Ruby - Misc #19123

Error handling of Struct#values_at and Array#values_at is slightly inconsistent

11/11/2022 05:29 PM - andrykonchin (Andrew Konchin)

Status:	Open
Priority:	Normal
Assignee:	

Description

Struct#values_at and `Array#values_at look pretty similar and handle all the complex cases of arguments (integer Ranges, list of Integers and mixing Ranges and Integers) in the same way.

Error handling is similar as well. In case of invalid Range argument they behave identically:

```
clazz = Struct.new(:name, :director, :year)
movie = clazz.new('Sympathy for Mr. Vengeance', 'Chan-wook Park', 2002)
array = [0, 1, 2]

# end is out of range
movie.values_at(0..4) # => ["Sympathy for Mr. Vengeance", "Chan-wook Park", 2002, nil, nil]
array.values_at(0..4) # => [0, 1, 2, nil, nil]

# beginning is out of range
movie.values_at(-5..4) # -5..4 out of range (RangeError)
array.values_at(-5..4) # -5..4 out of range (RangeError)
```

But when Integer argument is passed - they handle out of range cases differently:

```
# end is out of range
movie.values_at(4) # => offset 4 too large for struct(size:3) (IndexError)
array.values_at(4) # => [nil]

# beginning is out of range
movie.values_at(-5) # => offset -5 too small for struct(size:3) (IndexError)
array.values_at(-5) # => [nil]
```

So I am wondering what is the reason of such inconvenience. I suppose there may be a reason of this difference. But I see some benefits in consistent error handling.

History

#1 - 11/11/2022 05:29 PM - andrykonchin (Andrew Konchin)

- Tracker changed from Bug to Misc

- Backport deleted (2.7: UNKNOWN, 3.0: UNKNOWN, 3.1: UNKNOWN)