Ruby master - Bug #18899
Inconsistent argument handling in IO#set_encoding
07/06/2022 12:34 PM - javanthropus (Jeremy Bopp)

Status: Open
Priority: Normal
Assignee:
Target version:

<table>
<thead>
<tr>
<th>ruby -v:</th>
<th>Backport:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ruby 3.1.2p20 (2022-04-12 revision 4491bb740a) [x86_64-linux]</td>
<td>2.7: UNKNOWN, 3.0: UNKNOWN, 3.1: UNKNOWN</td>
</tr>
</tbody>
</table>

Description

IO#set_encoding behaves differently when processing a single String argument than it does when processing 2 arguments (whether Strings or Encodings) in the case where the external encoding is being set to binary and the internal encoding is being set to any other encoding.

This script demonstrates the resulting values of the external and internal encodings for an IO instance given different ways to equivalently call #set_encoding:

```ruby
#!/usr/bin/env ruby

def show(io, args)
    printf("args: %-50s external encoding: %-25s internal encoding: %-25s\n", args.inspect, io.external_encoding.inspect, io.internal_encoding.inspect

end

File.open('/dev/null') do |f|
    args = ['binary:utf-8']
    f.set_encoding(*args)
    show(f, args)

    args = ['binary', 'utf-8']
    f.set_encoding(*args)
    show(f, args)

    args = [Encoding.find('binary'), Encoding.find('utf-8')]
    f.set_encoding(*args)
    show(f, args)
end

This behavior is the same from Ruby 2.7.0 to 3.1.2.
```

History

#1 - 07/06/2022 01:53 PM - javanthropus (Jeremy Bopp)
- Description updated

#2 - 08/21/2022 02:24 PM - javanthropus (Jeremy Bopp)
Can anyone confirm if this is a bug or intended behavior? I've taken a look at the code that implements this, and there are 2 pretty independent code paths for handling the single string argument case and the multiple argument case. If this is confirmed to be a bug, I would like to write a patch to unify the behavior.

#3 - 08/23/2022 10:20 PM - jeremyevans0 (Jeremy Evans)
I think it is a bug. I submitted a pull request to fix it: [https://github.com/ruby/ruby/pull/6280](https://github.com/ruby/ruby/pull/6280). Not sure if the approach taken is the best way, though.