Ruby - Bug #1853

Cannot make constants using upper-case extended characters?

08/01/2009 05:00 PM - candlerb (Brian Candler)

Status:	Rejected		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 1.9.2dev (2009-07-18 trunk 24186) [i686-linux]	**Backport:**	

Description

=begin

```
SCHÖN = 1   # constant
=> 1
ÜBER = 2    # local variable!
=> 2
self.class.constants.grep(/SCH|BER/)
=> [:SCHÖN]
local_variables.grep(/SCH|BER/)
=> [:ÜBER]
```

I am not sure from the source code whether this is intentional or not.
In parse.c it uses rb_enc_isupper which understands encodings:

```
    else if (rb_enc_isupper(m[0], enc)) {
        id = ID_CONST;
    }
    else {
        id = ID_LOCAL;
```

This is in rb_intern3. And yet it is called from rb_intern2 which says:

```
ID
rb_intern2(const char *name, long len)
{
return rb_intern3(name, len, rb_usascii_encoding());
}
```

If this is intentional, it seems like an arbitrary restriction. Unicode characters are unambiguously classified into upper-case, lower-case and neither. If they can be used anywhere within an identifier, why not at the start of a constant?
=end

History

#1 - 08/01/2009 05:35 PM - runpaint (Run Paint Run Run)

=begin

> If this is intentional, it seems like an arbitrary restriction. Unicode characters are unambiguously classified into upper-case, lower-case and neither. If they can be used anywhere within an identifier, why not at the start of a constant?

I believe it is intentional as per the discussion in http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-core/19378 . As matz wrote in that thread, to change this behaviour would be to make the semantics of the program locale dependent. Plus, outside of regexes, Ruby doesn't really grok concepts like Unicode properties yet.

=end

#2 - 08/01/2009 05:50 PM - candlerb (Brian Candler)

=begin
Thank you for the link. (I'm in the process of documenting some of this).

We may be at cross-purposes over terminology, but when you say "to change this behaviour would be to make the semantics of the program locale dependent", do you mean that the behaviour of an individual Encoding may be locale-dependent? If so, that's new to me.

After all, when I put

Encoding: UTF-8

at the top of my source, then the source encoding is fixed at UTF-8. So what I'm asking is, does the behaviour of Encoding::UTF_8 (e.g. which characters are considered upper-case) vary at run-time dependent on the program's locale, in the sense of setlocale(3)? I'd like to see an example.

Until now, I was under the impression that each Encoding had fixed behaviour, and the locale-dependent behaviour arose from Encoding.default_external pointing to different Encodings based on the locale.

=end

#3 - 08/03/2009 05:10 PM - runpaint (Run Paint Run Run)

=begin

> Thank you for the link. (I'm in the process of documenting some of this).

Documentation is sorely needed in this area, so that sounds wonderful.
:-) (Look at http://blog.grayproductions.net/categories/character_encodings
and http://yokolet.blogspot.com/2009/07/design-and-implementation-of-ruby-m17n.html
if you haven't already).

> We may be at cross-purposes over terminology, but when you say "to change this behaviour would be to make the semantics of the program locale dependent", do you mean that the behaviour of an individual Encoding may be locale-dependent?

I believe all that matz meant is that the definition of constants
would depend on the script encoding, which may be set externally, i.e.
via -K, the shebang, or the system locale if the source comes from
STDIN or via -e. How much of a problem this would be practically, I'm
unsure.

-Run Paint Run Run

=end

#4 - 08/25/2009 03:15 PM - duerst (Martin Dürst)

=begin
Sorry to comment on a very old post, just back from vacation recently.

I think saying that program behavior depends on locale wasn't exactly
accurate. The problem is that for UTF-8 as a program encoding, there is
quite some information available about e.g. upper/lower case characters,
but this information isn't available for all other encodings, so
anything else than the current "upper case for Ruby identifiers means
ASCII upper case" could lead to nasty surprises.

Regards, Martin.

On 2009/08/01 17:50, Brian Candler wrote:

> Issue #1853 has been updated by Brian Candler.
>
> Thank you for the link. (I'm in the process of documenting some of this).
>
> We may be at cross-purposes over terminology, but when you say "to change this behaviour would be to make the semantics of the program locale dependent", do you mean that the behaviour of an individual Encoding may be locale-dependent? If so, that's new to me.
>
> After all, when I put
>
> # Encoding: UTF-8

at the top of my source, then the source encoding is fixed at UTF-8. So what I'm asking is, does the behaviour of Encoding::UTF_8 (e.g. which characters are considered upper-case) vary at run-time dependent on the program's locale, in the sense of setlocale(3)? If so, that's new to me, and I'd like to see an example.

Until now, I was under the impression that each Encoding had fixed behaviour, and the locale-dependent behaviour arose from Encoding.default_external pointing to different Encodings based on the locale.

---

http://redmine.ruby-lang.org/issues/show/1853

---

http://redmine.ruby-lang.org

--
#-# Martin J. Dürst, Professor, Aoyama Gakuin University
#-# http://www.sw.it.aoyama.ac.jp mailto:duerst@it.aoyama.ac.jp

=end

#5 - 11/04/2009 10:46 PM - naruse (Yui NARUSE)

- Category set to M17N

- Status changed from Open to Rejected

=begin
This is spec.
=end