Ruby - Bug #14127

(CSV) generating UTF-16LE encoded file without BOM

11/22/2017 08:38 PM - laykou (Ladislav Gallay)

Status:	Rejected		
Priority:	Normal		
Assignee:	kou (Kouhei Sutou)		
Target version:			
ruby -v:	2.4.1	**Backport:**	2.3: UNKNOWN, 2.4: UNKNOWN

Description

This file should contain BOM information so that it is properly detected as UTF-16LE file.

How to generate such file:

```
file = CSV.generate(encoding: 'UTF-16LE') do |csv|
    csv << ['something', 'ľščťžýáié']
end
```

According to file -I file.csv this file is recognized as application/octet-stream; charset=binary because it is missing the BOM information.

According to Wikipedia https://en.wikipedia.org/wiki/UTF-16 it should contain "\xFF\xFE" on the beginning of the document so that everyone knows iths UTF-16LE.

Here is someone trying to fix this in the similiar way: https://stackoverflow.com/a/22950912/1632815 I did it: manually adding that BOM information.

```
## Adds BOM, albeit in a somewhat hacky way.
new_html_file = File.open(foo.txt, "w:UTF-8")
new_html_file << "\xFF\xFE".force_encoding('utf-16le') + some_text.force_encoding('utf-8').encode('utf-16le')
```

History

#1 - 11/23/2017 12:42 PM - nobu (Nobuyoshi Nakada)

laykou (Ladislav Gallay) wrote:

> This file should contain BOM information so that it is properly detected as UTF-16LE file.
>
> How to generate such file:
>
> ```
> file = CSV.generate(encoding: 'UTF-16LE') do |csv|
>     csv << ['something', 'ľščťžýáié']
> end
> ```

csv.rb seems having bugs in ASCII-incompatible encodings support.

> According to file -I file.csv this file is recognized as application/octet-stream; charset=binary because it is missing the BOM information.
>
> According to Wikipedia https://en.wikipedia.org/wiki/UTF-16 it should contain "\xFF\xFE" on the beginning of the document so that everyone knows iths UTF-16LE.

CSV.generate just builds a CSV string, doesn't create a file.
Writing the result to a file with BOM is an application's responsibility.

```
CSV.open("utf16.csv", "w:UTF-16LE:utf-8") do |csv|
  csv.to_io.write "\uFEFF"
  csv << ['something', 'ľščťžýáié']
end
```

> Here is someone trying to fix this in the similiar way: https://stackoverflow.com/a/22950912/1632815 I did it: manually adding that BOM information.

```
new_html_file = File.open("foo.txt", "w:UTF-16LE")
new_html_file << "\uFEFF" << some_text
```

#2 - 02/26/2018 11:16 AM - hsbt (Hiroshi SHIBATA)

- Status changed from Open to Assigned

- Assignee set to kou (Kouhei Sutou)

#3 - 02/27/2018 03:14 AM - kou (Kouhei Sutou)

- Status changed from Assigned to Rejected

nobu almost said.

You should write BOM by yourself when you use CSV.generate.

If you don't want to write BOM by yourself, you should use CSV.open(..., "w:UTF-16"):

```
CSV.open("utf16.csv", "w:UTF-16:utf-8") do |csv|
  csv << ['something', 'ľščťžýáié']
end
```

But it generates big-endian UTF-16.

#4 - 10/18/2018 03:11 PM - printercu (Max Melentiev)

WDYT about adding file_header option or something like this?

It's quite tricky to add this in streaming mode:

```
CSV.open(file, 'wb', encoding: 'utf-16le', headers: headers_row, write_headers: true) do |csv|
  bom_written = false
  for_each_row do |row|
    unless bom_written
      csv.to_io.write(BOM)
      bom_written = true
    end
    csv << row
  end
end
```

#5 - 10/19/2018 01:22 AM - kou (Kouhei Sutou)

Why do you need to use bom_written?

```
CSV.open(file, 'wb', encoding: 'utf-16le', headers: headers_row, write_headers: true) do |csv|
  csv.to_io.write(BOM)
  for_each_row do |row|
    csv << row
  end
end
```

#6 - 10/19/2018 06:58 AM - printercu (Max Melentiev)

It has different behaviour. In my example file is empty if csv.<< is never called, in suggested example it contains BOM anyway.