Ruby - Misc #20191

Deprecate magic encoding comment

01/17/2024 05:08 PM - kddnewton (Kevin Newton)

Status:	Rejected	
Priority:	Normal	
Assignee:		

Description

I would like to ask that we deprecate the magic encoding comment, and instead require all source files to be encoded in UTF-8.

There would be many benefits to the performance of both the parser and compiler. It would also help to simplify both. For example, right now a string literal in a file encoded in US-ASCII can result in 3 different encodings, depending on its internal bytes. If the file is encoded in UTF-8, it can only be a UTF-8 string.

The encoding comment itself is not very commonly used in gems. If you take the top 100 most downloaded gem versions from rubygems.org and look at the resolved encoding of all of the files, you get:

- UTF-8: 11554
- ASCII-8BIT: 35
- US-ASCII: 10

For all of the most recent versions of gems on rubygems.org, you get:

- UTF-8: 2967421
- US-ASCII: 20130
- ASCII-8BIT: 9237
- ISO-8859-1: 87
- Windows-1252: 45
- Shift_JIS: 32
- Windows-31J: 22
- Windows-1251: 15
- EUC-JP: 11
- GBK: 4
- KOI8-R: 3
- ISO-8859-15: 2
- UTF8-MAC: 1
- invalid: 33

Note that "invalid" here could have worked on some rubies < 3.2 if they used Encoding#replicate.

If we were to change this, the main breaking change concern would be the encoding of strings and symbols that would leave the context of the file by virtue of a constant read/method call. That's why I think it should first be deprecated in a minor release, then removed in the next major. At the moment this would mean for the top 100 gems we would be worried about 0.39% of files, and on rubygems.org as a whole we would be worried about 0.99% of files.

If deprecating the entire encoding comment is unacceptable from a compatibility point of view, I would suggest we try only allowing UTF-8, US-ASCII, and ASCII-8BIT. This would still have a lot of value/simplifications/performance opportunities, at the expense of still needing to be parsed and checked. On the top 100 gems this would mean no files would have to change, and on rubygems.org as a whole it would mean we would be worried about 0.009% of files. That being said, if we're going to deprecate this at all, we should probably just do it all the way to get the full benefit.

(In case you want to check the math, the script used to calculate these is attached.)

History

#1 - 01/18/2024 01:01 AM - hsbt (Hiroshi SHIBATA)

I strongly against this proposal. There is no benefit to break existence Ruby script specified magic comment for resolving their complex issues related encoding like EUC-JP, SHIFT_JIS or something.

#2 - 01/18/2024 01:06 AM - mrkn (Kenta Murata)

We should investigate the real-world application codebases instead of gems on rubygems.org.
I guess there can be unexpectedly many scripts in non-UTF8 encoding historically, such as EUC-JP and Windows-31J.
If so, your proposal will break those many non-UTF-8 applications.

#3 - 01/18/2024 03:14 AM - naruse (Yui NARUSE)

- Status changed from Open to Rejected

You also need to consider applications in addition to gems publicly available on GitHub.
Breaking compatibility forces such users/developers to work such unproductive work.
You must carefully compare the trade off between your development and maintenance cost and Ruby users unproductive cost.
I don't understand why you propose such big incompatible change without concrete evidence of "a lot of value/simplifications/performance opportunities".
Even if Ruby is known as a language which is aggressive to introduce incompatibility, we are always carefully discussing the trade off of the incompatibility and ensure the benefit of it is actually larger than the downside of the incompatibility.

Also note that in Ruby if they change the encoding of source code, the encoding of literals defined in it is also changed.
Affected applications will need to change the logic or convert those strings into the original encoding.
Those changes will be larger than you expect.

#4 - 01/18/2024 03:42 AM - duerst (Martin Dürst)

For the record, I agree with Hiroshi, Kenta, and Yui. The changes from Python 2 to Python 3 didn't work in favor of Python (summarizing Yehuda Katz). The above change would be of a similar magnitude, with similar implications.

The proposed change might work if announced very long-term, e.g. for 2030 or so. Just doing it now and "hope for the best" is a bad idea.

#5 - 01/18/2024 03:57 AM - kddnewton (Kevin Newton)

> The proposed change might work if announced very long-term, e.g. for 2030 or so. Just doing it now and "hope for the best" is a bad idea.

To be clear, that's exactly what I'm proposing. Matz has indicated Ruby 4 would be around 2030. I was suggesting to deprecate in a minor version and remove in a major version.

Also, I very much do not understand your analogy to python 2/3. Python 2/3 changed the default encoding for string literals. To be clear in case there was confusion, I'm talking about the encoding of the source file, not removing encoding support from ruby or changing the default encoding of string literals.

#6 - 01/24/2024 05:22 PM - rubyFeedback (robert heiler)

kddnewton wrote:

> Python 2/3 changed the default encoding for string literals.

I think you need to consider to include the totality of the situation. For instance, we had print() and print in python as another major change; I had to adjust most of my old python scripts just for that alone. (I have significantly more ruby files though).

To the actual topic at hand - personally I tend to use this header:

```
#!/usr/bin/ruby -w
# Encoding: UTF-8
# frozen_string_literal: true
# ============================================================================ #
```

I actually aliased that via the commandline, and automatically assign it to the xorg-buffer (on Linux), to then copy/paste it into a new .rb file. Or I just auto-generate a new .rb file from scratch, which fills up a default template I use for ruby classes. There may not be a huge need for the above four lines, but I kind of use that since ... I think ten years or so by now, something like that.

I don't have a huge pro or con reason on the proposed change itself, by the way. I use UTF-8 by default, and I think I have not used the only other encoding I still use these days (binary) for a very long time - at the least not as a main encoding. I did, however had, initially have some problems transitioning into UTF-8, and that was a bit annoying (my old YAML files were not encoded in UTF-8 either, so I had to change all of them as well; it is often easier when there are small-ish changes, than when everything comes down at once, which is one reason why ruby 1.8.x towards ruby 2.x was not so simple initially, and a lack of documentation too, by the way. For instance, barely any website gave good, thorough explanation on how to handle problems along the transition path, in particular when it came to problems one may encounter along the way).

There is one part I disagree with, though, and this is that only rubygems.org is evaluated. There is more ruby code out there than "merely" hosted on rubygems.org, so we need to somehow consider this as well.

Perhaps for a ruby 4 roadmap towards 2030 this can be kept in mind, but even then I think the objective trade-offs (advantages and benefits) should be kept in mind. Personally it won't affect me much, I think, as I use UTF-8 all the time. I'd probably not even change my auto-generating code, as I have gotten so used to it as a habit, even if it were no longer necessary. (Although, admittedly, IF it were not necessary, then one can argue that it could be omitted, thus saving some space from files. Even then I think it is a fairly minor inconvenience to keep it really. If we want to improve ruby for ruby 4, then perhaps we should consider a "ruby as fast as C", e. g. a crystal-like ruby, but not crystal. I digress though.)

Files

gems.rb	4.33 KB	01/17/2024	kddnewton (Kevin Newton)