Ruby - Bug #14458

RubyVM::InstructionSequence compilation loses Regexp encoding

02/08/2018 02:40 PM - dannyfallon (Danny Fallon)

Status:	Rejected		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.4.3p205 (2017-12-14 revision 61247) [x86_64-darwin16]	**Backport:**	2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: UNKNOWN

Description

We appear to be losing encoding information for a Regexp object when we pass it through the compiler:

```
irb(main):001:0> "Test".encoding
=> #<Encoding:UTF-8>
irb(main):002:0> RubyVM::InstructionSequence.compile("'Test'.encoding").eval
=> #<Encoding:UTF-8>
irb(main):003:0> /\p{Alnum}/.encoding
=> #<Encoding:UTF-8>
irb(main):004:0> RubyVM::InstructionSequence.compile("/\p{Alnum}/.encoding").eval
=> #<Encoding:US-ASCII>
```

I think the encoding should be retained, much like it is for strings. Adding /u to the Regexp object
does retain the encoding but that feels like a burden we shouldn't have to bear?

```
irb(main):005:0> RubyVM::InstructionSequence.compile("/\p{Alnum}/u.encoding").eval
=> #<Encoding:UTF-8>
```

History

#1 - 02/08/2018 08:04 PM - shevegen (Robert A. Heiler)

Aha!

I remember several months ago having had some problems with encoding + regexes. I do not remember what was the ultimate cause, and it may be totally unrelated to what you described above; but I also wholeheartedly agree with what you wrote here:

> I think the encoding should be retained, much like it is for strings.

It should be the same encoding IMO, or perhaps even better, users should be able to set it and have full control over regexp encodings. I think Martin Durst (or with regular "umlauts", Martin Dürst), also wrote some somewhat related suggestion a few months ago or so.

> Adding /u to the Regexp object does retain the encoding but that feels like a burden we shouldn't have to bear?

I can not answer this because I know way too little about regexes, but I can also tell that it can be confusing/frustrating to try to hold all strings under a specific encoding, when there may still be unexpected encodings anywhere.

We have quite a lot of control over the encoding used in ruby projects, e. g. see method calls like File.read() with the :encoding key, and similar. I also suggested some time ago some easy way to enforce one encoding for a whole project, based on the "namespace" (such as the ruby hacker saying to ruby, "hello ruby - in namespace Foobar, I want only UTF-8 encoding and nothing else").

At any rate, sorry for digressing here - I am very much for all ways to have more control and more consistency when it comes to encoding in general, in ruby, so +1.

The above may be a bug, but I think the general "feature" is that a ruby hacker should be able to have a uniform encoding or at the least expect it. The above can be surprising when your project deals with Unicode, and suddenly some other encoding appears (like US-ASCII in the above).

#2 - 02/12/2018 06:36 AM - znz (Kazuhiro NISHIYAMA)

I think /p{Alnum}/ is US-ASCII only, so encoding is US-ASCII.

```
% irb -r irb/completion --simple-prompt
>> puts "/\p{Alnum}/.encoding"
/p{Alnum}/.encoding
=> nil
>> eval "/\p{Alnum}/.encoding"
=> #<Encoding:US-ASCII>
```

You can use "/\\p{Alnum}/.encoding" or '/\p{Alnum}/.encoding'.

#3 - 02/12/2018 03:17 PM - dannyfallon (Danny Fallon)

znz (Kazuhiro NISHIYAMA) wrote:

> I think /p{Alnum}/ is US-ASCII only, so encoding is US-ASCII.
>
> ```
> % irb -r irb/completion --simple-prompt
> >> puts "/\p{Alnum}/.encoding"
> /p{Alnum}/.encoding
> => nil
> >> eval "/\p{Alnum}/.encoding"
> => #<Encoding:US-ASCII>
> ```
>
> You can use "/\\p{Alnum}/.encoding" or '/\p{Alnum}/.encoding'.

Well, colour me embarrassed. Thank you for pointing out that subtlety in my example. I'm pretty sure we can close this one as invalid, sorry for wasting your time.

#4 - 02/13/2018 09:08 AM - duerst (Martin Dürst)

- Status changed from Open to Rejected