Ruby master - Feature #12753

Useful operator to check bit-flag is true or false

09/12/2016 06:47 AM - tagomoris (Satoshi TAGOMORI)

Status:	Closed	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:	2.5	

Description

Ruby's 0 is truthy value. It's useful for many cases, but it's confusing and I made many bugs when I'm writing code to handle binary data, because my thought is almost same with one to write C code in such situation.

```
n = get_integer_value
if n & 0b10100000
  # code for the case when flag is true
else
  # never comes here :(
end
```

IMO it's very useful to have methods for such use-cases, like #and? and #xor? (#or? looks not so useful... I can't imagine the use case of this operator, but it's better to have for consistency).

```
n = get_integer_value
case
when n.and?(0b10000000)
  # negative signed char
when n.and?(0b01110000)
  # large positive
else
  # small positive
end
```

Associated revisions

Revision 0da34dbb - 12/12/2017 09:12 AM - naruse (Yui NARUSE)

Integer#allbits?, Integer#anybits?, Integer#nobits? [Feature #12753]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@61147 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 61147 - 12/12/2017 09:12 AM - naruse (Yui NARUSE)

Integer#allbits?, Integer#anybits?, Integer#nobits? [Feature #12753]

Revision 61147 - 12/12/2017 09:12 AM - naruse (Yui NARUSE)

Integer#allbits?, Integer#anybits?, Integer#nobits? [Feature #12753]

Revision 61147 - 12/12/2017 09:12 AM - naruse (Yui NARUSE)

Integer#allbits?, Integer#anybits?, Integer#nobits? [Feature #12753]

Revision 61305 - 12/17/2017 06:19 PM - marcandre (Marc-Andre Lafortune)

Integer#{any|all|no}_bits: Fix coercion. Add specs [#12753]

Revision 61305 - 12/17/2017 06:19 PM - marcandre (Marc-Andre Lafortune)

Integer#{any|all|no}_bits: Fix coercion. Add specs [#12753]

Revision 61305 - 12/17/2017 06:19 PM - marcandre (Marc-Andre Lafortune)

Integer#{any|all|no}_bits: Fix coercion. Add specs [#12753]

History

#1 - 09/12/2016 07:11 AM - shyouhei (Shyouhei Urabe)

- Description updated

#2 - 11/25/2016 07:28 AM - matz (Yukihiro Matsumoto)

I understand the demand. But and? is an unacceptable name.
Any idea?

Matz.

#3 - 11/25/2016 07:28 AM - naruse (Yui NARUSE)

What about bittest?

#4 - 11/25/2016 07:45 AM - matz (Yukihiro Matsumoto)

bittest? sounds reasonable. Accepted.

Matz.

#5 - 11/25/2016 07:59 AM - herwinw (Herwin Quarantainenet)

I can't say the usage of bittest? is directly clear to me. Does it test if resulting integer is not equal to 0? And would we have to use it this way?

```
if (n & 0b10100000).bittest?
```

I think a name like Integer#binary_and? (maybe shortened to #binand?) would result in cleaner code

```
if n.binary_and?(0b10100000)
```

Of course this would require several other implementations as well, for all other binary operators

#6 - 11/25/2016 03:01 PM - naruse (Yui NARUSE)

Herwin Quarantainenet wrote:

> I can't say the usage of bittest? is directly clear to me. Does it test if resulting integer is not equal to 0? And would we have to use it this way?
>
> ```
> if (n & 0b10100000).bittest?
> ```

Like

```
if n.bittest?(0b10100000)
```

> I think a name like Integer#binary_and? (maybe shortened to #binand?) would result in cleaner code
>
> ```
> if n.binary_and?(0b10100000)
> ```
>
> Of course this would require several other implementations as well, for all other binary operators

There's two AND, bitwise and logical.
Therefore it can be bit_and?, but there's no reason to write logical AND as a method, which can be written with &&.

#7 - 11/26/2016 02:51 PM - herwin (Herwin W)

```
if n.bittest?(0b10100000)
```

If I encountered that code without having the context of this case, I wouldn't know what what the equivalent behaviour would be:

```
if n & 0b10100000 != 0           #=> Is at least one bit of the argument set?
if n & 0b10100000 == 0b10100000  #=> Are all the bits of the argument set?
```

> There's two AND, bitwise and logical.
> Therefore it can be bit_and?, but there's no reason to write logical AND as a method, which can be written with &&.

I was actually thinking about the other bitwise/binary operators here, like | and ^

#8 - 12/01/2016 08:33 AM - naruse (Yui NARUSE)

Herwin W wrote:

> ```
> if n.bittest?(0b10100000)
> ```

> If I encountered that code without having the context of this case, I wouldn't know what what the equivalent behaviour would be:
>
> ```
> if n & 0b10100000 != 0           #=> Is at least one bit of the argument set?
> if n & 0b10100000 == 0b10100000  #=> Are all the bits of the argument set?
> ```

Above one.

> > There's two AND, bitwise and logical.
> > Therefore it can be bit_and?, but there's no reason to write logical AND as a method, which can be written with &&.
>
> I was actually thinking about the other bitwise/binary operators here, like | and ^

I can't show a use case of |.
^ is maybe useful but the name is difficult.

#9 - 12/01/2016 09:16 AM - shugo (Shugo Maeda)

Yui NARUSE wrote:

> Herwin W wrote:
>
> > ```
> > if n.bittest?(0b10100000)
> > ```
> >
> > If I encountered that code without having the context of this case, I wouldn't know what what the equivalent behaviour would be:
> >
> > ```
> > if n & 0b10100000 != 0           #=> Is at least one bit of the argument set?
> > if n & 0b10100000 == 0b10100000  #=> Are all the bits of the argument set?
> > ```
>
> Above one.

IBM InfoSphere and MS FoxPro have BITTEST(), but its second argument is the bit position
to be tested.

http://www.ibm.com/support/knowledgecenter/SSZJPZ_11.3.0/com.ibm.swg.im.iis.ds.basic.doc/topics/r_dsbasic_BITTEST_function.html
https://msdn.microsoft.com/en-us/library/aa977348(v=vs.71).aspx

This behavior seems to fit the name bittest, compared to the proposed one.

#10 - 12/01/2016 09:21 AM - shugo (Shugo Maeda)

Shugo Maeda wrote:

> IBM InfoSphere and MS FoxPro have BITTEST(), but its second argument is the bit position
> to be tested.
>
> http://www.ibm.com/support/knowledgecenter/SSZJPZ_11.3.0/com.ibm.swg.im.iis.ds.basic.doc/topics/r_dsbasic_BITTEST_function.html
> https://msdn.microsoft.com/en-us/library/aa977348(v=vs.71).aspx
>
> This behavior seems to fit the name bittest, compared to the proposed one.

I didn't mean to propose this behavior.
I just meant to point out that bittest? may not be suitable for the proposed behavior.

#11 - 09/15/2017 04:31 PM - tagomoris (Satoshi TAGOMORI)

How about bitmask_test? or bitflag_test?

shugo (Shugo Maeda) wrote:

> Shugo Maeda wrote:
>
> > IBM InfoSphere and MS FoxPro have BITTEST(), but its second argument is the bit position
> > to be tested.
> >
> > http://www.ibm.com/support/knowledgecenter/SSZJPZ_11.3.0/com.ibm.swg.im.iis.ds.basic.doc/topics/r_dsbasic_BITTEST_function.html
> > https://msdn.microsoft.com/en-us/library/aa977348(v=vs.71).aspx
> >
> > This behavior seems to fit the name bittest, compared to the proposed one.

> I didn't mean to propose this behavior.
> I just meant to point out that bittest? may not be suitable for the proposed behavior.

#12 - 10/19/2017 06:49 AM - akr (Akira Tanaka)

How about Integer#has_allbits?(n), Integer#has_somebits?(n) and Integer#has_nobits?(n) ?

```
class Integer
  def has_allbits?(n) self & n == n end
  def has_somebits?(n) self & n != 0 end
  def has_nobits?(n) self & n == 0 end
end
```

#13 - 10/19/2017 06:58 AM - matz (Yukihiro Matsumoto)

has_* is not acceptable. It's not compatible with other method names.
I vote for allbit?, anybit? and nobit?. I am not sure about plurality though.

Matz.

#14 - 10/19/2017 07:11 AM - knu (Akinori MUSHA)

Speaking of plurality, what about:

a.bit?(b) → a & b != 0
a.bits?(b) → a & b == b

#15 - 10/19/2017 10:13 AM - phluid61 (Matthew Kerwin)

I think plural makes most sense:

```
a.allbits? b #→ a & b == b
a.anybits? b #→ a & b != 0
a.nobits? b  #→ a & b == 0
```

It introduces a strange paradox, though:

```
a.allbits? 0 #→ true
a.nobits? 0  #→ true
```

#16 - 12/01/2017 12:31 AM - aycabta (aycabta .)

phluid61 (Matthew Kerwin) wrote:

> It introduces a strange paradox, though:
>
> ```
> a.allbits? 0 #→ true
> a.nobits? 0  #→ true
> ```

I discussed it with watson1978 (Shizuo Fujita) (Shizuo Fujita). We guess the behavior is not strange.

The allbits? means "The receiver checks that all standing bits of the argument don't sit on itself".

```
a.allbits? 0 #→ true
```

In this case, "all standing bits of the argument don't sit on the receiver " because "all standing bits of argument" is nothing. So it returns true. I think this is correct. If I have to choose a word, it's reasonable specification.

```
a.nobits? 0  #→ true
```

I think this is correct in the same way.

#17 - 12/01/2017 04:26 PM - naruse (Yui NARUSE)

- Target version set to 2.5

- Assignee set to matz (Yukihiro Matsumoto)

- Status changed from Open to Assigned

A patch is as follows:

```
diff --git a/numeric.c b/numeric.c
index 1858113c09..511155a3ac 100644
--- a/numeric.c
+++ b/numeric.c
@@ -3209,6 +3209,45 @@ int_even_p(VALUE num)
     return Qfalse;
 }

+/*
+ *  call-seq:
+ *     int.allbits?(mask)  ->  true or false
+ *
+ *  Returns +true+ if all bits of <code>+int+ & +mask+</code> is 1.
+ */
+
+static VALUE
+int_allbits_p(VALUE num, VALUE mask)
+{
+    return rb_int_equal(rb_int_and(num, mask), mask);
+}
+
+/*
+ *  call-seq:
+ *     int.anybits?(mask)  ->  true or false
+ *
+ *  Returns +true+ if any bits of <code>+int+ & +mask+</code> is 1.
+ */
+
+static VALUE
+int_anybits_p(VALUE num, VALUE mask)
+{
+    return num_zero_p(rb_int_and(num, mask)) ? Qfalse : Qtrue;
+}
+
+/*
+ *  call-seq:
+ *     int.nobits?(mask)  ->  true or false
+ *
+ *  Returns +true+ if no bits of <code>+int+ & +mask+</code> is 1.
+ */
+
+static VALUE
+int_nobits_p(VALUE num, VALUE mask)
+{
+    return num_zero_p(rb_int_and(num, mask));
+}
+
 /*
  *  Document-method: Integer#succ
  *  Document-method: Integer#next
@@ -5396,6 +5435,9 @@ Init_Numeric(void)
     rb_define_method(rb_cInteger, "integer?", int_int_p, 0);
     rb_define_method(rb_cInteger, "odd?", int_odd_p, 0);
     rb_define_method(rb_cInteger, "even?", int_even_p, 0);
+    rb_define_method(rb_cInteger, "allbits?", int_allbits_p, 1);
+    rb_define_method(rb_cInteger, "anybits?", int_anybits_p, 1);
+    rb_define_method(rb_cInteger, "nobits?", int_nobits_p, 1);
     rb_define_method(rb_cInteger, "upto", int_upto, 1);
     rb_define_method(rb_cInteger, "downto", int_downto, 1);
     rb_define_method(rb_cInteger, "times", int_dotimes, 0);
diff --git a/test/ruby/test_integer_comb.rb b/test/ruby/test_integer_comb.rb
index 80d08cac04..1ad13dd31b 100644
--- a/test/ruby/test_integer_comb.rb
+++ b/test/ruby/test_integer_comb.rb
@@ -457,6 +457,30 @@ def test_even_odd
     }
   end

+  def test_allbits_p
+    VS.each {|a|
+      VS.each {|b|
+        assert_equal((a & b) == b, a.allbits?(b), "(#{a}).allbits?(#{b}")
+      }
+    }
+  end
```

```
+
+  def test_anybits_p
+    VS.each {|a|
+      VS.each {|b|
+        assert_equal((a & b) != 0, a.anybits?(b), "(#{a}).anybits?(#{b}")
+      }
+    }
+  end
+
+  def test_nobits_p
+    VS.each {|a|
+      VS.each {|b|
+        assert_equal((a & b) == 0, a.nobits?(b), "(#{a}).nobits?(#{b}")
+      }
+    }
+  end
+
   def test_to_s
     2.upto(36) {|radix|
       VS.each {|a|
```

#18 - 12/01/2017 05:34 PM - aycabta (aycabta .)

Hi naruse (Yui NARUSE), matz said below.

matz (Yukihiro Matsumoto) wrote:

> I vote for allbit?, anybit? and nobit?. I am not sure about plurality though.

For your action, as you know, Dir.exists? and File.exists? are deprecated. What do you think about this?

#19 - 12/01/2017 08:56 PM - phluid61 (Matthew Kerwin)

aycabta (ayca bta) wrote:

> Hi naruse (Yui NARUSE), matz said below.
>
> matz (Yukihiro Matsumoto) wrote:
>
> > I vote for allbit?, anybit? and nobit?. I am not sure about plurality though.
>
> For your action, as you know, Dir.exists? and File.exists? are deprecated. What do you think about this?

exist/exists is tense, bit/bits is plurality, so it is unrelated.

#20 - 12/01/2017 10:34 PM - aycabta (aycabta .)

phluid61 (Matthew Kerwin) wrote:

> exist/exists is tense, bit/bits is plurality, so it is unrelated.

Oh, thank you...I understand.

#21 - 12/06/2017 05:10 PM - aycabta (aycabta .)

In Ruby 2.5, Ripper::Lexer::State is introduced:
https://bugs.ruby-lang.org/projects/ruby-trunk/repository/revisions/60945/entry/ext/ripper/lib/ripper/lexer.rb#L49

It is for lex_state of parse.y, and has #& and #| for bit operations with lex_state_bits:
https://bugs.ruby-lang.org/projects/ruby-trunk/repository/revisions/60945/entry/parse.y#L78

RDoc uses it:
https://bugs.ruby-lang.org/projects/ruby-trunk/repository/revisions/60945/entry/lib/rdoc/parser/ripper_state_lex.rb#L321

If Integer#allbit? is implemented at 2.5, it's good for Ripper::Lexer::State and I'll use it for RDoc on 2.5.

#22 - 12/12/2017 09:12 AM - naruse (Yui NARUSE)

- Status changed from Assigned to Closed

Applied in changeset trunk|r61147.

Integer#allbits?, Integer#anybits?, Integer#nobits? [Feature #12753]

#23 - 12/17/2017 06:21 PM - marcandre (Marc-Andre Lafortune)

When writing specs, I discovered that coercion was failing for allbits.

I modified all three methods to apply coercion with to_int if needed.