Ruby 初級者向けレッスン第 26 回

okkez @ Ruby 関西

2008 年 01 月 31 日

今回の内容

- Ruby1.9.1 の NEWS を読む
- Ruby1.9.1 を使用するにあたって注意すること

今回のゴール

- Ruby 1.9.1 での変更点を知る
- Ruby 1.9.1 で動くコードを書く

Ruby1.9.1 の NEWS を読む

互換性

言語コア

新しい文法とセマンティクス

- ブロック引数は常にブロックローカルになりました。

例:

```
01:
02: a = :a
03: [1, 2, 3].map{|a| a * 2 }
04: a # => 3   # 1.8.7
05:   # => :a  # 1.9.1
```

- defined? and local variables

例:

```
01:
02: a = 0
03: defined? a      # => "local-variable"
04: 1.times do |i|
05:   defined? i    # => "local-variable(in-block)" # 1.8.7
06:                 # => "local-variable"           # 1.9.1
07: end
08: defined? i      # => nil
```

ブロック内部のブロック変数に対する結果が変わっています。

- パーサにソースコードの正しい文字列エンコーディングを伝えるために magic comments を使用する必要があります。

例:

```
01: # -*- coding: utf-8 -*-
02: # vim:set fileencoding=UTF-8:
03: # encoding:UTF-8
04:
05: # 上記のいずれかが一行目に必要です。他にも書き方はあります。
06:
07: puts ’こんにちは’
```

- Object#instance_eval, Module#module_eval 内での定数の定義に関する動作が変わりました。

例:

```
01: # -*- coding: utf-8 -*-
02:
03: class A
04: end
05:
06: a = A.new
07: a.instance_eval{ B = ’constant B’ }
08: A.module_eval{ C = ’constant C’ }
09:
10: B # => "constant B"                          # 1.8.7
11:   # => uninitialized constant B (NameError)  # 1.9.1
12: C # => "constant C"                          # 1.8.7
13:   # => uninitialized constant C (NameError)  # 1.9.1
14:
15: (class << a ; self end)::B # => "constant B" # 1.9.1
16: A::C                       # => "constant C" # 1.9.1
17:
18: a.instance_eval{ ::D = ’constant D’ }
```

```
19: A.module_eval{ ::E = 'constant E' }
20:
21: D # => "constant D" # 共通
22: E # => "constant E" # 共通
```

組み込みクラスとオブジェクト

Kernel and Object

- Kernel#methods などが文字列の配列ではなくシンボルの配列を返すようになりました。
- 他には {global,local,instance,class}_variables, constants もシンボルを返します

Class and Module

- Module#attr が Module#attr_reader の alias になりました。
  - まだなってません。
- Module#instance_methods, #private_instance_methods, #public_instance_methods が文字列の配列ではなくシンボルの配列を返すようになりました。
- Extra subclassing check when binding UnboundMethods

例:

```
01: class Foo; def foo; end end
02: module Bar
03:   define_method(:foo, Foo.instance_method(:foo))
04: end # => ERROR: (eval):3:in `define_method'
05:     #    : bind argument must be a subclass of Foo
```

Exceptions

- 例外クラスを比較する場合、同じクラスに所属し、同じメッセージとバックトレースを持つ場合に真を返すようになりました。

例:

```
01:
02: begin
03:   raise 'm1'
04: rescue => e1
05:   begin
06:     raise 'm2'
07:   rescue => e2
08:     p e1 == e2 # => false
```

```
09:   end
10: end
11:
12: begin
13:   raise RuntimeError
14: rescue => e1
15:   begin
16:     raise RuntimeError
17:   rescue => e2
18:     p e1 == e2 # => false
19:   end
20: end
```

- SystemStackError のスーパークラスが StandardError ではなく Exception になりました。
- SecurityError も同様です。
- Removed Exception#to_str [Ruby2]

Enumerable and Enumerator

- Enumerable::Enumerator が削除されました。
- Enumerable#map, #collect_all がブロックなしで呼び出された場合、Enumerator のインスタンスを返すようになりました。
- 組み込みライブラリや標準添付ライブラリに含まれる多くのメソッドがブロックなしで呼び出された場合、Enumerator のインスタンスを返すようになりました。

Array

- Array#nitems は削除されました。

例:

```
01:
02: arr = [1, 2, nil, 3, 4, nil, 5]
03:
04: if arr.respond_to?(:nitems)
05:   p arr.nitems
06: else
07:   p arr.count{|e| !e.nil? }
08: end
09:
```

- Array#choice は削除されました。Array#sample を使用してください。
- Array#[m, n] = nil の動作が変わりました。

例:

```
01:
02: arr = [1, 2, 3, 4, 5]
03:
04: arr[1, 3] = nil
05:
06: p arr # => [1, 5]      # 1.8.7
07:       # => [1, nil, 5] # 1.9.1
```

Hash

- Hash#to_s が Hash#inspect と同じになりました。
- Hash#each, Hash#each_pair の動作が変わりました。

例:

```
01:
02: h = { :a => 1, :b => 2, :c => 3 }
03:
04: h.each{|k, v| p [k, v] }
05: h.each{|v| p v }
06:
07: h.each_pair{|k, v| p [k, v] }
08: h.each_pair{|v| p v }
09:
```

- Hash#select が Hash を返すようになりました。

例:

```
01:
02: hash = { :a => 1, :b => 2, :c => 3 }
03:
04: p hash.select{|k, v| v % 2 == 1 }
05: # => [[:c, 3], [:a, 1]] # 1.8.7
06: # => {:a=>1, :c=>3}     # 1.9.1
```

- Hash の順序が保存されるようになりました。キーが挿入された順に列挙されます。
- ENV, *DBM のような Hash に似たインターフェイスを持つライブラリにもこれらの変更は適用されています。

IO operations

- バイトを意識していた多くのメソッドが文字を意識するようになりました。
- IO#getc は数値ではなく文字を返すようになりました。
- Non-Blocking IO
- Kernel#open は第二引数のオープンモードに "t" を取るようになりました。
- Kernel#open はエンコーディングを指定できるようになりました。
- IO#initialize は IO である引数を一つ受けとるようになりました。
- IO は指定された場合は、文字コードを自動的に変換します。

例:

```
01:
02: File.open('euc.txt', 'r:euc-jp:utf-8'){|file|
03:   file.lines.each{|line|
04:     puts line
05:   }
06: }
```

- StringIO#readpartial が追加されました。

例:

```
01: require 'stringio'
02:
03: s = StringIO.new('foobarbaz')
04: p s.readpartial(2) # => "fo"
05: p s.readpartial(2) # => "oz"
```

- IO.try_onvert, IO.binread, IO.copy_stream, IO#binmode?,
- IO#colose_on_exec=, IO#close_on_exec?
- Limit input in IO#gets, IO#readline, IO#readlines, IO#each_line, IO#lines, IO.foreach, IO.readlines, StringIO#gets, StringIO#readline, StringIO#each, StringIO#readlines

例:

```
01:
02: print("> ")
03: p STDIN.gets(1)
```

- IO#ungetc, StringIO#ungetc
- IO#ungetbyte, StringIO#ungetbyte

- IO#internal_encoding, IO#external_encoding, IO#set_encoding
- IO.pipe takes encoding option
- Directive %u behaves like %d for negative values in printf-style formatting.

例:

```
01: printf("%d %u %d, %u\n", 1, 2, -3 , -4)
```

File and Dir operations

- 以下のメソッドで必要があれば #to_path が呼ばれます。
  - File.path, File.chmod, File.lchmod, File.chown, File.lchown,
  - File.utime, File.unlink, etc..

例:

```
01: s = Struct.new(:path){
02:   def to_path
03:     path
04:   end
05: }.new("foo")
06:
07: File.path(s) # => "foo"
```

- Dir.[], Dir.glob
  - '[' is no longer considered a normal char (ruby-dev:23291)
  - handling of escaped '{', '}' and ',' (ruby-dev:23376)

例:

```
01: File.open("/tmp/[", "w"){|f| f.puts "hi"}
02: Dir["/tmp/["] # => []
03:
04: File.open("/tmp/,", "w"){|f| f.puts "hi"}
05: Dir.glob('/tmp/{\,}') # => ["/tmp/,"]  instead of ["/tmp/", "/tmp/"]
06:
```

以下のメソッドが追加されました。

- File.world_readable?, File.world_writable?
- Dir.exist?, Dir.exists?

File::Stat

以下のメソッドが追加されました。

- File::Stat#world_readable?
- File::Stat#world_writable?

String and Regexp

- String は Enumerable を include しなくなりました。
- ?c が文字を返すようになりました。
- "One-char-wide" semantics for String#[] and String#[]=

例:

```
01: # -*- coding: utf-8 -*-
02:
03:
04: "a"[0]                # => 'a'
05: foo = "foo"
06: foo[0] = ?b
07: foo                   # => 'boo'
08:
09: hoge = 'ほげほげ'
10: hoge[0] = 'も'
11: p hoge                # => 'もげほげ'
```

- 多くのメソッドでバイト単位ではなく文字単位を意識するようになりました。
- Encoding-awareness
- Regexp は互換性のあるエンコーディング間でしかマッチしません。
- Regexp#kcode は削除されました。Regexp#encoding を使用してください。

Integer

- Integer(nil) raises TypeError

Fixnum

- Fixnum#id2name は削除されました。
- Fixnum#to_sym は削除されました。

Struct

- Struct#inspect の表示がシンプルになりました。

例:

```
$ forall-ruby -e 'p Struct.new(:a).new'
ruby 1.8.7 (2009-01-23 revision 21750) [i686-linux]
#<struct #<Class:0xb7d98a20> a=nil>
ruby 1.9.1p5000 (2009-01-24 trunk 21752) [i686-linux]
#<struct a=nil>
```

Time

- New format in Time#to_s

例:

```
$ forall-ruby -e 'p Time.new.to_s'
ruby 1.8.7 (2009-01-23 revision 21750) [i686-linux]
"Sun Jan 25 22:06:12 +0900 2009"
ruby 1.9.1p5000 (2009-01-24 trunk 21752) [i686-linux]
"2009-01-25 22:06:12 +0900"
```

- Timezone information preserved on Marshal.dump/load

$SAFE and bound methods

- New trusted/untrusted model in addition to tainted/untainted model.

Deprecation

- Kernel#to_a は削除されました。
- Kernel#getc, #gsub, #sub は削除されました。
- Kernel#callcc と Continuation は 'continuation' ライブラリになりました。
- Object#type は削除されました。
- Array#indices, Array#indexes, Hash#indices, Hash#indexes は削除されました。
- Hash#index は非推奨になりました。Hash#key を使用してください。
- ENV.index は非推奨になりました。ENV.key を使用してください。
- Process::Status#to_int は削除されました。
- Numeric#rdiv ???
- Precision は削除されました。でも泣かないで再設計されて戻ってくるから！

- Symbol#to_int, Symbol#to_i は削除されました。
- $KCODE は影響を持たなくなりました。Encoding に関連する機能やクラスを使用してください。
- VERSION とそれに類する定数が削除されました。
- $= (ignore case) は使用できなくなりました。

標準添付ライブラリ

Pathname

- Pathname#to_str, Pathname#=~ は削除されました。

time and date

- Time.parse, Date.parse はスラッシュ区切りの数値を ”dd/mm/yyyy” と解釈します。以前は、”mm/dd/yyyy” と解釈していました。

Readline

- If Readline uses libedit, Readline::HISTORY[0] returns the first of the history.

Continuation

- as above

Deprecation

- Complex#image は削除されました。Complex#imag, Complex#imaginary を使用してください。
- All SSL-related class methods in Net::SMTP ???
- Prime#cache, Prime#primes, Prime#primes_so_far は削除されました。
- mailread library は削除されました。 tmail gem を使用してください。
- cgi-lib library は削除されました。 cgi ライブラリを使用してください。
- date2 library は削除されました。 date ライブラリを使用してください。
- eregex library は削除されました。
- finalize library は削除されました。本当に必要な場合は ObjectSpace.define_finalizer を使用してください。
- ftools library は削除されました。fileutils を使用してください。
- generator library は削除されました。Enumerator クラスを使用してください。

- importenv library, Env library は削除されました。
- jcode library は削除されました。文字列の m17n 機能を使用してください。
- parsedate library 削除されました。
- ping library は削除されました。
- readbytes library は削除されました。
- getopts library, parsearg library は削除されました。optparse か getoptlong を使用してください。
- soap, wsdl and xsd libraries は削除されました。 soap4r gem を使用してください。
- Win32API library は削除されました。 dl ライブラリを使用してください。
- dl library は再実装されて API が変更されました。新しいバージョンの dl か ffi gem を使用してください。
- rubyunit library and runit library は削除されました。minitest, test/unit か好きな Gem を使用してください。
- test/unit は minitest として再実装されました。minitest は test/unit とは完全に互換というわけではありません。

言語コアの変更点

New syntax and semantics

- Magic comments to declare in which encoding your source code is written
- Hash の新しいリテラルとハッシュ形式のメソッド引数の新しい書き方が追加されました。

例:

```
01: h = { a:1, b:2, c:3 } # => { :a => 1, :b => 2, :c => 3 }
02:
03: def m(h)
04:   p h
05: end
06:
07: m(h)
08: m(a:1, b:2, c:3)
```

- lambda の新しい書き方が追加されました

例:

```
01: p  -> { }.call # => nil
02:
03: p -> a, b  { a + b }.call(1,2) # => 3
04:
05: c = 1
06: ->(a, b; c){ c = a + b }.call(1,2)
07: p c # => 1
08:
```

- .() and calling Procs without #call/#[]

例:

```
01: p lambda{|a, b| a + b }.(1, 2)
```

- ブロック引数の扱いが変わりました。
- ブロックローカルの変数を定義出来るようになりました。

例:

```
01: def m
02:   yield 1, 2
03: end
04: m{|v| p v }  # => [1,2] # 1.8.7
05: # => 1     # 1.9.1
06:
07: (1..10).map{|a;tmp|
08:   tmp = a.to_s
09:   [a, tmp]
10: }
11: p defined? tmp
12:
13:
```

- オプション引数の後に必須引数を許可するようになりました。Post Argument

例:

```
01:  def m(a, b=nil, *c, d)
02:    [a,b,c,d]
03:  end
04:  m(1,2)  # => [1, nil, [], 2]
```

- Multiple splats allowed

例:

```
01:
02: a = *[1,2,3], *[4,5,6]
03: p a # => [1,2,3,4,5,6]
```

- #[] can take splatted arguments, hash style arguments and a block.
- New directives in printf-style formatted strings (%).
- Newlines allowed before ternary colon

例:

```
01: # -*- coding: utf-8 -*-
02:
03: rand(100) % 2 == 0 ? 'even' : 'odd'
04: # 以下のように書けるようになった
05: rand(100) % 2 == 0 ? 'even'
06:                    : 'odd'
```

- Encoding.default_external and default_internal

ライブラリの変更点

builtin classes and objects

Kernel and Object

- BasicObject が追加されました。
- Object#=~ はデフォルトで false ではなく nil を返すようになりました。
- Kernel#define_singleton_method が追加されました。
- Kernel#load はデフォルトでもっともバージョン番号の大きい gem をロードします。

Class and Module

- Module#const_defined?, #const_get and #method_defined? がオプションの引数を受けとるようになりました。

例:

```
01:
02: module A
03:   AA = :AA
04:   def foo
05:   end
06: end
07:
```

```
08: module B
09:   include A
10: end
11:
12: p B.const_defined?(:AA)
13: p B.const_defined?(:AA, false)
14: p B.const_get(:AA)
15: # p B.const_get(:AA, false) # raise
16: p B.method_defined?(:foo)
17: # p B.method_defined?(:foo, false) # raise
```

- Module#class_variable_{set,get} は public になりました。
- Class of singleton classes
  - superclass が変わったっぽい

例:

```
class X;end; x=X.new; class << x; self < X; end
```

Errno::EXXX

- 全ての Errno::EXXX が定義されるようになりました。プラットフォームで実装されていないものは Errno::NOERROR のエイリアスになります。

Binding#eval

例:

```
a = 1
binding.eval("a") # => 1
```

Blocks and Procs

- Arity of blocks without arguments
- proc is now a synonym of Proc.new
- Proc#yield (Proc#call の別名)

例:

```
01: # Invokes the block, setting the block's parameters to the values in
02: # params in the same manner the yield statement does.
03:
04: a_proc = Proc.new {|a, *b| b.collect {|i| i*a }}
05: p a_proc.yield(9, 1, 2, 3)   # => [9, 18, 27]
06: p a_proc.yield([9, 1, 2, 3]) # => [9, 18, 27]
```

```
07: p a_proc.call(9, 1, 2, 3)    # => [9, 18, 27]
08: p a_proc.call([9, 1, 2, 3])  # => [9, 18, 27]
09: a_proc = Proc.new {|a,b| a }
10: p a_proc.yield(1,2,3)        # => [1]
```

- Passing blocks to #[]
- Proc#lambda?

例:

```
01: proc{|a,b| }.lambda?   # => false
02: lambda{|a,b| }.lambda? # => true
```

- Proc#curry

例:

```
01: a = proc{|a,b| a - b }
02: b = a.curry # proc{|a| lambda{|b| a + b } }
03: c = b.call(3)
04: p c.call(2) # =>  1
05: p c.call(4) # => -1
```

Fiber
coroutines/micro-threads のライブラリです。よくわからない人には必要無いそうです。

Thread
色々削除されました。

- Thread.critical and Thread.critical= removed
- Thread#exit!, Thread#kill! and Thread#terminate! removed.

Enumerable and Enumerator

- Enumerable#each_with_index can take optional arguments and passes them to #each.

例:

```
01:
02: class X
03:   include Enumerable
04:   def each(*args)
05:     args.each{|arg|
06:       yield arg.inspect
07:     }
08:   end
```

```
09: end
10: z = []
11: X.new.each_with_index(42, 45){|x, i| z << [x, i] }
12: p z
```

- Enumerable#each_with_object

例:

```
01: # -*- coding: utf-8 -*-
02: p (1..10).inject(0){|sum, elem| sum + elem }           # => 55
03: p (1..10).each_with_object(0){|elem, sum| sum + elem } # =>  0
04:
05: arr = %w[たろう じろう さぶろう うめ]
06: v = arr.each_with_object(Hash.new{|h, k| h[k] = [] }){|element, memo|
07:   memo[element.size] << element
08: }
09: p v
```

- Enumerator#with_object
- Enumerator.new { ... }

例:

```
01: fib = Enumerator.new { |y|
02:   a = b = 1
03:   loop {
04:     y << a
05:     a, b = b, a + b
06:   }
07: }
08:
09: p fib.take(10) #=> [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

Array

- Array#delete returns a deleted element rather than a given object
- Array#to_s is equivalent to Array#inspect
- Array.try_convert
- Array#pack('m0') complies with RFC 4648.

Hash

- preserving item insertion order
- Hash#default_proc=
- Hash#compare_by_identity and Hash#compare_by_identity?
- Hash.try_convert
- Hash#assoc
- Hash#rassoc
- Hash#flatten

Range

- Range#cover?
- Range#include? iterates over elements and compares the given value with each element unless the range is numeric. Use Range#cover? for the old behavior, i.e. comparison with boundary values.
- Range#min, Range#max

File and Dir operations

- New methods

Process

- Process.spawn
- Process.daemon

String

- String#clear
- String#ord
- String#getbyte, String#setbyte
- String#chars and String#each_char act as character-wise.
- String#codepoints, String#each_codepoint
- String#unpack with a block
- String#hash
- String.try_convert

- String#encoding
- String#force_encoding, String#encode and String#encode!
- String#ascii_only?
- String#valid_encoding?
- String#match

Symbol

- Zero-length symbols allowed
- Symbol#=== matches strings ????
- Symbol#intern
- Symbol#encoding
- Symbol methods similar to those in String

Regexp

- Regexp#=== matches symbols
- Regexp.try_convert
- Regexp#match
- Regexp#fixed_encoding?
- Regexp#encoding
- Regexp#named_captures
- Regexp#names

MatchData

- MatchData#names
- MatchData#regexp

Encoding, Encoding::Converter

- supports conversion between many encodings

Numeric

- Numeric#upto, #downto, #times, #step はブロックなしで呼ばれた場合、Enumerator のインスタンスを返します。
- Numeric#real?, Complex#real?
- Numeric#magnitude

Rational / Complex

- They are in the core library now

Math

- Math#log takes an optional argument.
- Math#log2
- Math#cbrt, Math#lgamma, Math#gamma

Time

- Time.times removed. Use Process.times.
- Time#sunday?
- Time#monday?
- Time#tuesday?
- Time#wednesday?
- Time#thursday?
- Time#friday?
- Time#saturday?
- Time#tv_nsec and Time#nsec

Misc. new methods

- RUBY_ENGINE to distinguish between Ruby processor implementation
- public_method
- public_send
- GC.count
- ObjectSpace.count_objects
- Method#hash, Proc#hash
- Method#source_location, UnboundMethod#source_location and Proc#source_location

例:

```
01:
02: class A
03:   def foo
04:     proc {|a, b| a + b }
05:   end
06: end
07:
```

例:

```
01: require 'source_location'
02: p A.new.foo.source_location
```

- __callee__

例:

```
01:
02:
03: class A
04:   def foo
05:     p __callee__
06:   end
07: end
08:
09: def bar
10:   p __callee__
11: end
12:
13: baz = lambda{ p __callee__ }
14:
15: A.new.foo # => :foo
16: bar       # => :bar
17: baz.call  # => nil
18:
19: class B
20:   %w[foo bar baz].each{|m|
21:     define_method(m){|n, memo|
22:       return memo if n <= 0
23:       __send__(__callee__, n - 1, memo * n)
24:     }
25:   }
26: end
27:
28: p B.new.foo(5, 1)
```

```
29: p B.new.bar(5, 1)
30: p B.new.baz(5, 1)
```

- Elements in $LOAD_PATH and $LOADED_FEATURES are expanded
- 行頭ドットによる行継続

例:

```
01: class A
02:   def very_long_method_name
03:     @count ||= 0
04:     p @count += 1
05:     self
06:   end
07: end
08:
09: A.new.very_long_method_name
10:      .very_long_method_name
11:      .very_long_method_name
12:      .very_long_method_name
13:      .very_long_method_name
```

- __ENCODING__

例:

```
01: p __ENCODING__
02:
```

例:

```
01: # -*- coding: utf-8 -*-
02: p __ENCODING__
```

例:

```
01: # -*- coding: euc-jp -*-
02: p __ENCODING__
```

- !=, !~, ! が再定義できるようになった

例:

```
01:
02: class A
03:   def ==(other)
04:     true
```

```
05:   end
06:
07:   def !=(other)
08:     true
09:   end
10: end
11:
12: a = A.new
13: b = A.new
14:
15: p a == b # => true
16: p a != b # => true
```

- if, case when などで then の代わりの : は使えなくなった
- begin ... end 以外での retry は廃止された
- eval の第二引数として Proc オブジェクトを渡せなくなった

bundled libraries

RubyGems

- Package management system for Ruby.
- Integrated with Ruby's library loader.

Rake

- Ruby make. A simple ruby build program with capabilities similar to make.

minitest

- Our new testing library which is faster, cleaner and easier to read than the old test/unit.
- You can introduce the old test/unit as testunit gem through RubyGems if you want.

CMath

- Complex number version of Math

Prime

- Extracted from Mathn and improved. You can easily enumerate prime numbers.
- Prime.new is obsolete. Use its class methods.

ripper

- Ruby script parser

Readline

- Readline.vi_editing_mode?
- Readline.emacs_editing_mode?
- Readline::HISTORY.clear

Tk

- TkXXX widget classes are removed and redefined as aliases of Tk::XXX classes.

RDoc

- Updated to version 2.2.2.
  - See: http://rubyforge.org/frs/shownotes.php?group_id=627&release_id=26434

commandline options

- -E, –encoding
- -U
- –enable-gems, –disable-gems
- –enable-rubyopt, –disable-rubyopt
- long options are allowed in RUBYOPT environment variable.

実装の変更点

Memory Diet

- Object Compaction - Object, Array, String, Hash, Struct, Class, Module
- st_table compaction (inlining small tables)

YARV

- Ruby codes are compiled into opcodes before executed.
- Native thread

Platform supports

Support levels

0. Supported 1. Best effort 2. Perhaps 3. Not supported

Dropped

- No longer supports djgpp, human68k, MacOS 9 or earlier, VMS nor Windows CE.

Ruby1.9.1 を使用するにあたって注意すること

コマンドラインオプションを全部見る方法

```
$ man ruby
```

スクリプトを実行するときの注意点

gem を使わないとき

コマンドラインオプション –disable-gems を付けると速くなります。ただし、何回も同じスクリプトを実行する場合の二度目以降はキャッシュに載るので差が小さくなります。

ちなみに私のマシンで irb を起動すると体感できるくらい速度差があります。

$SAFE

tDiary などのウェブアプリで現在進行形の問題ですが、$SAFE = 1 でうまく動かない問題があるようです。

以下の現象が確認されています。

- RubyGems を一つでも使用している場合
  - RubyGems は動的に $LOAD_PATH にパスを追加する。
  - $LOAD_PATH に汚染されたパスが含まれるので $LOAD_PATH 全体が汚染される。
  - 元々は、汚染されていなかったパスからの require でも SecurityError 発生。
  - 実行時に動的に $LOAD_PATH を追加するとアウト。
- $SAFE = 1 で拡張ライブラリが require できない。
  - openuri など内部的に拡張ライブラリを使用しているものは全部アウト。
  - バグ？

スクリプトを書くときの注意点

これまでに出てきた非互換性についてはもちろんのこと以下のことにも注意する必要があります。

m17n

特に注意しなければならないのは以下の三つです。

- スクリプトエンコーディング (ソースエンコーディング)
- 文字列や正規表現などのオブジェクトのエンコーディング
- IO (入出力) のエンコーディング

Ruby1.9 でスクリプトエンコーディングを決定するのは主に magic comment です。非推奨のコマンドラインオプション -K でも決めることはできますが magic comment で指定するようにしてください。(-K はスクリプトエンコーディングと default_external に影響を与えます。)

magic comment は以下のような形式でスクリプトの一行目に記述することができます。一行目が shebang の場合は二行目に記述します。

```
# encoding: euc-jp
# -*- coding: euc-jp -*-
# vim:set fileencoding=euc-jp:
# coding: utf-8
```

毎回、自分の手で書くのは面倒なのでエディタに自動的に挿入させるようにすると良いと思います。

保存時に自動で Ruby 1.9 の magic comment をつける

http://d.hatena.ne.jp/rubikitch/20080307/magiccomment

vim だとこんな感じらしい。(thx nanki)

```
# 保存時ではなく、新規作成時に挿入する。(~/.vimrc に書くといいでしょう)
au BufEnter *.rb if getline(1) == ""
  \| :call append(0, ['#!/usr/bin/env ruby', '# -*- coding: UTF-8 -*-;'])
\| endif
```

他のエディタでも出来るはずなので工夫してみてください。

次に、文字列や正規表現などのオブジェクトのエンコーディングですが、リテラルで書いた場合はほぼスクリプトエンコーディングと同じになります。

例外がいくつかありますが、詳細はるりまを参照してください。

最後に IO (入出力) のエンコーディングについて説明します。

まず Encoding.default_external はロケールで決まります。変更する場合は -E オプションで変更します。Encoding.default_internal はデフォルトでは nil です。Encoding.default_internal に値をセットすると、ファイル読み込み時に自動的にエンコーディングを変換します。しかし、危険なのでおすすめしません。

おすすめの方法は、ファイル読み込み時に明示的にエンコーディングを指定することです。

例:

```
01: # -*- coding: utf-8 -*-
02: s = 'あいう'
03: p __ENCODING__                  # => #<Encoding:UTF-8>
```

```
04: p File.read('euc.txt').encoding # => #<Encoding:UTF-8>
05: str = File.read('euc.txt')
06: p str.encoding                  # => #<Encoding:UTF-8> 実は euc-jp
07: p str.force_encoding('euc-jp').encode('utf-8')
08: File.open('euc.txt', 'r:euc-jp:utf-8') do |file|
09:   p file.read.encoding # => #<Encoding:UTF-8>
10: end
11: p File.read('utf.txt')
```

ファイルに書き出すときも同様に明示的にエンコーディングを指定するのがよいでしょう。

その他

- Windows 環境で Unicode ファイル名がうまく扱えない
- String#== はエンコーディングも含めて判定する

まとめ

- Ruby 1.9.1 はすごい
- いっぱい変更点があるけど把握しきれない
- でも普通に使う分にはたぶん大丈夫
- みんな Ruby 1.9.1 を使おう！

参考

Ruby の RDoc
リンクなし。

eigenclass - Changes in Ruby 1.9
http://eigenclass.org/hiki/Changes+in+Ruby+1.9

笹田耕一/笹田研の研究業績リスト
http://www.atdot.net/ko1/activities/#idx12

ruby 1.9 を日常的に使うぼくが 1.9 の新機能を寸評する - まめめも
http://d.hatena.ne.jp/ku-ma-me/20090126/p1

Ruby 1.9 でアプリケーションが速くなるとしたら - kwatch の日記
http://d.hatena.ne.jp/kwatch/20081212/1229094312

Ruby 1.9 m17n リファレンス (不完全版) - diary of a madman
http://d.hatena.ne.jp/macks/20080102/p1

[ruby-dev:37852] 1.9.1-rc2 の NEWS の内容

http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-dev/37852

[ruby-dev:37843] $SAFE=1 での require が SecurityError になる条件

http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-dev/37843

[ruby-list:45777] Ruby 1.9.1-rc1[mswin32] でマルチバイトを含むソースが実行できない

http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-list/45777

多言語化

http://doc.okkez.net/191/view/spec/m17n

class IO

http://doc.okkez.net/191/view/class/IO#m17n

Book:プログラミング言語 Ruby

http://www.oreilly.co.jp/books/9784873113944/

[ruby-list:45826] Re: unicode のファイル名の処理

http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-list/45826