### Ruby master - Feature #12752

**Unpacking a value from a binary requires additional `.first'**  
09/12/2016 06:38 AM - tagomoris (Satoshi TAGOMORI)

<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></td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
</tr>
</tbody>
</table>

**Description**

When I parsed a binary protocol (msgpack) by `String#unpack`, there are many similar expressions, like below:

```ruby
str = io.read(1)  
length = str.unpack('C').first  
case length  
  when 4  
    data = io.read(4).unpack('N').first  
  # ...
```

I know that `unpack` returns "1 or more" values, but I think there are too many cases to return just one object. Creating Array objects for return values requires additional cost to create Array object, and calling `.first'.

I'm happy if I can write the code above like this:

```ruby
str = io.read(1)  
length = str.unpack('C', 0) # index, or str.unpack('C', index: 0)  
case length  
  when 4  
    data = io.read(4).unpack('N', 0)  
  # ...
```

**Associated revisions**

<table>
<thead>
<tr>
<th>Revision</th>
<th>09/02/2022</th>
<th>1/2</th>
</tr>
</thead>
</table>

**History**

#1 - 09/12/2016 06:41 AM - shyouhei (Shyouhei Urabe)
There is more concise way:
```
length, = str.unpack('C')
```

You cannot use this in a method chain, though.

I actually understand your frustration, but I feel no difference between `str.unpack('C', 0)` and `str.unpack('C')[0]`.

--
Yusuke Endoh mame@ruby-lang.org

How about Array#unpack1?

unpack1 looks so ad-hoc but surely is attractive.

Matz.

Go ahead.

Matz.

With a qualifier or multiple directives given, would it raise an ArgumentError or just go ahead with the process and only return the first value?

Just after processing the first value, it returns.

String#unpack1 [Feature #12752]

Returns the first value of String#unpack.