String #uminus should de-dupe unconditionally

02/14/2018 10:45 PM - sam.saffron (Sam Saffron)

Status: Closed
Priority: Normal
Assignee:
Target version:

Description
continuing: https://bugs.ruby-lang.org/issues/14475

Current documentation for String uminus says: "If the string is frozen, then return the string itself."

Trouble is that there is no simple way to de-duplicate unconditionally without inefficiency:

Say x is an arbitrary string (either frozen or unfrozen) :

\[
\begin{align*}
  x &= -x \ # \text{ may return a non fstring} \\
  x &= -+x \ # \text{ will return fstring, but makes an unneeded copy} \\
  x &= -x.dup \ # \text{ fstring again, unneeded copy} \\
  x &= x.frozen? \ ? -+x : -x \ # \text{ too verbose, unneeded copy}
\end{align*}
\]

Instead why not change it so - is deduped unconditionally?

I would argue this is worth backporting, cause if we are making fstring optimisations now, we are going to be stuck with legacy inefficient code going forward.

An alternative may be a c-extension gem that adds #fstring to String but that just feel wrong.

I think the documentation should say:

String uminus says: "If the string is de-duplicated, then return the string itself."

Happy to make the change it is quite simple:(FL_TEST(str, RSTRING_FSTR)

Related issues:
Related to Ruby master - Bug #14475: String de-duplication is broken in files...

Associated revisions
Revision 256411b4 - 06/04/2018 11:26 PM - normal
String#uminus dedupes unconditionally
[Feature #14478] [ruby-core:85669]
Thanks-to: Sam Saffron sam.saffron@gmail.com
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63566 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 63566 - 06/04/2018 11:26 PM - normalperson (Eric Wong)
String#uminus dedupes unconditionally
[Feature #14478] [ruby-core:85669]
Thanks-to: Sam Saffron sam.saffron@gmail.com

Revision 63566 - 06/04/2018 11:26 PM - normal
String#uminus dedupes unconditionally
[Feature #14478] [ruby-core:85669]
Thanks to: Sam Saffron sam.saffron@gmail.com

History

#1 - 02/14/2018 11:25 PM - phluid61 (Matthew Kerwin)

Why not make it impossible to create a frozen string that isn't deduped? That way the current -x works fine by default.

(I thought that was already the case..?)

#2 - 02/14/2018 11:38 PM - sam.saffron (Sam Saffron)

Why not make it impossible to create a frozen string that isn't deduped?

This can not be done without enormous breakage, it would break the semantics of freeze, was already discussed with Matz and decided against.

```ruby
puts x.object_id
x.freeze
puts x.object_id # must not change
```

#3 - 02/15/2018 01:28 AM - phluid61 (Matthew Kerwin)

sam.saffron (Sam Saffron) wrote:

```ruby
puts x.object_id
x.freeze
puts x.object_id # must not change
```

That makes sense, thanks. It seems there are two parts to the issue:

1. we should expose deduplication in the language

2. -@ should be one such exposure

I agree with 1. I also agree with 2, because I can't think of anything legitimate that depends on frozen duplicated string ids not changing.

#4 - 02/15/2018 01:39 AM - duerst (Martin Dürst)

- Related to Bug #14475: String de-duplication is broken in files with frozen_string_literal: true added

#5 - 02/15/2018 02:04 AM - sam.saffron (Sam Saffron)

-@ should be one such exposure

Oh this is already the case, see the source:

```ruby
/*
 * call-seq:
 * -str -> str (frozen)
 *
 * If the string is frozen, then return the string itself.
 * *
 * If the string is not frozen, return a frozen, possibly pre-existing
 * copy of it.
 */
static VALUE
str_uminus(VALUE str)
{
    if (OBJ_FROZEN(str)) {
        return str;
    } else {
        return rb_fstring(str);
    }
}
```

I am proposing to change this to:
/*
 * call-seq:
 * -str -> str (frozen)
 *
 * If the string is de-duplicated, then return the string itself.
 *
 * If the string is not return a frozen, possibly pre-existing copy of it.
 */
static VALUE
str_uminus(VALUE str)
{
  if (FL_TEST(str, RSTRING_FSTR)) {
    return str;
  } else {
    return rb_fstring(str);
  }
}

#6 - 02/15/2018 02:13 AM - phluid61 (Matthew Kerwin)
sam.saffron (Sam Saffron) wrote:
   -@ should be one such exposure

Oh this is already the case, see the source:

That's an implementation detail. -@'s contract is about freezing, and says nothing about deduplication. What you're proposing is to change the contract, and specify the deduplication.

#7 - 02/15/2018 02:21 AM - sam.saffron (Sam Saffron)
The contract says "possibly pre-existing copy of it." ... where possibly means "if the string is not frozen"

So the contract now is "return a de-duplicated string if the string is not frozen" which is a bit of a crazy contract if you ask me, instead the contract should simply be "return a de-duplicated string always"

What I am asking for here is change str_uminus from:
"sometimes return a de-duplicated string"
To
"always return a de-duplicated string"

I feel the change is not risky at all and will not break anyone in practice, I also think it should be backported to 2.5 cause --x is an ugly and horrible pattern for us to be promoting.

Also related:
https://bugs.ruby-lang.org/issues/11782
and
https://bugs.ruby-lang.org/issues/13077

Also to quote Matz:

For the time being, let us make -@ to call rb_fstring.
If users want more descriptive name, let's discuss later.
In my opinion, fstring is not acceptable.

So there you have it it seems Matz expected this not to be a sometimes thing.

#8 - 02/15/2018 05:14 AM - ko1 (Koichi Sasada)
Summary of this issue:

Background
We changed to return dedup-ed string with String-@.  
[https://bugs.ruby-lang.org/issues/13077#note-7](https://bugs.ruby-lang.org/issues/13077#note-7)

For the time being, let us make -@ to call rb_fstring.

### Issue

However, it returns non-deduped string when self is frozen because of OBJ_FREEZE(str) check ([https://bugs.ruby-lang.org/issues/14478#note-5](https://bugs.ruby-lang.org/issues/14478#note-5)).

It is inconsistent. We can't use it for dedup functionality.

### Proposal

We always need to return dedup-ed strings with String#-@

This proposal makes sense for me, at least I understand "inconsistency".

#### #9 - 02/15/2018 06:51 AM - normalperson (Eric Wong)

To reduce compatibility risk, how about we do heap sharing with fstring for already-frozen strings when calling uminus?

It would deduplicate large strings which require separate allocations; but won't reduce object count or help with small strings, though.

#### #10 - 02/15/2018 09:43 AM - sam.saffron (Sam Saffron)

> how about we do heap sharing

I am very against this, I just don't see the compatibility risk I think myself and maybe only another like 5 people know about the uminus change in 2.5 :) I am willing to bet a very large sum of money nobody is depending on object_id stability when calling uminus on a frozen string. This would be people who

1. Do not depend on object_id stability when calling uminus on non frozen strings
2. Somewhat for some crazy reason depend on object_id stability when called on frozen strings

I also can not even think of an actual use case where you would depend on this inconsistent behavior

If anything I think we are not going far enough, I think uminus should unconditionally untaint, but that is a topic for another issue. Also I wish we would just remove taint and trust.

#### #11 - 02/15/2018 12:10 PM - Eregon (Benoit Daloze)

Since it's var = -str, it's very clear the return value should be used (and deduped) and we can't expect str to changed.

So if the only concern is returning str if str is frozen in -str, I agree that seems a very low compatibility risk.

#### #12 - 02/15/2018 04:48 PM - shevegen (Robert A. Heiler)

I don't have any pro or con opinion here but I wanted to comment on one statement:

maybe only another like 5 people know about the uminus change in 2.5 :) 

Actually I noticed it first when nobu was using it, several months ago. :)

Noticed it in some of the ruby changelog and the method was new to me.

I have not used uminus myself, mostly because I do not like the syntax, even though I understand that it is easier than adding checks for frozen strings and using .dup (which I actually am doing nowadays a lot, even though I could use uminus instead, but I like the .frozen? and .dup check, even though it is more code; I use more frozen strings these days, mostly because I have noticed a significant speed improvement in general; and for ruby scripts that do a lot of work, I actually try to "optimize" these simply because it can make a big difference between...
e.g. 30 seconds and 20 seconds execution run, for larger ruby scripts/ruby projects).

**#13 - 02/20/2018 05:54 AM - matz (Yukihiro Matsumoto)**
I agree with de-duping. I also want the reference document to be updated.

Matz.

**#14 - 02/27/2018 07:03 AM - sam.saffron (Sam Saffron)**
I just tried the trivial patch of:

```diff
diff --git a/string.c b/string.c
index ebf5618..a9b991f 100644
--- a/string.c
+++ b/string.c
@@ -2605,20 +2605,16 @@ str_uplus(VALUE str)
 * call-seq:
 * - str -> str (frozen)
 *
--
- If the string is frozen, then return the string itself.
- * Return a frozen, possibly pre-existing
- * copy of the string.
- *
- If the string is not frozen, return a frozen, possibly pre-existing
- * copy of it.
+ String will be deduplicated as long as it is not tainted,
+ * or has any instance vars set on it.
 */
static VALUE
str_uminus(VALUE str)
{
- if (OBJ_FROZEN(str)) {
- return str;
- }
- else {
- return rb_fstring(str);
- }
+ return rb_fstring(str);
}
```

But this seems to break the test/ruby/test_string.rb cause there is some edge case we are missing here.

**#15 - 02/27/2018 08:52 AM - normalperson (Eric Wong)**

```
sam.saffron@gmail.com wrote:
But this seems to break the test/ruby/test_string.rb cause there is some edge case we are missing here.

I didn't see a failure in test_string.rb, but got one rubyspec failure:

1)
String#-@ is an identity function if the string is frozen FAILED
Expected "this string is frozen"
to be identical to "this string is frozen"

$srcdir/ruby/spec/ruby/core/string/uminus_spec.rb:38:in `block (4 levels) in <top (required)>'
$srcdir/ruby/spec/ruby/core/string/uminus_spec.rb:4:in `block in <top (required)>'
$srcdir/ruby/spec/ruby/core/string/uminus_spec.rb:3:in `<top (required)>'

I guess this requires a change to rubyspec:
```

diff --git a/spec/ruby/core/string/uminus_spec.rb b/spec/ruby/core/string/uminus_spec.rb
index f18e6b1234..9474666e30 100644
--- a/spec/ruby/core/string/uminus_spec.rb
+++ b/spec/ruby/core/string/uminus_spec.rb
@@ -35,11 +35,9 @@
it "is an identity function if the string is frozen" do
 dynamic = %w(this string is frozen).join(' ').freeze
-     (-dynamic).should equal(dynamic)
-     
```

09/01/2022
String#uminus dedupes unconditionally

[Feature #14478] [ruby-core:85669]

Thanks-to: Sam Saffron sam.saffron@gmail.com

Files

<table>
<thead>
<tr>
<th>0001-String-uminus-dedupes-unconditionally.patch</th>
<th>2.03 KB</th>
<th>06/04/2018</th>
</tr>
</thead>
</table>

09/01/2022