Ruby - Feature #13780

String#each_grapheme

08/03/2017 06:35 PM - rbjl (Jan Lelis)

Status:	Closed	
Priority:	Normal	
Assignee:	naruse (Yui NARUSE)	
Target version:	2.5	

Description

Ruby's regex engine has support for graphemes via \X:

https://github.com/k-takata/Onigmo/blob/791140951eefcf17db4e762e789eb046ea8a114c/doc/RE#L117-L124

This is really useful when working with Unicode strings. However, code like string.scan(/\X/) is not so readable enough, which might lead people to use String#each_char, when they really should split by graphemes.

What I propose is two new methods:

- String#each_grapheme which returns an Enumerator of graphemes (in the same way like \X)

and

- String#graphemes which returns an Array of graphemes (in the same way like \X)

What do you think?

Resources

- Unicode® Standard Annex #29: Unicode Text Segmentation: http://unicode.org/reports/tr29/
- Related issue: https://bugs.ruby-lang.org/issues/12831

Related issues:

Related to Ruby - Feature #18563: Add "graphemes" and "each_grapheme" aliases	**Closed**

Associated revisions

Revision df49fc659e70a728e000f5a9756e3f483b00acd1 - 08/31/2017 06:35 AM - naruse (Yui NARUSE)

String#each_grapheme_cluster and String#grapheme_clusters

added to enumerate grapheme clusters [Feature #13780]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@59698 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision df49fc65 - 08/31/2017 06:35 AM - naruse (Yui NARUSE)

String#each_grapheme_cluster and String#grapheme_clusters

added to enumerate grapheme clusters [Feature #13780]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@59698 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision bb03f02805d1cd112c5fc7a3a8027ed2797c0074 - 08/31/2017 08:07 AM - nobu (Nobuyoshi Nakada)

string.c: adjust indent [ci skip]

- string.c (rb_str_enumerate_grapheme_clusters): adjust indent.
  [Feature #13780]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@59700 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision bb03f028 - 08/31/2017 08:07 AM - nobu (Nobuyoshi Nakada)

string.c: adjust indent [ci skip]

- string.c (rb_str_enumerate_grapheme_clusters): adjust indent.
  [Feature #13780]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@59700 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 71de56621ec4643bdfee102f56a153260bb97998 - 09/03/2017 01:47 AM - nobu (Nobuyoshi Nakada)

string.c: fix for non-Unicode encodings

- string.c (rb_str_enumerate_grapheme_clusters): should enumerate
  chars for non-Unicode encodings. [Feature #13780]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@59731 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 71de5662 - 09/03/2017 01:47 AM - nobu (Nobuyoshi Nakada)

string.c: fix for non-Unicode encodings

- string.c (rb_str_enumerate_grapheme_clusters): should enumerate
  chars for non-Unicode encodings. [Feature #13780]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@59731 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 08/03/2017 07:09 PM - shevegen (Robert A. Heiler)

My only concern is about the name "grapheme".

I don't know how it is for others but ... this is the first time that I even heard the
term.

#2 - 08/03/2017 09:11 PM - shan (Shannon Skipper)

shevegen (Robert A. Heiler) wrote:

> My only concern is about the name "grapheme".
>
> I don't know how it is for others but ... this is the first time that I even heard the
> term.

I think the term is correct and it complements #codepoints and #each_codepoint. In Elixir for example:

```
"□□□□□□□□□" |> String.codepoints #=> ["□□", "□□", "□□", "□□"]
"□□□□□□□□□" |> String.graphemes #=> ["□□□□", "□□□□"]
```

#3 - 08/04/2017 01:57 PM - naruse (Yui NARUSE)

- Status changed from Open to Assigned

- Assignee set to naruse (Yui NARUSE)

- Target version set to 2.5

Accepted.
I'll introduce this in Ruby 2.5.

#4 - 08/05/2017 10:50 AM - naruse (Yui NARUSE)

shan (Shannon Skipper) wrote:

> shevegen (Robert A. Heiler) wrote:
>
> > My only concern is about the name "grapheme".
> >
> > I don't know how it is for others but ... this is the first time that I even heard the
> > term.
>
> I think the term is correct and it complements #codepoints and #each_codepoint. In Elixir for example:

Elixir's grapheme and Swift's Character refer Unicode® Standard Annex #29's "Grapheme Cluster".
http://unicode.org/reports/tr29/
The document says grapheme clusters are “user-perceived characters”.

#5 - 08/12/2017 05:40 PM - naruse (Yui NARUSE)

```
diff --git a/NEWS b/NEWS
index 4bfca9240c..1e66e94879 100644
--- a/NEWS
+++ b/NEWS
@@ -94,6 +94,7 @@ with all sufficient information, see the ChangeLog file or Redmine
   * String#delete_prefix! is added to remove prefix destructively [Feature #12694]
   * String#delete_suffix is added to remove suffix [Feature #13665]
   * String#delete_suffix! is added to remove suffix destructively [Feature #13665]
+  * String#graphemes is added to enumerate grapheme clusters [Feature #13780]

 * Thread

diff --git a/string.c b/string.c
index daef497b3d..dd0daa27e9 100644
--- a/string.c
+++ b/string.c
@@ -8066,6 +8066,117 @@ rb_str_codepoints(VALUE str)
     return rb_str_enumerate_codepoints(str, 1);
 }

+static VALUE
+rb_str_enumerate_graphemes(VALUE str, int wantarray)
+{
+    regex_t *reg_grapheme = NULL;
+    static regex_t *reg_grapheme_utf8 = NULL;
+    int encidx = ENCODING_GET(str);
+    rb_encoding *enc = rb_enc_from_index(encidx);
+    int unicode_p = rb_enc_unicode_p(enc);
+    const char *ptr, *end;
+    VALUE ary;
+
+    if (!unicode_p) {
+ return rb_str_enumerate_codepoints(str, wantarray);
+    }
+
+    /* synchronize */
+    if (encidx == rb_utf8_encindex() && reg_grapheme_utf8) {
+ reg_grapheme = reg_grapheme_utf8;
+    }
+    if (!reg_grapheme) {
+ const OnigUChar source[] = "\\X";
+ int r = onig_new(&reg_grapheme, source, source + sizeof(source) - 1,
+  ONIG_OPTION_DEFAULT, enc, OnigDefaultSyntax, NULL);
+ if (r) {
+     rb_bug("cannot compile grapheme cluster regexp");
+ }
+ if (encidx == rb_utf8_encindex()) {
+     reg_grapheme_utf8 = reg_grapheme;
+ }
+    }
+
+    ptr = RSTRING_PTR(str);
+    end = RSTRING_END(str);
+
+    if (rb_block_given_p()) {
+ if (wantarray) {
+#if STRING_ENUMERATORS_WANTARRAY
+     rb_warn("given block not used");
+     ary = rb_ary_new_capa(str_strlen(str, enc)); /* str's enc*/
+#else
+     rb_warning("passing a block to String#codepoints is deprecated");
+     wantarray = 0;
+#endif
+ }
+    }
+    else {
+ if (wantarray)
+     ary = rb_ary_new_capa(str_strlen(str, enc)); /* str's enc*/
+ else
+     return SIZED_ENUMERATOR(str, 0, 0, rb_str_each_char_size);
+    }
+
+    while (ptr < end) {
+ VALUE grapheme;
+ OnigPosition len = onig_match(reg_grapheme,
```

```
+  (const OnigUChar *)ptr, (const OnigUChar *)end,
+  (const OnigUChar *)ptr, NULL, 0);
+ if (len == 0) break;
+ if (len < 0) {
+     break;
+ }
+ grapheme = rb_enc_str_new(ptr, len, enc);
+ if (wantarray)
+     rb_ary_push(ary, grapheme);
+ else
+     rb_yield(grapheme);
+ ptr += len;
+    }
+    if (wantarray)
+ return ary;
+    else
+ return str;
+}
+
+/*
+ *  call-seq:
+ *     str.each_grapheme {|cstr| block }    -> str
+ *     str.each_grapheme                    -> an_enumerator
+ *
+ *  Passes each grapheme cluster in <i>str</i> to the given block, or returns
+ *  an enumerator if no block is given.
+ *  Unlike String#each_char, this enumerates by grapheme clusters defined by
+ *  Unicode Standard Annex #29 http://unicode.org/reports/tr29/
+ *
+ *     "a\u0300".each_chars.to_a.size #=> 2
+ *     "a\u0300".each_grapheme.to_a.size #=> 1
+ *
+ */
+
+static VALUE
+rb_str_each_grapheme(VALUE str)
+{
+    return rb_str_enumerate_graphemes(str, 0);
+}
+
+/*
+ *  call-seq:
+ *     str.graphemes   -> an_array
+ *
+ *  Returns an array of grapheme clusters in <i>str</i>.  This is a shorthand
+ *  for <code>str.each_grapheme.to_a</code>.
+ *
+ *  If a block is given, which is a deprecated form, works the same as
+ *  <code>each_grapheme</code>.
+ */
+
+static VALUE
+rb_str_graphemes(VALUE str)
+{
+    return rb_str_enumerate_graphemes(str, 1);
+}

 static long
 chopped_length(VALUE str)
@@ -10477,6 +10588,7 @@ Init_String(void)
     rb_define_method(rb_cString, "bytes", rb_str_bytes, 0);
     rb_define_method(rb_cString, "chars", rb_str_chars, 0);
     rb_define_method(rb_cString, "codepoints", rb_str_codepoints, 0);
+    rb_define_method(rb_cString, "graphemes", rb_str_graphemes, 0);
     rb_define_method(rb_cString, "reverse", rb_str_reverse, 0);
     rb_define_method(rb_cString, "reverse!", rb_str_reverse_bang, 0);
     rb_define_method(rb_cString, "concat", rb_str_concat_multi, -1);
@@ -10532,6 +10644,7 @@ Init_String(void)
     rb_define_method(rb_cString, "each_byte", rb_str_each_byte, 0);
     rb_define_method(rb_cString, "each_char", rb_str_each_char, 0);
     rb_define_method(rb_cString, "each_codepoint", rb_str_each_codepoint, 0);
+    rb_define_method(rb_cString, "each_grapheme", rb_str_each_grapheme, 0);

     rb_define_method(rb_cString, "sum", rb_str_sum, -1);
```

```
diff --git a/test/ruby/test_string.rb b/test/ruby/test_string.rb
index e88d749123..e3b44725df 100644
--- a/test/ruby/test_string.rb
+++ b/test/ruby/test_string.rb
@@ -885,6 +885,46 @@ def test_chars
     end
   end

+  def test_each_grapheme
+    [
+      "\u{20 200d}",
+      "\u{600 600}",
+      "\u{600 20}",
+      "\u{261d 1F3FB}",
+      "\u{1f600}",
+      "\u{20 308}",
+      "\u{1F477 1F3FF 200D 2640 FE0F}",
+      "\u{1F468 200D 1F393}",
+      "\u{1F46F 200D 2642 FE0F}",
+      "\u{1f469 200d 2764 fe0f 200d 1f469}",
+    ].each do |g|
+      assert_equal [g], g.each_grapheme.to_a
+    end
+
+    assert_equal ["\u000A", "\u0308"], "\u{a 308}".each_grapheme.to_a
+    assert_equal ["\u000D", "\u0308"], "\u{d 308}".each_grapheme.to_a
+  end
+
+  def test_graphemes
+    [
+      "\u{20 200d}",
+      "\u{600 600}",
+      "\u{600 20}",
+      "\u{261d 1F3FB}",
+      "\u{1f600}",
+      "\u{20 308}",
+      "\u{1F477 1F3FF 200D 2640 FE0F}",
+      "\u{1F468 200D 1F393}",
+      "\u{1F46F 200D 2642 FE0F}",
+      "\u{1f469 200d 2764 fe0f 200d 1f469}",
+    ].each do |g|
+      assert_equal [g], g.graphemes
+    end
+
+    assert_equal ["\u000A", "\u0308"], "\u{a 308}".graphemes
+    assert_equal ["\u000D", "\u0308"], "\u{d 308}".graphemes
+  end
+
   def test_each_line
     save = $/
     $/ = "\n"
```

#6 - 08/13/2017 12:15 AM - nobu (Nobuyoshi Nakada)

naruse (Yui NARUSE) wrote:

```
+    if (!unicode_p) {
+ return rb_str_enumerate_codepoints(str, wantarray);
+    }
```

Why codepoints?

#7 - 08/14/2017 10:22 AM - naruse (Yui NARUSE)

nobu (Nobuyoshi Nakada) wrote:

> naruse (Yui NARUSE) wrote:
>
> ```
> +    if (!unicode_p) {
> + return rb_str_enumerate_codepoints(str, wantarray);
> +    }
> ```

Why codepoints?

Ah, it should be chars; thanks!

#8 - 08/14/2017 10:57 AM - rbjl (Jan Lelis)

Great to see this implemented!

One tiny thing I've noticed:

- For non-Unicode strings, \X will still match "\r\n" as a single grapheme. This should probably also be the case with String#each_grapheme - or the difference should be clearly documented

#9 - 08/14/2017 11:00 AM - rbjl (Jan Lelis)

And a typo in "a\u0300".each_chars.to_a.size #=> 2,
should be "a\u0300".each_char.to_a.size #=> 2

#10 - 08/31/2017 05:32 AM - matz (Yukihiro Matsumoto)

grapheme sounds like an element in the grapheme cluster. How about each_grapheme_cluster?
If everyone gets used to the grapheme as an alias of grapheme cluster, we'd love to add an alias each_grapheme.

Matz.

#11 - 08/31/2017 06:35 AM - naruse (Yui NARUSE)

- Status changed from Assigned to Closed

Applied in changeset trunk|r59698.

String#each_grapheme_cluster and String#grapheme_clusters

added to enumerate grapheme clusters [Feature #13780]

#12 - 02/01/2022 08:26 PM - mame (Yusuke Endoh)

- Related to Feature #18563: Add "graphemes" and "each_grapheme" aliases added