Ruby - Bug #15460

Behaviour of String#setbyte changed

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

Status:	Closed		
Priority:	Normal		
Assignee:	shyouhei (Shyouhei Urabe)		
Target version:			
ruby -v:	ruby 2.6.0p0 (2018-12-25 revision 66547) [x86_64-linux]	**Backport:**	2.4: DONTNEED, 2.5: DONTNEED, 2.6: DONE

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 - Bug #10453: NUM2CHR() does not perform additional bounds ch...	**Rejected**
Related to Ruby - Feature #20594: A new String method to append bytes while p...	**Closed**

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/strringio/stringio.c: ditto for StringIO#ungetbyte

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

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/strringio/stringio.c: ditto for StringIO#ungetbyte
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_6@66845 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

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.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_6@66952 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 12/26/2018 12:38 AM - ko1 (Koichi Sasada)

FYI

```
7213568733f673da0d82f95e8a1bccf79ba3f0d3
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 5a5cbb576c31350a416b2121e5efb85aaffa0676 e55c59136a49fb7d2d70805a65a4c7f56519f2e9 M    string.c
:040000 040000 2170f3be4ac9d27e26ce361c38beb7874c07436f e71fc9f00ed5602a30e9f504fc015b9bf7b4dad1 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.

```
% ruby -v -e 'p "foo".setbyte(0,18446744073709551616)'
ruby 1.9.3p551 (2014-11-13) [x86_64-darwin15.6.0]
-e:1:in `setbyte': bignum too big to convert into `long' (RangeError)
        from -e:1:in `<main>'
```

#4 - 12/26/2018 01:44 AM - shyouhei (Shyouhei Urabe)

We may want to define the behaviour of these methods without introducing fixnum / bignum distinction. One possible way is:

```
Index: io.c
===================================================================
--- io.c (revision 66566)
+++ io.c (working copy)
@@ -4259,8 +4259,8 @@ rb_io_ungetbyte(VALUE io, VALUE b)
     GetOpenFile(io, fptr);
     rb_io_check_byte_readable(fptr);
     if (NIL_P(b)) return Qnil;
-    if (FIXNUM_P(b)) {
-        int i = FIX2INT(b);
+    if (RB_TYPE_P(b, T_FIXNUM) || RB_TYPE_P(b, T_BIGNUM)) {
+        int i = NUM2INT(rb_int_modulo(b, INT2FIX(256)));
         if (0 <= i && i <= UCHAR_MAX) {
             unsigned char cc = i & 0xFF;
             b = rb_str_new((const char *)&cc, 1);
Index: string.c
===================================================================
--- string.c (revision 66566)
+++ string.c (working copy)
@@ -5411,7 +5411,7 @@ static VALUE
 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;
```

#5 - 12/26/2018 11:59 AM - Eregon (Benoit Daloze)

I think it's good to be strict here, i.e., to raise RangeError when str.setbyte(256) or higher as that could very well be a bug in the calling code.
Negative values also sound buggy in most situations.

Why are values higher in HexaPDF? Does it mean multiple bytes need to be written or is it enough to throw away the high bits?
I suspect the latter is rarely correct.

#6 - 12/26/2018 05:47 PM - naruse (Yui NARUSE)

- Backport changed from 2.4: UNKNOWN, 2.5: UNKNOWN, 2.6: UNKNOWN to 2.4: DONTNEED, 2.5: DONTNEED, 2.6: REQUIRED

#7 - 12/28/2018 08:32 AM - gettalong (Thomas Leitner)

Eregon (Benoit Daloze) wrote:

> Why are values higher in HexaPDF? Does it mean multiple bytes need to be written or is it enough to throw away the high bits?
> I suspect the latter is rarely correct.

This actually happens in the predictor filter of HexaPDF (see https://github.com/gettalong/hexapdf/blob/master/lib/hexapdf/filter/predictor.rb#L166) which is an implementation of the PNG filter types (see https://www.w3.org/TR/2003/REC-PNG-20031110/#9Filter-types).

The PNG spec says that all operations are to be taken as unsigned arithmetic modulo 256 and since this is what String#setbyte originally did, I just left out the additional module operation.

It is not a huge problem for me and I will release a new HexaPDF version with the fix soon.

As for whether the modulo operation should be done by String#setbyte: Many methods automatically convert arguments to an expected type if possible, e.g. with #to_str. Converting an integer to a byte using modulo 256 seems like the most straight-forward way to do this for String#setbyte.

#8 - 01/11/2019 02:33 PM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Assigned

- Assignee set to shyouhei (Shyouhei Urabe)

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/strringio/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

#13 - 09/11/2024 09:36 AM - Eregon (Benoit Daloze)

- Related to Feature #20594: A new String method to append bytes while preserving encoding added