Now that it has been made clear that Enumerable#count never calls #size and that we have Enumerable#lazy, let me propose again an API for a lazy way to get the size of an Enumerable: Enumerable#size.

- call-seq:
  - enum.size # => nil, Integer or Float::INFINITY

- Returns the number of elements that will be yielded, without going through the iteration (i.e. lazy), or +nil+ if it can't be calculated lazily.

- perm = (1..100).to_a.permutation(4)
- perm.size # => 94109400
- perm.each_cons(2).size # => 94109399
- loop.size # => Float::INFINITY
- [42].drop_while.size # => nil

About 66 core methods returning enumerators would have a lazy size, like each_slice, permutation or lazy.take.

A few would have size return nil:
- Array#{r}index, {take|drop}_while
- Enumerable#{find|index}, {take|drop}_while
- IO: all methods

Sized enumerators can also be created naturally by providing a block to to_enum/enum_for or a lambda to Enumerator.new.

Example for to_enum:

class Integer
def composition
  return to_enum(:composition){ 1 << (self - 1) } unless block_given?
  yield [] if zero?
  downto(1) do |i|
    (self - i).composition do |comp|
      yield [i, *comp]
    end
  end
end
end

4.composition.to_a
# => [[4], [3, 1], [2, 2], [2, 1, 1], [1, 3], [1, 2, 1], [1, 1, 2], [1, 1, 1, 1]]

42.composition.size # => 2199023255552

Example for Enumerator.new:

def lazy_product(*enums)
sizer = ->{(enums.inject(1) do |product, e|
  break if (size = e.size).nil?
  product * size
end}
  Enumerator.new(sizer) do |yielder|
    # ... generate combinations
  end
end
```ruby
end

lazy_product(1..4, (1..3).each_cons(2)).size #=> 8
lazy_product(1..4, (1..3).cycle).size #=> Float::INFINITY

<table>
<thead>
<tr>
<th>Related issues:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Related to Ruby master - Feature #3715: Enumerator#size and #size=</td>
<td>Rejected</td>
<td>08/19/2010</td>
</tr>
<tr>
<td>Related to Ruby master - Bug #7298: Behavior of Enumerator.new different between...</td>
<td>Closed</td>
<td>11/07/2012</td>
</tr>
</tbody>
</table>

**Associated revisions**

**Revision c73b6bd7 - 11/06/2012 05:09 PM - marcandre (Marc-Andre Lafortune)**

- enumerator.c (enumerator_initialize): Warn when using deprecated form [Feature #6636]

  git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37495 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

**Revision 37495 - 11/06/2012 05:09 PM - marcandre (Marc-Andre Lafortune)**

- enumerator.c (enumerator_initialize): Warn when using deprecated form [Feature #6636]

**Revision 37495 - 11/06/2012 05:09 PM - marcandre (Marc-Andre Lafortune)**

- enumerator.c (enumerator_initialize): Warn when using deprecated form [Feature #6636]

**Revision 37495 - 11/06/2012 05:09 PM - marcandre (Marc-Andre Lafortune)**

- enumerator.c (enumerator_initialize): Warn when using deprecated form [Feature #6636]

**Revision 37495 - 11/06/2012 05:09 PM - marcandre (Marc-Andre Lafortune)**

- enumerator.c (enumerator_initialize): Warn when using deprecated form [Feature #6636]

**Revision 37495 - 11/06/2012 05:09 PM - marcandre (Marc-Andre Lafortune)**

- enumerator.c (enumerator_initialize): Warn when using deprecated form [Feature #6636]

**Revision 37495 - 11/06/2012 05:09 PM - marcandre (Marc-Andre Lafortune)**

- enumerator.c (enumerator_initialize): Warn when using deprecated form [Feature #6636]

**Revision 37497 - 11/06/2012 05:10 PM - marcandre (Marc-Andre Lafortune)**

- enumerator: New method #size; constructor accepts size [Feature #6636]

  include/ruby/intern.h: RETURN_SIZED_ENUMERATOR for support of sized enumerators

  git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37497 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

**Revision 37497 - 11/06/2012 05:10 PM - marcandre (Marc-Andre Lafortune)**

- enumerator: New method #size; constructor accepts size [Feature #6636]

  include/ruby/intern.h: RETURN_SIZED_ENUMERATOR for support of sized enumerators

```
enumerator: New method #size; constructor accepts size
[Feature #6636]

include/ruby/intern.h: RETURN_SIZED_ENUMERATOR for support of sized enumerators
Revision 37498 - 11/06/2012 05:10 PM - marcandre (Marc-Andre Lafortune)
- enumerator.c (obj_to_enum): Have #to_enum accept a block [Feature #6636]

Revision 37498 - 11/06/2012 05:10 PM - marcandre (Marc-Andre Lafortune)
- enumerator.c (obj_to_enum): Have #to_enum accept a block [Feature #6636]

Revision 37498 - 11/06/2012 05:10 PM - marcandre (Marc-Andre Lafortune)
- enumerator.c (obj_to_enum): Have #to_enum accept a block [Feature #6636]

Revision 37498 - 11/06/2012 05:10 PM - marcandre (Marc-Andre Lafortune)
- enumerator.c (obj_to_enum): Have #to_enum accept a block [Feature #6636]

Revision 37498 - 11/06/2012 05:10 PM - marcandre (Marc-Andre Lafortune)
- enumerator.c (obj_to_enum): Have #to_enum accept a block [Feature #6636]

Revision 37499 - 11/06/2012 05:10 PM - marcandre (Marc-Andre Lafortune)
- enumerator.c: Support #size for enumerators created from enumerators [Feature #6636]

Revision 37499 - 11/06/2012 05:10 PM - marcandre (Marc-Andre Lafortune)
- enumerator.c: Support #size for enumerators created from enumerators [Feature #6636]

Revision 37499 - 11/06/2012 05:10 PM - marcandre (Marc-Andre Lafortune)
- enumerator.c: Support #size for enumerators created from enumerators [Feature #6636]

Revision 37499 - 11/06/2012 05:10 PM - marcandre (Marc-Andre Lafortune)
- enumerator.c: Support #size for enumerators created from enumerators [Feature #6636]

Revision 37499 - 11/06/2012 05:10 PM - marcandre (Marc-Andre Lafortune)
- enumerator.c: Support #size for enumerators created from enumerators [Feature #6636]

Revision 37499 - 11/06/2012 05:10 PM - marcandre (Marc-Andre Lafortune)
- enumerator.c: Support #size for enumerators created from enumerators [Feature #6636]

Revision 37500 - 11/06/2012 05:10 PM - marcandre (Marc-Andre Lafortune)
- array.c: Support for Enumerator#size in trivial cases: each, each_index, reverse_each, sort_by, collect, collect!, select, select!, keep_if, reject, reject!, delete_if [Feature #6636]

Revision 37500 - 11/06/2012 05:10 PM - marcandre (Marc-Andre Lafortune)
- array.c: Support for Enumerator#size in trivial cases: each, each_index, reverse_each, sort_by, collect, collect!, select, select!, keep_if, reject, reject!, delete_if [Feature #6636]
array.c: Support for Enumerator#size in trivial cases: each, each_index, reverse_each, sort_by, collect, collect!, select, select!, keep_if, reject, reject!, delete_if [Feature #6636]

array.c: Support for Enumerator#size in trivial cases: each, each_index, reverse_each, sort_by, collect, collect!, select, select!, keep_if, reject, reject!, delete_if [Feature #6636]

array.c: Support for Enumerator#size in trivial cases: each, each_index, reverse_each, sort_by, collect, collect!, select, select!, keep_if, reject, reject!, delete_if [Feature #6636]

array.c: Support for Enumerator#size in trivial cases: each, each_index, reverse_each, sort_by, collect, collect!, select, select!, keep_if, reject, reject!, delete_if [Feature #6636]

array.c: Support for Enumerator#size in trivial cases: each, each_index, reverse_each, sort_by, collect, collect!, select, select!, keep_if, reject, reject!, delete_if [Feature #6636]

array.c (rb_ary_permutation): Support for Array#permutation.size [Feature #6636]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37501 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

array.c (rb_ary_permutation): Support for Array#permutation.size [Feature #6636]

array.c (rb_ary_permutation): Support for Array#permutation.size [Feature #6636]

array.c (rb_ary_permutation): Support for Array#permutation.size [Feature #6636]

array.c (rb_ary_permutation): Support for Array#permutation.size [Feature #6636]

array.c (rb_ary_permutation): Support for Array#permutation.size [Feature #6636]

array.c (rb_ary_permutation): Support for Array#permutation.size [Feature #6636]

array.c (rb_ary_permutation): Support for Array#permutation.size [Feature #6636]

array.c (rb_ary_combination): Support for Array#combination.size [Feature #6636]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37502 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

array.c (rb_ary_permutation): Support for Array#permutation.size [Feature #6636]
• array.c (rb_ary_combination): Support for Array#combination.size [Feature #6636]

Revision 37502 - 11/06/2012 05:11 PM - marcandre (Marc-Andre Lafortune)

• array.c (rb_ary_combination): Support for Array#combination.size [Feature #6636]

Revision 37502 - 11/06/2012 05:11 PM - marcandre (Marc-Andre Lafortune)

• array.c (rb_ary_combination): Support for Array#combination.size [Feature #6636]

Revision 37502 - 11/06/2012 05:11 PM - marcandre (Marc-Andre Lafortune)

• array.c (rb_ary_combination): Support for Array#combination.size [Feature #6636]

Revision 37502 - 11/06/2012 05:11 PM - marcandre (Marc-Andre Lafortune)

• array.c (rb_ary_combination): Support for Array#combination.size [Feature #6636]

Revision 68c90c4a - 11/06/2012 05:11 PM - marcandre (Marc-Andre Lafortune)

• array.c (rb_ary_repeated_permutation): Support for repeated_permutation.size [Feature #6636]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37503 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 37503 - 11/06/2012 05:11 PM - marcandre (Marc-Andre Lafortune)

• array.c (rb_ary_repeated_permutation): Support for repeated_permutation.size [Feature #6636]

Revision 37503 - 11/06/2012 05:11 PM - marcandre (Marc-Andre Lafortune)

• array.c (rb_ary_repeated_permutation): Support for repeated_permutation.size [Feature #6636]

Revision 37503 - 11/06/2012 05:11 PM - marcandre (Marc-Andre Lafortune)

• array.c (rb_ary_repeated_permutation): Support for repeated_permutation.size [Feature #6636]

Revision 37503 - 11/06/2012 05:11 PM - marcandre (Marc-Andre Lafortune)

• array.c (rb_ary_repeated_permutation): Support for repeated_permutation.size [Feature #6636]

Revision 37503 - 11/06/2012 05:11 PM - marcandre (Marc-Andre Lafortune)

• array.c (rb_ary_repeated_permutation): Support for repeated_permutation.size [Feature #6636]

Revision 37503 - 11/06/2012 05:11 PM - marcandre (Marc-Andre Lafortune)

• array.c (rb_ary_repeated_permutation): Support for repeated_permutation.size [Feature #6636]

Revision 37503 - 11/06/2012 05:11 PM - marcandre (Marc-Andre Lafortune)

• array.c (rb_ary_repeated_permutation): Support for repeated_permutation.size [Feature #6636]

Revision ba59365d - 11/06/2012 05:11 PM - marcandre (Marc-Andre Lafortune)

• array.c (rb_ary_repeated_combination): Support for repeated_combination.size [Feature #6636]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37504 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 37504 - 11/06/2012 05:11 PM - marcandre (Marc-Andre Lafortune)
array.c (rb_ary_repeated_combination): Support for repeated_combination.size [Feature #6636]

Revision 37504 - 11/06/2012 05:11 PM - marcandre (Marc-Andre Lafortune)
array.c (rb_ary_repeated_combination): Support for repeated_combination.size [Feature #6636]

Revision 37504 - 11/06/2012 05:11 PM - marcandre (Marc-Andre Lafortune)
array.c (rb_ary_repeated_combination): Support for repeated_combination.size [Feature #6636]

Revision 37504 - 11/06/2012 05:11 PM - marcandre (Marc-Andre Lafortune)
array.c (rb_ary_repeated_combination): Support for repeated_combination.size [Feature #6636]

Revision 37504 - 11/06/2012 05:11 PM - marcandre (Marc-Andre Lafortune)
array.c (rb_ary_repeated_combination): Support for repeated_combination.size [Feature #6636]

Revision 37504 - 11/06/2012 05:11 PM - marcandre (Marc-Andre Lafortune)
array.c (rb_ary_repeated_combination): Support for repeated_combination.size [Feature #6636]

Revision b8b01ab9 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)
array.c (rb_ary_cycle): Support for Array#cycle.size [Feature #6636]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37505 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 37505 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)
array.c (rb_ary_cycle): Support for Array#cycle.size [Feature #6636]

Revision 37505 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)
array.c (rb_ary_cycle): Support for Array#cycle.size [Feature #6636]

Revision 37505 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)
array.c (rb_ary_cycle): Support for Array#cycle.size [Feature #6636]

Revision 37505 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)
array.c (rb_ary_cycle): Support for Array#cycle.size [Feature #6636]

Revision 37505 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)
array.c (rb_ary_cycle): Support for Array#cycle.size [Feature #6636]

Revision 37505 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)
array.c (rb_ary_cycle): Support for Array#cycle.size [Feature #6636]

Revision 727024fb - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)
vm_eval.c (rb_f_loop): Support for loop.size [Feature #6636]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37506 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 37506 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)
vm_eval.c (rb_f_loop): Support for loop.size [Feature #6636]
Revision 37506 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)

• vm_eval.c (rb_f_loop): Support for loop.size [Feature #6636]

Revision 37506 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)

• vm_eval.c (rb_f_loop): Support for loop.size [Feature #6636]

Revision 37506 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)

• vm_eval.c (rb_f_loop): Support for loop.size [Feature #6636]

Revision 37506 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)

• vm_eval.c (rb_f_loop): Support for loop.size [Feature #6636]

Revision 37506 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)

• vm_eval.c (rb_f_loop): Support for loop.size [Feature #6636]

Revision c82ad6d2 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)

• enum.c: Support for enumerators created by Enumerable with forwarding: find_all, reject, ... [Feature #6636]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37507 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 37507 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)

• enum.c: Support for enumerators created by Enumerable with forwarding: find_all, reject, ... [Feature #6636]

Revision 37507 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)

• enum.c: Support for enumerators created by Enumerable with forwarding: find_all, reject, ... [Feature #6636]

Revision 37507 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)

• enum.c: Support for enumerators created by Enumerable with forwarding: find_all, reject, ... [Feature #6636]

Revision 37507 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)

• enum.c: Support for enumerators created by Enumerable with forwarding: find_all, reject, ... [Feature #6636]

Revision 37507 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)

• enum.c: Support for enumerators created by Enumerable with forwarding: find_all, reject, ... [Feature #6636]

Revision 37507 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)

• enum.c: Support for enumerators created by Enumerable with forwarding: find_all, reject, ... [Feature #6636]

Revision df8451e6 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)

• enum.c (enum_each_slice): Support for Enumerable#each_slice.size [Feature #6636]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37508 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 37508 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)

• enum.c (enum_each_slice): Support for Enumerable#each_slice.size [Feature #6636]

Revision 37508 - 11/06/2012 05:12 PM - marcandre (Marc-Andre Lafortune)
- enum.c (enum_cycle): Support for Enumerable#cycle.size [Feature #6636]

Revision 37510 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafontune)
- enum.c (enum_cycle): Support for Enumerable#cycle.size [Feature #6636]

Revision 37510 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafontune)
- enum.c (enum_cycle): Support for Enumerable#cycle.size [Feature #6636]

Revision 37510 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafontune)
- enum.c (enum_cycle): Support for Enumerable#cycle.size [Feature #6636]

Revision 37510 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafontune)
- enum.c (enum_cycle): Support for Enumerable#cycle.size [Feature #6636]

Revision 37510 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafontune)
- enum.c (enum_cycle): Support for Enumerable#cycle.size [Feature #6636]

Revision 37510 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafontune)
- enum.c (enum_cycle): Support for Enumerable#cycle.size [Feature #6636]

Revision 19ed71c8 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafontune)
- hash.c: Support for enumerators created by Hash: delete_if, reject!, ... [Feature #6636]
- git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37511 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 37511 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafontune)
- hash.c: Support for enumerators created by Hash: delete_if, reject!, ... [Feature #6636]

Revision 37511 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafontune)
- hash.c: Support for enumerators created by Hash: delete_if, reject!, ... [Feature #6636]

Revision 37511 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafontune)
- hash.c: Support for enumerators created by Hash: delete_if, reject!, ... [Feature #6636]

Revision 37511 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafontune)
- hash.c: Support for enumerators created by Hash: delete_if, reject!, ... [Feature #6636]

Revision 37511 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafontune)
- hash.c: Support for enumerators created by Hash: delete_if, reject!, ... [Feature #6636]

Revision 37511 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafontune)
- hash.c: Support for enumerators created by Hash: delete_if, reject!, ... [Feature #6636]

Revision 37511 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafontune)
- hash.c: Support for enumerators created by Hash: delete_if, reject!, ... [Feature #6636]

Revision 37511 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafontune)
- hash.c: Support for enumerators created by Hash: delete_if, reject!, ... [Feature #6636]

Revision 37511 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafontune)
- hash.c: Support for enumerators created by Hash: delete_if, reject!, ... [Feature #6636]

Revision 17c0aff0 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafontune)
- hash.c: Support for enumerators created by ENV: each, each_value, ... [Feature #6636]
- git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37512 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 37512 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafontune)
- hash.c: Support for enumerators created by ENV: each, each_value, ... [Feature #6636]

Revision 37512 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafontune)
- hash.c: Support for enumerators created by ENV: each, each_value, ... [Feature #6636]
Revision 37512 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafortune)

- hash.c: Support for enumerators created by ENV: each, each_value, ... [Feature #6636]

Revision 37512 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafortune)

- hash.c: Support for enumerators created by ENV: each, each_value, ... [Feature #6636]

Revision 37512 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafortune)

- hash.c: Support for enumerators created by ENV: each, each_value, ... [Feature #6636]

Revision 37512 - 11/06/2012 05:13 PM - marcandre (Marc-Andre Lafortune)

- hash.c: Support for enumerators created by ENV: each, each_value, ... [Feature #6636]

Revision ce0bf94 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)

- struct.c: Support for Struct's enumerators #size [Feature #6636]

Revision 37513 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)

- struct.c: Support for Struct's enumerators #size [Feature #6636]

Revision 37513 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)

- struct.c: Support for Struct's enumerators #size [Feature #6636]

Revision 37513 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)

- struct.c: Support for Struct's enumerators #size [Feature #6636]

Revision 37513 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)

- struct.c: Support for Struct's enumerators #size [Feature #6636]

Revision 37514 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)

- numeric.c: Extract ruby_float_step_size [Feature #6636]

Revision 37514 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)

- numeric.c: Extract ruby_float_step_size [Feature #6636]

Revision 37514 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)

- numeric.c: Extract ruby_float_step_size [Feature #6636]
• numeric.c: Extract ruby_float_step_size [Feature #6636]

Revision 37514 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)
  • numeric.c: Extract ruby_float_step_size [Feature #6636]

Revision 37514 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)
  • numeric.c: Extract ruby_float_step_size [Feature #6636]

Revision 37514 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)
  • numeric.c: Extract ruby_float_step_size [Feature #6636]

Revision 37514 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)
  • numeric.c: Extract ruby_float_step_size [Feature #6636]

Revision 34be3a5d - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)
  • numeric.c (num_step): Support for Numeric#step.size [Feature #6636]

Revision 37515 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)
  • numeric.c (num_step): Support for Numeric#step.size [Feature #6636]

Revision 37515 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)
  • numeric.c (num_step): Support for Numeric#step.size [Feature #6636]

Revision 37515 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)
  • numeric.c (num_step): Support for Numeric#step.size [Feature #6636]

Revision 37515 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)
  • numeric.c (num_step): Support for Numeric#step.size [Feature #6636]

Revision 37515 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)
  • numeric.c (num_step): Support for Numeric#step.size [Feature #6636]

Revision 37515 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)
  • numeric.c (num_step): Support for Numeric#step.size [Feature #6636]

Revision 28d8bf90 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)
  • range.c: Support for Range#size and Range#each.size [Feature #6636]

Revision 37516 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)
  • range.c: Support for Range#size and Range#each.size [Feature #6636]

Revision 37516 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)
  • range.c: Support for Range#size and Range#each.size [Feature #6636]

Revision 37516 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)
- range.c: Support for Range#size and Range#each.size [Feature #6636]

Revision 37516 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)
- range.c: Support for Range#size and Range#each.size [Feature #6636]

Revision 37516 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)
- range.c: Support for Range#size and Range#each.size [Feature #6636]

Revision 37516 - 11/06/2012 05:14 PM - marcandre (Marc-Andre Lafortune)
- range.c: Support for Range#size and Range#each.size [Feature #6636]

Revision faed90d8 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)
- range.c: Support for range.step.size [Feature #6636]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37517 b2dd03c8-39d4-4d8f-98f1-823f69b080e

Revision 37517 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)
- range.c: Support for range.step.size [Feature #6636]

Revision 37517 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)
- range.c: Support for range.step.size [Feature #6636]

Revision 37517 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)
- range.c: Support for range.step.size [Feature #6636]

Revision 37517 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)
- range.c: Support for range.step.size [Feature #6636]

Revision 37517 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)
- range.c: Support for range.step.size [Feature #6636]

Revision 37517 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)
- range.c: Support for range.step.size [Feature #6636]

Revision 37517 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)
- range.c: Support for range.step.size [Feature #6636]

Revision 37518 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)
- numeric.c (int_upto, int_downto): Support for Integer#{down|up}to.size [Feature #6636]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37518 b2dd03c8-39d4-4d8f-98f1-823f69b080e

Revision 37518 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)
- numeric.c (int_upto, int_downto): Support for Integer#{down|up}to.size [Feature #6636]

Revision 37518 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)
- numeric.c (int_upto, int_downto): Support for Integer#{down|up}to.size [Feature #6636]

Revision 37518 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)
- numeric.c (int_upto, int_downto): Support for Integer#{down|up}to.size [Feature #6636]
Revision 37518 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)

- numeric.c (int_upto, int_downto): Support for Integer#{down|up}to.size [Feature #6636]

Revision 37518 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)

- numeric.c (int_upto, int_downto): Support for Integer#{down|up}to.size [Feature #6636]

Revision 37518 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)

- numeric.c (int_upto, int_downto): Support for Integer#{down|up}to.size [Feature #6636]

Revision 37519 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)

- numeric.c (int_dotimes): Support for Integer#times.size [Feature #6636]

Revision 37519 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)

- numeric.c (int_dotimes): Support for Integer#times.size [Feature #6636]

Revision 37519 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)

- numeric.c (int_dotimes): Support for Integer#times.size [Feature #6636]

Revision 37519 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)

- numeric.c (int_dotimes): Support for Integer#times.size [Feature #6636]

Revision 37519 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)

- numeric.c (int_dotimes): Support for Integer#times.size [Feature #6636]

Revision 37519 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)

- numeric.c (int_dotimes): Support for Integer#times.size [Feature #6636]

Revision 44374034 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)

- string.c: Support for String#{each_byte,each_char,each_codepoint}.size [Feature #6636]

Revision 44374034 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)

- string.c: Support for String#{each_byte,each_char,each_codepoint}.size [Feature #6636]
• string.c: Support for String#{each_byte,each_char,each_codepoint}.size [Feature #6636]

Revision 37520 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)
• string.c: Support for String#{each_byte,each_char,each_codepoint}.size [Feature #6636]

Revision 37520 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)
• string.c: Support for String#{each_byte,each_char,each_codepoint}.size [Feature #6636]

Revision c8426ce8 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)
• enumerator.c: Support for lazy.size [Feature #6636]

Revision 37521 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)
• enumerator.c: Support for lazy.size [Feature #6636]

Revision 37521 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)
• enumerator.c: Support for lazy.size [Feature #6636]

Revision 37521 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)
• enumerator.c: Support for lazy.size [Feature #6636]

Revision 37521 - 11/06/2012 05:15 PM - marcandre (Marc-Andre Lafortune)
• enumerator.c: Support for lazy.size [Feature #6636]

Revision 37522 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)
• enumerator.c: Support for lazy.{map|flat_map|...}.size [Feature #6636]

Revision 37522 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)
• enumerator.c: Support for lazy.{map|flat_map|...}.size [Feature #6636]

Revision 37522 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)
• enumerator.c: Support for lazy.{map|flat_map|...}.size [Feature #6636]

Revision 37522 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)
• enumerator.c: Support for lazy.{map|flat_map|...}.size [Feature #6636]
enumerator.c: Support for lazy.{map|flat_map|...}.size [Feature #6636]

Revision 37522 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)
  • enumerator.c: Support for lazy.{map|flat_map|...}.size [Feature #6636]

enumerator.c: Support for lazy.{map|flat_map|...}.size [Feature #6636]

Revision 37522 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)
  • enumerator.c: Support for lazy.{map|flat_map|...}.size [Feature #6636]

Revision 9aafa954 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)
  • enumerator.c: Support for lazy.take.size [Feature #6636]
  git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37523 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 37523 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)
  • enumerator.c: Support for lazy.take.size [Feature #6636]

Revision 37523 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)
  • enumerator.c: Support for lazy.take.size [Feature #6636]

Revision 37523 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)
  • enumerator.c: Support for lazy.take.size [Feature #6636]

Revision 37523 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)
  • enumerator.c: Support for lazy.take.size [Feature #6636]

Revision 37523 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)
  • enumerator.c: Support for lazy.take.size [Feature #6636]

Revision 37523 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)
  • enumerator.c: Support for lazy.take.size [Feature #6636]

Revision 37523 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)
  • enumerator.c: Support for lazy.take.size [Feature #6636]

Revision 7a310962 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)
  • enumerator.c: Add support for lazy.drop.size [Feature #6636]
  git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37524 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 37524 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)
  • enumerator.c: Add support for lazy.drop.size [Feature #6636]

Revision 37524 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)
  • enumerator.c: Add support for lazy.drop.size [Feature #6636]

Revision 37524 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)
  • enumerator.c: Add support for lazy.drop.size [Feature #6636]
Revision 37524 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)

- enumerator.c: Add support for lazy.drop.size [Feature #6636]

Revision 37524 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)

- enumerator.c: Add support for lazy.drop.size [Feature #6636]

Revision 0814c4ac - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)

- enumerator.c: Support for lazy.cycle.size [Feature #6636]
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37525 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 37525 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)

- enumerator.c: Support for lazy.cycle.size [Feature #6636]

Revision 37525 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)

- enumerator.c: Support for lazy.cycle.size [Feature #6636]

Revision 37525 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)

- enumerator.c: Support for lazy.cycle.size [Feature #6636]

Revision 37525 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)

- enumerator.c: Support for lazy.cycle.size [Feature #6636]

Revision 37525 - 11/06/2012 05:16 PM - marcandre (Marc-Andre Lafortune)

- enumerator.c: Support for lazy.cycle.size [Feature #6636]

Revision 37526 - 11/06/2012 05:17 PM - marcandre (Marc-Andre Lafortune)

- NEWS: Update for lazy size evaluation [Feature #6636]
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37526 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 37526 - 11/06/2012 05:17 PM - marcandre (Marc-Andre Lafortune)

- NEWS: Update for lazy size evaluation [Feature #6636]

Revision 37526 - 11/06/2012 05:17 PM - marcandre (Marc-Andre Lafortune)

- NEWS: Update for lazy size evaluation [Feature #6636]

Revision 37526 - 11/06/2012 05:17 PM - marcandre (Marc-Andre Lafortune)

- NEWS: Update for lazy size evaluation [Feature #6636]

Revision 37526 - 11/06/2012 05:17 PM - marcandre (Marc-Andre Lafortune)
• NEWS: Update for lazy size evaluation [Feature #6636]

Revision 37526 - 11/06/2012 05:17 PM - marcandre (Marc-Andre Lafortune)

• NEWS: Update for lazy size evaluation [Feature #6636]

**History**

### #1 - 07/01/2012 05:26 AM - marcandre (Marc-Andre Lafortune)

- File enumsize.pdf added

Attaching one-minute slide

### #2 - 07/02/2012 02:09 AM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

Received. Thank you!

---

Yusuke Endoh mame@tsg.ne.jp

### #3 - 07/21/2012 03:43 PM - naruse (Yui NARUSE)

How about adding Enumerator#receiver and define yourself with it.

```diff
+diff --git a/enumerator.c b/enumerator.c
+index f01ddd5..8e3ae9a 100644
++ a/enumerator.c
+++ b/enumerator.c
@@ -942,6 +942,32 @@ enumerator_inspect(VALUE obj)
 }
 /*
  * call-seq:
-+  * e.receiver  -> object
-+  *
-+  * Returns the receiver of this enumerator.
-+  * /
+  +static VALUE +enumerator_receiver(VALUE obj) {
+    struct enumerator *e;
+    VALUE eobj;
+    TypedData_Get_Struct(obj, struct enumerator, &enumerator_data_type, e);
+    if (e || e->obj == Qundef) {
+      return Qnil;
+    }
+    eobj = rb_attr_get(obj, id_receiver);
+    if (NIL_P(eobj)) {
+      eobj = e->obj;
+    }
+    return eobj; +}
+  */

+* Yielder */
static void @@ -1748,6 +1774,7 @@ InitVM_Enumerator(void) rb_define_method(rb_cEnumerator, "feed", enumerator_feed, 1);
    rb_define_method(rb_cEnumerator, "rewind", enumerator_rewind, 0);
    rb_define_method(rb_cEnumerator, "inspect", enumerator_inspect,
    0);

  
-  \* Lazy */
rb_cLazy = rb_define_class_under(rb_cEnumerator, "Lazy", rb_cEnumerator);

irb(main):007:0> e="abcde".enum_for(:each_byte)
=> #
irb(main):009:0> def e.size; receiver.bytesize; end
=> nil
irb(main):010:0> e.size
=> 5

### #4 - 07/22/2012 06:38 AM - marcandre (Marc-Andre Lafortune)

Hi,
On Sat, Jul 21, 2012 at 2:43 AM, naruse (Yui NARUSE) naruse@airemix.jp wrote:

How about adding Enumerator#receiver and define yourself with it.

I agree receiver could be helpful. One would also need the method and the arguments, as in my request #3714.

Still, it doesn't really address the issue.

If someone wants to write a library to output the progression, for example, it is still not possible for a general enumerable/enumerator.

The proposal is so that:

- it is standard so anyone can depend on it and also create their own enumerables/enumerators
- it can help in some calculations
- It can help to have generic progression reports
- etc.

Thanks

#5 - 07/23/2012 11:55 PM - mame (Yusuke Endoh)
- Status changed from Assigned to Feedback

Marc-Andre Lafontune,

We discussed your slide at the developer meeting (7/21).

Matz was positive to the spec of return value: Integer, Float::INFINITY, and nil.
However, we couldn't understand what API is proposed for creating an Enumerator with size.

So, please revise and elaborate your API according to these two points:

- Enumerator.new(size) is not acceptable because of compatibility:

```ruby
p Enumerator.new([1,2,3]).take(2) #=> [1, 2]
```

- We cannot determine the size of enumerator when creating it:

```ruby
a = [1]
e = a.permutation
a << 2
p e.to_a #=> [1, 2] [2, 1]
```

So, the API may need to receive a code fragment that calculates size, such as a Proc.

--
Yusuke Endoh mame@tsg.ne.jp

#6 - 07/24/2012 12:13 AM - marcandre (Marc-Andre Lafontune)
- Status changed from Feedback to Open

Hi,

mame (Yusuke Endoh) wrote:

Matz was positive to the spec of return value: Integer, Float::INFINITY, and nil.

:-)

However, we couldn't understand what API is proposed for creating an Enumerator with size.

So, please revise and elaborate your API according to these two points:

- Enumerator.new(size) is not acceptable because of compatibility:
p Enumerator.new([1,2,3]).take(2)  #=> [1, 2]

Agreed.
I am proposing Enumerator.new(size_lambda){ block }, i.e. only if a block is given, then the first argument can be a lambda/proc that can lazily compute the size.

The old syntax of Enumerator.new without a block does not change meaning.

- We cannot determine the size of enumerator when creating it:

```ruby
a = [1]  
e = a.permutation  
a << 2  
p e.to_a  #=> 1, 2, 1
```

So, the API may need to receive a code fragment that calculates size, such as a Proc.

Agreed.
This is why I propose that to_enum accepts a block that can calculate the size, and Enumerator.new with a block can accept a lambda/proc for the same.

Does this address the concerns?

I will be glad to propose a set of patches so we can experiment with this.

Marc-André

#7 - 07/24/2012 11:29 PM - mame (Yusuke Endoh)
Hello MarcAndre

2012/7/24, marcandre (Marc-Andre Lafortune) ruby-core@marc-andre.ca:

- Enumerator.new(size) is not acceptable because of compatibility:

```ruby
p Enumerator.new([1,2,3]).take(2)  #=> [1, 2]
```

Agreed.
I am proposing Enumerator.new(size_lambda){ block }, i.e. only if a block is given, then the first argument can be a lambda/proc that can lazily compute the size.

This is just my guess, but matz will not like such a method whose meaning of its argument varies depending on whether block is given or not.

The old syntax of Enumerator.new without a block does not change meaning.

Is it okay that there is no way to specify size in this case?

- We cannot determine the size of enumerator when creating it:

```ruby
a = [1]  
e = a.permutation  
a << 2  
p e.to_a  #=> 1, 2, 1
```

So, the API may need to receive a code fragment that calculates size, such as a Proc.

Agreed.
This is why I propose that to_enum accepts a block that can calculate the size, and Enumerator.new with a block can accept a lambda/proc for the same.

What argument(s) will the lambda/proc receive?
Hi,
mame (Yusuke Endoh) wrote:

I am proposing Enumerator.new(size_lambda){ block }, i.e. only if a block is given, then the first argument can be a lambda/proc that can lazily compute the size.

This is just my guess, but matz will not like such a method whose meaning of its argument varies depending on whether block is given or not.

I understand the concern.

It could still be acceptable here because the other form is already documented as 'discouraged'. Maybe we should deprecate it?

Other possibility would be to add a different creator, e.g. Enumerator.sized(size_lambda){|yielder| ... }.

The old syntax of Enumerator.new without a block does not change meaning.

Is it okay that there is no way to specify size in this case?

This old syntax is already discouraged and to_enum/enum_for should be used instead.

This is why I propose that to_enum accepts a block that can calculate the size, and Enumerator.new with a block can accept a lambda/proc for the same.

What argument(s) will the lambda/proc receive?

We could consider passing the receiver and/or any arguments passed to to_enum, but I would propose to keep it simple and pass no arguments.

This is because enumerators are immutable and all information held in Enumerators should be accessible from the block/lambda anyways.

Marc-André

#9 - 10/27/2012 07:11 AM - ko1 (Koichi Sasada)
- Assignee changed from matz (Yukihiro Matsumoto) to mame (Yusuke Endoh)

#10 - 10/27/2012 11:51 AM - mame (Yusuke Endoh)
- Status changed from Open to Assigned
- Assignee changed from mame (Yusuke Endoh) to matz (Yukihiro Matsumoto)
- Target version changed from 2.0.0 to 2.6

#11 - 10/29/2012 02:48 PM - marcandre (Marc-Andre Lafontune)

Hi.

My understanding was this new feature would make it into Ruby 2.0. Did I misunderstand?

The implementation can be seen here: [https://github.com/marcandre/ruby/compare/marcandre:trunk...marcandre:enum_size](https://github.com/marcandre/ruby/compare/marcandre:trunk...marcandre:enum_size)

Although the combined diff ([https://github.com/marcandre/ruby/compare/marcandre:trunk...marcandre:enum_size.diff](https://github.com/marcandre/ruby/compare/marcandre:trunk...marcandre:enum_size.diff)) is pretty big, there are really just two interesting commits. The second adds #size and extends constructor. The third extends to_enum to accept a block. See [https://github.com/marcandre/ruby/commit/add_enumerator_size](https://github.com/marcandre/ruby/commit/add_enumerator_size) and [https://github.com/marcandre/ruby/commit/sized_to_enum](https://github.com/marcandre/ruby/commit/sized_to_enum)

The first commit only warns on using the deprecated form with no block Enumerator.new(obj, *args), but compatibility is maintained.

The remaining commits add support for the different ways of creating enumerators that can evaluate lazily their size. A few remain to be implemented,
in particular the lazy ones.

#12 - 10/31/2012 01:20 PM - marcandre (Marc-Andre Lafortune)
I added support for lazy enumerators.

Yusuke, Matz, did I address your questions/concerns?

#13 - 11/05/2012 01:30 PM - matz (Yukihiro Matsumoto)
After skimming your modifies, I feel they are decent.
Sorry for being late to check.

Matz.

#14 - 11/06/2012 02:16 AM - marcandre (Marc-Andre Lafortune)
- Assignee changed from matz (Yukihiro Matsumoto) to marcandre (Marc-Andre Lafortune)
- Target version changed from 2.6 to 2.0.0

Hi,

matz (Yukihiro Matsumoto) wrote:

   After skimming your modifies, I feel they are decent.

Thanks for looking at them. Very happy to read this :)  

I'll then commit these and finish implementing size for ranges of strings.

Marc-André

#15 - 11/07/2012 02:09 AM - marcandre (Marc-Andre Lafortune)
- Status changed from Assigned to Closed
- % Done changed from 0 to 100

This issue was solved with changeset r37495.
Marc-André, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- enumerator.c (enumerator_initialize): Warn when using deprecated form [Feature #6636]

Files

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>enumsize.pdf</td>
<td>131 KB</td>
<td>07/01/2012</td>
<td>marcandre (Marc-Andre Lafortune)</td>
</tr>
</tbody>
</table>