Ruby master - Feature #18143
Add a new method to change GC.stress only in the given block such as GC.with_stress(flag) {...}
09/02/2021 02:45 AM - kou (Kouhei Sutou)

Status: Rejected
Priority: Normal
Assignee:
Target version:

Description
GC.stress = true is useful for detecting GC related crashes. We can use it for debugging GC related problems and testing the problem is solved.

Generally, we need to enable stress mode before the target code block and disable stress mode after the target code block:

```ruby
GC.stress = true
# ... something buggy codes ...
GC.stress = false
```

Or we just enable stress mode before the target code block when the target code block causes a crash:

```ruby
GC.stress = true
# ... something crash codes ...
```

In test code, we must disable stress mode because stress mode slows down test execution:

```ruby
def test_gc
  GC.stress = true
  # ... GC related code ...
  ensure
    GC.stress = false
end
```

We have an utility method in CRuby's test utility: EnvUtil.#under_gc_stress:

https://github.com/ruby/ruby/blob/ab63f6d8543903f177c46634f38e5428655f003b/tool/lib/envutil.rb#L236-L242

```ruby
def under_gc_stress(stress = true)
  stress, GC.stress = GC.stress, stress
  yield
  ensure
    GC.stress = stress
end
module_function :under_gc_stress
```

This feature is useful not only CRuby's test but also other libraries test and debugging a program that has a GC related problem.

How about adding a new singleton method that changes stress mode only in the given block? If we have the method, we don't need to implement a small utility method multiple times for the feature.

API candidates:

GC.with_stress(flag) {...}:

```ruby
module GC
  def self.with_stress(flag)
    flag_old = stress
    self.stress = flag
    yield
    ensure
      self.stress = flag_old
    end
  end
end
```
GC.under_stress {...}:

```ruby
module GC
  def self.under_stress
    flag_old = stress
    self.stress = true
    yield
    ensure
      self.stress = flag_old
    end
  end
end
```

GC.stress(flag = true) {...}:

```ruby
module GC
  def stress flag = true
    if block_given?
      flag_old = Primitive.gc_stress_get
      begin
        GC.stress = flag
        ensure
          GC.stress = flag_old
        end
      else
        Primitive.gc_stress_get
      end
    end
  end
end
```

Note:

- Disadvantage is, GC.stress is a getter method and GC.stress do end is setter method. It can be confusing.
- nobu (Nobuyoshi Nakada) also pointed out that the block is just ignored on the older Ruby versions.

Source of implementation and some discussions: [https://github.com/ruby/ruby/pull/4793](https://github.com/ruby/ruby/pull/4793)

Background:

I'm maintaining some default gems. They have copy of EnvUtil but I don't want to have it for maintainability. If Ruby provides this feature by default, we can use it instead of copying EnvUtil. I know that I need to implement/copy the feature in each default gem until Ruby 3.0 reaches EOL. But we don't have the feature now, I need to copy EnvUtil forever.

History

#1 - 09/02/2021 09:19 AM - byroot (Jean Boussier)

This pattern is so common (not GC.stress, but the "set and restore value in ensure" that I wonder if it wouldn't be worth having some kind of more general solution, e.g.

Object#with:

```ruby
GC.with(:stress, true) do
  # ....
end
```

#2 - 09/02/2021 12:22 PM - Dan0042 (Daniel DeLorme)

byroot (Jean Boussier) wrote in #note-1:

This pattern is so common (not GC.stress, but the "set and restore value in ensure" that I wonder if it wouldn't be worth having some kind of more general solution, e.g.

Object#with:

```ruby
GC.with(:stress, true) do
  # ....
end
```

Wow, that's a really good idea. Indeed the pattern is so common that it would be worth having this in ruby core.
It would probably need to be a keyword though to be efficient, and it could conflict with existing with methods. IMHO it looks much like Python’s with keyword (https://docs.python.org/3/reference/compound_stmts.html#the-with-statement), and I feel blocks are the natural replacement in Ruby.
By having the relevant method taking a block it also encourages/clarifies that some cleanup is typically needed.

I think GC.stress(true) { ... } is fine.
We already have many Foo.open which can both take a block or not. Maybe we make the argument required for clarity and avoiding it to be silently ignored on older Ruby versions?

NB: it was mostly a quick thought, I don’t plan to formally propose it.

That being said:

need to be a keyword though to be efficient

I don’t quite see why a keyword would be needed for efficiency, neither do I see why it would need to be particularly efficient. This pattern is generally used very high up the stack and rarely in hotspots. Even a pure Ruby implementation would be fine IMHO.

it could conflict with existing with methods.

If it’s defined in Object or Kernel, any existing with method would override it, so I don’t see how it could conflict. You could also argue that it would mostly be useful on classes and module and define it in Module.

IMHO it looks much like Python’s with keyword

Only vary superficially. Python’s with protocol allow you to emulate the Ruby pattern:

```
def foo
    # do something: Python's '__enter__()'
yield
ensure
    # cleanup: Python's '__exit__()'
end
```

I feel blocks are the natural replacement in Ruby.

Well, that’s what this suggestion is, a block. It’s just a general solution to setting and restoring an attribute instead of an ad hoc solution for just GC.stress.

Some people may expect that this API enables the GC stress mode only on the current thread, but actually the configuration would be process-global like Dir.chdir.

```
Thread.new { 
    sleep 1
    # The GC stress mode is enabled here
}
```

Is it okay?

Some people may expect that this API enables the GC stress mode only on the current thread, but actually the configuration would be process-global like Dir.chdir.

```
Thread.new { 
    sleep 0.1
}
```

I think that it’s OK. Dir.chdir has also the same behavior:

```
original_pwd = Dir.pwd
Thread.new { 
    loop do
        p Dir.pwd #=> original_pwd then "/tmp"
    end
}
```
Dir.chdir("/tmp") { 
  sleep 1 
}

We can document the behavior explicitly if it's a problem.

#7 - 09/16/2021 05:02 AM - matz (Yukihiro Matsumoto)
- Status changed from Open to Rejected

I reject this proposal, for several reasons:

- it's very easy to implement the method using existing GC.stress and begin ... ensure by yourself
- if it's built-in, users may expect too much (thread-safety is one of them)

Matz.