Ruby master - Feature #17470

Introduce non-blocking `Timeout.timeout`

12/26/2020 07:42 AM - ioquatix (Samuel Williams)

| Status:    | Closed                     |
| Priority:  | Normal                     |
| Assignee:  | ioquatix (Samuel Williams) |
| Target version: |                      |

Description

In this bug report, user complained that Timeout.timeout does not work correctly with scheduler:

https://github.com/socketry/async-io/issues/43

We should introduce non-blocking timeout.

I propose the following:

```
rb_fiber_scheduler_with_timeout(VALUE scheduler, VALUE timeout, VALUE block)
```

We can directly modify Timeout.timeout to invoke this hook.

Related issues:

Related to Ruby master - Feature #17363: Timeouts

History

#1 - 12/26/2020 12:48 PM - Eregon (Benoit Daloze)

How would that work, what would be a sample implementation of the hook?

Timeout.timeout must be able to interrupt non-IO operations as well.

#2 - 12/26/2020 09:36 PM - ioquatix (Samuel Williams)

Timeout.timeout must be able to interrupt non-IO operations as well.

Actually, that's not necessarily true, even for the current implementation, see getaddrinfo for example.

#3 - 12/27/2020 06:44 AM - jsc (Justin Collins)

It appears that Timeout.timeout does not work when a Fiber scheduler is set, regardless of the implementation, even when Fibers are not explicitly in use.

Simple example using the sample scheduler:

```
require 'timeout'
require 'fiber'
require_relative 'scheduler'

scheduler = Scheduler.new
Fiber.set_scheduler scheduler

Timeout.timeout 1 do
  puts "hello"
end
```

This code will print "hello" then hang.

#4 - 12/27/2020 04:24 PM - Eregon (Benoit Daloze)

ioquatix (Samuel Williams) wrote in #note-2:

Actually, that's not necessarily true, even for the current implementation, see getaddrinfo for example.

You're completely evading my question :p
Could you answer it please?
That's one of the very rare cases, maybe even the only case in core methods that cannot be interrupted by Thread.raise.

Is the idea to simply give up on interrupting anything that's not handled by the scheduler if there is a Fiber.schedular, or something better? That seems too incompatible to me.
I think a new API to interrupt only scheduler things might make more sense (related to #17363).

#5 - 12/27/2020 04:25 PM - Eregon (Benoit Daloze)
- Related to Feature #17363: Timeouts added

#6 - 02/09/2021 09:57 PM - jeremyevans0 (Jeremy Evans)
- Backport deleted (2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN)
- Tracker changed from Bug to Feature

#7 - 02/11/2021 11:23 PM - ioquatix (Samuel Williams)
*Eregon (Benoit Daloze)* without preemptive scheduling it won't be possible. There are other cases where the GVL is not released, e.g. the sqlite3 gem. In the case of the scheduler, Timeout.timeout is completely broken, so this is an improvement any way you cut it.

#8 - 02/12/2021 02:05 AM - ioquatix (Samuel Williams)

*jsc (Justin Collins)* I tried your example but it seems to work fine?

```ruby
MESSAGE = "Hello World"

def test_timeout_on_main_fiber
  message = nil
  thread = Thread.new do
    scheduler = Scheduler.new
    Fiber.set_scheduler scheduler

    Timeout.timeout(1) do
      message = MESSAGE
    end
  end

  thread.join

  assert_equal MESSAGE, message
end
```

#9 - 03/30/2021 05:38 AM - ioquatix (Samuel Williams)
- Assignee set to ioquatix (Samuel Williams)
- Status changed from Open to Closed

Implemented in [https://github.com/ruby/ruby/pull/4173](https://github.com/ruby/ruby/pull/4173).