Ruby master - Bug #6203

Array#values_at does not handle ranges with end index past the end of the array

03/26/2012 01:50 PM - ferrous26 (Mark Rada)

Status: Closed
Priority: Normal
Assignee: nobu (Nobuyoshi Nakada)
Target version: 2.0.0
ruby -v: trunk

Description
=begin
When I use Array#values_at I expect that it would be the same as successive calls to (({Array#[]})).

There is one case where this does not hold:

a = [0,1,2,3,4,5]
a[4..6] # => [4, 5]
a.values_at(4..6) # => [4,5,nil]

I think this is an inconsistency in the design of (({Array#values_at})). We can look at a more extreme case:

a[4..100] # => [4, 5]
a.values_at 4..100 # => [4, 5, nil]

And now it doesn't make any sense.

I think the best solution would be to make (({Array#values_at})) be equivalent to successive calls to (({Array#[]})). I have patched (({rb_range_beg_len()})) to handle the extra case and opened a pull request on github.
=end

Associated revisions
Revision 7a654240 - 07/14/2012 09:18 AM - nobu (Nobuyoshi Nakada)
array.c: fill with nil
- array.c (rb_get_values_at): fill with nil out of range. [ruby-core:43678] [Bug #6203]
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@36393 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 36393 - 07/14/2012 09:18 AM - nobu (Nobuyoshi Nakada)
array.c: fill with nil
- array.c (rb_get_values_at): fill with nil out of range. [ruby-core:43678] [Bug #6203]

Revision 36393 - 07/14/2012 09:18 AM - nobu (Nobuyoshi Nakada)
array.c: fill with nil
- array.c (rb_get_values_at): fill with nil out of range. [ruby-core:43678] [Bug #6203]

Revision 36393 - 07/14/2012 09:18 AM - nobu (Nobuyoshi Nakada)
array.c: fill with nil
- array.c (rb_get_values_at): fill with nil out of range. [ruby-core:43678] [Bug #6203]

Revision 36393 - 07/14/2012 09:18 AM - nobu (Nobuyoshi Nakada)
array.c: fill with nil
- array.c (rb_get_values_at): fill with nil out of range. [ruby-core:43678] [Bug #6203]

Revision 36393 - 07/14/2012 09:18 AM - nobu (Nobuyoshi Nakada)
array.c: fill with nil
- array.c (rb_get_values_at): fill with nil out of range. [ruby-core:43678] [Bug #6203]

Revision 36393 - 07/14/2012 09:18 AM - nobu (Nobuyoshi Nakada)
array.c: fill with nil

- array.c (rb_get_values_at): fill with nil out of range. [ruby-core:43678] [Bug #6203]

Revision 36393 - 07/14/2012 09:18 AM - nobu (Nobuyoshi Nakada)
array.c: fill with nil

- array.c (rb_get_values_at): fill with nil out of range. [ruby-core:43678] [Bug #6203]

Revision 37769 - 11/20/2012 02:53 PM - marcandre (Marc-Andre Lafortune)

- NEWS: List incompatible change for Array#values_at [#6203]

Revision 37769 - 11/20/2012 02:53 PM - marcandre (Marc-Andre Lafortune)

- NEWS: List incompatible change for Array#values_at [#6203]

Revision 37769 - 11/20/2012 02:53 PM - marcandre (Marc-Andre Lafortune)

- NEWS: List incompatible change for Array#values_at [#6203]

Revision 37769 - 11/20/2012 02:53 PM - marcandre (Marc-Andre Lafortune)

- NEWS: List incompatible change for Array#values_at [#6203]

Revision 37769 - 11/20/2012 02:53 PM - marcandre (Marc-Andre Lafortune)

- NEWS: List incompatible change for Array#values_at [#6203]

Revision 37769 - 11/20/2012 02:53 PM - marcandre (Marc-Andre Lafortune)

- NEWS: List incompatible change for Array#values_at [#6203]

Revision 37775 - 11/21/2012 04:23 AM - marcandre (Marc-Andre Lafortune)

- range.c (rb_range_beg_len): Fix potential bug for limit case [#6203]

Revision 37775 - 11/21/2012 04:23 AM - marcandre (Marc-Andre Lafortune)

- range.c (rb_range_beg_len): Fix potential bug for limit case [#6203]

Revision 37775 - 11/21/2012 04:23 AM - marcandre (Marc-Andre Lafortune)

- range.c (rb_range_beg_len): Fix potential bug for limit case [#6203]

Revision 37775 - 11/21/2012 04:23 AM - marcandre (Marc-Andre Lafortune)

- range.c (rb_range_beg_len): Fix potential bug for limit case [#6203]

Revision 37775 - 11/21/2012 04:23 AM - marcandre (Marc-Andre Lafortune)

- range.c (rb_range_beg_len): Fix potential bug for limit case [#6203]

Revision 37775 - 11/21/2012 04:23 AM - marcandre (Marc-Andre Lafortune)

- range.c (rb_range_beg_len): Fix potential bug for limit case [#6203]

Revision 37775 - 11/21/2012 04:23 AM - marcandre (Marc-Andre Lafortune)

- range.c (rb_range_beg_len): Fix potential bug for limit case [#6203]

Revision 37775 - 11/21/2012 04:23 AM - marcandre (Marc-Andre Lafortune)

- range.c (rb_range_beg_len): Fix potential bug for limit case [#6203]

Revision 37775 - 11/21/2012 04:23 AM - marcandre (Marc-Andre Lafortune)

- range.c (rb_range_beg_len): Fix potential bug for limit case [#6203]

Revision 37775 - 11/21/2012 04:23 AM - marcandre (Marc-Andre Lafortune)

- range.c (rb_range_beg_len): Fix potential bug for limit case [#6203]

History
#1 - 03/26/2012 02:12 PM - nobu (Nobuyoshi Nakada)
- Tracker changed from Bug to Feature
It's definitely not a bug.

#2 - 03/26/2012 02:12 PM - nobu (Nobuyoshi Nakada)
- Description updated

#3 - 03/26/2012 10:46 PM - marcandre (Marc-Andre Lafortune)

Hi,

nobu (Nobuyoshi Nakada) wrote:

   It's definitely not a bug.

It's not?
How do you explain:

[1, 2, 3].values_at(2...42) # => [3]
[1, 2, 3].values_at(2..41) # => [3, nil]

I feel that both must return the same result.
Moreover, the only acceptable results are either [3] or [3, nil, nil, ... nil]

#4 - 03/29/2012 01:57 AM - marcandre (Marc-Andre Lafortune)
- Tracker changed from Feature to Bug

Moving back to "bug", as no explanation from Nobu.

#5 - 03/29/2012 05:14 AM - marcandre (Marc-Andre Lafortune)
- Assignee set to marcandre (Marc-Andre Lafortune)
- Target version changed from 1.9.3 to 2.0.0
- ruby -v set to trunk

The patch from Mark Rada never made it through, but I concur that the problem is in rb_range_beg_len.
I'll commit the fix below unless there's objection.
I've already fixed RubySpec:
https://github.com/rubyspec/rubyspec/commit/558a40edd5cd43a503641238cabe9481ce9a723

diff --git a/range.c b/range.c
index 61fb643..aa0628d 100644
--- a/range.c
+++ b/range.c
@@ -746,16 +746,16 @@ rb_range_beg_len(VALUE range, long *begp, long *lenp, long len, int err)
     if (begp < 0)
 goto out_of_range;
 }
 if (end < 0)
 end += len;
 if (!excl)
     end++;                  /* include end point */  if (err == 0 || err == 2) {     if (beg > len)         goto out_of_range;     if (end > len)         end = len;  }
 if (end < 0)
 end += len;
 if (!excl)
     end++;                  /* include end point */  len = end - beg;  if (len < 0)     len = 0;

#6 - 03/29/2012 05:22 AM - ferrous26 (Mark Rada)

marcandre (Marc-Andre Lafortune) wrote:

   The patch from Mark Rada never made it through, but I concur that the problem is in rb_range_beg_len.

Hello Marc,

Sorry for not linking to the pull request here, though I did open a pull request on Github here: https://github.com/ruby/ruby/pull/109

Though I think your solution is much nicer. Perhaps the tests I added are still useful.
Is this a bug?

#8 - 03/29/2012 11:11 AM - marcandre (Marc-Andre Lafortune)

Hi,

naruse (Yui NARUSE) wrote:

Is this a bug?

I'm very curious: how could it not be bug? How would you explain that values_at(2...42) != values_at(2..41)?

I simply want to lighten the load for Matz; he has so many issues assigned to him or waiting on him already.

#9 - 03/29/2012 12:27 PM - naruse (Yui NARUSE)

marcandre (Marc-Andre Lafortune) wrote:

I simply want to lighten the load for Matz; he has so many issues assigned to him or waiting on him already.

You must wait (or press) matz if a fix of an issue is clear and no other option of the new behavior.

#10 - 04/12/2012 03:28 AM - ferrous26 (Mark Rada)

naruse (Yui NARUSE) wrote:

marcandre (Marc-Andre Lafortune) wrote:

I simply want to lighten the load for Matz; he has so many issues assigned to him or waiting on him already.

You must wait (or press) matz if a fix of an issue is clear and no other option of the new behavior.

It has been 2 weeks since the last update. Has this issue been forgotten?

#11 - 07/14/2012 05:26 PM - matz (Yukihiro Matsumoto)

- Assignee changed from matz (Yukihiro Matsumoto) to nobu (Nobuyoshi Nakada)

The trailing nil must be a bug.

Matz.

#12 - 07/14/2012 06:18 PM - nobu (Nobuyoshi Nakada)

- Status changed from Assigned to Closed
- % Done changed from 0 to 100

This issue was solved with changeset r36393.
Mark, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

array.c: fill with nil

- array.c (rb_get_values_at): fill with nil out of range. [ruby-core:43678] [Bug #6203]

#13 - 07/16/2012 11:48 PM - marcandre (Marc-Andre Lafortune)

matz (Yukihiro Matsumoto) wrote:

The trailing nil must be a bug.
Was there a reason to change the behavior of[].values_at(42..100)? Could this not lead to incompatibilities?

Nobu: is there a reason not to fix rb_range_beg_len also as per [ruby-core:43811]?

#14 - 07/23/2012 09:11 AM - marcandre (Marc-Andre Lafortune)
- Status changed from Closed to Open

Marking as open, since fix doesn't seem to correspond to what Matz wrote.

marcandre (Marc-Andre Lafortune) wrote:

matz (Yukihiro Matsumoto) wrote:

The trailing nil must be a bug.

Was there a reason to change the behavior of[].values_at(42..100)? Could this not lead to incompatibilities?

Nobu: is there a reason not to fix rb_range_beg_len also as per [ruby-core:43811]?

#15 - 11/19/2012 02:13 PM - marcandre (Marc-Andre Lafortune)

Matz, could you please confirm what return value is correct for[].values_at(1..3)?

1) It could be []. This would be compatible with Ruby 1.8 & 1.9.
2) It could be [nil, nil, nil]. This is current behavior in trunk after Nobu's commit, but with potential incompatibility.

If 1, trunk must be changed
If 2, NEWS must list incompatibility.

Nobu: in either case, any reason not to commit my patch in [#5]?

Thanks

#16 - 11/20/2012 06:26 PM - matz (Yukihiro Matsumoto)

This is a spec change in 2.0 unless any big compatibility problem arise. Array#values_at should be define as:

def values_at(*idx)
idx.map{|i| self[i]}
end

So it should be written in the NEWS file.

Matz.

#17 - 11/21/2012 01:25 PM - marcandre (Marc-Andre Lafortune)
- Status changed from Open to Closed

Great, thanks.

NEWS updated.