Ruby trunk - Feature #12882

Add caller/file/line information to internal Kernel#warn calls

10/28/2016 09:01 PM - jeremyevans0 (Jeremy Evans)

Status:	Closed
Priority:	Normal
Assignee:	matz (Yukihiro Matsumoto)
Target version:	

Description

Most internal uses of Kernel#warn do not begin with caller/file/line information, making debugging such warnings more difficult, and hindering filtering/processing of warnings on a per-file/directory/gem basis when overriding Warning.warn (a new feature in ruby 2.4).

I think it would be better if internal calls to Kernel#warn in ruby code included caller information, so the warnings generated are similar to those generated by rb_warn in C code.

I'm attaching an initial attempt at this. There was a lot of internal inconsistency in existing warning messages that include caller information. I choose to standardize on caller(1,1).first. Specifying 1 as the length argument to caller provides a 5-10x speedup compared to the default of nil as the length argument, and since warning messages only use the first line, there's no need to generate additional lines. I also benchmarked using caller_locations, but that didn't seem to perform better, probably because the result is just converted to a string anyway.

Some existing warning messages that include caller information use #caller but attempt to strip out the method information via gsub. I'm guessing it would be better to use #caller_locations. However, that's kind of cumbersome right now, if we want to do that, we should probably add a method to Thread::Backtrace::Location that returns just the absolute_path and lineno as a string, so we could standardize on caller_locations(1,1).first.file_and_line (or whatever the method is called).

There are a few questionable changes in the attached patch, such as the use of caller(0,1) in certain cases (when the changes were at top level), or including caller information in warning messages that were only output in DEBUG mode.

The attached patch does not include changes for rdoc and rubygems, as those are maintained in separate repositories. Assuming that this patch or a similar one is accepted, I can submit patches to both rdoc and rubygems for their warning messages.

Related issues:

Related to Ruby trunk - Bug #14262: ArgumentError (negative level (-1)) when ...	**Closed**

Associated revisions

Revision 9fa87ebe - 12/12/2017 12:10 PM - shyouhei (Shyouhei Urabe)

NEWS entry for [Feature #12882]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@61156 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 61156 - 12/12/2017 12:10 PM - shyouhei (Shyouhei Urabe)

NEWS entry for [Feature #12882]

Revision 61156 - 12/12/2017 12:10 PM - shyouhei (Shyouhei Urabe)

NEWS entry for [Feature #12882]

Revision 61156 - 12/12/2017 12:10 PM - shyouhei (Shyouhei Urabe)

NEWS entry for [Feature #12882]

History

#1 - 11/25/2016 09:35 AM - matz (Yukihiro Matsumoto)

LGTM.

Matz.

#2 - 11/25/2016 09:41 AM - matz (Yukihiro Matsumoto)

LGTM, as long as we add a functionality to warn to include those info.

Matz.

#3 - 11/26/2016 04:19 AM - jeremyevans0 (Jeremy Evans)

matz, did you want to modify the Kernel#warn API? Currently, I don't think modifying the Kernel#warn API can be done in a backwards compatible way, as Kernel#warn accepts an arbitrary number of arguments and converts them all to strings. We could support this with keyword arguments, using an API like warn("message", caller: 1), but it wouldn't be completely backward compatible as that currently prints the message and {:caller=>1} to stderr. If such a change is acceptable, I can work on implementing this. Since Kernel#warn would add an extra call frame, we would have to add 1 to the :caller option.

Did you want to modify Kernel#warn's behavior to automatically assume caller: 1 if no :caller keyword argument is given? Most of the stdlib calls to Kernel#warn use caller(1,1) before calling Kernel#warn. This would be a bigger break to backwards compatibility, as external callers to Kernel#warn (outside the stdlib) that included caller information would result in the caller information appearing twice in the error output.

If you want something completely backwards compatible, we either need to continue to include the caller information manually in error messages passed to Kernel#warn (as this patch does), or we need to add a new method (which seems undesireable to me). Assuming you want to keep backwards compatibility and include caller information manually in calls to Kernel#warn, what are your thoughts on adding Thread::Backtrace::Location#file_and_line (or similar method) and then using caller_locations(1,1).first.file_and_line in most of the warning messages?

#4 - 12/12/2016 03:33 PM - jeremyevans0 (Jeremy Evans)

matz, can you update this ticket and let me know which of these four options you prefer:

1) Add :caller keyword argument to Kernel#warn (not backwards compatible API-wise).
2) Change Kernel#warn to automatically prepend caller(1,1) (not backwards compatible output-wise).
3) Add new method (e.g Kernel#warning) that accepts :caller keyword argument or automatically prepends caller(1,1).
4) Prepend caller information to strings passed to Kernel#warn (as this patch does).

#5 - 12/13/2016 05:08 AM - duerst (Martin Dürst)

- Assignee set to matz (Yukihiro Matsumoto)

#6 - 10/19/2017 05:52 AM - matz (Yukihiro Matsumoto)

- Status changed from Open to Feedback

I prefer option 1. The issue is the name (and semantics) of the keyword argument.
caller: seems ambiguous. How about uplevel:n to prepend caller(n,1) to the message.

Matz.

#7 - 10/19/2017 06:19 AM - nobu (Nobuyoshi Nakada)

```
diff --git a/error.c b/error.c
index 9bd8c31386..499212cdd8 100644
--- a/error.c
+++ b/error.c
@@ -321,10 +321,28 @@ end_with_asciichar(VALUE str, int c)
 static VALUE
 rb_warn_m(int argc, VALUE *argv, VALUE exc)
 {
-    if (!NIL_P(ruby_verbose) && argc > 0) {
+    VALUE opts, uplevel = Qnil;
+
+    if (!NIL_P(ruby_verbose) && argc > 0 &&
+   (argc = rb_scan_args(argc, argv, "*:", NULL, &opts)) > 0) {
     VALUE str = argv[0];
-   if (argc > 1 || !end_with_asciichar(str, '\n')) {
-       str = rb_str_tmp_new(0);
+   if (!NIL_P(opts)) {
+       static ID kwds[1];
+       if (!kwds[0]) CONST_ID(kwds[0], "uplevel");
+       rb_get_kwargs(opts, kwds, 0, 1, &uplevel);
+       if (uplevel == Qundef) {
+       uplevel = Qnil;
+       }
+       else if (!NIL_P(uplevel)) {
+       uplevel = LONG2NUM((long)NUM2ULONG(uplevel) + 1);
+       uplevel = rb_vm_thread_backtrace(1, &uplevel, GET_THREAD()->self);
+       if (!NIL_P(uplevel)) {
+           uplevel = rb_ary_entry(uplevel, 0);
+       }
+       }
+   }
+   if (argc > 1 || !NIL_P(uplevel) || !end_with_asciichar(str, '\n')) {
+       str = NIL_P(uplevel) ? rb_str_tmp_new(0) : rb_str_cat_cstr(uplevel, ": ");
```

```
        RBASIC_SET_CLASS(str, rb_cWarningBuffer);
        rb_io_puts(argc, argv, str);
        RBASIC_SET_CLASS(str, rb_cString);
```

#8 - 10/24/2017 10:55 PM - jeremyevans0 (Jeremy Evans)

- File 0001-Add-uplevel-keyword-to-Kernel-warn-and-use-it.patch added

Attached is a patch to add the uplevel keyword to Kernel#warn. It is based on nobu's patch, but I've modified it so that it only uses the path and lineno of the backtrace location (not the method name/label), and it also prepends the string "warning: " before the message. This way if you use the uplevel keyword, you get the same format that rb_warn would use.

This patch includes changes to lib (except for bundler, rubygems, and rdoc) to use the uplevel keyword when calling Kernel#warn. It also contains changes to net/ftp and net/imap to use Kernel#warn for warnings instead of operating directly on $stderr, and changes to lib/cgi/core and tempfile to use $stderr directly instead of Kernel#warn for debug messages that are not warnings.

#9 - 12/12/2017 12:44 AM - jeremyevans0 (Jeremy Evans)

The previously posted patch still applies and the tests still pass with it. I would like to get this committed before 2.5.0, as it makes the change for Kernel#warn to call Warning.warn (#12944) much more useful.

#10 - 12/12/2017 12:10 PM - shyouhei (Shyouhei Urabe)

- Status changed from Feedback to Closed

Applied in changeset trunk|r61156.

NEWS entry for [Feature #12882]

#11 - 12/31/2017 10:33 AM - znz (Kazuhiro NISHIYAMA)

- Related to Bug #14262: ArgumentError (negative level (-1)) when `warn "test message", uplevel: -2` added

Files

0001-Prepend-warning-messages-with-caller-information.patch	30.6 KB	10/28/2016	jeremyevans0 (Jeremy Evans)
0001-Add-uplevel-keyword-to-Kernel-warn-and-use-it.patch	26 KB	10/24/2017	jeremyevans0 (Jeremy Evans)