Ruby master - Bug #16497

StringIO#internal_encoding is broken (more severely in 2.7)

01/10/2020 11:18 AM - zverok (Victor Shepelev)

Status:	Assigned		
Priority:	Normal		
Assignee:	nobu (Nobuyoshi Nakada)		
Target version:			
ruby -v:		**Backport:**	2.5: DONTNEED, 2.6: DONTNEED, 2.7: REQUIRED

Description

To the best of my understanding from Encoding docs, the following is true:

- external encoding (explicitly specified or taken from Encoding.default_external) specifies how the IO understands input and stores it internally
- internal encoding (explicitly specified or taken from Encoding.default_internal) specifies how the IO converts what it reads.

Demonstration with regular files:

```
# prepare data
File.write('test.txt', 'Україна'.encode('KOI8-U'), encoding: 'KOI8-U') #=> 7

def test(io)
  str = io.read
  [io.external_encoding, io.internal_encoding, str, str.encoding]
end

# read it:
test(File.open('test.txt', 'r:KOI8-U'))
# => [#<Encoding:KOI8-U>, nil, "\xF5\xCB\xD2\xC1\xA7\xCE\xC1", #<Encoding:KOI8-U>]

# We can specify internal encoding when opening the file:
test(File.open('test.txt', 'r:KOI8-U:UTF-8'))
# => [#<Encoding:KOI8-U>, #<Encoding:UTF-8>, "Україна", #<Encoding:UTF-8>]

# ...or when it is already opened
test(File.open('test.txt').tap { |f| f.set_encoding('KOI8-U', 'UTF-8') })
# => [#<Encoding:KOI8-U>, #<Encoding:UTF-8>, "Україна", #<Encoding:UTF-8>]

# ...or with Encoding.default_internal
Encoding.default_internal = 'UTF-8'
test(File.open('test.txt', 'r:KOI8-U'))
# => [#<Encoding:KOI8-U>, #<Encoding:UTF-8>, "Україна", #<Encoding:UTF-8>]
```

But with StringIO, **internal encoding can't be set** in Ruby **2.6**:

```
require 'stringio'
Encoding.default_internal = nil
str = 'Україна'.encode('KOI8-U')

# Simplest form:
test(StringIO.new(str))
# => [#<Encoding:KOI8-U>, nil, "\xF5\xCB\xD2\xC1\xA7\xCE\xC1", #<Encoding:KOI8-U>]

# Try to set via mode
test(StringIO.new(str, 'r:KOI8-U:UTF-8'))
# => [#<Encoding:KOI8-U>, nil, "\xF5\xCB\xD2\xC1\xA7\xCE\xC1", #<Encoding:KOI8-U>]

# Try to set via set_encoding:
test(StringIO.new(str, 'r:KOI8-U:UTF-8').tap { |f| f.set_encoding('KOI8-U', 'UTF-8') })
# => [#<Encoding:KOI8-U>, nil, "\xF5\xCB\xD2\xC1\xA7\xCE\xC1", #<Encoding:KOI8-U>]
```

```
# Try to set via Enoding.default_internal:
Encoding.default_internal = 'UTF-8'
test(StringIO.new(str))
# => [#<Encoding:KOI8-U>, nil, "\xF5\xCB\xD2\xC1\xA7\xCE\xC1", #<Encoding:KOI8-U>]
```

So, in 2.6, any attempt to do something with StringIO's internal encoding are **just ignored**.

In **2.7**, though, matters became much worse:

```
require 'stringio'
Encoding.default_internal = nil
str = 'Україна'.encode('KOI8-U')

# Behaves same as 2.6
test(StringIO.new(str))
# => [#<Encoding:KOI8-U>, nil, "\xF5\xCB\xD2\xC1\xA7\xCE\xC1", #<Encoding:KOI8-U>]

# Try to set via mode: WEIRD behavior starts
test(StringIO.new(str, 'r:KOI8-U:UTF-8'))
# => [#<Encoding:UTF-8>, nil, "\xF5\xCB\xD2\xC1\xA7\xCE\xC1", #<Encoding:UTF-8>]

# Try to set via set_encoding: still just ignored
test(StringIO.new(str, 'r:KOI8-U:UTF-8').tap { |f| f.set_encoding('KOI8-U', 'UTF-8') })
# => [#<Encoding:KOI8-U>, nil, "\xF5\xCB\xD2\xC1\xA7\xCE\xC1", #<Encoding:KOI8-U>]

# Try to set via Enoding.default_internal: WEIRD behavior again
Encoding.default_internal = 'UTF-8'
test(StringIO.new(str))
# => [#<Encoding:UTF-8>, nil, "\xF5\xCB\xD2\xC1\xA7\xCE\xC1", #<Encoding:UTF-8>]
```

So, **2.7** not just ignores attempts to set **internal** encoding, but erroneously sets it to **external** one, so strings are not recoded, but their encoding is forced to change.

I believe it is severe bug (more severe than 2.6's "just ignoring").

This Reddit thread shows how it breaks existing code:

- the author uses StringIO to work with ASCII-8BIT strings;
- the code is performed in Rails environment (which sets internal_encoding to UTF-8 by default);
- under **2.7**, StringIO#read returns ASCII-8BIT content in Strings saying their encoding is UTF-8.

Associated revisions

Revision e257c08f - 03/15/2020 09:43 AM - byroot (Jean Boussier)

[ruby/stringio] StringIO#initialize default to the source string encoding

[Bug #16497]

https://github.com/ruby/stringio/commit/4958a5ccab

Revision 47b08728 - 03/15/2020 11:53 AM - byroot (Jean Boussier)

[ruby/stringio] StringIO#initialize default to the source string encoding

[Bug #16497]

https://github.com/ruby/stringio/commit/4958a5ccab
(cherry picked from commit e257c08f2ec27e2d66cdfa7e2415deb492522e22)

Revision 1ad9b231 - 03/15/2020 01:15 PM - nobu (Nobuyoshi Nakada)

Added guard against [Bug #16497]

History

#1 - 01/10/2020 01:18 PM - nobu (Nobuyoshi Nakada)

- Backport changed from 2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN to 2.5: DONTNEED, 2.6: DONTNEED, 2.7: REQUIRED

- Assignee set to nobu (Nobuyoshi Nakada)

- Status changed from Open to Assigned

#2 - 01/11/2020 03:16 AM - jrochkind (jonathan rochkind)

Note the StringIO is not transcoding. it is simply changing the encoding "tagging" of the String without changing any bytes.

If the string was ASCII-8BIT (ie BINARY), there is really *no way* to transcode. But here's an example where it *could* have transcoded, but did not -- it only changed the encoding tag, making the result invalid.

```
require 'stringio'
Encoding.default_internal = Encoding::UTF_8

str = "é".encode("WINDOWS-1252")
#=> "\xE9" # that is an é in WINDOWS-1252, no problem
sio = StringIO.new(str)
read_string = sio.read
read_string.encoding # => #<Encoding:UTF-8>
read_string.valid_encoding? # => false
read_string # => "\xE9"
```

The current behavior can't possibly be correct.

But I believe that the ruby 2.6.x behavior was actually correct/desirable/most useful, and should be returned. If a StringIO must support some kind of transcoding behavior, it should not be it's *default* behavior, by default it should do what it did in 2.6.0 -- ie, default to the encoding already set on the string passed in as argument to StringIO initializer.

If StringIO is to support these notions of encodings, I am unclear on what the difference is between the "internal" and "external" encoding for a StringIO -- it may be a bug where it is taking Encoding.default_internal and using it as an *external* encoding? Not sure. Or it may be wrongly thinking the *default* internal encoding should be applied to StringIO input -- it should not be, it is just a *default*, where the StringIO initializer argument, as it is already a ruby string (unlike "real" IO), it already *has* a known character encoding attached to it, no resort to Encoding.default_internal is required. Aha, I think this may in fact be the nature of the bug?

#3 - 01/12/2020 12:36 PM - zverok (Victor Shepelev)

A bit more discussion on fixing the behavior, after discussing on Reddit.
Basically, it is two ways to fix it:

1. Just return to 2.6 behavior
2. Fully implement internal_encoding for StringIO

I believe **2 (make it use internal_encoding) is more useful**. One example where current behavior is not enough is substituting StringIO (for example, in tests) in code where File is usually used. For example, Hunspell's spellchecker data files use SET <encoding> tag as their first line. So, the code reading this file, could look this way:

```
def read_aff(io)
  encoding = read_encoding(io)
  io.set_encoding(encoding, 'UTF-8')
# read in whatever encoding it has, recode to internal encoding of the software
  # now, I assume
  io.read_line
  # will be equivalent to
  <read_bytes_from_io>.force_encoding(encoding).encode('UTF-8')
  # ...which is true for File and false for StringIO
end
```

However, **option 1 (just make it behave like 2.6)** is more **backward-compatible**. Otherwise (full fix) this simple code will change behavior under Rails (which set default_internal to UTF-8):

```
str = 'foo'.force_encoding('Windows-1251')
io = StringIO.new(str)
io.read
```

In 2.6 it returns string in Windows-1251 encoding, and some code may rely on it. If we'll make StringIO respect default_internal, it will start to return recoded to UTF-8 string.

So, maybe this report should be **split into two:**

1. At least fix the bug (by returning to the previous behavior) in 2.7.1
2. Make StringIO respect internal_encoding in 2.8/3.0

WDYT?

#4 - 01/13/2020 05:33 PM - jrochkind (jonathan rochkind)

StringIO has been documented for a while to *ignore* it's own internal encoding, but respect it's own external encoding.

I am not sure this ever made any sense. It might have made more sense to respect an internal encoding, but ignore an external encoding. external encoding is normally used to assume what encoding IO bytes are, from for example a file system or network. But since StringIO starts with a ruby string that already has a tagged encoding, it is unlike other standard IO, and has no situation where it has to use an exteranl_encoding to assume a character encoding in the face of unknown encoding of IO bytes.

However, an internal encoding *could* be useful. The internal encoding is used to transcode a value to that internal encoding *after* reading.

I don't understand what was changed in ruby 2.7.0. I don't understand why the value of Encoding.default_internal effects StringIO operation. I believe whatever was changed was buggy, and not what was intended.

However, even if it were done *right*, it could still cause backwards compatibility problems. It's possible someone was trying to fix the fact that (in 2.6.x), a StringIO says it has an external_encoding of (by default) Encoding.default_external (which is often UTF8 by default), but may not actually do that. If it were "fixed" to do what 2.7.0 is doing based only on value of Encoding.default_external, not default_internal -- that might be "correct", but it would *still* be a backwards incompatibility problem. And I don't believe it would actually be a useful feature to anyone, as I don't think supporting external encoding on StringIO is useful.

I believe that this behavior should be reverted for 2.7.x, so as not to introduce a backwards incompat of unclear usefulness in 2.7.x.

I think for ruby 3.0.0, the intended/desired behavior of StringIO with regard to encoding should be thought through more carefully, to make sure what it's doing is useful, before introducing backwards breaking change.

#5 - 03/12/2020 01:03 PM - byroot (Jean Boussier)

I have a potential fix for this issue: https://github.com/ruby/ruby/pull/2960

#6 - 03/15/2020 09:44 AM - byroot (Jean Boussier)

- Status changed from Assigned to Closed

Applied in changeset git|e257c08f2ec27e2d66cdfa7e2415deb492522e22.

[ruby/stringio] StringIO#initialize default to the source string encoding

[Bug #16497]

https://github.com/ruby/stringio/commit/4958a5ccab

#7 - 03/15/2020 01:07 PM - naruse (Yui NARUSE)

- Backport changed from 2.5: DONTNEED, 2.6: DONTNEED, 2.7: REQUIRED to 2.5: DONTNEED, 2.6: DONTNEED, 2.7: DONE

ruby_2_7 47b08728cf3d0441a3da4dc1dcdd578817b0e036.

#8 - 03/15/2020 05:27 PM - zverok (Victor Shepelev)

naruse (Yui NARUSE) one of my two "weird" cases is not fixed yet:

```
def test(io)
  str = io.read
  [io.external_encoding, io.internal_encoding, str, str.encoding]
end

str = 'Укpаїна'.encode('KOI8-U')

test(StringIO.new(str, 'r:KOI8-U:UTF-8'))
# => [#<Encoding:UTF-8>, nil, "\xF5\xCB\xD2\xC1\xA7\xCE\xC1", #<Encoding:UTF-8>]
```

(Tried just now on the freshest master)

#9 - 03/16/2020 04:37 AM - nobu (Nobuyoshi Nakada)

- Backport changed from 2.5: DONTNEED, 2.6: DONTNEED, 2.7: DONE to 2.5: DONTNEED, 2.6: DONTNEED, 2.7: REQUIRED

- Status changed from Closed to Assigned