Ruby trunk - Feature #14143

Thread.report_on_exception should be true by default

11/29/2017 05:51 PM - Eregon (Benoit Daloze)

Status:	Closed	
Priority:	Normal	
Assignee:	Eregon (Benoit Daloze)	
Target version:	2.5	

Description

Extracted from #6647 to focus on the default value now that the feature is implemented.

I strongly believe we should have Thread.report_on_exception = true by default.

It only adds some extra stderr output for apps which let threads die, which is very rarely intended.
If it is intended, then one can use Thread.current.report_on_exception = false
to clarify it's OK for that thread to die and the failure is handled by the app on Thread#join.

I enabled Thread.report_on_exception=true by default in ruby/spec, see https://github.com/ruby/spec/pull/517,
the only cases needing Thread.current.report_on_exception=false
are the specs testing report_on_exception itself and Thread#join/value/status/raise.

Enabling it for test-all shows a fair amount of extra output and failures, which I would bet some of them are bugs in the tests (I already found one, r60854 & r60870),
and other tests should simply more carefully test what they expect
(for instance assert_raise() inside the Thread just around the code raising an exception and join the Thread).

I am willing to help to reduce the extra output and failures in test-all,
but I would like a OK from Matz to try enabling Thread.report_on_exception by default.

Dear Matz, do you think it is reasonable to show exceptions killing threads on stderr by default,
instead of silently swallowing them until Thread#join ?
(if there is ever a Thread#join ..., often not or too late, when the rest of the application has crashed)

Associated revisions

Revision 5a3c024d - 12/12/2017 06:43 PM - Eregon (Benoit Daloze)

Set Thread.report_on_exception=true by default to report exceptions in Threads

- [Feature #14143] [ruby-core:83979]
- vm.c (vm_init2): Set Thread.report_on_exception to true.
- thread.c (thread_start_func_2): Add indication the message is caused by report_on_exception = true.
- spec/ruby: Specify the new behavior.
- test/ruby/test_thread.rb: Adapt and improve tests for Thread.report_on_exception and Thread#report_on_exception.
- test/ruby/test_thread.rb, test/ruby/test_exception.rb: Unset report_on_exception for tests expecting no extra output.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@61183 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 61183 - 12/12/2017 06:43 PM - Eregon (Benoit Daloze)

Set Thread.report_on_exception=true by default to report exceptions in Threads

- [Feature #14143] [ruby-core:83979]
- vm.c (vm_init2): Set Thread.report_on_exception to true.
- thread.c (thread_start_func_2): Add indication the message is caused by report_on_exception = true.
- spec/ruby: Specify the new behavior.
- test/ruby/test_thread.rb: Adapt and improve tests for Thread.report_on_exception and Thread#report_on_exception.
- test/ruby/test_thread.rb, test/ruby/test_exception.rb: Unset report_on_exception for tests expecting no extra output.

Revision 61183 - 12/12/2017 06:43 PM - Eregon (Benoit Daloze)

Set Thread.report_on_exception=true by default to report exceptions in Threads

- [Feature #14143] [ruby-core:83979]
- vm.c (vm_init2): Set Thread.report_on_exception to true.
- thread.c (thread_start_func_2): Add indication the message is caused by report_on_exception = true.
- spec/ruby: Specify the new behavior.

- test/ruby/test_thread.rb: Adapt and improve tests for Thread.report_on_exception and Thread#report_on_exception.
- test/ruby/test_thread.rb, test/ruby/test_exception.rb: Unset report_on_exception for tests expecting no extra output.

Revision 61183 - 12/12/2017 06:43 PM - Eregon (Benoit Daloze)

Set Thread.report_on_exception=true by default to report exceptions in Threads

- [Feature #14143] [ruby-core:83979]
- vm.c (vm_init2): Set Thread.report_on_exception to true.
- thread.c (thread_start_func_2): Add indication the message is caused by report_on_exception = true.
- spec/ruby: Specify the new behavior.
- test/ruby/test_thread.rb: Adapt and improve tests for Thread.report_on_exception and Thread#report_on_exception.
- test/ruby/test_thread.rb, test/ruby/test_exception.rb: Unset report_on_exception for tests expecting no extra output.

Revision 2dfbc64f - 12/12/2017 06:44 PM - Eregon (Benoit Daloze)

Rescue expected Interrupt in TupleSpaceTestModule#test_take_bug_8215

- test/rinda/test_rinda.rb (test_take_bug_8215): add rescue for expected exception, which removes the warning by Thread.report_on_exception [Feature #14143].

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@61185 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 61185 - 12/12/2017 06:44 PM - Eregon (Benoit Daloze)

Rescue expected Interrupt in TupleSpaceTestModule#test_take_bug_8215

- test/rinda/test_rinda.rb (test_take_bug_8215): add rescue for expected exception, which removes the warning by Thread.report_on_exception [Feature #14143].

Revision 61185 - 12/12/2017 06:44 PM - Eregon (Benoit Daloze)

Rescue expected Interrupt in TupleSpaceTestModule#test_take_bug_8215

- test/rinda/test_rinda.rb (test_take_bug_8215): add rescue for expected exception, which removes the warning by Thread.report_on_exception [Feature #14143].

Revision 61185 - 12/12/2017 06:44 PM - Eregon (Benoit Daloze)

Rescue expected Interrupt in TupleSpaceTestModule#test_take_bug_8215

- test/rinda/test_rinda.rb (test_take_bug_8215): add rescue for expected exception, which removes the warning by Thread.report_on_exception [Feature #14143].

Revision 6671a826 - 12/12/2017 06:44 PM - Eregon (Benoit Daloze)

Fix tests which fail with extra stderr output when a Thread dies

- [Feature #14143] [ruby-core:83979].

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@61186 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 61186 - 12/12/2017 06:44 PM - Eregon (Benoit Daloze)

Fix tests which fail with extra stderr output when a Thread dies

- [Feature #14143] [ruby-core:83979].

Revision 61186 - 12/12/2017 06:44 PM - Eregon (Benoit Daloze)

Fix tests which fail with extra stderr output when a Thread dies

- [Feature #14143] [ruby-core:83979].

Revision 61186 - 12/12/2017 06:44 PM - Eregon (Benoit Daloze)

Fix tests which fail with extra stderr output when a Thread dies

- [Feature #14143] [ruby-core:83979].

Revision 15689ed7 - 12/12/2017 06:44 PM - Eregon (Benoit Daloze)

Fix test-all tests to avoid creating report_on_exception warnings

- The warnings are shown by Thread.report_on_exception defaulting to true. [Feature #14143] [ruby-core:83979]

- Improves tests by narrowing down the scope where an exception is expected.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@61188 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 61188 - 12/12/2017 06:44 PM - Eregon (Benoit Daloze)

Fix test-all tests to avoid creating report_on_exception warnings

- The warnings are shown by Thread.report_on_exception defaulting to true. [Feature #14143] [ruby-core:83979]
- Improves tests by narrowing down the scope where an exception is expected.

Revision 61188 - 12/12/2017 06:44 PM - Eregon (Benoit Daloze)

Fix test-all tests to avoid creating report_on_exception warnings

- The warnings are shown by Thread.report_on_exception defaulting to true. [Feature #14143] [ruby-core:83979]
- Improves tests by narrowing down the scope where an exception is expected.

Revision 61188 - 12/12/2017 06:44 PM - Eregon (Benoit Daloze)

Fix test-all tests to avoid creating report_on_exception warnings

- The warnings are shown by Thread.report_on_exception defaulting to true. [Feature #14143] [ruby-core:83979]
- Improves tests by narrowing down the scope where an exception is expected.

Revision 2e315baf - 12/14/2017 01:08 PM - Eregon (Benoit Daloze)

The main Thread should have report_on_exception=true for consistency

- Adapt test and add specs.
- See [Feature #14143] [ruby-core:84227]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@61237 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 61237 - 12/14/2017 01:08 PM - Eregon (Benoit Daloze)

The main Thread should have report_on_exception=true for consistency

- Adapt test and add specs.
- See [Feature #14143] [ruby-core:84227]

Revision 61237 - 12/14/2017 01:08 PM - Eregon (Benoit Daloze)

The main Thread should have report_on_exception=true for consistency

- Adapt test and add specs.
- See [Feature #14143] [ruby-core:84227]

Revision 61237 - 12/14/2017 01:08 PM - Eregon (Benoit Daloze)

The main Thread should have report_on_exception=true for consistency

- Adapt test and add specs.
- See [Feature #14143] [ruby-core:84227]

Revision 0b81d101 - 12/14/2017 07:26 PM - Eregon (Benoit Daloze)

Add [Feature #14143] to NEWS

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@61265 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 61265 - 12/14/2017 07:26 PM - Eregon (Benoit Daloze)

Add [Feature #14143] to NEWS

Revision 61265 - 12/14/2017 07:26 PM - Eregon (Benoit Daloze)

Add [Feature #14143] to NEWS

Revision 61265 - 12/14/2017 07:26 PM - Eregon (Benoit Daloze)

Add [Feature #14143] to NEWS

History

#1 - 11/29/2017 07:07 PM - Eregon (Benoit Daloze)

I think it is time to do this, and most rubyists using threads with Ruby seem to agree with this
(see #6647 and I can say at least JRuby & TruffleRuby implementers want this).

Other threads should write a backtrace just like the main thread when they die due to an exception,
and not hope there will be a Thread#join soon enough.
Threads are heavyweight in Ruby, they are rarely if ever used as an isolation mechanism,
and we cannot assume Thread#join will always be called.
Proper exception handling in threads should anyway not rely on Thread#join semantics
but deal with the exception and communicate the failure with other threads explicitly.

A one-liner example to illustrate:

```
$ ruby -e 'q=Queue.new; t=Thread.new{ q.push "abc".starts_with?("foo") }; p q.pop; t.join'
```

This small example will show a deadlock message, but not what is the source of the problem:

```
-e:1:in `pop': No live threads left. Deadlock? (fatal)
1 threads, 1 sleeps current:0x00005599e5464bf0 main thread:0x00005599e512e5e0
* #<Thread:0x00005599e5161eb0 sleep_forever>
   rb_thread_t:0x00005599e512e5e0 native:0x00007f4493ac8700 int:0
   -e:1:in `pop'
   -e:1:in `<main>'
    from -e:1:in `<main>'
```

Thread.report_on_exception=true by default would clearly show the misspelled method name:

```
#<Thread:0x000055bbcaf3cb00@-e:1 run> terminated with exception:
-e:1:in `block in <main>': undefined method `starts_with?' for "abc":String (NoMethodError)
Did you mean?  start_with?
```

-d/--debug is not a good fit here because:

- it changes the semantics to abort_on_exception.
- it outputs a lot of extra exceptions (in bigger programs), yet without showing the backtrace of exceptions.
- it does not differentiate between exceptions which killed a Thread and exceptions which are rescue'd.

report-on-GC as proposed in #6647 is of no use here since the Thread might be GC-reachable long after it dies,
and it adds extra non-determinism which has no place in error reporting.

#2 - 11/29/2017 08:26 PM - headius (Charles Nutter)

I think everyone knows where I stand.

#3 - 11/29/2017 08:33 PM - headius (Charles Nutter)

Perhaps "on by default" would be more palatable to people if we could also specify *how* to handle these unhandled exceptions?

In Java, all threads will report a bubbled-out exception if you do not specify a handler for those exceptions. The handler can be set on a global basis or per-thread. So basically, the default is that bubbled exceptions get fed to a default handler that reports thread death.

In Ruby, it might look like this:

```
Thread.report_on_exception = true # default
Thread.on_exception {|ex| handle ex in app-appropriate way}
```

Report would then basically mean "send the exception to the default handler" and you can change the handler to something that logs to disk or to some online service or whatever.

PROS:

- Consistent reporting of exceptions from threads that specify no handler.
- Threads won't quietly die anymore.
- Threads can be made to quietly die or do something else with bubbled exceptions, on a local OR global basis

CONS:

- Existing code that depends on threads quietly dying will now see some error reporting (I'd recommend one line, with full trace in verbose mode)

#4 - 11/29/2017 08:43 PM - headius (Charles Nutter)

I am really liking the flow of

```
Thread.on_exception do
  some stuff
```

```
end
```

#5 - 11/29/2017 09:49 PM - Eregon (Benoit Daloze)

Real-world libraries like Sidekiq implement their own thread exception handler:
https://github.com/mperham/sidekiq/blob/a60a91d3dd857592a532965f0701d285f13f28f1/lib/sidekiq/util.rb#L15-L27

Interestingly, this is named "safe_thread", probably alluding to the fact Thread.new is unsafe/dangerous as it just ignores errors silently.
This would be much nicer with Thread.on_exception(&handler).
However, I think Thread.on_exception should be a separate issue, extending on good defaults from this issue (then the default handler would just be to print to stderr).

What matters most to me and many rubyists is that by default Ruby Threads do not die silently,
and instead the programmer is given a good clue when a part of the program raises an error and it's never handled.
Thread.report_on_exception = true by default would mean, with no extra code, there is already a good handling of exceptions in threads.

Nobody should have to remember to always add Thread.report_on_exception = true or Thread.abort_on_exception = true
on every Ruby program using Thread, or otherwise waste a lot of debugging time.

I am telling this from experience: I am a PhD student working for 3 years on Ruby & concurrency, as well as a test suite maintainer.

#6 - 11/30/2017 08:28 AM - shevegen (Robert A. Heiler)

I have too little experience with threads to meaningfully comment on this.

But I have used Thread-methods before, in particular in ruby-gtk.

I distinctly remember having had to set:

```
Thread.abort_on_exception = true
```

in some of the .rb files.

I do not remember 100% as to why but I believe it was because if I would not do so, I would not be able to see which error caused the ruby GUI app to crash (this started my investigation there, by the way, since I saw crashes that I did not understand). Or the error was somewhere hidden. Ruby-GTK can really generate very long core dumps or exception messages. :)

For short code examples, it was simple to see what was going on but I found that the more complex the widgets become, the harder it is to debug them (another reason why I try to write as simple as code as possible, when I can get away with it).

After I did set Thread to abort on exceptions to true, and read up on Threads, I understand it to some extent so that I can use it just fine ... or use it in a "useful way". But initially when I first noticed problems, I had no real idea what was going on. It should also be said that the ruby-gtk stuff uses some kind of internal main loop as well, which I think has somehow to do again with threads (in glib; my apologies for this bad description here but I really do not know much about threads; I only know that glib has its own loop e. g.:

```
GLib::MainLoop.new(GLib::MainContext.default, true)
```

)

So to me, that first initial step, to find out how to deal with threads in Ruby, was not trivial to find out. In that context I agree with Benoit's comment "Nobody should have to remember" which I think is a fine statement, in particular by people who are new to ruby. It's not so difficult for more experienced, older ruby people but for newcomers, making threads as simple as possible would be good, IMO. (I still have not really looked into fibers and mutexes ... it all seems to be a huge chunk of domain-specific knowledge here to have to learn :D).

#7 - 11/30/2017 10:25 AM - Eregon (Benoit Daloze)

shevegen (Robert A. Heiler) (Robert A. Heiler) wrote:

> So to me, that first initial step, to find out how to deal with threads in Ruby, was not trivial to find out. In that context

> I agree with Benoit's comment "Nobody should have to remember"
> which I think is a fine statement, in particular by people who
> are new to ruby. It's not so difficult for more experienced,
> older ruby people but for newcomers, making threads as simple
> as possible would be good, IMO.

I think it is also a problem even for experienced rubyists.
I often forget to add Thread.{report or abort}_on_exception = true.
It is not easy to remember that every Thread.new{} has to use these extra lines at the beginning
or threads might crash silently with no easy way to debug.

headius (Charles Nutter) was saying on IRC #jruby yesterday that he experiences this problem at least once a month.
enebo (Thomas Enebo) was talking about a similar scenario than yours:
GUI programming with threads in Ruby is harder than it should because errors go unnoticed.

#8 - 12/01/2017 05:23 PM - enebo (Thomas Enebo)

I think clarification on what the semantics should be would be helpful.

Should **ANY** exception raised in a Thread be considered a reasonable way of ending a thread? Yes or No

Yes - You execute your program and it does not work right. Wondering why after doing weird stuff like method ensure blocks to print out the backtrace (yes I did that at one time) you find: Thread.abort_on_exception = true. Once enabling and re-running (assuming it is predictable) you realize you made a typo and a NoMethodError killed your thread.

No - When your program does not work right you see a stack trace on a thread which died from an exception raise.

In No, the semantics is there is no reasonable exception to be thrown which will not generate a backtrace, but compared to Yes we do not get hidden death. Hidden death makes developers frustrated and until they discover abort_on_exception they wonder why this is like this. Even after they learn this they still wonder why this is like this... :)

Besides asking a thread to kill we could always add a special exception which allows a thread to go away silently. I personally think that is too much though.

#9 - 12/12/2017 05:58 AM - matz (Yukihiro Matsumoto)

- Target version set to 2.5

- Assignee set to matz (Yukihiro Matsumoto)

Hi,

OK, it will be merged to 2.5RC1. If something (bad) happens, we will revert it.

Matz.

#10 - 12/12/2017 10:27 AM - Eregon (Benoit Daloze)

matz (Yukihiro Matsumoto) wrote:

> OK, it will be merged to 2.5RC1. If something (bad) happens, we will revert it.

Thank you!
I will provide a patch as soon as possible.

#11 - 12/12/2017 01:56 PM - Eregon (Benoit Daloze)

- Assignee changed from matz (Yukihiro Matsumoto) to Eregon (Benoit Daloze)

We discussed with ko1 (Koichi Sasada) and usa (Usaku NAKAMURA) to add a hint to the warning produced by Thread.report_on_exception.
I am thinking to go with usa (Usaku NAKAMURA)'s suggestion. The hint is between "()":

```
#<Thread:0x000055c6660d0b10@report.rb:3 run> terminated with exception (report_on_exception is true):
Traceback (most recent call last):
    4: from report.rb:4:in `block in <main>'
    3: from report.rb:4:in `times'
    2: from report.rb:5:in `block (2 levels) in <main>'
    1: from report.rb:5:in `times'
report.rb:6:in `block (3 levels) in <main>': unhandled exception
```

Then I think we can discuss possible ways to fix this warning (fix a bug, add rescue, set report to false) in more details
in the documentation of Thread.report_on_exception.

#12 - 12/12/2017 02:07 PM - Eregon (Benoit Daloze)

- File 0001-Set-Thread.report_on_exception-true-by-default-to-re.patch added

The current patch and progress is available at https://github.com/ruby/ruby/compare/trunk...eregon:thread_report_on_exception_by_default

Only the first commit (also attached as patch) is needed to "make exam"
We found a bug in DRb with report_on report_on_exception: #14171.
The second commit fixes it.

The first commit sets Thread.report_on_exception=false in test-all to avoid too many warnings,
due to tests letting threads die, often in purpose. I'm working on fixing them.

#13 - 12/12/2017 06:43 PM - Eregon (Benoit Daloze)

- Status changed from Open to Closed

Applied in changeset trunk|r61183.

Set Thread.report_on_exception=true by default to report exceptions in Threads

- [Feature #14143] [ruby-core:83979]
- vm.c (vm_init2): Set Thread.report_on_exception to true.
- thread.c (thread_start_func_2): Add indication the message is caused by report_on_exception = true.
- spec/ruby: Specify the new behavior.
- test/ruby/test_thread.rb: Adapt and improve tests for Thread.report_on_exception and Thread#report_on_exception.
- test/ruby/test_thread.rb, test/ruby/test_exception.rb: Unset report_on_exception for tests expecting no extra output.

#14 - 12/12/2017 06:51 PM - Eregon (Benoit Daloze)

I committed the change in r61182 and adapted test-all to fix warnings in r61188.
The tests look improved by this change, and few tests need report_on_exception = false.

I noticed oddly the main thread returns false for Thread#report_on_exception:

```
$ ruby -e 'p Thread.report_on_exception'
true
$ ruby -e 'p Thread.current.report_on_exception'
false
$ ruby -e 'Thread.new { p Thread.current.report_on_exception }.join'
true
```

I think it should return true for consistency (even though it doesn't use the same mechanism for reporting errors, it does still reports exceptions finishing its execution).
I'll fix that detail tomorrow.

#15 - 12/14/2017 01:10 PM - Eregon (Benoit Daloze)

Documentation updated and improved in r61216.
The main Thread now also has report_on_exception=true for consistency with r61237 (it is not used though).
I think this feature is ready for 2.5.

#16 - 12/15/2017 02:55 AM - ko1 (Koichi Sasada)

(bikeshed)

Eregon (Benoit Daloze) wrote:

> # terminated with exception (report_on_exception is true):

I reconsider about this line and
#<Thread:0x000055c6660d0b10@report.rb:3 run> terminated with exception (report_on_exception):
is not enough?

We can understand this report is related to report_on_exception (and can google it).

#17 - 12/15/2017 01:11 PM - Eregon (Benoit Daloze)

ko1 (Koichi Sasada) wrote:

> I reconsider about this line and
> #<Thread:0x000055c6660d0b10@report.rb:3 run> terminated with exception (report_on_exception):

> is not enough?
>
> We can understand this report is related to report_on_exception (and can google it).

Ideally, I think referring to the documentation of Thread::report_on_exception directly would be the most helpful, so something like:

```
#<Thread:0x000055c6660d0b10@report.rb:3 run> terminated with exception (see Thread.report_on_exception)
```

Note that if one of your concerns is the line length, we at least can remove the status as it is useless.

```
#<Thread:0x000055c6660d0b10@report.rb:3> terminated with exception (see Thread.report_on_exception)
```

The address seems of little use beyond identifying which Thread is is, which we could use a simpler numbering, or simply skip leading zeros.
We could also show an extra backtrace line for Thread.new, so there is no need to show the source location on the first line:

```
#<Thread:0x000055c6660d0b10> terminated with exception (see Thread.report_on_exception)
Traceback (most recent call last):
    5: from report.rb:3:in `Thread.new'
    4: from report.rb:4:in `block in <main>'
    3: from report.rb:4:in `times'
    2: from report.rb:5:in `block (2 levels) in <main>'
    1: from report.rb:5:in `times'
report.rb:6:in `block (3 levels) in <main>': unhandled exception
```

But it might be harder to correlate with other p Thread.current / Thread#inspect output.

#18 - 10/01/2018 07:01 PM - stefan.sedich (Stefan Sedich)

headius (Charles Nutter) wrote:

> I am really liking the flow of
>
> ```
> Thread.on_exception do
>   some stuff
> end
> ```

Has these been any more discussion/movement around such a feature? my google-fu is failing me, and this is something that would be very useful so that I can push these errors through our structured logger and not just end up with STDERR output.

Files

0001-Set-Thread.report_on_exception-true-by-default-to-re.patch	5.27 KB	12/12/2017	Eregon (Benoit Daloze)