Ruby master - Bug #14240

warn four special variables: $; $, $/ $\

12/26/2017 08:08 AM - akr (Akira Tanaka)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:		**Backport:**	2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: UNKNOWN

Description

I think the four special variables for separators should be deprecated.

```
$/    input record separator (default argument for "gets")
$\    output record separator ("print" prints it at last)
$,    default separator for Array#join and print
$;    default separator for String#split
```

I feel many program doesn't work if they are set to non-default value.

Since they are global, not thread local,
we can not change these variables safely in a multi threaded program.

So, I think we should warn them (and delete them in future).

Related issues:

Related to Ruby master - Feature #14138: Define English.rb aliases by default...	**Closed**	
Related to Ruby master - Feature #5977: Remove $, and avoid perlish global va...	**Rejected**	**02/07/2012**

History

#1 - 12/26/2017 08:12 AM - matz (Yukihiro Matsumoto)

- Related to Feature #14138: Define English.rb aliases by default and eliminate the library added

#2 - 12/26/2017 08:13 AM - matz (Yukihiro Matsumoto)

Agreed.

Besides that, we should warn for $= and $., I think.

Matz.

#3 - 12/26/2017 09:08 AM - normalperson (Eric Wong)

Shouldn't English posts be on ruby-core instead of ruby-dev?

matz@ruby-lang.org wrote:

> Agreed.
>
> Besides that, we should warn for $= and $., I think.

I find $., $\, and $/ useful for oneliners, at least. $.
especially

I'm fine with awk-compatible English.rb names ($NR, $ORS, $RS)
by default, but I do not like the long names in English.rb.

I like having some awk and Perl-isms in Ruby :>

#4 - 12/26/2017 09:08 AM - normalperson (Eric Wong)

Shouldn't English posts be on ruby-core instead of ruby-dev?

matz@ruby-lang.org wrote:

> Agreed.
>
> Besides that, we should warn for $= and $., I think.

I find $., $\, and $/ useful for oneliners, at least. $.
especially

I'm fine with awk-compatible English.rb names ($NR, $ORS, $RS)
by default, but I do not like the long names in English.rb.

I like having some awk and Perl-isms in Ruby :>

#5 - 12/26/2017 09:36 AM - akr (Akira Tanaka)

2017-12-26 17:55 GMT+09:00 Eric Wong normalperson@yhbt.net:

> Shouldn't English posts be on ruby-core instead of ruby-dev?

Oops. Sorry.

> > Besides that, we should warn for $= and $., I think.
>
> I find $., $\, and $/ useful for oneliners, at least. $.
> especially

Hm. One idea to support oneliners is that warn the
special variables when -e option is not given.
--
Tanaka Akira

#6 - 12/26/2017 09:36 AM - akr (Akira Tanaka)

2017-12-26 17:55 GMT+09:00 Eric Wong normalperson@yhbt.net:

> Shouldn't English posts be on ruby-core instead of ruby-dev?

Oops. Sorry.

> > Besides that, we should warn for $= and $., I think.
>
> I find $., $\, and $/ useful for oneliners, at least. $.
> especially

Hm. One idea to support oneliners is that warn the
special variables when -e option is not given.
--
Tanaka Akira

#7 - 12/26/2017 11:08 AM - shevegen (Robert A. Heiler)

By the way, the awk-inspired names such as $NR, $ORS, $RS,
also do not tell me anything. :-)

In fairness, I also have to admit that the english names also
do not always tell me that much more. Depends on the name.

http://ruby-doc.org/stdlib/libdoc/English/rdoc/English.html

```
$\ = ' -- '
"waterbuffalo" =~ /buff/
print $', $$, "\n"
```

versus

```
require "English"

$OUTPUT_FIELD_SEPARATOR = ' -- '
"waterbuffalo" =~ /buff/
```

```
print $POSTMATCH, $PID, "\n"
```

The second variant is a tiny bit more readable to me, but
I also can not tell you what $OUTPUT_FIELD_SEPARATOR really
means, without having to look at the docu. :)
And $POSTMATCH hmm... I can try to make a guess, but I am
not sure. I'd have to look at the docu. :D

Only $PID I can infer at once... to mean the PID of a
process. I like that name. Please tell me that it does
not mean anything else than PID ... :D

My brain takes more time for the first code variant and
in general, I try to write only very simply code in ruby
(which is also why I am absolutely bad at golfing and
one-liners, but I understand if people want to be able
to be super-succinct ... I remember the xmas tree in
code golfing, that's more like art than code though).

What actually makes me not use "English" there is the require
line - see also headius' suggestion elsewhere. I liked the recent
change in regards to pp for similar reason, less to type. :)
(Although I was using pp a LOT nonetheless, it is too useful to me
to not make use of it. It just is less typing for me now
past 2.5.x)

Anyway, I digress so I better stop now.

#8 - 12/26/2017 08:08 PM - normalperson (Eric Wong)

Tanaka Akira akr@fsij.org wrote:

> > Besides that, we should warn for $= and $., I think.
>
> 2017-12-26 17:55 GMT+09:00 Eric Wong normalperson@yhbt.net:
>
> > I find $., $\, and $/ useful for oneliners, at least. $.
> > especially
>
> Hm. One idea to support oneliners is that warn the
> special variables when -e option is not given.

Can we have the warnings forever? (No removal, ever).
I would be happy with that :)

One important thing about oneliners is you won't find many
examples of them in code search engines to judge popularity.

#9 - 12/26/2017 08:08 PM - normalperson (Eric Wong)

Tanaka Akira akr@fsij.org wrote:

> > Besides that, we should warn for $= and $., I think.
>
> 2017-12-26 17:55 GMT+09:00 Eric Wong normalperson@yhbt.net:
>
> > I find $., $\, and $/ useful for oneliners, at least. $.
> > especially
>
> Hm. One idea to support oneliners is that warn the
> special variables when -e option is not given.

Can we have the warnings forever? (No removal, ever).
I would be happy with that :)

One important thing about oneliners is you won't find many
examples of them in code search engines to judge popularity.

#10 - 12/26/2017 09:15 PM - Eregon (Benoit Daloze)

I agree.
These global variables can only be useful in tiny scripts,
and even then I believe none of them are idiomatic Ruby code.

All of these are shortcuts saving at best a couple characters,
but also threaten any usage of gets/print/split/join methods in larger programs.
Giving an argument gets/print/split/join is much clearer and safer.

I was already thinking along the same lines 6 years ago about $, in https://bugs.ruby-lang.org/issues/5977,
after being caught by a bug of some library setting $, and breaking the rest in very surprising ways iirc.

#11 - 12/26/2017 09:15 PM - Eregon (Benoit Daloze)

- Related to Feature #5977: Remove $, and avoid perlish global variables added

#12 - 01/24/2018 08:22 AM - akr (Akira Tanaka)

We discussed this issue at today's developper meeting.

We (including matz) agree produce warnings for 5 variables ($; $, $/ $\ $.) when it is written in
ruby code except -e argument.

The warning is produced at compile time.
So the warning is produced once for each occurrence (even if it occurs in a loop).

Note that $= already produces a warning since ruby 1.9.

#13 - 01/25/2018 10:09 AM - nobu (Nobuyoshi Nakada)

I wonder that aliased variables also should be warned, $-0, $-F, and aliases in English.rb.

Currently, aliases of $KCODE are also warned.
In other words, the feature of $KCODE is warned (and has no effect now).

Should we warn these four variable names, or their features?

#14 - 07/26/2019 11:19 PM - jeremyevans0 (Jeremy Evans)

- File warn-5-gvars.patch added

Attached is a patch that adds a deprecation warning for the 5 global variables ($; $, $/ $\ $.), unless inside ruby -e. However, there is use of these variables inside Ruby that should be fixed if we plan to make this change. I've already seen the following issues:

```
/path/ruby/lib/erb.rb:907: warning: global variable $. is deprecated
../.././ext/ripper/tools/preproc.rb: warning: global variable $/ is deprecated
/path/ruby/lib/rubygems/specification.rb:14: warning: global variable $. is deprecated
```

It is likely there are more issues than just these three.

#15 - 07/27/2019 01:54 AM - nobu (Nobuyoshi Nakada)

It does not match to test rb_warn with assert_warning.
Should be rb_warn+assert_warn or rb_warning+assert_warning.

And non-default $; and $, are warned now.
Do you think these **names** should be warned too?

#16 - 07/29/2019 07:03 PM - jeremyevans0 (Jeremy Evans)

- File warn-5-gvars-v2.patch added

nobu (Nobuyoshi Nakada) wrote:

> It does not match to test rb_warn with assert_warning.
> Should be rb_warn+assert_warn or rb_warning+assert_warning.

Thanks. I switched to assert_warn, as I think deprecation warnings should be a regular warning, not a verbose warning. Verbose warnings are best used for warnings like unused variables. I only think verbose deprecation warnings should be used if there is a plan to follow it up with a non-verbose deprecation warning before removing it.

> And non-default $; and $, are warned now.

> Do you think these **names** should be warned too?

I don't feel strongly about it. The $; and $, are currently run-time warnings based on the specific values passed when setting the values. akr's previous comment indicated that the warnings of the five variables should be at compile time.

One problem with the implementation in my previous patch is that it warns at the wrong location. Switching from rb_warn to rb_compile_warn fixes that issue.

Attached is an updated patch, as well as a few fixes where the global variables are currently used. Another file where they are also currently used is in:

```
/path/ruby/lib/rubygems/stub_specification.rb:114: warning: global variable $. is deprecated
/path/ruby/lib/rubygems/stub_specification.rb:134: warning: global variable $. is deprecated
```

But a fix there should be filed upstream.

There are probably additional cases where these global variables are used internally. At the very least all tests that use them probably need modification to hide the warnings.

#17 - 07/30/2019 01:52 AM - nobu (Nobuyoshi Nakada)

jeremyevans0 (Jeremy Evans) wrote:

> > And non-default $; and $, are warned now.
> > Do you think these **names** should be warned too?
>
> I don't feel strongly about it. The $; and $, are currently run-time warnings based on the specific values passed when setting the values. akr's previous comment indicated that the warnings of the five variables should be at compile time.

There are tons of that warnings, and it is not easy to suppress parser warning within the given file only.

> One problem with the implementation in my previous patch is that it warns at the wrong location. Switching from rb_warn to rb_compile_warn fixes that issue.

It should be rb_warn1 and WARN_I(c) for ripper.
And a typo f.line_no in template/encdb.h.tmpl.

#18 - 08/26/2019 12:58 AM - jeremyevans0 (Jeremy Evans)

- File warn-5-gvars-v3.patch added

nobu (Nobuyoshi Nakada) wrote:

> jeremyevans0 (Jeremy Evans) wrote:
>
> > > And non-default $; and $, are warned now.
> > > Do you think these **names** should be warned too?
> >
> > I don't feel strongly about it. The $; and $, are currently run-time warnings based on the specific values passed when setting the values. akr's previous comment indicated that the warnings of the five variables should be at compile time.
>
> There are tons of that warnings, and it is not easy to suppress parser warning within the given file only.

Agreed. I'm fine dropping these compile warnings if the run-time warnings are considered sufficient.

> > One problem with the implementation in my previous patch is that it warns at the wrong location. Switching from rb_warn to rb_compile_warn fixes that issue.
>
> It should be rb_warn1 and WARN_I(c) for ripper.
> And a typo f.line_no in template/encdb.h.tmpl.

Thanks for the review. Attached is an updated patch that fixes these issues.

#19 - 08/26/2019 04:33 AM - nobu (Nobuyoshi Nakada)

jeremyevans0 (Jeremy Evans) wrote:

nobu (Nobuyoshi Nakada) wrote:

> jeremyevans0 (Jeremy Evans) wrote:
>
> > And non-default $; and $, are warned now.
> > Do you think these **names** should be warned too?
>
> I don't feel strongly about it. The $; and $, are currently run-time warnings based on the specific values passed when setting the values. akr's previous comment indicated that the warnings of the five variables should be at compile time.
>
> There are tons of that warnings, and it is not easy to suppress parser warning within the given file only.

Agreed. I'm fine dropping these compile warnings if the run-time warnings are considered sufficient.

Another point is that compile warnings are for the names.
Use of aliases of them like English.rb will be still silent.

So, this question.

> Should we warn these four variable names, or their features?

Files

warn-5-gvars.patch	2.31 KB	07/26/2019	jeremyevans0 (Jeremy Evans)
warn-5-gvars-v2.patch	5.44 KB	07/29/2019	jeremyevans0 (Jeremy Evans)
warn-5-gvars-v3.patch	5.4 KB	08/26/2019	jeremyevans0 (Jeremy Evans)