Ruby master - Bug #17221

Relax the Fiber#transfer's limitation

10/07/2020 05:28 PM - ko1 (Koichi Sasada)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:		**Backport:**	2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN

Description

Using Fiber#transfer with Fiber#resume for a same Fiber is limited (once Fiber#transfer is called for a fiber, the fiber can not be resumed more).

```
require 'fiber'
f1 = nil
f2 = Fiber.new{
  f1.transfer
}
f1 = Fiber.new{
  f2.transfer
  Fiber.yield 10
  Fiber.yield 20
}
p f1.resume #=> 10
p f1.resume #=> `resume': cannot resume transferred Fiber (FiberError)
```

This restriction was introduced to protect the resume/yield chain, but we realized that it is too much to protect the chain.

Instead of the current restriction, we introduce some other protections.

(1) can not transfer to the resuming fiber.

```
require 'fiber'

root = Fiber.current
f1 = f2 = nil

f1 = Fiber.new{
  f2 = Fiber.new{
    root.transfer(10)
  }
  f2.resume
}

p f1.transfer #=> 10

# root <-----+
#  |         |
#  v         | transfer
#  f1 -> f2 -+ # resume/yield chain


# horizontal direction: resume
# vertical direction: transfer

p f1.transfer #=> attempt to transfer to a resuming fiber (FiberError)

# f1 has it's own resume/yield chain, and f1.transfer breaks the chain

# root <-----+
#  || (error)|
#   vv         |
#   f1 -> f2 -+ # resume/yield chain
```

(2) can not transfer to the yielding fiber.

```
require 'fiber'

f1 = f2 = nil

f1 = Fiber.new{
  f2 = Fiber.new{
    Fiber.yield
  }
  f2.resume
  10
}

p f1.transfer #=> 10

# root
# | ^
# | | transfer
# v |
# f1 --> f2 # resume/yield chain
#    <--

p f2.transfer #=> `transfer': attempt to transfer to an yielding fiber (FiberError)

# f2 is waiting for the resume, so the transfer is not allowed.

# root --+
# | ^    | transfer (error)
# | |    |
# v |    v
# f1 --> f2 # resume/yield chain
#    <--
```

(3) can not resume transferring fiber.

```
require 'fiber'

f1 = f2 = nil

f2 = Fiber.new{
  f1.resume #=> attempt to resume the transferring fiber (FiberError)
}
f1 = Fiber.new{
  f2.transfer
}
f1.transfer

# root
# |
# v
# f1 <-+
# |    |
# v    | resume (error)
# f2 --+

# f1 seems waiting for transfer from other fibers.
```

(4) can not yield from not-resumed fiber

```
require 'fiber'

f2 = Fiber.new do
  Fiber.yield #=> `yield': attempt to yield on not resumed fiber (FiberError)
```

```
end

f1 = Fiber.new
  f2.transfer
end

p f1.transfer

#      root
#      |
#      v
#      f1
#      |
#      v
#   <- f2
#   yield to where ...? (2.7 switches to root fiber)
```

and remove current restriction. The first example works fine:

```
require 'fiber'
f1 = nil
f2 = Fiber.new{
  f1.transfer
}
f1 = Fiber.new{
  f2.transfer
  Fiber.yield 10
  Fiber.yield 20
}
p f1.resume #=> 10
p f1.resume #=> 20


# root -> f1 <-+
#         |    |
#         v    |
#         f2 --+
```

The basic idea is respect *programmer's intention*.

For (1), resuming fiber should be switched by the Fiber.yield.
For (2), yielding fiber should be switched by the Fiber#resume.
For (3), transferring fiber should be switched by the Fiber#transfer.

Mainly (1) can keep the resume/yield chain. Also (2) and (3) makes the chain and relationships with fibers cleanly.

Also at the end of a transferred fiber, it had continued on root fiber.

However, if the root fiber resumed a fiber (and that fiber can resumed another fiber), this behavior also breaks the resume/yield chain.
So at the end of a transferred fiber, switch to the edge of resume chain from root fiber.
For example, root fiber resumed f1 and f1 resumed f2, transferred to f3 and f3 terminated, then continue from the fiber f2 (it was continued
from root fiber without this patch).

```
require 'fiber'
f3 = Fiber.new{
  10
}
f2 = Fiber.new{
  f3.transfer + 20
}
f1 = Fiber.new{
  f2.resume
}
p f1.resume #=> 30
```

```
# without this patch:
#
# root -> f1 -> f2
#  ^             |
#  | exit        v
#  +----------- f3

# with this patch:
#
# root -> f1 -> f2 <-+  # keep resume/yield chain
#                |    |
#                v    |
#                f3 --+ exit
```

The patch is: https://github.com/ruby/ruby/pull/3636

Associated revisions

Revision bf3b2a43 - 10/12/2020 01:58 PM - ko1 (Koichi Sasada)

relax Fiber#transfer's restriction

Using Fiber#transfer with Fiber#resume for a same Fiber is
limited (once Fiber#transfer is called for a fiber, the fiber
can not be resumed more). This restriction was introduced to
protect the resume/yield chain, but we realized that it is too much
to protect the chain. Instead of the current restriction, we
introduce some other protections.

(1) can not transfer to the resuming fiber.
(2) can not transfer to the yielding fiber.
(3) can not resume transferred fiber.
(4) can not yield from not-resumed fiber.

[Bug #17221]

Also at the end of a transferred fiber, it had continued on root fiber.
However, if the root fiber resumed a fiber (and that fiber can resumed
another fiber), this behavior also breaks the resume/yield chain.
So at the end of a transferred fiber, switch to the edge of resume
chain from root fiber.
For example, root fiber resumed f1 and f1 resumed f2, transferred to
f3 and f3 terminated, then continue from the fiber f2 (it was continued
from root fiber without this patch).

History

#1 - 10/07/2020 05:35 PM - ko1 (Koichi Sasada)

- Description updated

#2 - 10/07/2020 05:57 PM - ko1 (Koichi Sasada)

- File clipboard-202010080257-u2lbv.png added

- File clipboard-202010080248-9wdyk.png added

Implementation note

Each fiber has "resuming_fiber" and we can check (1).

clipboard-202010080257-u2lbv.png

#3 - 10/07/2020 06:07 PM - ko1 (Koichi Sasada)

- File deleted (clipboard-202010080248-9wdyk.png)

#4 - 10/07/2020 08:03 PM - Eregon (Benoit Daloze)

This sounds great to me!

#5 - 10/07/2020 10:31 PM - ioquatix (Samuel Williams)

Great work everyone!

#6 - 10/07/2020 11:46 PM - ko1 (Koichi Sasada)

Note for (2). It can break compatibility, but transferred fibers can not be resumed more, so maybe nobody rely on mixing them.

```
require 'fiber'

f = Fiber.new do
  loop do
    Fiber.yield :ok
  end
end

p f.resume
p f.transfer #=> 3.0  `transfer': attempt to transfer to an yielding fiber (FiberError)
p f.resume   #=> 2.7  `resume': cannot resume transferred Fiber (FiberError)
```

#7 - 10/08/2020 12:48 AM - ko1 (Koichi Sasada)

- Description updated

I found "(4) can not yield from not-resumed fiber", so I updated the proposal body with (4).

#8 - 10/08/2020 09:34 AM - Eregon (Benoit Daloze)

For (2), can we use the error message attempt to transfer to a yielding fiber (FiberError) (yielding instead of resuming)? That seems clearer since f2 is not resuming but waiting for a resume and inside Fiber.yield (= yielding).

#9 - 10/09/2020 07:18 PM - ko1 (Koichi Sasada)

- Description updated

Eregon (Benoit Daloze) wrote in #note-8:

> For (2), can we use the error message attempt to transfer to a yielding fiber (FiberError) (yielding instead of resuming)? That seems clearer since f2 is not resuming but waiting for a resume and inside Fiber.yield (= yielding).

It was my mistake. Implementation says

> `transfer': attempt to transfer to an yielding fiber (FiberError)

Description is updated.

#10 - 10/09/2020 07:19 PM - ko1 (Koichi Sasada)

BTW, "a yielding"? "an yielding"?

#11 - 10/09/2020 08:53 PM - Eregon (Benoit Daloze)

I believe it's a yielding Fiber (the first sound in yield-ing is not a vowel sound)

#12 - 10/11/2020 03:13 AM - duerst (Martin Dürst)

Eregon (Benoit Daloze) wrote in #note-11:

> I believe it's a yielding Fiber (the first sound in yield-ing is not a vowel sound)

A Yes indeed for this question. (not an yes :-)

#13 - 10/12/2020 12:16 AM - ko1 (Koichi Sasada)

Thanks!

#14 - 10/12/2020 12:20 AM - ko1 (Koichi Sasada)

Next English question :)

"attempt to transfer to a yielding fiber" or "attempt to transfer to the yielding fiber" (a/the) on error message (error message when calling fib.transfer and fib is yielding).

#15 - 10/12/2020 09:31 AM - ko1 (Koichi Sasada)

BTW I already got Matz's approval so I'll merge it as soon as possible.

#16 - 10/12/2020 01:58 PM - ko1 (Koichi Sasada)

- Status changed from Open to Closed

Applied in changeset git|bf3b2a43741e4f72be21bc6acf24d37e7fcff61c.

---

relax Fiber#transfer's restriction

Using Fiber#transfer with Fiber#resume for a same Fiber is
limited (once Fiber#transfer is called for a fiber, the fiber
can not be resumed more). This restriction was introduced to
protect the resume/yield chain, but we realized that it is too much
to protect the chain. Instead of the current restriction, we
introduce some other protections.

(1) can not transfer to the resuming fiber.
(2) can not transfer to the yielding fiber.
(3) can not resume transferred fiber.
(4) can not yield from not-resumed fiber.

[Bug #17221]

Also at the end of a transferred fiber, it had continued on root fiber.
However, if the root fiber resumed a fiber (and that fiber can resumed
another fiber), this behavior also breaks the resume/yield chain.
So at the end of a transferred fiber, switch to the edge of resume
chain from root fiber.
For example, root fiber resumed f1 and f1 resumed f2, transferred to
f3 and f3 terminated, then continue from the fiber f2 (it was continued
from root fiber without this patch).

#17 - 10/12/2020 04:16 PM - shan (Shannon Skipper)

It seems like both "a" and "the" work here. I might say, "cannot transfer to a yielding Fiber" or "attempted transfer to a yielding Fiber."

Files

clipboard-202010080257-u2lbv.png	25.5 KB	10/07/2020	ko1 (Koichi Sasada)