Ruby - Feature #13381

[PATCH] Expose rb_fstring and its family to C extensions

03/29/2017 05:29 AM - eagletmt (Kohei Suzuki)

Status:	Closed	
Priority:	Normal	
Assignee:		
Target version:		

Description

https://github.com/ruby/ruby/pull/1559

Currently, C extensions cannot use fstrings. I'd like to use rb_fstring_cstr instead of rb_str_new_cstr for static strings in C extensions to avoid excess allocation.

I think there's several use cases.

- https://github.com/k0kubun/hamlit/blob/v2.8.0/ext/hamlit/hamlit.c#L508-L512
- https://bitbucket.org/ged/ruby-pg/src/e5eb92cca97abc0c6fc168acfad993c2ad314589/ext/pg_connection.c?at=v0.20.0&fileviewer=file-view-default#pg_connection.c-3679
- https://bitbucket.org/ged/ruby-pg/src/e5eb92cca97abc0c6fc168acfad993c2ad314589/ext/pg_copy_coder.c?at=v0.20.0&fileviewer=file-view-default#pg_copy_coder.c-38

Related issues:

Related to Ruby - Feature #17147: New method to get frozen strings from Strin...	**Feedback**
Has duplicate Ruby - Feature #16029: Expose fstring related APIs to C-extensions	**Closed**

Associated revisions

Revision ef19fb111a8c8bf1a71d46e6fcf34b227e086845 - 11/17/2020 12:39 AM - byroot (Jean Boussier)

Expose the rb_interned_str_* family of functions

Fixes [Feature #13381]

Revision ef19fb111a8c8bf1a71d46e6fcf34b227e086845 - 11/17/2020 12:39 AM - byroot (Jean Boussier)

Expose the rb_interned_str_* family of functions

Fixes [Feature #13381]

Revision ef19fb11 - 11/17/2020 12:39 AM - byroot (Jean Boussier)

Expose the rb_interned_str_* family of functions

Fixes [Feature #13381]

Revision 6bef49427ab2a9d3bc338f1cffcd086153a59f44 - 11/30/2020 08:33 AM - byroot (Jean Boussier)

Fix rb_interned_str_* functions to not assume static strings

Fixes [Feature #13381]

When passed a fake_str, register_fstring would create new strings with str_new_static. That's not what was expected, and answer almost no use cases.

Revision 6bef49427ab2a9d3bc338f1cffcd086153a59f44 - 11/30/2020 08:33 AM - byroot (Jean Boussier)

Fix rb_interned_str_* functions to not assume static strings

Fixes [Feature #13381]

When passed a fake_str, register_fstring would create new strings with str_new_static. That's not what was expected, and answer almost no use cases.

Revision 6bef4942 - 11/30/2020 08:33 AM - byroot (Jean Boussier)

Fix rb_interned_str_* functions to not assume static strings

Fixes [Feature #13381]

When passed a fake_str, register_fstring would create new strings
with str_new_static. That's not what was expected, and answer
almost no use cases.

History

#1 - 03/29/2017 05:36 AM - ko1 (Koichi Sasada)

- Status changed from Open to Feedback

I can understand use cases but we shouldn't expose the name "fstring".

#2 - 03/29/2017 06:32 AM - eagletmt (Kohei Suzuki)

OK, I've read comments of #13077.

What do you think of renaming fstring to "deduped" string? "Deduped" strings are implicitly frozen.

- Rename rb_fstring to rb_str_deduped
- Rename rb_fstring_new to rb_str_deduped_new
- Rename rb_fstring_cstr to rb_str_deduped_cstr
- Rename rb_fstring_enc_new to rb_enc_str_deduped_new
- Rename rb_fstring_enc_cstr to rb_enc_str_deduped_cstr
  - I think enc should come first for consistency

#3 - 06/16/2017 07:58 AM - ko1 (Koichi Sasada)

- Status changed from Feedback to Assigned

- Assignee set to ko1 (Koichi Sasada)

#4 - 06/16/2017 08:00 AM - nobu (Nobuyoshi Nakada)

- Assignee deleted (ko1 (Koichi Sasada))

How about fixed_str?

#5 - 12/31/2019 09:56 AM - k0kubun (Takashi Kokubun)

- Has duplicate Feature #16029: Expose fstring related APIs to C-extensions added

#6 - 07/17/2020 04:19 AM - ko1 (Koichi Sasada)

Nobu and I discussed the name, we want to choose rb_str_pool terminology.

- rb_str_pool prefix
  - rb_str_pool(VALUE)
  - rb_str_pool_buff(const char *ptr, long len)
  - rb_str_pool_cstr(const char *ptr)
  - rb_enc_str_pool_buff(const char *ptr, long len, rb_encoding *enc)
  - rb_enc_str_pool_cstr(const char *ptr, rb_encoding *enc)

- FYI: rb_str_new series:
  - rb_str_new(const char *ptr, long len)
  - rb_str_new_cstr(const char *ptr)
  - rb_enc_str_new(const char *ptr, long len, rb_encoding *enc)
  - rb_enc_str_new_cstr(const char *ptr, rb_encoding *enc)

If there is no comments, we will merge it soon.

#7 - 07/17/2020 09:22 AM - Eregon (Benoit Daloze)

I'd suggest buff => buffer, I don't think it's worth abbreviating that.
BTW, there is rb_alloc_tmp_buffer so buff would be inconsistent.

Is rb_str_pool(VALUE) useful?

I think it could be rb_str_uminus() (or just rb_funcall(str, rb_intern("-@"))).
And then we could be consistent with rb_str_new*, and drop confusing _buff suffixes (not really buffer (=> sounds mutable), just a C string with known length).

So I think this is better:

```
rb_str_uminus(VALUE) // or just rb_funcall(str, rb_intern("-@"))
rb_str_pool(const char *ptr, long len)
rb_str_pool_cstr(const char *ptr)
rb_enc_str_pool(const char *ptr, long len, rb_encoding *enc)
rb_enc_str_pool_cstr(const char *ptr, rb_encoding *enc)
```

#8 - 07/20/2020 07:25 AM - nobu (Nobuyoshi Nakada)

@ko1 (Koichi Sasada) suggests rb_str_pool_value(VALUE) for the first type.

As these may not create a new object, I thought of using pool as a verb instead of new however, it doesn't feel fit much.
Does anyone have other ideas?

#9 - 07/20/2020 07:51 AM - byroot (Jean Boussier)

> Does anyone have other ideas?

A common term for this in intern / interning / interned. However that terminology is already used for symbols (rb_str_intern).

This would look like:

```
rb_interned_str(VALUE)
rb_interned_str(const char *ptr, long len)
rb_interned_str_cstr(const char *ptr)
rb_enc_interned_str(const char *ptr, long len, rb_encoding *enc)
rb_enc_interned_str_cstr(const char *ptr, rb_encoding *enc)
```

#10 - 08/12/2020 06:36 AM - ko1 (Koichi Sasada)

> However that terminology is already used for symbols (rb_str_intern).

Yes. This is why we didn't propose it.

#11 - 09/04/2020 10:00 AM - naruse (Yui NARUSE)

- Related to Feature #17147: New method to get frozen strings from String objects added

#12 - 09/23/2020 06:07 AM - ko1 (Koichi Sasada)

I'm now positive "interned_str" instead of "interned".

#13 - 09/23/2020 07:23 AM - byroot (Jean Boussier)

Would you like a patch?

#14 - 09/25/2020 01:01 AM - Dan0042 (Daniel DeLorme)

Any of the earlier suggestions were good, such as "pool" or "deduped". But while "interned" is technically correct, it will lead to confusion with symbols. It's better to keep a separate terminology imho.

#15 - 11/17/2020 12:39 AM - byroot (Jean Boussier)

- Status changed from Assigned to Closed

Applied in changeset git|ef19fb111a8c8bf1a71d46e6fcf34b227e086845.

Expose the rb_interned_str_* family of functions

Fixes [Feature #13381]

#16 - 11/18/2020 03:34 PM - byroot (Jean Boussier)

After trying to use the new functions in json and messagepack I realized we overlooked something entirely.

The internal rb_fstring_* will build new strings with str_new_static, so will directly re-use the string pointer that was provided. That's pretty much unusable, and unsafe.

I submitted another pull request to fix this: https://github.com/ruby/ruby/pull/3786

#17 - 11/26/2020 07:41 AM - nobu (Nobuyoshi Nakada)

It sounds like that fstring doesn't match that purpose.
Is it really better to divert fstring than separated string pools?

#18 - 11/26/2020 05:44 PM - byroot (Jean Boussier)

> It sounds like that fstring doesn't match that purpose.

I'm not sure why it wouldn't. Ultimately the prupose is the same than String#-@, but from the C API and by passing a char *.

> Is it really better to divert fstring than separated string pools?

It would be way less efficient, consider the following case:

```
objects = JSON.load_file('path/to.json') # [{"field": 1}, {"field": 2}, ...]
objects.map { |o| o['field'] }
```

Here some_field since it is a literal is part of the fstring table. As of Ruby 2.7 the json extension has to rb_str_new() many times, before calling hash_aset which will then deduplicate these strings.

In some use cases like ours, this generates a huge amount of GC pressure that could be avoided if the json extension (and some others) could directly lookup interned strings from a char pointer.

#19 - 11/26/2020 06:42 PM - byroot (Jean Boussier)

@alanwu (Alan Wu) just pointed to me that the confusion might come from the fact that this ticket need isn't quite the same than the one I originally opened: https://bugs.ruby-lang.org/issues/16029

They are close but not quite duplicates. @eagletmt wanted a fast API to convert static C strings into Ruby strings, I wanted an API for basically rb_str_new_frozen() that would deduplicate at the same time without having to allocate.

#20 - 11/26/2020 07:02 PM - byroot (Jean Boussier)

Also if that helps, here's how it would be used by json: https://github.com/flori/json/pull/451