Ruby master - Feature #9713

__FILE__ return unexpected encoding - breaks Dir.glob

04/07/2014 05:47 PM - thomthom (Thomas Thomassen)

Status:	Assigned	
Priority:	Normal	
Assignee:	cruby-windows	
Target version:	3.0	

Description

C:/□□□/FILE.rb:

```
# encoding: UTF-8
puts "Encoding.find 'filesystem': #{Encoding.find('filesystem').inspect}"
puts "Encoding.find 'locale': #{Encoding.find('locale').inspect}"
puts "Encoding.default_internal: #{Encoding.default_internal.inspect}"
puts "Encoding.default_external: #{Encoding.default_external.inspect}"
puts "Encoding.locale_charmap: #{Encoding.locale_charmap.inspect}"
puts "__FILE__: #{__FILE__.encoding.inspect}"
puts "'foobar': #{'foobar'.encoding.inspect}"
```

C:/FILE.rb:

```
# encoding: UTF-8
puts "Encoding.find 'filesystem': #{Encoding.find('filesystem').inspect}"
puts "Encoding.find 'locale': #{Encoding.find('locale').inspect}"
puts "Encoding.default_internal: #{Encoding.default_internal.inspect}"
puts "Encoding.default_external: #{Encoding.default_external.inspect}"
puts "Encoding.locale_charmap: #{Encoding.locale_charmap.inspect}"
puts "__FILE__: #{__FILE__.encoding.inspect}"
puts "'foobar': #{'foobar'.encoding.inspect}"

puts ""
puts "Loading C:/□□□/FILE.rb ..."
require "C:/□□□/FILE.rb"
```

Results:

media-20140407.png

```
c:\ruby-220\usr\bin>ruby "C:\FILE.rb"
Encoding.find 'filesystem': #<Encoding:Windows-1252>
Encoding.find 'locale': #<Encoding:IBM437>
Encoding.default_internal: nil
Encoding.default_external: #<Encoding:IBM437>
Encoding.locale_charmap: "CP437"
__FILE__: #<Encoding:IBM437>
'foobar': #<Encoding:UTF-8>

Loading C:/???/FILE.rb ...
Encoding.find 'filesystem': #<Encoding:Windows-1252>
Encoding.find 'locale': #<Encoding:IBM437>
Encoding.default_internal: nil
Encoding.default_external: #<Encoding:IBM437>
Encoding.locale_charmap: "CP437"
__FILE__: #<Encoding:UTF-8>
'foobar': #<Encoding:UTF-8>

c:\ruby-220\usr\bin>
```

Now, lets see how this affects Dir.glob:

Test scenario - a folder structure like this:

```
C:/test/
```

```
C:/test/foo/
C:/test/□□□/
```

C:/FILE.rb

```
# encoding: UTF-8
puts "Encoding.find 'filesystem': #{Encoding.find('filesystem').inspect}"
puts "Encoding.find 'locale': #{Encoding.find('locale').inspect}"
puts "Encoding.default_internal: #{Encoding.default_internal.inspect}"
puts "Encoding.default_external: #{Encoding.default_external.inspect}"
puts "Encoding.locale_charmap: #{Encoding.locale_charmap.inspect}"
puts "__FILE__: #{__FILE__.encoding.inspect}"
puts "'foobar': #{'foobar'.encoding.inspect}"

puts ""
pattern = File.join(File.dirname(__FILE__), "test", "*")
puts "pattern.encoding: #{pattern.encoding.inspect}"
result = Dir.glob(pattern)
p result
p result.map { |file| file.encoding }

puts ""
puts "force encoding:"
pattern.force_encoding("UTF-8")
result = Dir.glob(pattern)
p result
p result.map { |file| file.encoding }
```

Result:

```
c:\ruby-220\usr\bin>ruby "C:\FILE.rb"
Encoding.find 'filesystem': #<Encoding:Windows-1252>
Encoding.find 'locale': #<Encoding:IBM437>
Encoding.default_internal: nil
Encoding.default_external: #<Encoding:IBM437>
Encoding.locale_charmap: "CP437"
__FILE__: #<Encoding:IBM437>
'foobar': #<Encoding:UTF-8>

pattern.encoding: #<Encoding:IBM437>
["C:/test/foo", "C:/test/???"]
[#<Encoding:IBM437>, #<Encoding:IBM437>]

force encoding:
["C:/test/foo", "C:/test/\u3066\u3059\u3068"]
[#<Encoding:UTF-8>, #<Encoding:UTF-8>]

c:\ruby-220\usr\bin>
```

Observe how when Dir.glob is fed a string based on **FILE** it will return strings in the same encoding, even though the string should include Unicode characters. The Unicode characters are replaced by question marks. (Actual ASCII bytes for question mark: 63) Just by forcing the input string to UTF-8 will make Dir.glob return the expected strings with correct Unicode characters.

I'm unsure of where the bug lies, but in terms of what I expected I would not have expected **FILE** to return different encoding depending on the executing file containing Unicode characters. All files have been marked as UTF-8 in the file header.

Associated revisions

Revision 2c12deaf - 04/09/2014 06:17 AM - nobu (Nobuyoshi Nakada)

encoding.c: fix rdoc of __FILE__

- encoding.c (rb_enc_default_internal): fix rdoc. __FILE__ is in filesystem encoding but not default_internal. [ruby-core:61894] [Bug #9713]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@45539 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 45539 - 04/09/2014 06:17 AM - nobu (Nobuyoshi Nakada)

encoding.c: fix rdoc of __FILE__

- encoding.c (rb_enc_default_internal): fix rdoc. __FILE__ is in filesystem encoding but not default_internal. [ruby-core:61894] [Bug #9713]

Revision 45539 - 04/09/2014 06:17 AM - nobu (Nobuyoshi Nakada)

encoding.c: fix rdoc of __FILE__

- encoding.c (rb_enc_default_internal): fix rdoc. __FILE__ is in filesystem encoding but not default_internal. [ruby-core:61894] [Bug #9713]

Revision 45539 - 04/09/2014 06:17 AM - nobu (Nobuyoshi Nakada)

encoding.c: fix rdoc of __FILE__

- encoding.c (rb_enc_default_internal): fix rdoc. __FILE__ is in filesystem encoding but not default_internal. [ruby-core:61894] [Bug #9713]

Revision 45539 - 04/09/2014 06:17 AM - nobu (Nobuyoshi Nakada)

encoding.c: fix rdoc of __FILE__

- encoding.c (rb_enc_default_internal): fix rdoc. __FILE__ is in filesystem encoding but not default_internal. [ruby-core:61894] [Bug #9713]

Revision 45539 - 04/09/2014 06:17 AM - nobu (Nobuyoshi Nakada)

encoding.c: fix rdoc of __FILE__

- encoding.c (rb_enc_default_internal): fix rdoc. __FILE__ is in filesystem encoding but not default_internal. [ruby-core:61894] [Bug #9713]

Revision 45539 - 04/09/2014 06:17 AM - nobu (Nobuyoshi Nakada)

encoding.c: fix rdoc of __FILE__

- encoding.c (rb_enc_default_internal): fix rdoc. __FILE__ is in filesystem encoding but not default_internal. [ruby-core:61894] [Bug #9713]

History

#1 - 04/07/2014 06:38 PM - usa (Usaku NAKAMURA)

The encoding of the results of Dir.glob are the same encoding with it's parameter.
So, there is no bug about the second case.

But, the first case, the encoding of **FILE** should be Windows-1252 (filesystem encoding)
or UTF-8 (script's encoding), I think.
It may be a bug.

#2 - 04/07/2014 07:46 PM - thomthom (Thomas Thomassen)

Usaku NAKAMURA wrote:

> But, the first case, the encoding of **FILE** should be Windows-1252 (filesystem encoding)
> or UTF-8 (script's encoding), I think.
> It may be a bug.

Seeing how the Windows file system can use Unicode characters I would expect **FILE** to be unicode encoded. Even if the file encoding was different. The file system doesn't store file names in Windows-1252 encoded data, that's just the fallback compatibility code page for programs that doesn't declare them selves as Unicode capable. Ruby doesn't do this - it doesn't seem to declare the UNICODE flag, but instead explicitly calls the *W variant of the file functions.

If I need to represent a file name in the UI some way, or write to file, in a different encoding then I can do the appropriate transposing. But I don't see any reason why Ruby's file related functions under Windows should yield any strings that are not Unicode.

#3 - 04/07/2014 08:30 PM - usa (Usaku NAKAMURA)

Thomas Thomassen wrote:

> If I need to represent a file name in the UI some way, or write to file, in a different encoding then I can do the appropriate transposing. But I don't see any reason why Ruby's file related functions under Windows should yield any strings that are not Unicode.

Because Ruby is 21 years old. She was born before Unicode spread.
When we began to introduce m17n features into Rubym there are many many l10n scripts existed.
In order to maintain compatibility with these scripts, we designed conservatively about the elements which are subject to the influence of encoding.
Specifications, such as Dir.glob, are performed by such judgment.

#4 - 04/07/2014 08:45 PM - thomthom (Thomas Thomassen)

Looking at how Ruby determines the filesystem encoding:

http://rxr.whitequark.org/mri/source/encoding.c#1267
static int enc_set_filesystem_encoding(void)

```
1266      char cp[sizeof(int) * 8 / 3 + 4];
1267      snprintf(cp, sizeof cp, "CP%d", AreFileApisANSI() ? GetACP() : GetOEMCP());
1268      idx = rb_enc_find_index(cp);
1269      if (idx < 0) idx = ENCINDEX_ASCII;
```

It's asking between OEM CP and ASCII CP - both of which are not Unicode. So Ruby will under Windows always try to return using ASCII or the OEM code page?
I can understand the desire for compatibility, but I'd wish for some better control - switches when you compile it so it was possible to set up Ruby under Windows where it wasn't necessary to juggle all these different encoding types.

For **FILE** to use UTF-8 selectively when it contains bytes outside of the filesystem CP seems very erratic. And as can be seen with the Dir.glob function it causes failure cascading further down the ruby scripts as some functions use inconsistent encoding.

#5 - 04/08/2014 12:15 PM - thomthom (Thomas Thomassen)

Referringthe docs (http://www.ruby-doc.org/core-2.1.1/Encoding.html#method-c-default_external-3D) to Encoding.default_internal=, __FILE__ should return strings according to the default internal - but when using -E to set it I don't see this behaviour.

#6 - 04/08/2014 01:45 PM - thomthom (Thomas Thomassen)

I'm starting to wonder if the three bugs I recently files could all go under one: that the behaviour described under Encoding.default_internal= isn't happening for all the elements it lists. The issues I'm experiencing appear to would have been working fine if encoding behaved as described for that function.

#7 - 04/09/2014 06:17 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

- % Done changed from 0 to 100

Applied in changeset r45539.

---

encoding.c: fix rdoc of __FILE__

- encoding.c (rb_enc_default_internal): fix rdoc. __FILE__ is in filesystem encoding but not default_internal. [ruby-core:61894] [Bug #9713]

#8 - 04/09/2014 09:03 AM - thomthom (Thomas Thomassen)

Nobuyoshi Nakada wrote:

> - encoding.c (rb_enc_default_internal): fix rdoc. __FILE__ is in filesystem encoding but not default_internal.

In my test __FILE__ is returned in the OEM encoding - not filesystem encoding.
And is it by design that __FILE__ will return a different encoding depending on it's content? And is there no way to configure it to return a consistent encoding?

#9 - 04/09/2014 09:07 AM - usa (Usaku NAKAMURA)

- Status changed from Closed to Assigned

Thomas Thomassen wrote:

> In my test __FILE__ is returned in the OEM encoding - not filesystem encoding.

So, reopened.

#10 - 12/04/2015 01:35 PM - thomthom (Thomas Thomassen)

Revisiting this issue again. Is there a resolution to what can be done to improve this and still satisfy compatibility concerns?

#11 - 12/04/2015 02:15 PM - usa (Usaku NAKAMURA)

What can I say now is that we are planning to use UTF-8 as filesystem encoding on Windows at Ruby 3.0.

#12 - 12/04/2015 02:31 PM - thomthom (Thomas Thomassen)

Usaku NAKAMURA wrote:

> What can I say now is that we are planning to use UTF-8 as filesystem encoding on Windows at Ruby 3.0.

That's very promising to hear. I'll keep an eye out for that.
Though, Ruby 3 is quite a bit away, isn't it? Anything that can be done to the v2 branch to mitigate issues? I'd be willing to offer my help.

#13 - 01/14/2020 09:11 AM - naruse (Yui NARUSE)

- Backport deleted (2.0.0: UNKNOWN, 2.1: UNKNOWN)

- ruby -v deleted (ruby 2.2.0dev (2014-04-07 trunk 45528) [i386-mswin32_100])

- Target version set to 3.0

- Tracker changed from Bug to Feature

Files

media-20140407.png	83.1 KB	04/07/2014	thomthom (Thomas Thomassen)