Can't create valid Cyrillic-named class/module

Status: Closed  Priority: Normal  Assignee: matz (Yukihiro Matsumoto)

Description
Can't create a valid Cyrillic-named class. If I have a file мир.rb:

```ruby
# Content of мир.rb:
class Мир
  def приветствовать
    "Привет, Мир!"
  end
end
```

and do in bash:

```bash
ruby мир.rb
```

I get the error (SyntaxError):

```
мир.rb:1: class/module name must be CONSTANT
```

Same error with module creating and same behavior in irb (please, see attachment). At the same time Cyrillic-named constants and methods work fine.

It is expected that creating Cyrillic-named class/modules should work without error.

Related issues:
- Related to Ruby master - Bug #11859: Regexp matching with \p{Upper} and \p{Lo... Rejected
- Has duplicate Ruby master - Bug #15524: Unicode not Supported in Class Names Rejected

Associated revisions

Revision f852a0e - 04/10/2018 12:41 AM - nobu (Nobuyoshi Nakada)
symbol.c: non-ASCII constant names

- symbol.c (rb_sym_constant_char_p): support for non-ASCII constant names. [Feature #13770]
- object.c (rb_mod_const_get, rb_mod_const_defined): support for non-ASCII constant names.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63130 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 63130 - 04/10/2018 12:41 AM - nobu (Nobuyoshi Nakada)
symbol.c: non-ASCII constant names

- symbol.c (rb_sym_constant_char_p): support for non-ASCII constant names. [Feature #13770]
- object.c (rb_mod_const_get, rb_mod_const_defined): support for non-ASCII constant names.
symbol.c: non-ASCII constant names

- symbol.c (rb_sym_constant_char_p): support for non-ASCII constant names. [Feature #13770]
- object.c (rb_mod_const_get, rb_mod_const_defined): support for non-ASCII constant names.

History

#1 - 07/26/2017 12:46 PM - nobu (Nobuyoshi Nakada)
Constant names must start with an upper case in ASCII.

#2 - 07/26/2017 02:25 PM - matz (Yukihiro Matsumoto)
And maybe it's time to relax the limitation for Non-ASCII capital letters to start constant names.

Matz.

#3 - 07/26/2017 05:32 PM - shevegen (Robert A. Heiler)
Martin Dürst could then create classes for all Emojis in Unicode. :D

#4 - 07/27/2017 12:40 AM - shyouhei (Shyouhei Urabe)
matz (Yukihiro Matsumoto) wrote:

And maybe it's time to relax the limitation for Non-ASCII capital letters to start constant names.

What do you think of Titlecase? Are they allowed?

http://unicode.org/faq/casemap_charprop.html#4

#5 - 07/27/2017 06:19 AM - phluid61 (Matthew Kerwin)
shyouhei (Shyouhei Urabe) wrote:

matz (Yukihiro Matsumoto) wrote:

And maybe it's time to relax the limitation for Non-ASCII capital letters to start constant names.

What do you think of Titlecase? Are they allowed?

http://unicode.org/faq/casemap_charprop.html#4

Isn't titlecase a mapping property, rather than an attribute? That is, how a character would be converted to titlecase is orthogonal to whether it's uppercase.

#6 - 07/27/2017 06:27 AM - shyouhei (Shyouhei Urabe)
phluid61 (Matthew Kerwin) wrote:

shyouhei (Shyouhei Urabe) wrote:

matz (Yukihiro Matsumoto) wrote:

And maybe it's time to relax the limitation for Non-ASCII capital letters to start constant names.

What do you think of Titlecase? Are they allowed?

http://unicode.org/faq/casemap_charprop.html#4

Isn't titlecase a mapping property, rather than an attribute? That is, how a character would be converted to titlecase is orthogonal to whether it's uppercase.

Can I ask you whether U+01C8 is a valid Constant name or not in your opinion? and why?
shyouhei (Shyouhei Urabe) wrote:

Isn’t titlecase a mapping property, rather than an attribute? That is, how a character would be converted to titlecase is orthogonal to whether it’s uppercase.

Can I ask you whether U+01C8 is a valid Constant name or not in your opinion? and why?

Oh, you’re right, I had misread the documentation.

I think that if Ruby accepts all Lu characters as constants, it could also accept all Lt. In the case of U+01C8 I’m not overly concerned because it’s not common any more (but I think Ljudevit is just as valid as Ljudevit); however for U+01F2 it could be reasonable for someone in Macedonia to name a constant Dze, for example.

OK, I see. Thank you.

shevegen (Robert A. Heiler) wrote:

Well, it’s unclear whether emoji (note the Japanese plural!) are upper-case or lower-case.

I thought maybe we could make a distinction between children (lower-case) and adults (upper-case), but there are not many children, tons of adults, and tons of other stuff (not to say gunk).

Martin Dürst could then create classes for all Emojis in Unicode. :D

I agree. Here are some pointers for implementation:

The distinction between constants (tCONSTANT) and identifiers (tIDENTIFIER) is made at parse.c:7830 using macro ISUPPER. Some other uses of ISUPPER (but not all of them) seem to be related to this distinction, e.g. the one at symbol.c:281.

ISUPPER is defined using rb_isupper in include/ruby/ruby.h, the later being defined inline in the same file, as ‘A’ <= c && c <= ‘Z’. This would have to be replaced with a call to ONIGENC_IS_CODE_CTYPE or so, which would work for legacy encodings. For Unicode-based encodings, where we want to into account titlecase (thanks, Shyouhei!), it may be slightly more complicated.

A question we might want to check for is if there’s any code out there that currently uses non-ASCII upper-case variable names.

Another question is whether we might want to have some convention for Japanese, e.g. Katakana for class names. Just thinking out loud (and ducking).

To distinguish non-ASCII upper/lower cases would lead non-ASCII punctuations too.

ASCII punctuations cannot be a part of identifiers, will non-ASCII versions be same?

BTW, I think Japanese has no or little concept of plural, except that some words imply "many" and some suffixes.

I’m uncertain about the usage of mbc_case_fold.

diff --git i/parse.y w/parse.y
index 02d9412a2c..96f25d893e 100644
--- i/parse.y
+++ w/parse.y
@@ -7790,6 +7790,8 @@
 return result;
 }
+int rb_enc_const_id_char_p(const char *name, const char *end, rb_encoding *enc);
static enum yytokentype
parse_ident(struct parser_params *parser, int c, int cmd_state)
{
@@ -7827,7 +7829,9 @@ parse_ident(struct parser_params *parser, int c, int cmd_state)
pushback(c);
}
- if (result == 0 && ISUPPER(tok()[0]))
+ if (result == 0 &&
+ ISUPPER(tok()[0]) ||
+ rb_enc_const_id_char_p(tok(), tok()+toklen(), current_enc)) {
+ return tCONSTANT;
+ }
else {
    diff --git i/symbol.c w/symbol.c
    index f4516ebbe4..490cae0127 100644
    --- i/symbol.c
    +++ w/symbol.c
@@ -198,6 +198,28 @@ rb_enc_symname_p(const char *name, rb_encoding *enc)
           return rb_enc_symname2_p(name, strlen(name), enc);
 }
 +int
+rb_enc_const_id_char_p(const char *name, const char *end, rb_encoding *enc)
+{
+    + int c, len;
+    + if (end <= name) return FALSE;
+    + if (ISASCII(*name)) return ISUPPER(*name);
+    + c = rb_enc_codepoint_len(name, end, &len, enc);
+    + if (c < 0) return FALSE;
+    + if (rb_enc_isupper(c, enc)) return TRUE;
+    +
+    + OnigUChar fold[ONIGENC_GET_CASE_FOLD_CODES_MAX_NUM];
+    + const OnigUChar *beg = (const OnigUChar *)name;
+    + int r = enc->mbc_case_fold(ONIGENC_CASE_FOLD, 
+    +     &beg, (const OnigUChar *)end,
+    +     fold, enc);
+    + if (r > 0 && (r != len || memcmp(fold, name, r)))
+        return TRUE;
+    +
+    + return FALSE;
+}
+
+#define IDSET_ATTRSET_FOR_SYNTAX ((1U<<ID_LOCAL)|(1U<<ID_CONST))
+#define IDSET_ATTRSET_FOR_INTERN (~(1<<ID_SCOPE_SHIFT)) & ~(1U<<ID_ATTRSET))
@@ -278,7 +300,7 @@ rb_enc_symname_type(const char *name, long len, rb_encoding *enc, unsigned int a
    break;
    default:
        - type = ISUPPER(*m) ? ID_CONST : ID_LOCAL;
- type = rb_enc_const_id_char_p(m, e, enc) ? ID_CONST : ID_LOCAL;
    id:
        if (m >= e || (*m != '_' && !ISALPHA(*m) && ISASCII(*m))) {
            if (len > 1 && *(e-1) == '=') {
#13 - 08/01/2017 11:21 AM - nobu (Nobuyoshi Nakada)
- Backport deleted (2.2: UNKNOWN, 2.3: UNKNOWN, 2.4: UNKNOWN)
- ruby -v deleted (2.4.1p111 (2017-03-22 revision 58053) [x86_64-linux])
- Tracker changed from Bug to Feature
#14 - 08/18/2017 12:41 PM - nobu (Nobuyoshi Nakada)
- Assignee set to matz (Yukihiro Matsumoto)
- Status changed from Open to Assigned
#15 - 08/31/2017 09:06 AM - duerst (Martin Dürst)
In the patch, I suggest adding something like
          if (rb_enc_islower(c, enc)) return FALSE;
if (rb_enc_isupper(c, enc)) return TRUE;

#16 - 09/06/2017 09:45 AM - duerst (Martin Dürst)
I have checked for upper-case letters without corresponding lower-case letters, with the following short script:

```
ruby -n -e 'l=$_.split(/;/); if l[2]="Lu" && l[13]=="" then puts l[1];end' <UnicodeData.txt
```

Somewhat contrary to my expectations, this turned up quite a number of characters (471 of them). Most are MATHEMATICAL symbols in the range U+1D400 to U+1D7FF. My understanding is that they don't have mappings because mathematicians use upper-case and lower-case symbols with different meanings.

There are some other upper-case characters without defined lower-case equivalents, but most of the correspond to empty slots in the MATHEMATICAL symbols charts.

The above patch would treat all identifiers starting with upper-case, even MATHEMATICAL symbols, as class names. Unless we want to forbid such characters in identifiers, I think that's the right thing to do.

What's more important for the above patch is that there are no title-case characters without lower-case mappings, so

```ruby
+ if (r > 0 && (r != len || memcmp(fold, name, r)))
+     return TRUE;
+
```
in the patch will work correctly.

What is more important for the above patch is whether

#17 - 09/06/2017 09:54 AM - duerst (Martin Dürst)
sb (Sergey Borodanov) wrote:

```
Same error with module creating and same behavior in irb (please, see attachment). At the same time Cyrillic-named constants and methods work fine.
```

Methods indeed should work fine, because currently all non-ASCII characters are lumped together as lower-case. But I don't think constants work fine; it may only look so.

Please try e.g.

```
ruby -e 'Мир = 55; Мир = 77'
```

You should get a warning saying the the constant was already initialized. I don't get such a warning, which means that Мир here is treated as a variable, not as a constant.

#18 - 09/06/2017 10:02 AM - duerst (Martin Dürst)
- Related to Bug #11859: Regexp matching with \p{Upper} and \p{Lower} for EUC-JP doesn't work. added

#19 - 09/06/2017 10:04 AM - duerst (Martin Dürst)
As mentioned at the last committers' meeting, I think the patch will not work e.g. for upper-case characters in three-byte EUC-JP (characters from JIS X 0212) because the necessary data isn't there (see #11859).

#20 - 09/07/2017 12:12 PM - nobu (Nobuyoshi Nakada)
duerst (Martin Dürst) wrote:

```
In the patch, I suggest adding something like

if (rb_enc_islower(c, enc)) return FALSE;

immediately before or after

if (rb_enc_isupper(c, enc)) return TRUE;
```

I changed these code as followings:

```ruby
if (rb_enc_isalpha(c, enc)) { /* non-lower case alphabet should be upper/title case */
  if (!rb_enc_islower(c, enc)) return TRUE;
```
The whole patch is [https://github.com/nobu/ruby/tree/feature/13770-nonasci-const-name](https://github.com/nobu/ruby/tree/feature/13770-nonasci-const-name)

- Status changed from Assigned to Closed

Applied in changeset trunk|r63130.

symbol.c: non-ASCII constant names

- symbol.c (rb_sym_constant_char_p): support for non-ASCII constant names. [Feature #13770]

- object.c (rb_mod_const_get, rb_mod_const_defined): support for non-ASCII constant names.

- Has duplicate Bug #15524: Unicode not Supported in Class Names added

**Files**

| Screenshot from 2017-07-26 19-08-14.png | 64.2 KB | 07/26/2017 | sb (Sergey Borodanov) |