Ruby - Bug #18955

Kernel#sprintf - %c ignores a non-ASCII character's encoding

08/06/2022 11:25 AM - andrykonchin (Andrew Konchin)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	3.0.3	**Backport:**	2.7: UNKNOWN, 3.0: UNKNOWN, 3.1: UNKNOWN

Description

I haven't found any similar existing issue so decided to create a new one.

I noticed that sprintf("%c", string) doesn't handle (in an expected way) a case when encodings of format sequence and string argument aren't the same and the string argument contains non-ASCII character.

In this case it seems to me that sprintf just uses binary representation of a character and assigns (or interprets with) encoding of the format sequence string.

I would expect that sprintf negotiates encoding and converts everything (the character and the format string) to the chosen one. And raises error when negotiation fails.

Examples to illustrate this behavior:

```
format = "%c".encode("Windows-1251")
string = "Й".encode(Encoding::KOI8_U)
r = sprintf(format, string)
r.encoding
# => #<Encoding:Windows-1251>

r == "Й".encode("Windows-1251")
# => false

r.codepoints
# => [234]
string.codepoints
# => [234]
```

In this example the result's encoding is a format's encoding. But codepoint isn't changed and equals a codepoint of the character in the original string's encoding. But it should be different:

```
"Й".encode("Windows-1251").codepoints
# => [201]
```

Another example:

```
string = "À".encode(Encoding::CP1252)
sprintf("%c", string)
# => in `sprintf': invalid byte sequence in UTF-8 (ArgumentError)
```

In this example the error means that sprintf doesn't encode properly a codepoint (of string's encoding) in UTF-8. It uses just raw bytes.

Associated revisions

Revision ce384ef5a95b809f248e089c1608e60753dabe45 - 08/19/2022 06:57 PM - nobu (Nobuyoshi Nakada)

[Bug #18955] Check length of argument for %c in proper encoding

Revision 1ef49de83483e6f78bfe9c795a473ccfb29db150 - 08/19/2022 06:57 PM - nobu (Nobuyoshi Nakada)

[Bug #18955] format single character for %c

Revision ce384ef5a95b809f248e089c1608e60753dabe45 - 08/19/2022 06:57 PM - nobu (Nobuyoshi Nakada)

[Bug #18955] Check length of argument for %c in proper encoding

Revision 1ef49de83483e6f78bfe9c795a473ccfb29db150 - 08/19/2022 06:57 PM - nobu (Nobuyoshi Nakada)

[Bug #18955] format single character for %c

Revision ce384ef5 - 08/19/2022 06:57 PM - nobu (Nobuyoshi Nakada)

[Bug #18955] Check length of argument for %c in proper encoding

Revision 1ef49de8 - 08/19/2022 06:57 PM - nobu (Nobuyoshi Nakada)

[Bug #18955] format single character for %c

History

#1 - 08/08/2022 01:53 PM - andrykonchin (Andrew Konchin)

- Description updated

#2 - 08/08/2022 01:54 PM - andrykonchin (Andrew Konchin)

- Subject changed from Kernel#sprintf - %c doesn't convert non-ASCII characters to Kernel#sprintf - %c ignores a non-ASCII character's encoding

#3 - 08/08/2022 01:55 PM - andrykonchin (Andrew Konchin)

- Description updated

#4 - 08/08/2022 01:55 PM - andrykonchin (Andrew Konchin)

- Description updated

#5 - 08/08/2022 01:56 PM - andrykonchin (Andrew Konchin)

- Description updated

#6 - 08/12/2022 02:51 PM - andrykonchin (Andrew Konchin)

- Description updated

#7 - 08/12/2022 02:52 PM - andrykonchin (Andrew Konchin)

- Description updated

#8 - 08/16/2022 06:02 AM - nobu (Nobuyoshi Nakada)

A codepoint is expected for %c, then the former examples are currently expected behaviors, I think.

The latter example is a bug.

#9 - 08/18/2022 09:42 AM - mame (Yusuke Endoh)

At the dev-meeting, @akr (Akira Tanaka) proposed that the format %c behaves like %s (with the one-codepoint restriction) and @matz (Yukihiro Matsumoto) agreed with it.

#10 - 08/19/2022 08:30 PM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

Applied in changeset git|ce384ef5a95b809f248e089c1608e60753dabe45.

[Bug #18955] Check length of argument for %c in proper encoding