How to specify `frozen_string_literal: true`

Kazuhiro NISHIYAMA

RubyKaigi 2017 LT
2017/09/19

Powered by Rabbit 2.2.1

self.introduce

- One of Ruby committers
- Twitter, GitHub: `@znz`

What’s `frozen_string_literal` magic comment

- Specify string literals are frozen or not per file
- Frozen string literals are faster
  - Generate less objects
  - Reduce GC

Example

- After coding if exist

example:

```
# -*- coding: utf-8 -*-
# frozen_string_literal: true
p ''.frozen? #=> true
```

Past release

- Add `frozen_string_literal: false` for almost `*.rb` files when Ruby 2.3.0
- For compatibility with `--enable=frozen-string-literal` command line option of ruby

Changes in this year

- Recent Ruby programs tend to specify `true`, but ruby standard libraries are `false`.
- So I changed to `frozen_string_literal: true` in some files which assigned to no maintainer in doc/maintainers.rdoc.

Review points before change

Almost `RuntimeError: can't modify frozen String` points are:

- `String#<<`
- bang methods (e.g. `String#sub!`)

Overlooked modification

- `IO#read(length, out_buffer)`
  - this buffer is overlooked

Find modified string literal

- --debug=frozen-string-literal command line option is very useful

examples:

```
ruby --debug=frozen-string-literal test.rb
make test-all TESTS='-v base64' RUBYOPT=--debug=frozen-string-literal
```

Tests

- `--debug=frozen-string-literal` needs to run codes.
- So tests are very important.

No change if can’t

- If no tests, there is no need to forcibly change it.
- I think working codes are useful than broken faster codes.

mkmf.rb

- mkmf.rb has many tests
- But mkmf.rb is too complex to me in a short time
- So I remain false

Conclusion

- No tests may overlook modifications
- Use `frozen_string_literal: true` if you can
- You can use `frozen_string_literal: false` in some cases for compatibility for `--enable=frozen-string-literal`