 12:24 PM, shyouhei (Shyouhei Urabe)
> shyouhei@ruby-lang.org wrote:

> I'm not against these constant. Just wanted to note that (for instance) max Fixnum can portally be obtained via:
>
> 1 << (1.size * 8 - 2) - 1

Not on JRuby. JRuby's Fixnum is always a full signed 64-bit integer.

Oh, good to know that. So you always use boxed types and unboxing is left to the VM.

#6 - 12/06/2012 02:29 PM - headius (Charles Nutter)

On Wed, Dec 5, 2012 at 4:11 PM, Urabe Shyouhei shyouhei@ruby-lang.org wrote:

> Oh, good to know that. So you always use boxed types and unboxing is left to the VM.

Correct.

#7 - 12/06/2012 11:28 PM - matz (Yukihiro Matsumoto)

- Assignee set to mame (Yusuke Endoh)

- Target version changed from 2.6 to 2.0.0

Endo san, since it's a small and useful change, can I merge this for 2.0 even after spec freeze?
Of course, you can reject as a release manager. In that case, file this proposal as "next minor" again.

Matz.

#8 - 12/07/2012 12:00 AM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

- Assignee changed from mame (Yusuke Endoh) to matz (Yukihiro Matsumoto)

matz (Yukihiro Matsumoto) wrote:

> Endo san, since it's a small and useful change, can I merge this for 2.0 even after spec freeze?
> Of course, you can reject as a release manager. In that case, file this proposal as "next minor" again.

As a release manager, okay. Because matz accepted the proposal :-)

Personally, however, I'm not sure when it is useful.
I guess that you want to avoid an boxed integer for saving memory on an embedded system, right?
But I don't know how it is helpful.

In addition, I don't understand why we should distinguish between Fixnum and Bignum.
Isn't the difference just an implementation-defined technicality?
I hope that they will be integrated to one class and that the difference will become invisible to users.
I'm afraid if Fixnum::Max will make the integration difficult.

--
Yusuke Endoh mame@tsg.ne.jp

#9 - 12/07/2012 12:38 AM - matz (Yukihiro Matsumoto)

It is especially useful for mruby that does not have Bignum, but I believe it's useful to tell how big fixnum is portable among Ruby implementations.

Matz.

#10 - 12/09/2012 09:07 PM - mame (Yusuke Endoh)

Still, I'm not sure what problem (in mruby?) is resolved by the existence of Fixnum::MAX.

How:

```
n3 = n1 + n2
raise "unboxed" if n3 > Fixnum::MAX
```

differs from:

```
n3 = n1 + n2
raise "unboxed" if !n3.instance_of?(Fixnum)
```

? Just easier to read?

--
Yusuke Endoh mame@tsg.ne.jp

#11 - 12/10/2012 02:29 PM - Anonymous

Hi,

In message "Re: [ruby-core:50708] [ruby-trunk - Feature #7517] Fixnum::MIN,MAX"
on Sun, 9 Dec 2012 21:07:37 +0900, "mame (Yusuke Endoh)" mame@tsg.ne.jp writes:

|Still, I'm not sure what problem (in mruby?) is resolved by the existence of Fixnum::MAX.
|
|How:
|
| n3 = n1 + n2
| raise "unboxed" if n3 > Fixnum::MAX
|
|differs from:
|
| n3 = n1 + n2
| raise "unboxed" if !n3.instance_of?(Fixnum)
|
|? Just easier to read?

For cases like the following:

printf "integers %d .. %d are unboxed\n", Fixnum::MIN, Fixnum::MAX

you don't have to create Fixnum instances.

matz.

#12 - 12/11/2012 11:04 PM - naruse (Yui NARUSE)

What is the use case?

If this is an mruby context, I suspect they should be Integer::MAX and Integer::MIN.

#13 - 12/12/2012 08:23 AM - Anonymous

Hi,

In message "Re: [ruby-core:50760] [ruby-trunk - Feature #7517] Fixnum::MIN,MAX"
on Tue, 11 Dec 2012 23:04:30 +0900, "naruse (Yui NARUSE)" naruse@airemix.jp writes:

|What is the use case?

See #11 in portable way (i.e. portable among CRuby, JRuby, mruby, etc.)

|If this is an mruby context, I suspect they should be Integer::MAX and Integer::MIN.

Maybe, since mruby doesn't have Bignum. But CRuby cannot have
Integer::MAX, so my intention to be portable will not be satisfied.

matz.

#14 - 12/13/2012 10:34 AM - tarui (Masaya Tarui)

Hi,

It cannot imagine except the use-case of liking to know how far an integer being treated.
so, I think Integer::MAX better then Fixnum::MAX.

CRuby's Integer::MAX is INFINITY,isn't it?
If I introduce a Integer subclass for extending a digit number on mruby, is it a Fixnum?
Currently, Fixnum can have instance, and It is an immediate class.
But, Is Fixnum changed into an abstract class?

#15 - 12/13/2012 02:53 PM - Anonymous

Hi,

In message "Re: [ruby-core:50849] [ruby-trunk - Feature #7517] Fixnum::MIN,MAX"

on Thu, 13 Dec 2012 10:34:07 +0900, "tarui (Masaya Tarui)" tarui@prx.jp writes:

|Hi,
|
|It cannot imagine except the use-case of liking to know how far an integer being treated.
|so, I think Integer::MAX better then Fixnum::MAX.
|
|CRuby's Integer::MAX is INFINITY,isn't it?

INFINITY is a float. Are you proposing introducing Bignum::INFINITY?
In any way, Integer::MAX being (Bignum::)INFINITY do not have any
additional info than having Bignum. So I don't consider it useful.

|If I introduce a Integer subclass for extending a digit number on mruby, is it a Fixnum?
|Currently, Fixnum can have instance, and It is an immediate class.
|But, Is Fixnum changed into an abstract class?

Fixnum is by definition immediate number, so that it has bound limit
(thus I proposal MIN and MAX for it). Integer is not. Even if I add
another immediate integer (say SmallInt a la Smalltalk) to mruby, it
has no relation to Fixnum.

```
                          matz.
```

#16 - 01/29/2013 05:16 PM - ko1 (Koichi Sasada)

- Target version changed from 2.0.0 to 2.6

#17 - 08/30/2014 10:30 PM - cremno (cremno phobia)

Even if the use-cases aren't really convincing, I think the informational purpose alone is a good enough argument to add it. If you search for “ruby fixnum max” or something similar, you'll find the interest is there, but the solution(s) usually only work in CRuby and are not as easy to understand as the proposed constants.

```
diff --git a/numeric.c b/numeric.c
index 1e971f4..b57c686 100644
--- a/numeric.c
+++ b/numeric.c
@@ -4062,6 +4062,15 @@ Init_Numeric(void)

     rb_cFixnum = rb_define_class("Fixnum", rb_cInteger);

+    /*
+     * The minimum value of a Fixnum.
+     */
+    rb_define_const(rb_cFixnum, "MIN", LONG2FIX(FIXNUM_MIN));
+    /*
+     * The maximum value of a Fixnum.
+     */
+    rb_define_const(rb_cFixnum, "MAX", LONG2FIX(FIXNUM_MAX));
+
     rb_define_method(rb_cFixnum, "to_s", fix_to_s, -1);
     rb_define_alias(rb_cFixnum, "inspect", "to_s");

diff --git a/test/ruby/test_fixnum.rb b/test/ruby/test_fixnum.rb
index 8b2cf2e..8c896ec 100644
--- a/test/ruby/test_fixnum.rb
+++ b/test/ruby/test_fixnum.rb
@@ -312,4 +312,20 @@ class TestFixnum < Test::Unit::TestCase
     assert_equal(1, 5.remainder(4))
     assert_predicate(4.remainder(Float::NAN), :nan?)
   end
+
+  def test_min
+    assert_kind_of(Fixnum, Fixnum::MIN)
+    assert_kind_of(Bignum, Fixnum::MIN - 1)
+    if RUBY_ENGINE == 'ruby'
+      assert_equal(-2 ** (1.size * 8 - 2), Fixnum::MIN)
+    end
+  end
+
+  def test_max
+    assert_kind_of(Fixnum, Fixnum::MAX)
+    assert_kind_of(Bignum, Fixnum::MAX + 1)
+    if RUBY_ENGINE == 'ruby'
```

```
+      assert_equal(2 ** (1.size * 8 - 2) - 1, Fixnum::MAX)
+    end
+  end
 end
```

#18 - 09/01/2014 04:49 PM - headius (Charles Nutter)

An obvious use case would be to make algorithms know how large an integer they can represent in Fixnum without overflowing to Bignum. This is strictly a Fixnum thing, too, so I think having the constants on Fixnum is exactly right. In JRuby, Fixnum::MAX and MIN would be 64-bit signed integer max and min.

#19 - 01/10/2015 01:19 PM - akr (Akira Tanaka)

- Related to Feature #10728: Warning for Fixnum#size to use RbConfig::SIZEOF['long'] added

#20 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)