Ruby master - Feature #11782

String#+@ and String#-@

12/07/2015 10:13 AM - ko1 (Koichi Sasada)

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

Description

Matz said

> In fact, my best choice is introducing String#+ that returns a mutable copy of a string.
> [ruby-core:71879] [Ruby trunk - Bug #11759]

So that this is a ticket for that.
I'll commit it ASAP to check this methods before 2.3.

Specification:

- +'foo' returns modifiable string.
- -'foo' returns frozen string (because wasters will freeze below 0 degree in Celsius).

Associated revisions

Revision 6d8bf54c - 12/07/2015 03:10 PM - ko1 (Koichi Sasada)

- string.c: introduce String#+@ and String#-@ to control String mutability. [Feature #11782]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@52917 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 52917 - 12/07/2015 03:10 PM - ko1 (Koichi Sasada)

- string.c: introduce String#+@ and String#-@ to control String mutability. [Feature #11782]

Revision 52917 - 12/07/2015 03:10 PM - ko1 (Koichi Sasada)

- string.c: introduce String#+@ and String#-@ to control String mutability. [Feature #11782]

Revision 52917 - 12/07/2015 03:10 PM - ko1 (Koichi Sasada)

- string.c: introduce String#+@ and String#-@ to control String mutability. [Feature #11782]

Revision 52917 - 12/07/2015 03:10 PM - ko1 (Koichi Sasada)

- string.c: introduce String#+@ and String#-@ to control String mutability. [Feature #11782]

Revision 52917 - 12/07/2015 03:10 PM - ko1 (Koichi Sasada)

- string.c: introduce String#+@ and String#-@ to control String mutability. [Feature #11782]

History

#1 - 12/07/2015 12:58 PM - matz (Yukihiro Matsumoto)

It's OK if we don't have strong objection.

Matz.

#2 - 12/07/2015 02:05 PM - mame (Yusuke Endoh)

+1

--
Yusuke Endoh mame@ruby-lang.org

#3 - 12/07/2015 03:10 PM - ko1 (Koichi Sasada)

- Status changed from Open to Closed

Applied in changeset r52917.

- string.c: introduce String#+@ and String#-@ to control String mutability. [Feature #11782]

#4 - 12/07/2015 03:14 PM - ko1 (Koichi Sasada)

- Status changed from Closed to Assigned

Endo-san pointed out that +foo.upcase! is evaluated as +(foo.upcase!).
It can be problem. But any practical examples?

Mutate operations such as str = +'foo' << bar << baz works (because +@ is stronger than <<).

Evaluation with current implementation (no optimization)

```
require 'benchmark'

N = 10_000_000
Benchmark.bm{|x|
  x.report{
    N.times{ 'foo'.dup }
  }
  x.report{
    N.times{ +'foo' }
  }
  x.report{
    N.times{ 'foo'.freeze }
  }
  x.report{
    N.times{ -'foo' }
  }
}

__END__
# frozen-string-literal: true
       user     system      total        real
   5.850000   0.010000   5.860000 (  5.858450)
   0.780000   0.000000   0.780000 (  0.780911)
   0.340000   0.000000   0.340000 (  0.351072)
   0.530000   0.000000   0.530000 (  0.529234)

# frozen-string-literal: false
       user     system      total        real
   2.540000   0.000000   2.540000 (  2.533344)
   0.750000   0.000000   0.750000 (  0.752019)
   0.350000   0.000000   0.350000 (  0.349776)
   1.180000   0.000000   1.180000 (  1.180882)
```

#5 - 12/07/2015 03:19 PM - ko1 (Koichi Sasada)

The purpose of this proposal is to encourage frozen string literals and provide better way than "foo".dup. So that String#-@ can be redundancy.

#6 - 12/08/2015 01:05 AM - ko1 (Koichi Sasada)

Quoted from DevelopersMeeting20151021Japan log:
https://bugs.ruby-lang.org/projects/ruby/wiki/DevelopersMeeting20151021Japan
https://docs.google.com/document/d/1axnQv1A2SdRExw--_RzXXJAPrRyvN7MCIB0WrKcZaSE/pub

> matz: i have another idea +"..." -> mutable, -"..." -> immutable. it is more clean than "...".freeze. how about that, instead of the magic comment? i don't care about pull-requests, but care about the representation of source code.
> akr: …
> matz: I don't like magic comment, so that I hesitate to introduce it.
> a_matsuda: + and - seems string operation. << -"foo" can be seen as a here document.
> matz: I understand. + and - is not good idea.
> matz: another idea: make ‘’’foo’’’ frozen. incompatibility is negligible.
> matz: yet another idea: make single quoted strings frozen. This is incompatible.

Matz said "+ and - is not good idea.".
So that please check it.

#7 - 12/08/2015 05:38 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

how about splitting strings like this?

```
long_string = 'first part ' +
  'second part ' +
  'last part'
```

Should long_string be optimized by the compiler to generate 'first part second part last part' as a frozen string, or should it actually perform the additions and generate a modified string?

#8 - 12/28/2015 04:36 PM - bughit (bug hit)

What is primary use case for -'string'? I initially though it was to be a shorthand for 'string'.freeze, avoiding allocation (due to special compiler support), producing an interned frozen string. But it does a dup, so does -'string' lack compiler support, thus first allocating a normal string then duping and freezing a copy which is not interned? So instead of 0 allocations ('string'.freeze) you get 2 (-'string')?