## StringScanner#pos returns wrong character position if used with multibyte chars

**06/26/2010 01:29 AM - Quintus (Marvin Gülker)**

<table>
<thead>
<tr>
<th>Status:</th>
<th>Rejected</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>2.0.0</td>
</tr>
<tr>
<td>ruby -v:</td>
<td>ruby 1.9.2dev (2010-05-31 revision 28117) [x86_64-linux]</td>
</tr>
<tr>
<td>Backport:</td>
<td></td>
</tr>
</tbody>
</table>

### Description

The `StringScanner` class from 1.9's stdlib works on bytes rather than on characters. That means, if you want to extract substrings from the original string by use of the return value of `StringScanner#pos` you get incorrect results:

```ruby
irb(main):001:0> require "strscan"
=> true
irb(main):002:0> str = "abcädeföghi"
=> "abcädeföghi"
irb(main):003:0> ss = StringScanner.new(str)
=> #
irb(main):004:0> ss.scan_until(/ä/)  # after the first scan_until
=> "abcä"
irb(main):005:0> ss.pos              # after the second scan_until
=> 5
irb(main):006:0> ss.scan_until(/ö/)
=> "defö"
irb(main):007:0> ss.pos
=> 10
irb(main):008:0>
```

After the first `scan_until` I expected the position to be 4, after the second to be 8, which means we finally have an offset of 2 here.

My Ruby version is ruby 1.9.1p378 (2010-01-10 revision 26273) [x86_64-linux], but I also get the same behaviour with the 1.9.2-preview3 (ruby 1.9.2dev (2010-05-31 revision 28117) [x86_64-linux]).

### Related issues:

- Related to Ruby master - Feature #1159: StringScanner に文字ベースでのインデックスを返すメソッドがほしい
  - Rejected
  - 02/14/2009

### History

**#1 - 06/26/2010 08:39 AM - mame (Yusuke Endoh)**

```
- Status changed from Open to Rejected

=begin
Hi,

It is a spec. See rdoc of StringScanner#pos.

FYI, IO#pos is also byte-oriented.
I guess this is because #pos is supposed to be byte-oriented.

--
Yusuke Endoh mame@tsg.ne.jp
=end
```