Ruby master - Bug #10584

String.valid_encoding?, String.ascii_only? fails to account for BOM.

12/10/2014 12:55 AM - geoff-codes (Geoff Nixon)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.2.0preview2 (2014-11-28 trunk 48628) [x86_64-darwin14]	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN

Description

IMO:

- A Unicode (UTF-16, UTF-32) string with a valid BOM should not be considered a valid encoding if endianness is changed.
- A UTF-8 string with BOM should not consider the BOM as a codepoint.

```
> file utf-16be-file
utf-16be-file: POSIX shell script, Big-endian UTF-16 Unicode text executable

> file utf-16le-file
utf-16le-file: POSIX shell script, Little-endian UTF-16 Unicode text executable

> file utf-8-with-bom-file
utf-8-with-bom-file: POSIX shell script, UTF-8 Unicode (with BOM) text executable

> ruby -e "p File.binread('utf-16le-file').force_encoding('UTF-16BE').valid_encoding?"
true # false

> ruby -e "p File.binread('utf-16be-file').force_encoding('UTF-16LE').valid_encoding?"
true # false

> ruby -e "p File.read('utf-8-with-bom-file').ascii_only?"
false # true

> ruby -e "p File.read('utf-8-with-bom-file')[0]"
"" # '#'
```

No?

History

#1 - 12/10/2014 11:21 AM - duerst (Martin Dürst)

This isn't as simple as you describe it. With respect to BOMs, there is a clear distinction between external data and internal data. A BOM is often very helpful in external data (e.g. a file). On the other hand, it's not only useless, but actually highly counterproductive for internal data (just think about concatenation).

The problem currently is that Ruby doesn't absorb that difference, it leaves it to the programmer. The reason for this is that it's difficult to define a clear external/internal boundary (the file example is the easy one). Also, some cases require a BOM (e.g. UTF-16 in XML) whereas others forbid it and others allow it and so on. It might be possible to deal with some of this as options on methods reading from files, but that would require careful analysis.

Because U+FFFE isn't a valid codepoint in Unicode, your first two examples could be made true, and might indeed catch some errors. For your third example, a string with a BOM is definitely not ASCII, so ascii_only? should definitely return false. This is not only the definition of ASCII, but also tightly linked to Ruby's internals (including optimizations).

For your forth example, once internal, it's unclear whether the BOM is actually a BOM or a zero-width non-breaking space. The later can appear at the start of a piece of text easily. Although explicitly deprecated, it's still effective, I just used it recently in a Web page.

#2 - 01/05/2018 09:01 PM - naruse (Yui NARUSE)

- Target version deleted (2.2.0)

Files

utf-8-with-bom-file	14 Bytes	12/10/2014	geoff-codes (Geoff Nixon)
utf-16le-file	2.46 KB	12/10/2014	geoff-codes (Geoff Nixon)
utf-16be-file	2.45 KB	12/10/2014	geoff-codes (Geoff Nixon)