Ruby trunk - Feature #8579

Frozen string syntax

06/29/2013 09:52 PM - charliesome (Charlie Somerville)

Status:	Closed
Priority:	Normal
Assignee:	charliesome (Charlie Somerville)
Target version:	2.1.0

Description

I'd like to propose a new type of string literal - %f().

Because Ruby strings are mutable, every time a string literal is evaluated a new String object must be duped.

It's quite common to see code that stores a frozen String object into a constant which is then reused for performance reasons. Example: https://github.com/rack/rack/blob/master/lib/rack/methodoverride.rb

A new %f() string literal would instead evaluate to the same frozen String object every time. The benefit of this syntax is that it removes the need to pull string literals away from where they are used.

Here's an example of the proposed %f() syntax in action:

```
def foo
  ["bar".object_id, %f(bar).object_id]
end

p foo # might print "[123, 456]"

p foo # might print "[789, 456]"
```

These string literals could also be stored into a global refcounted table for deduplication across the entire program, futher reducing memory usage.

If this proposal is accepted, I can handle implementation work.

Related issues:

Related to Ruby trunk - Feature #8923: Frozen nil/true/false	**Closed**	**09/19/2013**
Related to Ruby trunk - Feature #8906: Freeze Symbols	**Closed**	**09/14/2013**
Related to Ruby trunk - Feature #8909: Expand "f" frozen suffix to literal ar...	**Rejected**	**09/14/2013**
Related to Ruby trunk - Feature #8992: Use String#freeze and compiler tricks ...	**Closed**	**10/08/2013**
Related to Ruby trunk - Feature #8976: file-scope freeze_string directive	**Closed**	

Associated revisions

Revision a056098c - 09/02/2013 07:11 AM - charliesome (Charlie Somerville)

- NEWS: Add note about frozen string literals
- compile.c (case_when_optimizable_literal): optimize NODE_LIT strings in when clauses of case statements
- ext/ripper/eventids2.c: add tSTRING_SUFFIX
- parse.y: add 'f' suffix on string literals for frozen strings
- test/ripper/test_scanner_events.rb: add scanner tests
- test/ruby/test_string.rb: add frozen string tests

[Feature #8579] [ruby-core:55699]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@42773 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 42773 - 09/02/2013 07:11 AM - charliesome (Charlie Somerville)

- NEWS: Add note about frozen string literals
- compile.c (case_when_optimizable_literal): optimize NODE_LIT strings in when clauses of case statements
- ext/ripper/eventids2.c: add tSTRING_SUFFIX
- parse.y: add 'f' suffix on string literals for frozen strings
- test/ripper/test_scanner_events.rb: add scanner tests
- test/ruby/test_string.rb: add frozen string tests

[Feature #8579] [ruby-core:55699]

Revision 42773 - 09/02/2013 07:11 AM - charliesome (Charlie Somerville)

- NEWS: Add note about frozen string literals
- compile.c (case_when_optimizable_literal): optimize NODE_LIT strings in when clauses of case statements
- ext/ripper/eventids2.c: add tSTRING_SUFFIX
- parse.y: add 'f' suffix on string literals for frozen strings
- test/ripper/test_scanner_events.rb: add scanner tests
- test/ruby/test_string.rb: add frozen string tests

[Feature #8579] [ruby-core:55699]

Revision 42773 - 09/02/2013 07:11 AM - charliesome (Charlie Somerville)

- NEWS: Add note about frozen string literals
- compile.c (case_when_optimizable_literal): optimize NODE_LIT strings in when clauses of case statements
- ext/ripper/eventids2.c: add tSTRING_SUFFIX
- parse.y: add 'f' suffix on string literals for frozen strings
- test/ripper/test_scanner_events.rb: add scanner tests
- test/ruby/test_string.rb: add frozen string tests

[Feature #8579] [ruby-core:55699]

Revision 42773 - 09/02/2013 07:11 AM - charliesome (Charlie Somerville)

- NEWS: Add note about frozen string literals
- compile.c (case_when_optimizable_literal): optimize NODE_LIT strings in when clauses of case statements
- ext/ripper/eventids2.c: add tSTRING_SUFFIX
- parse.y: add 'f' suffix on string literals for frozen strings

- test/ripper/test_scanner_events.rb: add scanner tests
- test/ruby/test_string.rb: add frozen string tests

[Feature #8579] [ruby-core:55699]

Revision 42773 - 09/02/2013 07:11 AM - charliesome (Charlie Somerville)

- NEWS: Add note about frozen string literals
- compile.c (case_when_optimizable_literal): optimize NODE_LIT strings in when clauses of case statements
- ext/ripper/eventids2.c: add tSTRING_SUFFIX
- parse.y: add 'f' suffix on string literals for frozen strings
- test/ripper/test_scanner_events.rb: add scanner tests
- test/ruby/test_string.rb: add frozen string tests

[Feature #8579] [ruby-core:55699]

Revision 42773 - 09/02/2013 07:11 AM - charliesome (Charlie Somerville)

- NEWS: Add note about frozen string literals
- compile.c (case_when_optimizable_literal): optimize NODE_LIT strings in when clauses of case statements
- ext/ripper/eventids2.c: add tSTRING_SUFFIX
- parse.y: add 'f' suffix on string literals for frozen strings
- test/ripper/test_scanner_events.rb: add scanner tests
- test/ruby/test_string.rb: add frozen string tests

[Feature #8579] [ruby-core:55699]

History

#1 - 06/29/2013 11:23 PM - judofyr (Magnus Holm)

+1.

What about interpolation? %F() would be useful although they can't be deduplicated.

// Magnus Holm

#2 - 06/30/2013 10:56 AM - mame (Yusuke Endoh)

I'm neutral for the proposal itself.
Instead of a new kind of %-notation, it would be better to introduce a modifier like regexp literal:

%r(foo)o
%q(foo)o

--
Yusuke Endoh mame@tsg.ne.jp

#3 - 07/01/2013 06:14 AM - Anonymous

+1 to mame's proposal, literals already take long to learn for us average Joe users, only operator precedence is worse.

#4 - 07/01/2013 04:49 PM - charliesome (Charlie Somerville)

mame, is there a precedent of using modifiers on non-regexp literals?

I'm not against your proposal, but it would be odd for this particular feature to introduce a new syntax.

Also, if we used a modifier, how would that affect other types of percent literals like %w or %i?

#5 - 07/01/2013 05:00 PM - nobu (Nobuyoshi Nakada)

charliesome (Charlie Somerville) wrote:

> Also, if we used a modifier, how would that affect other types of percent literals like %w or %i?

As for %i, it doesn't make sense to freeze symbols, so it would freeze result array if it were introduced.
And all modifiers don't have to be applied to all kinds of literals.

#6 - 08/20/2013 08:57 AM - charliesome (Charlie Somerville)

- File FrozenStringSyntax.pdf added

- Assignee changed from matz (Yukihiro Matsumoto) to ko1 (Koichi Sasada)

ko1 - please see attached a slide for the upcoming developer meeting in Japan.

#7 - 08/20/2013 10:23 AM - ko1 (Koichi Sasada)

(2013/08/20 8:57), charliesome (Charlie Somerville) wrote:

> ko1 - please see attached a slide for the upcoming developer meeting in Japan.

My position is negative to introduce new %f() syntax.

I like suffix that mame proposed [ruby-core:55705].

There are two dimmention:
(1) frozen.
(2) once.

mame-san proposed (2) feature. And `once' feature doesn't return frozen,
but return same object.

Try with once suffix with regexp.

```
##
3.times{|i|
r = /foo/o
p [r.object_id, r.frozen?]
r.instance_variable_set(:@foo, i)
p r.instance_variable_get(:@foo)
}
#=>
[23212812, false]
0
[23212812, false]
1
[23212812, false]
2
##
```

I believe you (charliesome) *don't* care about syntax, but you want a
feature to introduce frozen, or same string literal. right?

My idea is to introduce frozen suffix for (1) and once suffix for (2).

```
frozen_str = "foo"f
once_str = "foo"o
3.times{|i|
once_dynamic_str = "#{i}"o #=> every time it returns "0".
}
frozen_and_once_str = "foo"of
```

I think %f() is not good syntax.

... "f" and "o" suffix are also bad? :P

--
// SASADA Koichi at atdot dot net

#8 - 08/20/2013 10:23 AM - ko1 (Koichi Sasada)

(2013/08/20 10:01), SASADA Koichi wrote:

> My idea is to introduce frozen suffix for (1) and once suffix for (2).

Because we already introduced "i" (imaginary number literal) and "r"
(rational number) suffixes.

overuse? :)

--
// SASADA Koichi at atdot dot net

#9 - 08/20/2013 10:47 AM - charliesome (Charlie Somerville)

> I believe you (charliesome) *don't* care about syntax, but you want a
> feature to introduce frozen, or same string literal. right?

Correct. I have a preference toward %f (for consistency with other string types), but I am happy as long as the same string literal feature is accepted.

> My idea is to introduce frozen suffix for (1) and once suffix for (2).

I'm ok with frozen suffix, but I'm not so sure about once suffix. I believe matz is negative towards once:

```
17:10 matz: I don't recommend /re/o behavior
17:11 matz: because it requires more complex cache mechanism
```

Anyway, I will revise my slide and post it here later.

#10 - 08/20/2013 11:29 AM - charliesome (Charlie Somerville)

- File FrozenStringSyntax_2.pdf added

Updated slide with f suffix syntax

#11 - 08/20/2013 06:20 PM - alexeymuranov (Alexey Muranov)

=begin
Just two put in my 2 cents, i have suggested in a comment to #7791 to have an intermediate class between String and Symbol, but maybe frozen strings are better. What would you say about a new symbol-like literal syntax, like, for example,

|'this is a frozen string'

?
Since frozen strings are likely to be used as hash keys, in simple cases the quotes may be dropped:

|this_is_a_frozen_string

Just a thought.
=end

#12 - 08/24/2013 11:46 PM - charliesome (Charlie Somerville)

I have found a problem with using f-suffix syntax.

What should happen in this case?

```
"hello "f "world"
```

Note that this is not a problem when using %f syntax as it is a syntax error to have adjacent percent-strings like this:

```
%f(hello ) %f(world)
```

#13 - 08/25/2013 04:55 AM - kstephens (Kurt Stephens)

How about something more generic? A prefix operator that memoizes and freezes any expression result in a thread-safe manner on first eval:

```
%f'a frozen string'                 #
%f"a frozen #{interpolated} string" #
%f{a: 'frozen', hash: 'value'}      # A frozen, inline memoized Hash.
%f[:a, 'frozen', :array, 'value']   # A frozen, inline, memoized Array.
%f(some(:method, 'call'))           # You get the idea.
```

#14 - 08/26/2013 01:23 PM - funny_falcon (Yura Sokolov)

24.08.2013 23:55 пользователь "kstephens (Kurt Stephens)" <
redmine@ruby-lang.org> написал:

> Issue #8579 has been updated by kstephens (Kurt Stephens).
>
> How about something more generic? A prefix operator that memoizes and
> freezes any expression result in a thread-safe manner on first eval:
>
> ```
> %f'a frozen string'                 #
> %f"a frozen #{interpolated} string" #
> %f{a: 'frozen', :hash 'value'}      # A frozen, inline memoized Hash.
> %f[:a, 'frozen', :array, 'value']   # A frozen, inline, memoized
> ```
>
> Array.
> %f(some(:method, 'call')) # You get the idea.

Looks pretty! +1

> Feature #8579: Frozen string syntax
> https://bugs.ruby-lang.org/issues/8579#change-41341
>
> Author: charliesome (Charlie Somerville)
> Status: Open
> Priority: Normal
> Assignee: ko1 (Koichi Sasada)
> Category: syntax
> Target version: current: 2.1.0
>
> I'd like to propose a new type of string literal - %f().
>
> Because Ruby strings are mutable, every time a string literal is
> evaluated a new String object must be duped.
>
> It's quite common to see code that stores a frozen String object into a
> constant which is then reused for performance reasons. Example:
> https://github.com/rack/rack/blob/master/lib/rack/methodoverride.rb
>
> A new %f() string literal would instead evaluate to the same frozen
> String object every time. The benefit of this syntax is that it removes the
> need to pull string literals away from where they are used.
>
> Here's an example of the proposed %f() syntax in action:
>
> ```
> def foo
>   ["bar".object_id, %f(bar).object_id]
> end
>
> p foo # might print "[123, 456]"
>
> p foo # might print "[789, 456]"
> ```
>
> These string literals could also be stored into a global refcounted table
> for deduplication across the entire program, futher reducing memory usage.
>
> If this proposal is accepted, I can handle implementation work.
>
> --
> http://bugs.ruby-lang.org/

#15 - 08/31/2013 03:21 PM - matz (Yukihiro Matsumoto)

I accept the suffix idea ("frozen string"f), and string concatenation for frozen strings will be invalid ("foo"f "bar" would be syntax error).

Matz.

#16 - 08/31/2013 04:20 PM - charliesome (Charlie Somerville)

- Assignee changed from ko1 (Koichi Sasada) to charliesome (Charlie Somerville)

#17 - 09/02/2013 04:11 PM - charliesome (Charlie Somerville)

- Status changed from Open to Closed

- % Done changed from 0 to 100

This issue was solved with changeset r42773.
Charlie, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- NEWS: Add note about frozen string literals
- compile.c (case_when_optimizable_literal): optimize NODE_LIT strings
  in when clauses of case statements
- ext/ripper/eventids2.c: add tSTRING_SUFFIX
- parse.y: add 'f' suffix on string literals for frozen strings
- test/ripper/test_scanner_events.rb: add scanner tests
- test/ruby/test_string.rb: add frozen string tests

[Feature #8579] [ruby-core:55699]

#18 - 09/02/2013 05:53 PM - ko1 (Koichi Sasada)

(2013/08/31 15:21), matz (Yukihiro Matsumoto) wrote:

> I accept the suffix idea ("frozen string"f), and string concatenation for frozen strings will be invalid ("foo"f "bar" would be syntax error).

There are comments on this change. For example, it is confusing about
the character 'f'. For example, it remind "Float".

e.g.: "1.0"f is not 1.0 (Flaot).

--
// SASADA Koichi at atdot dot net

#19 - 10/08/2013 02:53 AM - ko1 (Koichi Sasada)

(2013/08/31 16:20), charliesome (Charlie Somerville) wrote:

> Feature #8579: Frozen string syntax

Just another syntax idea:

FROZEN{ 'foo' }

Advantage:

- Can implement on Ruby level in 2.0 or before
- Can extend for other literals such as Array
- Similar to BEGIN{ ... } / END{ ... }

Disadvantage:

- Long
- Not in Ruby style?
- A few compatibility issue for programs using FROZEN() method

--
// SASADA Koichi at atdot dot net

#20 - 10/08/2013 06:28 AM - headius (Charles Nutter)

See also http://bugs.ruby-lang.org/issues/8992 which proposes just making "literal string".freeze do the right thing in the compiler.

FROZEN { } is not terrible syntax, but it's the longest one suggested.

Files

FrozenStringSyntax.pdf	41 KB	08/20/2013	charliesome (Charlie Somerville)
FrozenStringSyntax_2.pdf	33.7 KB	08/20/2013	charliesome (Charlie Somerville)