Ruby - Feature #19314

String#bytesplice should support partial copy

01/06/2023 11:06 AM - shugo (Shugo Maeda)

Status:	Closed	
Priority:	Normal	
Assignee:		
Target version:		

Description

String#bytesplice should support partial copy without temporary String objects.

For example, given x = "0123456789", either of the following replaces the contents of x with "0167856789":

```
x.bytesplice(2, 3, x, 6, 3)
x.bytesplice(2..4, x, 6..8)
```

Considerations

- What should be the return value?
  - The return value should be the whole source string for performance and consistency with bytesplice(offset, len, s).
- Can the source and destination ranges overlap?
  - Yes.
- Can the source and destination lengths be different?
  - Yes.
- Can range form and offset/length form be mixed in the source and destination?
  - No.
- What should happen when any offset doesn't land on character boundary in text strings.
  - IndexError should be raised.
- Can the length be omitted in the destination?
  - Maybe yes, but it may be confusing.

Use cases

- Gapped buffer implementation for text editors
- NAT implementation
  - https://twitter.com/kazuho/status/1611279616098070532

Related issues:

Related to Ruby - Feature #19315: Lazy substrings in CRuby	**Open**

Associated revisions

Revision f7b72462aa27716370c6bea1f2c240983aca9a55 - 01/19/2023 08:13 AM - shugo (Shugo Maeda)

String#bytesplice should return self

In Feature #19314, we concluded that the return value of String#bytesplice
should be changed from the source string to the receiver, because the source
string is useless and confusing when extra arguments are added.

This change should be included in Ruby 3.2.1.

Revision f7b72462aa27716370c6bea1f2c240983aca9a55 - 01/19/2023 08:13 AM - shugo (Shugo Maeda)

String#bytesplice should return self

In Feature #19314, we concluded that the return value of String#bytesplice
should be changed from the source string to the receiver, because the source
string is useless and confusing when extra arguments are added.

This change should be included in Ruby 3.2.1.

Revision f7b72462 - 01/19/2023 08:13 AM - shugo (Shugo Maeda)

String#bytesplice should return self

In Feature #19314, we concluded that the return value of String#bytesplice
should be changed from the source string to the receiver, because the source
string is useless and confusing when extra arguments are added.

This change should be included in Ruby 3.2.1.

Revision 373e62248c9dceb660e95f1cf05fa2a4a469cd64 - 01/20/2023 03:24 AM - naruse (Yui NARUSE)

merge revision(s) f7b72462aa27716370c6bea1f2c240983aca9a55: [Backport #19356]

```
    String#bytesplice should return self

    In Feature #19314, we concluded that the return value of String#bytesplice
    should be changed from the source string to the receiver, because the source
    string is useless and confusing when extra arguments are added.

    This change should be included in Ruby 3.2.1.
    ---
     string.c                 | 4 ++--
     test/ruby/test_string.rb | 2 +-
     2 files changed, 3 insertions(+), 3 deletions(-)
```

Revision 373e62248c9dceb660e95f1cf05fa2a4a469cd64 - 01/20/2023 03:24 AM - naruse (Yui NARUSE)

merge revision(s) f7b72462aa27716370c6bea1f2c240983aca9a55: [Backport #19356]

```
    String#bytesplice should return self

    In Feature #19314, we concluded that the return value of String#bytesplice
    should be changed from the source string to the receiver, because the source
    string is useless and confusing when extra arguments are added.

    This change should be included in Ruby 3.2.1.
    ---
     string.c                 | 4 ++--
     test/ruby/test_string.rb | 2 +-
     2 files changed, 3 insertions(+), 3 deletions(-)
```

Revision 373e6224 - 01/20/2023 03:24 AM - naruse (Yui NARUSE)

merge revision(s) f7b72462aa27716370c6bea1f2c240983aca9a55: [Backport #19356]

```
    String#bytesplice should return self

    In Feature #19314, we concluded that the return value of String#bytesplice
    should be changed from the source string to the receiver, because the source
    string is useless and confusing when extra arguments are added.

    This change should be included in Ruby 3.2.1.
    ---
     string.c                 | 4 ++--
     test/ruby/test_string.rb | 2 +-
     2 files changed, 3 insertions(+), 3 deletions(-)
```

Revision cce3960964784e57cba14762503c5fdd688e9919 - 01/20/2023 09:02 AM - shugo (Shugo Maeda)

[Feature #19314] Add new arguments of String#bytesplice

bytesplice(index, length, str, str_index, str_length) -> string
bytesplice(range, str, str_range) -> string

In these forms, the content of +self+ is replaced by str.byteslice(str_index, str_length) or str.byteslice(str_range); however the substring of +str+ is not allocated as a new string.

Revision cce3960964784e57cba14762503c5fdd688e9919 - 01/20/2023 09:02 AM - shugo (Shugo Maeda)

[Feature #19314] Add new arguments of String#bytesplice

bytesplice(index, length, str, str_index, str_length) -> string
bytesplice(range, str, str_range) -> string

In these forms, the content of +self+ is replaced by str.byteslice(str_index, str_length) or str.byteslice(str_range); however the substring of +str+ is not allocated as a new string.

Revision cce39609 - 01/20/2023 09:02 AM - shugo (Shugo Maeda)

[Feature #19314] Add new arguments of String#bytesplice

bytesplice(index, length, str, str_index, str_length) -> string
bytesplice(range, str, str_range) -> string

In these forms, the content of +self+ is replaced by str.byteslice(str_index, str_length) or str.byteslice(str_range); however the substring of +str+ is not allocated as a new string.

Revision c948a081367300f46af41905c6bf4813d12e95bc - 01/20/2023 12:41 PM - shugo (Shugo Maeda)

Add a NEWS entry for [Feature #19314] [ci skip]

Revision c948a081367300f46af41905c6bf4813d12e95bc - 01/20/2023 12:41 PM - shugo (Shugo Maeda)

Add a NEWS entry for [Feature #19314] [ci skip]

Revision c948a081 - 01/20/2023 12:41 PM - shugo (Shugo Maeda)

Add a NEWS entry for [Feature #19314] [ci skip]

History

#1 - 01/06/2023 12:20 PM - Eregon (Benoit Daloze)

I think this is too hard to read and parse for a human and 5 arguments seems way too much for a core method.
It feels like a full memcpy/arraycopy which I don't think in general is a good idea for String.
The implementation complexity in []= and similar already hurts Ruby too much.

This is probably the 3rd or more workaround I see to have proper lazy substrings in CRuby, i.e., "abcdef"[1..3] must not copy bytes.
That is what needs to be solved (it already works in TruffleRuby).
Yes, it means RSTRING_PTR() might need to allocate to \0-terminate, so be it, it's worth it.

So I am strongly against this, it's a nth workaround for something simpler to solve which is much more helpful in general.

#2 - 01/06/2023 12:30 PM - Eregon (Benoit Daloze)

- Related to Feature #19315: Lazy substrings in CRuby added

#3 - 01/06/2023 12:41 PM - naruse (Yui NARUSE)

I agree that this is a workaround and a VM should solve this as an optimization.

But your proposal: Lazy substrings is not a solution because it also creates an object especially for small strings which is embedded in RVALUE.

I agree that this is memcpy/arraycopy.
Therefore this proposal should add a description how large this workaround contributes performance in such use cases as memcpy on Ruby.

#4 - 01/06/2023 12:54 PM - Eregon (Benoit Daloze)

naruse (Yui NARUSE) wrote in #note-3:

> But your proposal: Lazy substrings is not a solution because it also creates an object especially for small strings which is embedded in RVALUE.

Yes it creates a String instance reusing the same buffer.
That shouldn't cost much compared to copying many bytes.
It should be insignificant on a benchmark with a long string to copy/move, for a short string perf shouldn't matter much anyway (it won't the be bottleneck of the program).

If it's still too much overhead, it sounds like allocations in CRuby need to be better optimized, or escape analysis should be implemented.
Again, those 2 are more general and benefits are much wider than this one method change that would be used for very few Ruby programs and only handles one specific case.

#5 - 01/06/2023 01:02 PM - Eregon (Benoit Daloze)

Ah, something I missed though is that with lazy substrings, there would still need to be a copy of the bytes to "unshare" the string when writing to it.
That copy would also be needed if the string was shared before (e.g. with .dup), but that's unknown in our case.
This does depend on how sharing is implemented, maybe CRuby can see it's only String instances sharing that buffer, and actually both strings are involved in this operation and so there is only need to copy the bytes of the substring.

> It feels like a full memcpy/arraycopy which I don't think in general is a good idea for String.

To expand on that, I dislike that because it's using String as a byte array.
If anything, such operation should be supported on Array before String.

Now that we have IO::Buffer and there is https://docs.ruby-lang.org/en/master/IO/Buffer.html#method-i-copy, why not use that?

#6 - 01/06/2023 01:14 PM - Eregon (Benoit Daloze)

Eregon (Benoit Daloze) wrote in #note-5:

> Now that we have IO::Buffer and there is https://docs.ruby-lang.org/en/master/IO/Buffer.html#method-i-copy, why not use that?

So this does what you want I believe:

```
x = "0123456789"
IO::Buffer.for(x) do |buffer|
  buffer.copy(buffer, 2, 3, 6)
end
p x # => "0167856789"
```

I think there is no need to change String#bytesplice therefore (there is even not a need for String#bytesplice due to that, which I think we shouldn't have added).
And IO::Buffer seems better suited for byte-buffer-like operations.

#7 - 01/07/2023 10:48 AM - naruse (Yui NARUSE)

> That shouldn't cost much compared to copying many bytes.

This proposal shows two use cases: text editor and NAT, which doesn't copy many bytes.

#8 - 01/19/2023 08:19 AM - shugo (Shugo Maeda)

- Status changed from Open to Closed

Applied in changeset git|f7b72462aa27716370c6bea1f2c240983aca9a55.

String#bytesplice should return self

In Feature #19314, we concluded that the return value of String#bytesplice
should be changed from the source string to the receiver, because the source
string is useless and confusing when extra arguments are added.

This change should be included in Ruby 3.2.1.

#9 - 01/19/2023 08:30 AM - shugo (Shugo Maeda)

- Status changed from Closed to Open

#10 - 01/19/2023 09:14 AM - matz (Yukihiro Matsumoto)

Accepted.

Matz.

#11 - 01/20/2023 08:14 AM - naruse (Yui NARUSE)

- Status changed from Open to Closed

Applied in changeset git|373e62248c9dceb660e95f1cf05fa2a4a469cd64.

---

merge revision(s) f7b72462aa27716370c6bea1f2c240983aca9a55: [Backport #19356]

```
    String#bytesplice should return self

    In Feature #19314, we concluded that the return value of String#bytesplice
    should be changed from the source string to the receiver, because the source
    string is useless and confusing when extra arguments are added.

    This change should be included in Ruby 3.2.1.
    ---
     string.c                 | 4 ++--
     test/ruby/test_string.rb | 2 +-
     2 files changed, 3 insertions(+), 3 deletions(-)
```