Ruby trunk - Bug #9229

[patch] expose rb_fstring() as String#dedup

12/08/2013 04:12 PM - tmm1 (Aman Gupta)

Status:	Closed		
Priority:	Normal		
Assignee:	matz (Yukihiro Matsumoto)		
Target version:	2.1.0		
ruby -v:	trunk	**Backport:**	1.9.3: UNKNOWN, 2.0.0: UNKNOWN

Description

After recent commits, ruby is using the new rb_fstring() API extensively inside the VM to de-duplicate internal strings.
This technique has proven very successful, and reduced the majority of long-lived strings in large applications.

I think we should expose this functionality to ruby as well.

This api would allow gem/library maintainers to de-duplicate strings in any long-lived objects they create.
For example, many gems today contain large constant lookup tables that contain many strings. These tables are often loaded via yaml or json from disk:

Addressable::IDNA::UNICODE_DATA
MIME::Types.instance_variable_get(:@**types**)
TZInfo::Timezone.class_variable_get(:@@loaded_zones)
ActiveSupport::Multibyte::UCD
TTFunk::Table::Post::Format10::POSTSCRIPT_GLYPHS
Money::Currency::TABLE
Rack::Utils::HTTP_STATUS_CODES

In our app, strings in these tables account for a huge portion of long-lived strings in our runtime.
Another example is strings referenced by long-lived rubygem specifications. From a ObjectSpace.dump_all snapshot:

$ grep '"MIT"' heap.json | wc -l
73

With the proposed patch, a user (or ideally library maintainer) can easily de-duplicate strings in known long-lived objects:

```
Gem::Specification._all.each{ |s| s.license = s.license.dedup if s.license }.size
=> 304
```

A simple implementation follows.

diff --git a/string.c b/string.c
index f8dd03d..8294c78 100644
--- a/string.c
+++ b/string.c
@@ -145,7 +145,7 @@ fstr_update_callback(st_data_t *key, st_data_t *value, st_data_t arg, int existi
return ST_STOP;
}

- if (STR_SHARED_P(str)) {
- if (STR_SHARED_P(str) || RBASIC_CLASS(str) != rb_cString) { /* str should not be shared */ str = rb_enc_str_new(RSTRING_PTR(str), RSTRING_LEN(str), STR_ENC_GET(str)); OBJ_FREEZE(str); @@ -8278,6 +8278,20 @@ str_scrub_bang(int argc, VALUE *argv, VALUE str) return str; }

+/*

- * call-seq:
- * str.dedup -> str
- *
- * Returns a frozen version of this string. If possible, an existing
- * object with the same value will be returned.

- */ + +static VALUE +str_dedup(VALUE self) +{
- return rb_fstring(self); +} + /**
  - Document-class: Symbol * @@ -8768,6 +8782,7 @@ Init_String(void) rb_define_method(rb_cString, "scrub", str_scrub, -1); rb_define_method(rb_cString, "scrub!", str_scrub_bang, -1); rb_define_method(rb_cString, "freeze", rb_obj_freeze, 0);

- rb_define_method(rb_cString, "dedup", str_dedup, 0);

  rb_define_method(rb_cString, "to_i", rb_str_to_i, -1);
  rb_define_method(rb_cString, "to_f", rb_str_to_f, 0);
  diff --git a/test/ruby/test_string.rb b/test/ruby/test_string.rb
  index 7ce1c06..d8c414b 100644
  --- a/test/ruby/test_string.rb
  +++ b/test/ruby/test_string.rb
  @@ -600,6 +600,13 @@ class TestString < Test::Unit::TestCase
  end
  end

- def test_dedup

- fstr = "foobar".freeze
  +

- assert_same fstr, S("foobar").dedup

- assert_same fstr, S("foobar").dup.dedup

- end
  +
  def test_each
  save = $/
  $/ = "\n"

Related issues:

Is duplicate of Ruby trunk - Feature #8977: String#frozen that takes advantag...	**Assigned**

History

#1 - 12/08/2013 06:12 PM - shyouhei (Shyouhei Urabe)

It sounds a bit too rash. For instance other Ruby implementations might not need such thing.

I prefer that kind of optimization to happen automatically when needed, than force programmer to cast a spell.

#2 - 12/09/2013 05:31 AM - phluid61 (Matthew Kerwin)

Would this be better as MRI's implementation of String#freeze ?

#3 - 12/09/2013 05:51 AM - tmm1 (Aman Gupta)

> I prefer that kind of optimization to happen automatically when needed, than force programmer to cast a spell.

I agree, but I'm not sure how MRI can perform this optimization automatically. For instance, in the rubygems case:

```
Gem::Specification.new do |s|
s.license = "MIT"
end
```

Above, there is no way for the VM to know that the string "MIT" should be immutable. The VM must expect that the user might try to modify it's value, and so it will always return a duplicated copy of the underlying frozen string.

For this case, the user can write "MIT".freeze. But since this only works on string literals, it cannot be used to perform de-duplication on strings loaded from json/yaml files or other external sources.

> Would this be better as MRI's implementation of String#freeze ?

This was discussed in #8992 and I think there was some resistance to #freeze returning a new object.

#4 - 12/09/2013 06:49 AM - tmm1 (Aman Gupta)

- Status changed from Open to Closed

This is a dupe of #8977. The proposal there is to use String#frozen, which I like better as well.

#5 - 12/09/2013 06:54 AM - phluid61 (Matthew Kerwin)

tmm1 (Aman Gupta) wrote:

> > Would this be better as MRI's implementation of String#freeze ?
>
> This was discussed in #8992 and I think there was some resistance to #freeze returning a new object.

We could always introduce #freeze! which is guaranteed to return the same object. Any old code that relies on receiving the same object reference would have to be updated, but I'm okay with that (not that it's up to me.)