Enable to get "--backtrace-limit" value by "$-B"

There is currently no way to get "--backtrace-limit" value when outputing backtraces from my script, so I have to ignore this value and output backtraces.

In order to be able to output according to "--backtrace-limit" value, I need an API to get this value from the Ruby side.

Proposal

I propose a readonly built-in variable "$-B" to get "--backtrace-limit" value.

```ruby
# ruby --backtrace-limit=3 script.rb
puts $-B # => 3
```

History

#1 - 12/27/2020 11:06 AM - mame (Yusuke Endoh)
+1 if we can provide -B option too.

#2 - 12/27/2020 04:03 PM - Eregon (Benoit Daloze)
What's the use case?
I think we should encourage to use Exception#full_message (instead of manually formatting backtraces), and that could default to apply the limit.
Related: https://github.com/ruby/ruby/commit/f6a080047e960e8b1821a97850830286ce9dee1#commitcomment-45451705

#3 - 12/28/2020 02:47 PM - neg_hide (Hidenori Negishi)
I am making a library that makes error messages easier to read. Therefore, instead of using Exception#full_message, I need to analyze the Exception and output the error message myself.

For example, suppose I have a script like this:

```ruby
def foo
  require "typo_library_name"
end

def bar
  foo
end

bar
```

When I execute ruby --backtrace-limit=1 script.rb in a normal environment, only the backtrace of kernel_require.rb is displayed as shown below, which makes it difficult to understand the true cause of the error. (The same is true for the string obtained from Exception#full_message.)

```
/home/myname/.rbenv/versions/3.0.0/lib/ruby/3.0.0/rubygems/core_ext/kernel_require.rb:85:in `require': cannot
load such file -- typo_library_name (LoadError)
from /home/ynname/.rbenv/versions/3.0.0/lib/ruby/3.0.0/rubygems/core_ext/kernel_require.rb:85:in `require'
... 3 levels...

When I use my library, it omits the backtraces of kernel_require.rb and outputs them, so the error message looks like this.

ruby.rb:2:in `foo': cannot load such file -- typo_library_name (LoadError)
from ruby.rb:2:in `foo'
from ruby.rb:6:in `bar'
from ruby.rb:9:in `<main>'

This process is impossible for Exception#full_message. So I have to output the error message myself.

However, I want the output to look like this:

ruby.rb:2:in `foo': cannot load such file -- typo_library_name (LoadError)
from ruby.rb:2:in `foo'
... 2 levels...

That's why I need a value for "--backtrace-limit".

#4 - 01/13/2021 05:50 AM - matz (Yukihiro Matsumoto)
I am OK with introducing a way to get the limit. But I am strongly opposed to $-B. Any other idea?

Matz.

#5 - 01/13/2021 07:10 AM - jeremyevans0 (Jeremy Evans)
matz (Yukihiro Matsumoto) wrote in #note-4:

I am OK with introducing a way to get the limit. But I am strongly opposed to $-B. Any other idea?

If we are OK with this being a constant and not a variable, RUBY_BACKTRACE_LIMIT.

From the dev meeting log, Thread::Backtrace.limit was decided upon, which sounds better to me.

#6 - 01/14/2021 01:20 PM - Eregon (Benoit Daloze)
neg_hide (Hidenori Negishi) wrote in #note-3:

I am making a library that makes error messages easier to read. Therefore, instead of using Exception#full_message, I need to analyze the Exception and output the error message myself.

The message is included in Exception#full_message. And Exception#backtrace is just Strings too anyway.
I think duplicating the code to format an exception is just a high chance to reimplement it incorrectly (e.g., not handling causes, highlighting, etc correctly).

I see your point that Exception#full_message defaults to use the --backtrace-limit.
I think what we need is a backtrace_limit option to Exception#full_message, and a way to get the value (Thread::Backtrace.limit sounds fine).

#7 - 01/15/2021 02:44 PM - neg_hide (Hidenori Negishi)
I agree with Thread::Backtrace.limit because I think it's easy to understand.

#8 - 02/15/2021 09:29 AM - nobu (Nobuyoshi Nakada)
- Status changed from Open to Closed

Applied in changeset dc47d655e357a7f09943f199969b437a25665ed2.

Added Thread::Backtrace.limit [Feature #17479]