Add String#bytesplice

02/22/2022 09:21 AM - shugo (Shugo Maeda)

Status: Closed
Priority: Normal
Assignee:
Target version:

Description
I withdrew the proposal of String#bytesplice in #13110 because it may cause problems if the specified offset does not land on character boundary.

But how about to raise IndexError in such cases?

```ruby
# encoding: utf-8
s = "あいうえおかきくけこ"
s.bytesplice(9, 6, "xx")
p s #=> "あいうxxかきくけこ"
s.bytesplice(2, 3, "x") #=> offset 2 does not land on character boundary (IndexError)
s.bytesplice(3, 4, "x") #=> offset 7 does not land on character boundary (IndexError)
```

Pull request
https://github.com/ruby/ruby/pull/5584

Spec

bytesplice(index, length, str) -> string
bytesplice(range, str) -> string

Replaces some or all of the content of +self+ with +str+, and returns +str+.
The portion of the string affected is determined using the same criteria as String#byteslice, except that +length+ cannot be omitted.
If the replacement string is not the same length as the text it is replacing, the string will be adjusted accordingly.
The form that take an Integer will raise an IndexError if the value is out of range; the Range form will raise a RangeError.
If the beginning or ending offset does not land on character (codepoint) boundary, an IndexError will be raised.

Motivation

On a text editor Textbringer, the content of a buffer is represented by a String whose encoding is ASCII-8BIT, and force_encoding(Encoding::UTF_8) is called when necessary.
It's because point (cursor position) and marks are represented by byte offsets for performance, and currently there is no way to modify UTF-8 strings with byte offsets.
If String#bytesplice is introduced, the content of a text buffer can be represented by a UTF-8 string, and force_encoding can be removed. [https://github.com/shugo/textbringer/pull/31/files](https://github.com/shugo/textbringer/pull/31/files)

Associated revisions

Revision 2fd4d499 - 03/18/2022 02:53 AM - shugo (Shugo Maeda)
Add a NEWS entry about [Feature #18598] [ci skip]

History

#1 - 02/22/2022 09:21 AM - shugo (Shugo Maeda)
- Description updated

#2 - 02/22/2022 10:31 AM - shugo (Shugo Maeda)
- Description updated

#3 - 02/22/2022 10:48 AM - Eregon (Benoit Daloze)
Shouldn't a text editor use the ropes representation for Strings instead? ([https://en.wikipedia.org/wiki/Rope_%28data_structure%29](https://en.wikipedia.org/wiki/Rope%28data_structure%29))
This sounds very inefficient because bytesplice will need to copy everything after the insert if the inserted_bytes.length \(!=\) length.

That's more of a personal opinion but I always found splice arguments and semantics confusing, also in JavaScript. 
\([\_]\) at least makes it much clearer, but s.bytesplice(2, 3, "x") sounds like a C API to me.

If we do add this I would suggest only adding the Range version for simplicity.

I think for byteindex & byteoffset in #13110 there was good motivation, and Ruby internally would anyway need to use byte offsets so exposing those to the user seemed relatively harmless, and it needed as you showed very complex hacks.  
But here I question the need for it, because the code before bytesplice seems reasonable enough, i.e., the code before https://github.com/shugo/textbringer/pull/31/files seems fine enough.

It's also a very specific use case, I would like to see other use cases if we add a core method to String.

There are also other ways to solve this, where I think you semantically want a byte array/buffer which can be shown as text and searched:

- Use UTF-32LE/UTF-32BE to have constant indexing of Strings, then \([\_]\) works fine
- Can the String be kept as Encoding::BINARY all the time, why does it need to be UTF-8? Can it just be reencoded to UTF-8 in the few places which really need it?
- Do not use String and e.g. use an Array of byte values or a C extension
- Use Ropes or similar implemented in Ruby, which would avoid extra copying and might not need to use byte offsets at all
- Add some way to have a "cursor object" in a String, which knows both the byte index and the character index, and have its own methods, that would be much more general and could help improve the performance in far more cases (e.g., could also yield such a cursor in some each_char_with_cursor method). It's probably too tricky to implement correctly when the String is mutable though.

#4 - 02/22/2022 11:40 PM - shugo (Shugo Maeda)

Eregon (Benoît Daloze) wrote in #note-3:

Shouldn't a text editor use the ropes representation for Strings instead? (https://en.wikipedia.org/wiki/Rope_(data_structure))

This sounds very inefficient because bytesplice will need to copy everything after the insert if the inserted_bytes.length \(!=\) length.

In general ropes may be a good choice, but I prefer gap buffers (https://en.wikipedia.org/wiki/Gap_buffer) implemented by String.

Because it's simple and fast enough for regular files.

It's hard to implement time-efficient data structure in pure Ruby, but the built-in class String is fast and has rich features like regular expressions.

That's more of a personal opinion but I always found splice arguments and semantics confusing, also in JavaScript. 
\([\_]\) at least makes it much clearer, but s.bytesplice(2, 3, "x") sounds like a C API to me.

If we do add this I would suggest only adding the Range version for simplicity.

I prefer \([\_]\) too, but bytesplice is a low level API, so I think it's not a big issue.

Only adding the Range version is acceptable for me, but we already have String#byteslice and it supports the Integer version, so it may better to support the Integer version for consistency. However, omitting the second argument length is harmful for String#byteslice (if the default length is 1 byte, multibyte strings may get broken), so length shouldn't be omitted.

I think for byteindex & byteoffset in #13110 there was good motivation, and Ruby internally would anyway need to use byte offsets so exposing those to the user seemed relatively harmless, and it needed as you showed very complex hacks.  
But here I question the need for it, because the code before bytesplice seems reasonable enough, i.e., the code before https://github.com/shugo/textbringer/pull/31/files seems fine enough.

It's also a very specific use case, I would like to see other use cases if we add a core method to String.

I'd like to hear other's opinions about use cases. I agree that Textbringer is a specific use case, but it's important for me, and it's good to introduce String#byteslice if it's useful for others.

There are also other ways to solve this, where I think you semantically want a byte array/buffer which can be shown as text and searched:

- Use UTF-32LE/UTF-32BE to have constant indexing of Strings, then \([\_]\) works fine

UTF-32 is not ASCII compatible and cannot be used as a script encoding, so I prefer UTF-8.

- Can the String be kept as Encoding::BINARY all the time, why does it need to be UTF-8? Can it just be reencoded to UTF-8 in the few places which really need it?

It's possible and that's what I do in the implementation of Textbringer. But such encoding changes are unnecessary if Ruby supports byte-based operations on strings with text encodings. I've heard from Naruse-san that he has the similar idea. He may have more use cases.

- Do not use String and e.g. use an Array of byte values or a C extension

I wouldn't like to implement regular expressions on Array.
Use Ropes or similar implemented in Ruby, which would avoid extra copying and might not need to use byte offsets at all

I prefer String for the reasons stated above.

Add some way to have a "cursor object" in a String, which knows both the byte index and the character index, and have its own methods, that would be much more general and could help improve the performance in far more cases (e.g., could also yield such a cursor in some each_char_with_cursor method). It's probably too tricky to implement correctly when the String is mutable though.

Such a cursor object can be alternative of byte-based methods, but a cursor object is only valid for a particular String at a particular time, and it may bring different issues from byte-based methods.

#5 - 03/18/2022 01:43 AM - ko1 (Koichi Sasada)
Matz said "go ahead".

#6 - 03/18/2022 02:55 AM - shugo (Shugo Maeda)
- Status changed from Open to Closed

Applied in changeset git|2fd7d499db489db9eb4046849aa785c3bd382761.

Add a NEWS entry about [Feature #18598] [ci skip]

#7 - 03/18/2022 03:01 AM - shugo (Shugo Maeda)
String#bytesplice has been added by git|1107839a7fed31339fc947995b7b45b8eaf4041b.