Ruby master - Feature #4447

add String#byteslice() method

02/25/2011 05:52 PM - sunaku (Suraj Kurapati)

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

Description

=begin
Please add a String#byteslice() method to the Ruby 1.9 core API.

Without that method, I am forced to *inefficiently* perform byte-based string slicing by (1) unpacking the entire string into an Array (with String#unpack or worse: my_string.bytes.to_a) then (2) slicing that Array and finally (3) joining the sliced Array into a string (with Array#pack or worse: my_array.map(&:chr).join), all as shown below:

class String
unless method_defined? :byteslice
##
Does the same thing as String#slice but
operates on bytes instead of characters.
#
def byteslice(*args)*
unpack('C').slice(*args*).pack('C*')
end
end
end

Thanks for your consideration.
=end

Associated revisions

Revision d301b4d8 - 03/01/2011 04:28 AM - naruse (Yui NARUSE)

Fix rdoc of String#byteslice. Feature #4447

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@30992 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 30992 - 03/01/2011 04:28 AM - naruse (Yui NARUSE)

Fix rdoc of String#byteslice. Feature #4447

Revision 30992 - 03/01/2011 04:28 AM - naruse (Yui NARUSE)

Fix rdoc of String#byteslice. Feature #4447

Revision 30992 - 03/01/2011 04:28 AM - naruse (Yui NARUSE)

Fix rdoc of String#byteslice. Feature #4447

Revision 30992 - 03/01/2011 04:28 AM - naruse (Yui NARUSE)

Fix rdoc of String#byteslice. Feature #4447

Revision 30992 - 03/01/2011 04:28 AM - naruse (Yui NARUSE)

Fix rdoc of String#byteslice. Feature #4447

Revision 30992 - 03/01/2011 04:28 AM - naruse (Yui NARUSE)

Fix rdoc of String#byteslice. Feature #4447

History

#1 - 02/25/2011 06:17 PM - radarek (Radosław Bułat)

=begin
Hm, can't you just force it to binary encoding?

```
s = "ąęć"
=> "ąęć"
s.slice(1,2)
=> "ęć"
s.force_encoding("binary").slice(1,2)
=> "\x85\xC4"
```

=end

#2 - 02/25/2011 07:23 PM - duerst (Martin Dürst)

=begin
string.force_encoding(ENCODING::BINARY).slice almost does what you want,
very efficiently. The problem is that the string will then be marked as
BINARY. This can be set back, but it may affect operations that run in
parallel. To expand on this, something like

class String
def temporarily_binary
encoding = self.encoding
self.force_encoding ENCODING::BINARY
yield
self.force_encoding encoding
end
end

would be more general. It would be used like so:

string.temporarily_binary { |s| s.slice(...) }

But this is still not thead-safe.

Regards, Martin.

On 2011/02/25 17:49, Suraj Kurapati wrote:

> Issue #4447 has been reported by Suraj Kurapati.
>
> ---
>
> Feature #4447: add String#byteslice() method
> http://redmine.ruby-lang.org/issues/4447
>
> Author: Suraj Kurapati
> Status: Open
> Priority: Normal
> Assignee:
> Category:
> Target version:
>
> Please add a String#byteslice() method to the Ruby 1.9 core API.
>
> Without that method, I am forced to *inefficiently* perform byte-based
> string slicing by (1) unpacking the entire string into an Array (with
> String#unpack or worse: my_string.bytes.to_a) then (2) slicing that
> Array and finally (3) joining the sliced Array into a string (with
> Array#pack or worse: my_array.map(&:chr).join), all as shown below:
>
> class String
> unless method_defined? :byteslice
> ##
> # Does the same thing as String#slice but
> # operates on bytes instead of characters.
> #
> def byteslice(*args)
> *unpack('C*').slice(*args).pack('C*')
> end
> end
> end

Thanks for your consideration.

--
#-# Martin J. Dürst, Professor, Aoyama Gakuin University
#-# http://www.sw.it.aoyama.ac.jp mailto:duerst@it.aoyama.ac.jp
=end

#3 - 02/26/2011 07:23 AM - RickDeNatale (Rick DeNatale)

=begin
On Fri, Feb 25, 2011 at 3:42 PM, Gary Wright gwtmp01@mac.com wrote:

> On Feb 25, 2011, at 4:52 AM, Martin J. DÃ¼rst wrote:
>
> > string.force_encoding(ENCODING::BINARY).slice almost does what you want, very efficiently. The problem is that the string will then be marked as BINARY. This can be set back, but it may affect operations that run in parallel. To expand on this, something like
>
> Isn't any manipulation of a string via two threads simultaneously already not thread safe?
>
> That is to say, toggling the encoding doesn't make the situation any worse than it is with String#slice all by itself.

No, the issue is that String#force_encoding mutates the string, but
String#slice does not.

str
=end

#4 - 02/28/2011 03:42 PM - naruse (Yui NARUSE)

- Category set to M17N

- Status changed from Open to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

=begin
This request sounds reasonable.
A patch is following:

```
diff --git a/string.c b/string.c
index 23784ab..cea9028 100644
--- a/string.c
+++ b/string.c
@@ -3987,6 +3987,95 @@ rb_str_setbyte(VALUE str, VALUE index, VALUE value)
return value;
}

+static VALUE
+str_byte_substr(VALUE str, long beg, long len)
+{
    char *p, *s = RSTRING_PTR(str), *e = s + RSTRING_LEN(str);
    VALUE str2;
    if (beg > RSTRING_LEN(str)) return Qnil;
    if (beg < 0) {
    beg += RSTRING_LEN(str);
    if (beg < 0) return Qnil;
    }
    if (beg + len > RSTRING_LEN(str))
    len = RSTRING_LEN(str) - beg;
    if (len <= 0) {
    len = 0;
    p = 0;
    }
    else
    p = s + beg; +
    if (len > RSTRING_EMBED_LEN_MAX && beg + len == RSTRING_LEN(str)) {
    str2 = rb_str_new4(str);
    str2 = str_new3(rb_obj_class(str2), str2);
    RSTRING(str2)->as.heap.ptr += RSTRING(str2)->as.heap.len - len;
    RSTRING(str2)->as.heap.len = len;
    }
    else {
```

- str2 = rb_str_new5(str, p, len);
- OBJ_INFECT(str2, str);
- } +
- return str2; +} + +static VALUE +str_byte_aref(VALUE str, VALUE indx) +{
- long idx;
- switch (TYPE(indx)) {
- case T_FIXNUM:
- idx = FIX2LONG(indx); +
- num_index:
- str = str_byte_substr(str, idx, 1);
- if (!NIL_P(str) && RSTRING_LEN(str) == 0) return Qnil;
- return str; +
- default:
- /* check if indx is Range */
- {
- long beg, len = RSTRING_LEN(str);
- VALUE tmp; +
- switch (rb_range_beg_len(indx, &beg, &len, len, 0)) {
- case Qfalse:
- break;
- case Qnil:
- return Qnil;
- default:
- tmp = str_byte_substr(str, beg, len);
- return tmp;
- }
- }
- idx = NUM2LONG(indx);
- goto num_index;
- }
- return Qnil;   /* not reached */ +} + +
- * call-seq:
- *   str.byteslice() -> new_str
- *
- *   "hello".byteslice(1)   #=> "e"
- *   "hello".byteslice(1, 2) #=> "el"
- *   "\u3042".byteslice(1, 2) #=> "\x81\x82"
- */ + +static VALUE +rb_str_byteslice(int argc, VALUE *argv, VALUE str) +{
- if (argc == 2) {
- return str_byte_substr(str, NUM2LONG(argv[0]), NUM2LONG(argv[1]));
- }
- if (argc != 1) {
- rb_raise(rb_eArgError, "wrong number of arguments (%d for 1..2)", argc);
- }
- return str_byte_aref(str, argv[0]); +} + /*
- call-seq:
- str.reverse   -> new_str @@ -7649,6 +7738,7 @@ Init_String(void) rb_define_method(rb_cString, "chr", rb_str_chr, 0); rb_define_method(rb_cString, "getbyte", rb_str_getbyte, 1); rb_define_method(rb_cString, "setbyte", rb_str_setbyte, 2);
- rb_define_method(rb_cString, "byteslice", rb_str_byteslice, -1);

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 c5d3a53..f18c814 100644
--- a/test/ruby/test_string.rb
+++ b/test/ruby/test_string.rb
@@ -1944,4 +1944,33 @@ class TestString < Test::Unit::TestCase
assert_equal(S("hello world"), a)
assert_equal(S("hello "), b)
end
+

- def b(str)
- str.force_encoding(Encoding::ASCII_8BIT)
- end +
- def test_byteslice
- assert_equal(b("h"), "hello".byteslice(0))
- assert_equal(nil, "hello".byteslice(5))
- assert_equal(b("o"), "hello".byteslice(-1))
- assert_equal(nil, "hello".byteslice(-6)) +
- assert_equal(b(""), "hello".byteslice(0, 0))
- assert_equal(b("hello"), "hello".byteslice(0, 6))
- assert_equal(b("hello"), "hello".byteslice(0, 6))
- assert_equal(b(""), "hello".byteslice(5, 1))
- assert_equal(b("o"), "hello".byteslice(-1, 6))

- assert_equal(nil, "hello".byteslice(-6, 1)) +
- assert_equal(b("h"), "hello".byteslice(0..0))
- assert_equal(b(""), "hello".byteslice(5..0))
- assert_equal(b("o"), "hello".byteslice(4..5))
- assert_equal(nil, "hello".byteslice(6..0))
- assert_equal(b(""), "hello".byteslice(-1..0))
- assert_equal(b("llo"), "hello".byteslice(-3..5)) +
- assert_equal(b("\x81"), "\u3042".byteslice(1))
- assert_equal(b("\x81\x82"), "\u3042".byteslice(1, 2))
- assert_equal(b("\x81\x82"), "\u3042".byteslice(1..2))
- end end =end

#5 - 03/01/2011 12:23 AM - matz (Yukihiro Matsumoto)

=begin
Hi,

OK, please commit it to the trunk.

```
                          matz.
```

In message "Re: [ruby-core:35392] [Ruby 1.9 - Feature #4447] [Assigned] add String#byteslice() method"
on Mon, 28 Feb 2011 15:42:55 +0900, Yui NARUSE redmine@ruby-lang.org writes:

|Issue #4447 has been updated by Yui NARUSE.
|
|Category set to M17N
|Status changed from Open to Assigned
|Assignee set to Yukihiro Matsumoto
|
|This request sounds reasonable.
|A patch is following:
|
| diff --git a/string.c b/string.c
| index 23784ab..cea9028 100644
=end

#6 - 03/01/2011 03:24 AM - headius (Charles Nutter)

=begin
JRuby bug added: http://jira.codehaus.org/browse/JRUBY-5547
=end

#7 - 04/20/2011 10:02 PM - sunaku (Suraj Kurapati)

=begin
Was NARUSE's patch committed to trunk? Can we close this issue? Thanks.
=end

#8 - 04/20/2011 10:27 PM - naruse (Yui NARUSE)

- Status changed from Assigned to Closed

=begin

=end