Ruby master - Bug #16143

BOM UTF-8 is not removed after rewind

09/04/2019 08:28 AM - Dirk (Dirk Meier-Eickhoff)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.6.2p47 (2019-03-13 revision 67232) [x86_64-darwin17]	**Backport:**	2.5: UNKNOWN, 2.6: UNKNOWN

Description

I have a CSV file with "forced quotes" and UTF-8 BOM (\xEF\xBB\xBF) which CSV can not read after a rewind. I get "CSV::MalformedCSVError: Illegal quoting in line 1."

My UTF-8 CSV file with BOM:

```
File.open('bom_test.csv', 'w') do |io|
  io.write("\xEF\xBB\xBF\"Name\",\"City\"\n\"John Doe\",\"New York\"")
end
```

Reproduce error:

```
# Case 1
csv = CSV.open('bom_test.csv', 'r:BOM|UTF-8', {headers: true})
csv.shift
# => #<CSV::Row "Name":"John Doe" "City":"New York">
csv.rewind
csv.shift
# => CSV::MalformedCSVError (Illegal quoting in line 1.)

# Case 2
csv = CSV.open('bom_test.csv', 'r:BOM|UTF-8', {headers: true})
csv.readline
# => #<CSV::Row "Name":"John Doe" "City":"New York">
csv.rewind
csv.readline
# => CSV::MalformedCSVError (Illegal quoting in line 1.)
```

Sutou Kouhei has posted other reproducable code to my first issue at CSV gem: https://github.com/ruby/csv/issues/103

```
File.open("/tmp/a.txt", "w") do |x|
  x.puts("\xEF\xBB\xBFa,b,c")
end
File.open("/tmp/a.txt", "r:BOM|UTF-8") do |x|
  p x.gets.unpack("U*") # => [97, 44, 98, 44, 99, 10]
  x.rewind
  p x.gets.unpack("U*") # => [65279, 97, 44, 98, 44, 99, 10]
end
```

He said: "This [CSV] library rely on Ruby's BOM processing. It seems that Ruby's BOM processing doesn't support rewind."

My expectation is that reading a file with BOM always return the same content, regardless of first reading or after a rewind.

History

#1 - 09/05/2019 01:57 PM - nobu (Nobuyoshi Nakada)

I'm afraid if the spec of BOM is such simple and obvious.

Implemented but I'm sure that something is overlooked.
https://github.com/ruby/ruby/pull/2430

#2 - 10/10/2019 09:59 PM - kou (Kouhei Sutou)

I've reviewed the pull request. I found a problem.

#3 - 10/17/2019 06:30 AM - akr (Akira Tanaka)

I feel changing the default behavior of IO#rewind is dangerous.

We use IO#rewind when we modify a file in place.

```
open(filename, "r+") {|f|
  f.read
  f.rewind
  f.truncate(0)
  f.write "..."
}
```

If IO#rewind moves the file pointer to just after BOM,
BOM is changed to NULs in above code.

I think adding a keyword argument for IO#rewind is better for compatibility.