Ruby master - Feature #6670

str.chars.last should be possible

06/30/2012 02:52 AM - yhara (Yutaka HARA)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>mame (Yusuke Endoh)</td>
</tr>
<tr>
<td>Target version:</td>
<td>3.0</td>
</tr>
</tbody>
</table>

**Description**

```
=begin
Since str.chars returns an Enumerator, we need explicit to_a for some operations:

str.chars.to_a.last
str.chars.to_a[1,3]

But often I forget that and write:

str.chars.last
str.chars[1,3]

Besides that, I feel it is hard to explain why to_a is needed here when I'm writing articles for Ruby beginners.

Simplest way to achieve this is to make String#chars (also #lines, #bytes and #codepoints) return an Array. Since arrays have most of the methods defined in Enumerator, this will not be a big change. For programs like str.chars.next, you can use each_char instead.
=end
```

**Related issues:**

Related to Ruby master - Bug #7440: IO#lines etc. should return Array

Rejected 11/26/2012

**Associated revisions**

Revision 3960936 - 11/24/2012 06:46 PM - knu (Akinori MUSHA)

String#(lines,chars,codepoints,bytes) now return an array.

- string.c (rb_str_each_line, rb_str_lines): String#lines now returns an array instead of an enumerator. Passing a block is deprecated but still supported for backwards compatibility. Based on the patch by yhara. [Feature #6670]
- string.c (rb_str_each_char, rb_str_chars): Ditto for String#chars.
- string.c (rb_str_each_codepoint, rb_str_codepoints): Ditto for String#codepoints.
- string.c (rb_str_each_byte, rb_str_bytes): Ditto for String#bytes.
- NEWS: Add notes for the above changes.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37838 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 37838 - 11/24/2012 06:46 PM - knu (Akinori MUSHA)

String#(lines,chars,codepoints,bytes) now return an array.

- string.c (rb_str_each_line, rb_str_lines): String#lines now returns an array instead of an enumerator. Passing a block is deprecated but still supported for backwards compatibility. Based on the patch by yhara. [Feature #6670]
- string.c (rb_str_each_char, rb_str_chars): Ditto for String#chars.
- string.c (rb_str_each_codepoint, rb_str_codepoints): Ditto for String#codepoints.
- string.c (rb_str_each_byte, rb_str_bytes): Ditto for String#bytes.
- NEWS: Add notes for the above changes.

Revision 37838 - 11/24/2012 06:46 PM - knu (Akinori MUSHA)

String#(lines, chars, codepoints, bytes) now return an array.

- string.c (rb_str_each_line, rb_str_lines): String#lines now returns an array instead of an enumerator. Passing a block is deprecated but still supported for backwards compatibility. Based on the patch by yhara. [Feature #6670]
- string.c (rb_str_each_char, rb_str_chars): Ditto for String#chars.
- string.c (rb_str_each_codepoint, rb_str_codepoints): Ditto for String#codepoints.
- string.c (rb_str_each_byte, rb_str_bytes): Ditto for String#bytes.
- NEWS: Add notes for the above changes.

Revision 37838 - 11/24/2012 06:46 PM - knu (Akinori MUSHA)

String#(lines, chars, codepoints, bytes) now return an array.

- string.c (rb_str_each_line, rb_str_lines): String#lines now returns an array instead of an enumerator. Passing a block is deprecated but still supported for backwards compatibility. Based on the patch by yhara. [Feature #6670]
- string.c (rb_str_each_char, rb_str_chars): Ditto for String#chars.
- string.c (rb_str_each_codepoint, rb_str_codepoints): Ditto for String#codepoints.
- string.c (rb_str_each_byte, rb_str_bytes): Ditto for String#bytes.
- NEWS: Add notes for the above changes.

Revision 37838 - 11/24/2012 06:46 PM - knu (Akinori MUSHA)

String#(lines, chars, codepoints, bytes) now return an array.

- string.c (rb_str_each_line, rb_str_lines): String#lines now returns an array instead of an enumerator. Passing a block is deprecated but still supported for backwards compatibility. Based on the patch by yhara. [Feature #6670]
- string.c (rb_str_each_char, rb_str_chars): Ditto for String#chars.
- string.c (rb_str_each_codepoint, rb_str_codepoints): Ditto for String#codepoints.
- string.c (rb_str_each_byte, rb_str_bytes): Ditto for String#bytes.
- NEWS: Add notes for the above changes.
String#{lines,chars,codepoints,bytes} now return an array.

- string.c (rb_str_each_line, rb_str_lines): String#lines now returns an array instead of an enumerator. Passing a block is deprecated but still supported for backwards compatibility. Based on the patch by yhara. [Feature #6670]
- string.c (rb_str_each_char, rb_str_chars): Ditto for String#chars.
- string.c (rb_str_each_codepoint, rb_str_codepoints): Ditto for String#codepoints.
- string.c (rb_str_each_byte, rb_str_bytes): Ditto for String#bytes.

NEWS: Add notes for the above changes.

Revision 37838 - 11/24/2012 06:46 PM - knu (Akinori MUSHA)

String#{lines,chars,codepoints,bytes} now return an array.

- string.c (rb_str_each_line, rb_str_lines): String#lines now returns an array instead of an enumerator. Passing a block is deprecated but still supported for backwards compatibility. Based on the patch by yhara. [Feature #6670]
- string.c (rb_str_each_char, rb_str_chars): Ditto for String#chars.
- string.c (rb_str_each_codepoint, rb_str_codepoints): Ditto for String#codepoints.
- string.c (rb_str_each_byte, rb_str_bytes): Ditto for String#bytes.

NEWS: Add notes for the above changes.

Revision 37838 - 11/24/2012 06:46 PM - knu (Akinori MUSHA)

String#{lines,chars,codepoints,bytes} now return an array.

- string.c (rb_str_each_line, rb_str_lines): String#lines now returns an array instead of an enumerator. Passing a block is deprecated but still supported for backwards compatibility. Based on the patch by yhara. [Feature #6670]
- string.c (rb_str_each_char, rb_str_chars): Ditto for String#chars.
- string.c (rb_str_each_codepoint, rb_str_codepoints): Ditto for String#codepoints.
- string.c (rb_str_each_byte, rb_str_bytes): Ditto for String#bytes.

NEWS: Add notes for the above changes.

Revision c47c095b - 12/22/2012 05:22 PM - knu (Akinori MUSHA)

Deprecate #{lines,bytes,chars,codepoints} of IO-likes.

- io.c (rb_io_lines, rb_io_bytes, rb_io_chars, rb_io_codepoints): Deprecate IO#{lines,bytes,chars,codepoints} and those of ARGF. [Feature #6670]
- ext/stringio/stringio.c (strio_lines, strio_bytes, strio_chars) (strio_codepoints): Deprecate StringIO#{lines,bytes,chars,codepoints}. [Feature #6670]
- ext/zlib/zlib.c (rb_gzreader_lines, rb_gzreader_bytes): Deprecate Zlib::GzipReader#{lines,bytes}. [Feature #6670]

Revision 38563 - 12/22/2012 05:22 PM - knu (Akinori MUSHA)

Deprecate #{lines,bytes,chars,codepoints} of IO-likes.

- io.c (rb_io_lines, rb_io_bytes, rb_io_chars, rb_io_codepoints): Deprecate IO#{lines,bytes,chars,codepoints} and those of ARGF. [Feature #6670]
- ext/stringio/stringio.c (strio_lines, strio_bytes, strio_chars) (strio_codepoints): Deprecate StringIO#{lines,bytes,chars,codepoints}. [Feature #6670]
- ext/zlib/zlib.c (rb_gzreader_lines, rb_gzreader_bytes): Deprecate Zlib::GzipReader#{lines,bytes}. [Feature #6670]
Deprecate #(lines,bytes,chars,codepoints) of IO-likes.

- io.c (rb_io_lines, rb_io_bytes, rb_io_chars, rb_io_codepoints):
  Deprecate IO#(lines,bytes,chars,codepoints) and those of ARGF.
  [Feature #6670]

- ext/stringio/stringio.c (strio_lines, strio_bytes, strio_chars)
  (strio_codepoints): Deprecate StringIO#(lines,bytes,chars,codepoints).
  [Feature #6670]

- ext/zlib/zlib.c (rb_gzreader_lines, rb_gzreader_bytes):
  Deprecate Zlib::GzipReader#(lines,bytes).
  [Feature #6670]
Deprecate IO#{lines,bytes,chars,codepoints} and those of ARGF.
[Feature #6670]

- ext/stringio/stringio.c (strio_lines, strio_bytes, strio_chars)
  (strio_codepoints): Deprecate
StringI#{lines,bytes,chars,codepoints}. [Feature #6670]

- ext/zlib/zlib.c (rb_gzreader_lines, rb_gzreader_bytes):
  Deprecate Zlib::GzipReader#{lines,bytes}. [Feature #6670]

Revision 94bdc4ed - 12/26/2018 02:43 PM - mame (Yusuke Endoh)
string.c: remove the deprecation warnings of String#bytes with block
And its friends: lines, chars, grapheme_clusters, and codepoints.
[Feature #6670] [ruby-core:90729]
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@66579 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 66579 - 12/26/2018 02:43 PM - mame (Yusuke Endoh)
string.c: remove the deprecation warnings of String#bytes with block
And its friends: lines, chars, grapheme_clusters, and codepoints.
[Feature #6670] [ruby-core:90728]

History
#1 - 06/30/2012 02:54 AM - yhara (Yutaka HARA)
- File 6670.pdf added
Adding presentation slide for the feature request meeting ([ruby-dev:45708])

#2 - 07/02/2012 01:41 AM - mame (Yusuke Endoh)
- Status changed from Open to Assigned
- Assignee set to matz (Yukihiro Matsumoto)

Received. Thank you!
You really want just #last?
--
Yusuke Endoh mame@tsg.ne.jp

#3 - 07/02/2012 02:59 AM - trans (Thomas Sawyer)
While I realize it doesn't exactly fit with the whole iteration thing particularly well, I nonetheless do not think it’s unreasonable for Enumerator to
support something like #last(n=1). In most cases that just means it has to iterate on down to the end and deliver the result. In some cases it might be
able to optimize, say if the enumerable has a fixed size.
This issue might also have some relation to the issue about Enumerable#size, #6636.

#4 - 07/02/2012 12:07 PM - marcandre (Marc-Andre Lafortune)
Hi,
trans (Thomas Sawyer) wrote:

  This issue might also have some relation to the issue about Enumerable#size, #6636.

Not directly. Being able to calculate lazily a size does not mean it is easy to have random access, and the api I propose would be quite a bit more complicated if it was to allow for random access.
I feel that the lack of #last is a good thing. It’s there to remind you that it might be expensive, and that (for example) it would be worth storing in a local var instead of calling it twice...

#5 - 07/24/2012 10:53 PM - mame (Yusuke Endoh)
- Assignee changed from matz (Yukihiro Matsumoto) to yhara (Yutaka HARA)
Yutaka Hara,

I'm happy to inform you that matz has accepted your proposal, as making #chars, #lines, etc. return an Array (and keep #each_char, etc as is).

"foo".chars #=> ['f', 'o', 'o']
"foo".each_char #=> #

Yhara-san, could you create a patch?

--
Yusuke Endoh mame@tsg.ne.jp

#6 - 08/03/2012 07:53 PM - yhara (Yutaka HARA)

I'm happy to inform you that matz has accepted your proposal, as making #chars, #lines, etc. return an Array (and keep #each_char, etc as is).

Thank you. I'm happy too :-)

I will make a patch in this weekend.

#7 - 08/15/2012 09:30 PM - yhara (Yutaka HARA)

=begin
Hello,

I wrote a patch for String#lines, #chars, #bytes and #codepoints.

- [https://github.com/ruby/ruby/pull/158.diff](https://github.com/ruby/ruby/pull/158.diff)

Seemingly the following methods also need to be fixed (right?)

- IO#lines, chars, bytes, codepoints
- StringIO#lines, chars, bytes, codepoints
- ARGF.lines, chars, bytes
  - Is it intentional that ARGF.codepoints is missing? =end

#8 - 10/27/2012 07:13 AM - ko1 (Koichi Sasada)

ping.
status?

#9 - 10/29/2012 05:43 PM - knu (Akinori MUSHA)

If #lines is to become no longer an alias for #each_line, there should be no point in supporting lines {} any more. It should emit a deprecation warning and get unsupported in the future.

#10 - 11/19/2012 11:59 AM - zzak (Zachary Scott)

- File string_bytes_to_array.patch added
- Assignee changed from yhara (Yutaka HARA) to mame (Yusuke Endoh)

I've added Yutaka-san's patch from github, please continue discussion here.

mame, could you please review this?

#11 - 11/20/2012 03:14 AM - mame (Yusuke Endoh)

- Assignee changed from mame (Yusuke Endoh) to knu (Akinori MUSHA)

knu, could you please review this? :-)

--
Yusuke Endoh mame@tsg.ne.jp

#12 - 11/24/2012 11:16 AM - mame (Yusuke Endoh)

- Assignee changed from knu (Akinori MUSHA) to nagachika (Tomoyuki Chikanaga)
nagachika-san, could you please review this?

--
Yusuke Endoh mame@tsg.ne.jp

#13 - 11/24/2012 07:38 PM - knu (Akinori MUSHA)
Sorry, my mail filter was so buggy I failed to be notified.

I've reviewed and already given a comment above, and another on Twitter that yieldp can be dropped in favor of NIL_P(ary), without any of them responded to.
Should I revise the patch myself?

#14 - 11/24/2012 10:18 PM - nagachika (Tomoyuki Chikanaga)
- Assignee changed from nagachika (Tomoyuki Chikanaga) to knu (Akinori MUSHA)

Hello,
Sorry for late reply.

OK, I assign this ticket to knu san again.
Anyway I'm willing to review the patch :)

BTW, I cannot apply string_bytes_to_array.patch on trunk r37835.

#15 - 11/24/2012 10:24 PM - mame (Yusuke Endoh)
Thanks!
Should I revise the patch myself?

Could you do it, please?
If you are not willing, please pass the ball to nagachika-san.

--
Yusuke Endoh mame@tsg.ne.jp

#16 - 11/25/2012 03:46 AM - knu (Akinori MUSHA)
- Status changed from Assigned to Closed
- % Done changed from 0 to 100

This issue was solved with changeset r37838.
Yutaka, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

String#[lines,chars,codepoints,bytes] now return an array.

- string.c (rb_str_each_line, rb_str_lines): String#lines now returns an array instead of an enumerator. Passing a block is deprecated but still supported for backwards compatibility. Based on the patch by yhara. [Feature #6670]
- string.c (rb_str_each_char, rb_str_chars): Ditto for String#chars.
- string.c (rb_str_each_codepoint, rb_str_codepoints): Ditto for String#codepoints.
- string.c (rb_str_each_byte, rb_str_bytes): Ditto for String#bytes.
- NEWS: Add notes for the above changes.

#17 - 11/26/2012 02:23 AM - brixen (Brian Shirai)
What about IO, StringIO, ARGF as mentioned above?

Thanks,
Brian

#18 - 11/26/2012 06:01 AM - drbrain (Eric Hodel)

=begin
An IO may be infinite (((open "dev/zero" do |o| o.chars.to_a))). and so may ARGF (((%ruby -e 'ARGF.chars.to_a' /dev/zero))).

It doesn't make sense to add it to StringIO, but you make work around the omission through StringIO#string.
==end

#19 - 11/27/2012 05:36 AM - naruse (Yui NARUSE)

- Status changed from Closed to Assigned
- Assignee changed from knu (Akinori MUSHA) to matz (Yukihiro Matsumoto)

Now we have Enumerable#size, so we can know whether the last character is available or not by it. So how about changing String#chars to not Array but Enumerator with size and define last method?

#20 - 11/27/2012 08:15 AM - marcandre (Marc-Andre Lafortune)

#size will return nil for all enumerators based on IO.

Maybe the best is to have chars return an array for strings and deprecate it with a warning for IO.

#21 - 11/27/2012 12:21 PM - mame (Yusuke Endoh)

- Assignee changed from matz (Yukihiro Matsumoto) to yhara (Yutaka HARA)

Okay, I understand that it may be harmful to change the behavior right now. Then, let's warn a user to use 'each_line' instead in 2.0.0, and change it in future.

Yhara-san, could you create the following type of patch for all methods?

diff --git a/io.c b/io.c
index bacc6fc..26d3970 100644
--- a/io.c
+++ b/io.c
@@ -10795,6 +10795,13 @@ argf_each_line(int argc, VALUE *argv, VALUE argf)
 }
 }
+static VALUE
+argf_lines(int argc, VALUE *argv, VALUE argf)
+{
+   rb_warn("ARGF#lines will return an Array in future; you should use 'each_line' instead");
+   return argf_each_line(argc, argv, argv, VALUE argf);
+}
+
+static VALUE
+argf_lines(int argc, VALUE *argv, VALUE argf)
+
++
+   rb_warn("ARGF#lines will return an Array in future; you should use 'each_line' instead");
+   return argf_each_line(argc, argv, argv, VALUE argf);
+}
+
+call-seq:
+ARGF.bytes \{[byte] block \} -> ARGF @@ -11557,7 +11564,7 @@ Init_IO(void) rb_define_method(rb_cARGF, "each_line",
+argf_each_line, -1); rb_define_method(rb_cARGF, "each_byte", "each_byte", argf_each_byte, 0); rb_define_method(rb_cARGF, "each_char",
+argf_each_char, 0);
+}
+
+rb_define_method(rb_cARGF, "lines", argf_each_line, -1);
+rb_define_method(rb_cARGF, "lines", argf_lines, -1);
+rb_define_method(rb_cARGF, "bytes", argf_each_byte, 0);
++
+Yusuke Endoh  mame@tsg.ne.jp

#22 - 11/27/2012 08:14 PM - knu (Akinori MUSHA)

It may be an idea to simply obsolete IO#lines, chars, codepoints, bytes for now.

First of all, line wise operation is known to be popular and we've already got #readlines since a long time ago. So, there is no need for a new IO#lines that returns an array unlike String.

For chars, codepoints and bytes, there is no known need for turning a whole file or stream into an on-memory array because we haven't offered such methods to date. That may be because character or byte wise operation on a stream is typically done using a seek pointer as it reads lazily.

If this should be the case, we don't have an urgent need for a new IO#chars, #codepoints or #bytes. However, we, including matz, confirmed in this issue that a method that does not return an array having a name in the plural form is not intuitive because most Enumerator methods have a name consisting of each_ + singular noun, and lines etc. are breaking that convention.

02/21/2020
So, what about simply obsoleting IO#lines, chars, codepoints, bytes in the current shape?

We can re-add them later in another shape perhaps after we introduce an indexable enumerator (lazy array) that would satisfy everyone who might take both performance and intuitiveness seriously.

#23 - 11/27/2012 10:21 PM - trans (Thomas Sawyer)
If I understand correctly, this is going to break a lot of code?

#24 - 11/27/2012 11:56 PM - knu (Akinori MUSHA)
- File 0001-Deprecate-lines-bytes-chars-codepoints-of-IO-likes.patch added

Here's a patch to deprecate #lines, #bytes, #chars and #codepoints of IO-likes.

#25 - 11/28/2012 01:32 AM - yhara (Yutaka HARA)
trans (Thomas Sawyer) wrote:

If I understand correctly, this is going to break a lot of code?

For String, the impact will be limited.

- String#lines returns Array, which has most of the methods defined in Enumerator.
- Exceptions are #next, #peek, #with_index, etc.
  If you have a code like str.lines.with_index', you need to change it to str.each_line.with_index'.
- When you have a huge string, str.lines' will become slower and consume more memory.
  I think this is a rare case because we usually avoid File.read(path_to_huge_file').
  Even when you really need huge_str.lines to return Enumerator, you can use huge_str.each_line instead.

For IO/StringIO/ARGF/GzipReader, I'd like to +1 for showing deprecation warning in 2.0.0.

#26 - 11/28/2012 09:38 AM - duerst (Martin Dürst)
Instead of this proposal, what about adding some/most/all of the Array methods to Enumerator?

E.g. like so:

module Enumerator
  def to_a[pos]
  end
end

Of course, this is just the simplest case of [], and the simplest (and maybe slowest) implementation. This is just an idea, but I think it would be way better than having to distinguish lines/each_line, chars/each_char,... and so on.

#27 - 11/28/2012 12:20 PM - yhara (Yutaka HARA)
duerst (Martin Dürst) wrote:

Instead of this proposal, what about adding some/most/all of the Array methods to Enumerator?

It is not easy to define behavior of Enumerator#[].

For example:

File.open(path)[[]]
ls = l.lines
p lines[0]  #=> Prints first line
p lines[0]  #=> Prints nil, if [] is equivalent to to_a[pos]

#28 - 11/29/2012 02:13 PM - duerst (Martin Dürst)
yhara (Yutaka HARA) wrote:

For String, the impact will be limited.

- String#lines returns Array, which has most of the methods defined in Enumerator.
Exceptions are #next, #peek, #with_index, etc.
If you have a code like str.lines.with_index', you need to change it to str.each_line.with_index'.

For with_index in particular, wouldn't it make sense to either add it to Enumerable or deprecate it on Enumerator? That would eliminate one more difference.

#29 - 11/29/2012 05:12 PM - knu (Akinori MUSHA)
- Assignee changed from yhara (Yutaka HARA) to matz (Yukihiro Matsumoto)

We don't have much time left before 2.0 to decide how to change IO#lines, #chars, etc. Can we deprecate them for now as a first step as proposed above?

#30 - 11/30/2012 06:08 PM - yhara (Yutaka HARA)

knu (Akinori MUSHA) wrote:

Here's a patch to deprecate #lines, #bytes, #chars and #codepoints of IO-likes.

Maybe we can change StringIO#lines to return Array now because it does not have problems like IO and ARGF. (problems = return value of IO#lines may be huge or even infinite)

#31 - 12/01/2012 01:28 AM - drbrain (Eric Hodel)

I often use StringIO as an IO substitute for tests. I would prefer matching behavior.

You can get the last line by stringio.string.lines.last. Is this acceptable?

#32 - 12/01/2012 02:02 AM - yhara (Yutaka HARA)

duerst (Martin Dürst) wrote:

For with_index in particular, wouldn't it make sense to either add it to Enumerable or deprecate it on Enumerator? That would eliminate one more difference.

If we add Enumerable#with_index, we could write str.with_index. This does not make sense because String#each is not defined.

drbrain (Eric Hodel) wrote:

I often use StringIO as an IO substitute for tests. I would prefer matching behavior.

You can get the last line by stringio.string.lines.last. Is this acceptable?

Yes. And I got a reply from knu (Akinori MUSHA): https://twitter.com/knu/status/274442544518156288
He said "We should not fix API of StringIO before fixing that of IO" and I agreed.

#33 - 12/23/2012 02:22 AM - knu (Akinori MUSHA)
- Status changed from Assigned to Closed

This issue was solved with changeset r38563.
Yutaka, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

Deprecate #lines,bytes,chars,codepoints of IO-likes.

- io.c (rb_io_lines, rb_io_bytes, rb_io_chars, rb_io_codepoints): Deprecate IO#lines,bytes,chars,codepoints] and those of ARGF. [Feature #6670]
- ext/stringio/stringio.c (strio_lines, strio_bytes, strio_chars) (strio_codepoints): Deprecate StringIO#lines,bytes,chars,codepoints]. [Feature #6670]
- ext/zlib/zlib.c (rb_gzreader_lines, rb_gzreader_bytes): Deprecate Zlib::GzipReader#lines,bytes]. [Feature #6670]
I have forgotten this ticket completely, and I have used String#lines with a block many times. Will the methods really stop yielding each line in 3.0? I think that the current behavior (callback only if block is given, and always return an array) is somewhat reasonable, at least, not harmful. So, how about keeping the current behavior as is?

While I myself have not needed str.chars.last, I can understand why Yutaka HARA suggested it. Makes sense to me personally:

One comment on this:

I feel that the lack of #last is a good thing. It's there to remind you that it might be expensive, and that (for example) it would be worth storing in a local var instead of calling it twice.

In general, in ruby, speed considerations are not the first typical thought that may come to every ruby hacker. And there is a big difference between (a) something that is slow, but possible, and (b) something that may be slow, but is not possible. I think in this case, (a) applies and the speed consideration comes (or should come) secondary - so I don't feel that using speed as an example, should be something that speaks against the proposal itself. IMO the question should be whether this is sufficiently useful or not. I can't answer this either, but from that point of view, I think the proposal by Yutaka HARA makes sense. Matz will probably say something about the suggestion in the upcoming developer meeting. :)

PS: The documentation can also mention whether there are speed considerations, so I think this is another reason why speed considerations should not come primary as opposed to functionality that may be useful (whether it is, I do not know; just mentioning this in general).

Sorry for long absent. We discussed this ticket at DevelopersMeeting20180419Japan, and matz decided to withdraw the deprecation of "String#bytes with block" and friends. I'll remove the deprecation warnings soon.

I committed r66575. I forgot to write the ticket number...

<table>
<thead>
<tr>
<th>Files</th>
<th>Date</th>
<th>Size</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>6670.pdf</td>
<td>06/30/2012</td>
<td>39.4 KB</td>
<td>yhara (Yutaka HARA)</td>
</tr>
<tr>
<td>string_bytes_to_array.patch</td>
<td>11/19/2012</td>
<td>27.4 KB</td>
<td>zzak (Zachary Scott)</td>
</tr>
<tr>
<td>0001-Deprecate-lines-bytes-chars-codepoints-of-IO-likes.patch</td>
<td>11/27/2012</td>
<td>21.9 KB</td>
<td>knu (Akinori MUSHA)</td>
</tr>
</tbody>
</table>