Ruby master - Bug #17254

ENV.replace may set nil instead of the proper value

10/08/2020 10:20 AM - znz (Kazuhiro NISHIYAMA)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.7.2p137 (2020-10-01 revision 5445e04352) [x86_64-linux]	**Backport:**	2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN

Description

On docs.ruby-lang.org, it uses snap ruby, and it failed to run rdoc.

```
Oct 07 13:20:08 docs-2020.ruby-lang.org env[6183]: rdoc --title Documentation for Ruby master --ma
in README.md --output /var/www/docs.ruby-lang.org/releases/20200916140300/master -U --all --encodi
ng=UTF-8 .
Oct 07 13:20:08 docs-2020.ruby-lang.org env[6183]: <internal:gem_prelude>:1:in `require': cannot l
oad such file -- rubygems.rb (LoadError)
Oct 07 13:20:08 docs-2020.ruby-lang.org env[6183]:         from <internal:gem_prelude>:1:in `<inte
rnal:gem_prelude>'
Oct 07 13:20:08 docs-2020.ruby-lang.org env[6183]: rake aborted!
```

I investigate it, it caused by setting nil instead of the proper value in ENV.replace.

```
vagrant@buster:/tmp/t$ cat Gemfile
# frozen_string_literal: true

source "https://rubygems.org"

git_source(:github) {|repo_name| "https://github.com/#{repo_name}" }

# gem "rails"
vagrant@buster:/tmp/t$ env PATH=/var/www/docs.ruby-lang.org/shared/bundle/ruby/2.7.0/bin:/snap/bin
:$PATH DEBIAN_DISABLE_RUBYGEMS_INTEGRATION=1 bundle exec env -u RUBYOPT ruby -r/snap/ruby/189/lib/
ruby/gems/2.7.0/gems/bundler-2.1.4/lib/bundler/setup -e 'p ENV["RUBYLIB"]'
"/snap/ruby/189/lib/ruby/gems/2.7.0/gems/bundler-2.1.4/lib"
```

Calling ENV.clear before ENV.replace resolves this issue.

```
vagrant@buster:/tmp/t$ cat /tmp/clear-before-replace.rb
class << ENV
  alias orig_replace replace
  def replace(h)
    clear
    orig_replace(h)
  end
end
vagrant@buster:/tmp/t$ env PATH=/var/www/docs.ruby-lang.org/shared/bundle/ruby/2.7.0/bin:/snap/bin
:$PATH DEBIAN_DISABLE_RUBYGEMS_INTEGRATION=1 bundle exec env -u RUBYOPT ruby -r/tmp/clear-before-r
eplace -r/snap/ruby/189/lib/ruby/gems/2.7.0/gems/bundler-2.1.4/lib/bundler/setup -e 'p ENV["RUBYLI
B"]'
"/snap/ruby/189/lib/ruby/gems/2.7.0/gems/bundler-2.1.4/lib:/snap/ruby/189/lib/ruby/2.7.0:/snap/rub
y/189/lib/ruby/2.7.0/amd64"
```

Where should call ENV.clear?
In ENV.replace or caller of ENV.replace?

Associated revisions

Revision c0aeb98a - 10/29/2020 03:08 PM - jeremyevans (Jeremy Evans)

Make ENV.replace handle multiple environ entries with the same key

While it is expected that all environment keys are unique, that is not enforced. It is possible by manipulating environ directly you can call a process with an environment with duplicate keys. If ENV.replace was passed a hash with a key where environ had a duplicate for that key, ENV.replace would end up deleting the key from environ.

The fix in this case is to not assume that the environment key list has unique keys, and continue processing the entire key list in keylist_delete.

Fixes [Bug #17254]

History

#1 - 10/16/2020 02:45 AM - jeremyevans0 (Jeremy Evans)

Looking at the ENV.replace implementation, the reason it doesn't call clear is it isn't thread-safe. ENV.replace tries to set all ENV keys in the replacement hash (the argument), then remove keys for ENV that aren't in the replacement hash. This ensures that at no point does the environment have a missing key if ENV already has the key and the replacement hash also has the key.

So having ENV.replace call ENV.clear is not good as it would break thread safety. It sounds like a bug in ENV.replace if it is actually setting an environment key to a nil value. I couldn't recreate the error by doing:

```
ENV.clear
ENV.replace(ENV_VALUE_FROM_YOUR_EXAMPLE)
ENV.replace(h_VALUE_FROM_YOUR_EXAMPLE)
```

One thing I noted is the ENV in your example before the replace has multiple entries for "GEM_HOME", "GEM_PATH", and "RUBYLIB". That seems like a bug itself, and it is possibly related to the cause of ENV.replace resulting in a nil ENV value. Possibly the issue is due to multiple ENV keys with the same content but different encodings?

#2 - 10/22/2020 06:44 AM - znz (Kazuhiro NISHIYAMA)

I dump duplicated keys with encoding.
Encodings of duplicated keys are same.

```
vagrant@buster:/tmp/t$ cat /tmp/dump2.rb
class << ENV
  alias orig_replace replace
  def replace(h)
    key_count = ENV.keys.tally
    duplicated_keys = []
    ENV.each_pair.each do |k, v|
      next if key_count[k] == 1
      duplicated_keys << k
      puts "#{k.dump}(#{k.encoding})=#{v.dump}"
    end
    puts
    duplicated_keys.uniq.each do |k|
      puts "#{k.dump}(#{k.encoding})=#{ENV[k].dump}"
    end
    puts
    orig_replace(h)
  end
end
vagrant@buster:/tmp/t$ env PATH=/var/www/docs.ruby-lang.org/shared/bundle/ruby/2.7.0/bin:/snap/bin:$PATH DEBIA
N_DISABLE_RUBYGEMS_INTEGRATION=1 bundle exec env -u RUBYOPT ruby -r/tmp/dump2 -r/snap/ruby/189/lib/ruby/gems/2
.7.0/gems/bundler-2.1.4/lib/bundler/setup -e 'p ENV["RUBYLIB"]'
"GEM_HOME"(UTF-8)="/home/vagrant/.gem"
"RUBYLIB"(UTF-8)="/snap/ruby/189/lib/ruby/gems/2.7.0/gems/bundler-2.1.4/lib:/snap/ruby/189/lib/ruby/2.7.0:/sna
p/ruby/189/lib/ruby/2.7.0/amd64"
"GEM_PATH"(UTF-8)="/home/vagrant/.gem:/snap/ruby/189/lib/ruby/gems/2.7.0"
"GEM_HOME"(UTF-8)="/home/vagrant/.gem"
"GEM_PATH"(UTF-8)="/home/vagrant/.gem:/snap/ruby/189/lib/ruby/gems/2.7.0"
"RUBYLIB"(UTF-8)="/snap/ruby/189/lib/ruby/2.7.0:/snap/ruby/189/lib/ruby/2.7.0/amd64:/snap/ruby/189/lib/ruby/ge
ms/2.7.0/gems/bundler-2.1.4/lib:/snap/ruby/189/lib/ruby/2.7.0:/snap/ruby/189/lib/ruby/2.7.0/amd64"

"GEM_HOME"(UTF-8)="/home/vagrant/.gem"
"RUBYLIB"(UTF-8)="/snap/ruby/189/lib/ruby/gems/2.7.0/gems/bundler-2.1.4/lib:/snap/ruby/189/lib/ruby/2.7.0:/sna
p/ruby/189/lib/ruby/2.7.0/amd64"
"GEM_PATH"(UTF-8)="/home/vagrant/.gem:/snap/ruby/189/lib/ruby/gems/2.7.0"

"/snap/ruby/189/lib/ruby/gems/2.7.0/gems/bundler-2.1.4/lib"
```

#3 - 10/22/2020 03:05 PM - jeremyevans0 (Jeremy Evans)

znz (Kazuhiro NISHIYAMA) wrote in #note-2:

> I dump duplicated keys with encoding.
> Encodings of duplicated keys are same.

Interesting. So it may be that duplicate keys in ENV in general are in issue? We generally think of ENV having unique keys, but I'm guessing that is not enforced by the kernel.

Looking at the implementation of setenv(3)/unsetenv(3) on OpenBSD, it does specify that that there can be multiple environment entries with the same key, so Ruby's ENV implementation should probably not assume there is at most one entry per key.

#4 - 10/29/2020 03:09 PM - jeremyevans (Jeremy Evans)

- Status changed from Open to Closed

Applied in changeset git|c0aeb98aa903334f06758d39c772cb22440d8a4e.

Make ENV.replace handle multiple environ entries with the same key

While it is expected that all environment keys are unique, that is not enforced. It is possible by manipulating environ directly you can call a process with an environment with duplicate keys. If ENV.replace was passed a hash with a key where environ had a duplicate for that key, ENV.replace would end up deleting the key from environ.

The fix in this case is to not assume that the environment key list has unique keys, and continue processing the entire key list in keylist_delete.

Fixes [Bug #17254]

Files

full-env-log.txt	8.96 KB	10/08/2020	znz (Kazuhiro NISHIYAMA)
rdoc-failed-log.txt	2.57 KB	10/08/2020	znz (Kazuhiro NISHIYAMA)