Ruby master - Bug #15460

Behaviour of String#setbyte changed

12/25/2018 10:22 AM - gettalong (Thomas Leitner)

<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>shyouhei (Shyouhei Urabe)</td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
</tr>
<tr>
<td>ruby -v:</td>
<td>ruby 2.6.0p0 (2018-12-25 revision 66547) [x86_64-linux]</td>
</tr>
<tr>
<td>Backport:</td>
<td>2.4: DONTNEED, 2.5: DONTNEED, 2.6: DONE</td>
</tr>
</tbody>
</table>

Description

I just installed Ruby 2.6.0 for benchmarking reasons and found that the change string.c: setbyte silently ignores upper bits broke my library/application HexaPDF.

Before using String#setbyte I tested how it would respond to values lower than 0 or greater than 255 and found that it automatically performed the needed modulo 256 operation (at least up to Ruby 2.5.3). Therefore I left out the explicit modulo operation for performance reasons.

Would it make sense to change the String#setbyte implementation to perform the modulo operation? This would restore compatibility with prior Ruby versions and may be what people would expect.

Related issues:

Related to Ruby master - Bug #10453: NUM2CHR() does not perform additional bo... Rejected

Associated revisions

Revision d154bec0 - 01/15/2019 06:41 AM - shyouhei (Shyouhei Urabe)

setbyte / ungetbyte allow out-of-range integers

- string.c: String#setbyte to accept arbitrary integers [Bug #15460]
- io.c: ditto for IO#ungetbyte
- ext/stringio/stringio.c: ditto for StringIO#ungetbyte

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@66824 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 66824 - 01/15/2019 06:41 AM - shyouhei (Shyouhei Urabe)

setbyte / ungetbyte allow out-of-range integers

- string.c: String#setbyte to accept arbitrary integers [Bug #15460]
- io.c: ditto for IO#ungetbyte
- ext/stringio/stringio.c: ditto for StringIO#ungetbyte

Revision 4cb618e7 - 01/17/2019 09:36 PM - naruse (Yui NARUSE)

merge revision(s) 66760,66761,66824: [Backport #15460]

Follow behaviour of IO#ungetbyte

see r65802 and [Bug #14359]

* expand tabs.

setbyte / ungetbyte allow out-of-range integers

* string.c: String#setbyte to accept arbitrary integers [Bug #15460]
* io.c: ditto for IO#ungetbyte
* ext/stringio/stringio.c: ditto for StringIO#ungetbyte
Revision 66845 - 01/17/2019 09:36 PM - naruse (Yui NARUSE)
merge revision(s) 66760,66761,66824: [Backport #15460]

Follow behaviour of IO#ungetbyte

see r65802 and [Bug #14359]
* expand tabs.
setbyte / ungetbyte allow out-of-range integers
* string.c: String#setbyte to accept arbitrary integers [Bug #15460]
* io.c: ditto for IO#ungetbyte
* ext/stringio/stringio.c: ditto for StringIO#ungetbyte

Revision b228a084 - 01/30/2019 01:09 PM - naruse (Yui NARUSE)
merge revision(s) 66888: [Backport #15460]

  * Fix rubyspec to follow IO#ungetbyte's fix
    Merge CRuby r66824
    With fixing actual spec and the version the change applied.

Revision 66952 - 01/30/2019 01:09 PM - naruse (Yui NARUSE)
merge revision(s) 66888: [Backport #15460]

  * Fix rubyspec to follow IO#ungetbyte's fix
    Merge CRuby r66824
    With fixing actual spec and the version the change applied.

History
#1 - 12/26/2018 12:38 AM - ko1 (Koichi Sasada)
FYI
7213568733f673da0d82f95e8albcecf79ba3f0d3
Author: shyouhei <shyouhei@b2dd03c8-39d4-4d8f-98ff-823fe69b080e>
Date: Mon Nov 19 09:52:46 2018 +0000

    string.c: setbyte silently ignores upper bits

    The behaviour of String#setbyte has been depending on the width
    of int, which is not portable. Must check explicitly.

    git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@65804 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

    100644 100644 5a5cbb576e30e1d2e5e6eb85aaf6a0676 e55c59136a49fb72d270850a65a4c7f56519f2e9 M string.c
    040000 040000 2170f3be4ac9d27e26ce361c38beb7874c07436f e71fc9f0ed5602a30e9f504f0c15b9bf74dad1 M test

#2 - 12/26/2018 12:48 AM - shyouhei (Shyouhei Urabe)
I feel sorry for the situation. I didn't expect any actual usage of that corner case.

However the previous behaviour was strange. It did actually raise exception when you pass a very huge number to it:

    % ruby -v -e 'p IO.pipe[0].ungetbyte(18446744073709551616)'
    ruby 1.9.3p551 (2014-11-13) [x86_64-darwin15.6.0]
    -e:1:in 'ungetbyte': can't convert Bignum into String (TypeError)
      from -e:1:in `<main>'

The *TypeError* in the error message is because the testing ruby is old. Now that Bignum is wiped out, I don't think this restriction is reasonable.

#3 - 12/26/2018 01:24 AM - shyouhei (Shyouhei Urabe)
Apology for the previous comment. It shows IO#ungetbyte example. I confused them because I fixed them the same day for the same reason. The same thing happens for String#setbyte.
We may want to define the behaviour of these methods without introducing fixnum / bignum distinction. One possible way is:

```c
rb_str_setbyte(VALUE str, VALUE index, VALUE value)
{
    long pos = NUM2LONG(index);
    int byte = NUM2INT(value);
    int byte = NUM2INT(rb_int_modulo(value, INT2FIX(256)));
    long len = RSTRING_LEN(str);
    char *head, *left = 0;
    unsigned char *ptr;
```
I had a chance this week to ask matz if he wants to allow bigger inputs or not for those methods. He answered yes. He prefers mod 256 behaviour for larger numbers. I will fix them again.

#9 - 01/15/2019 06:42 AM - shyouhei (Shyouhei Urabe)
- Status changed from Assigned to Closed

Applied in changeset trunk\r66824.

setbyte / ungetbyte allow out-of-range integers
- string.c: String#setbyte to accept arbitrary integers [Bug #15460]
- io.c: ditto for IO#ungetbyte
- ext/stringio/stringio.c: ditto for StringIO#ungetbyte

#10 - 01/17/2019 09:36 PM - naruse (Yui NARUSE)
- Backport changed from 2.4: DONTNEED, 2.5: DONTNEED, 2.6: REQUIRED to 2.4: DONTNEED, 2.5: DONTNEED, 2.6: DONE

ruby_2_6 r66845 merged revision(s) 66760,66761,66824.

#11 - 12/02/2019 12:41 AM - nobu (Nobuyoshi Nakada)
On the 2.6 branch, this causes a declaration-after-statement warning.

#12 - 12/03/2019 03:38 PM - mame (Yusuke Endoh)
- Related to Bug #10453: NUM2CHR() does not perform additional bounds checks added