Pthread fibers become invalid on fork - different from normal fibers.

Fork is notoriously hard to use correctly and in most cases we should be encouraging Process#spawn. However, it does have use cases for example pre-fork model of server design. So there are some valid usage at least.

We recently introduced non-native fiber based on pthread which is generally more compatible than copy coroutine w.r.t. the overall burden on the implementation. However, it has one weak point, which is that pthreads become invalid on fork, and thus fibers become invalid on fork. That means that the following program can become invalid:

```ruby
Fiber.new do
  fork
end.resume
```

It will create two threads, the main thread and the thread for the fiber. When the child begins execution, it will be within the child pthread, but the parent pthread is no longer valid, i.e. it's gone.

I see a couple of options here (not mutually exclusive):

- Combining Fibers and fork is invalid, Fork only works from main fiber.
- Ignore the problem and expect users of fork to be aware that the program can potentially enter an invalid state - okay for fork-exec but not much else.
- Terminate all non-current fibers as we do for threads, and possibly fail if the current fiber exits for some reason.

Because pthread coroutine should be very uncommon, I don't think we should sacrifice the general good qualities of the fiber semantic model for some obscure case. Maybe it would be sufficient to have a warning (not printed by default unless running on pthread coroutines), that fork within a non-main fiber can have undefined results.

History

#1 - 07/09/2021 02:49 PM - jeremyevans0 (Jeremy Evans)
I'm fine having fork raise an exception when called from non-main fiber if the pthread coroutine implementation is used, and issue a verbose warning in other cases.

#2 - 08/19/2021 06:57 AM - matz (Yukihiro Matsumoto)
Using fork from within non-main fibers should be prohibited only on architectures where it is unreliable (e.g. OpenBSD).

Matz.

#3 - 08/19/2021 07:05 AM - ioquatix (Samuel Williams)
Thanks matz (Yukihiro Matsumoto), ko1 (Koichi Sasada), on my PC, the following program segfaults. I tried it on 2.7 and 3 (head).

```ruby
require 'fiber'

fiber = Fiber.new do
  while true
    puts "Hello World"
    fork
    Fiber.yield
    end
end

fiber.resume
puts "Exiting"
```
I'm not sure, do we expect this to work? It seems child process crash. But the program seems well-formed.

Hello World
Exiting
[BUG] Segmentation fault at 0x0000000000000040
ruby 2.7.3p183 (2021-04-05 revision 6847ee089d) [x86_64-linux]

-- Control frame information -----------------------------------------------
cr:0001 p:---- s:0003 e:000002 (none) [FINISH]

-- Machine register context -----------------------------------------------
RIP: 0x0000560912d70169 RBP: 0x00005609149f05d0 RSP: 0x00005609f972fa09d60
RAX: 0x00005609149f05d0 RBX: 0x000056091484ed8b RCX: 0x00005609149f05d0
RDI: 0x00005609149f05d0 RSI: 0x00005609149f05d0
R8: 0x00005609149f05d0 R8: 0x00005609149f05d0 R9: 0x00005609149f05d0 R10: 0x00005609149f05d0
R11: 0x00005609149f05d0 R12: 0x00005609149f05d0 R13: 0x00005609149f05d0
R14: 0x00005609149f05d0 R15: 0x00005609149f05d0

-- C level backtrace information -------------------------------------------
ruby(0x2143a8) [0x560912f573a8]
/home/samuel/.rubies/ruby-2.7.3/bin/ruby(rb_bug_for_fatal_signal+0xe4) [0x560912ff1f74]
/home/samuel/.rubies/ruby-2.7.3/bin/ruby(sigsegv+0x49) [0x560912f852f9]
[0x7f9732719540]
/home/samuel/.rubies/ruby-2.7.3/bin/ruby(rb_ec_tag_jump+0x9) [0x560912d70169]
/home/samuel/.rubies/ruby-2.7.3/bin/ruby(rb_longjmp+0x82) [0x560912d71fd2]
/home/samuel/.rubies/ruby-2.7.3/bin/ruby(rb_print_undefined+0x40) [0x560912d74790]
/home/samuel/.rubies/ruby-2.7.3/bin/ruby(rb_vraise+0x4c) [0x560912ff01c]
/home/samuel/.rubies/ruby-2.7.3/bin/ruby(rb_raise+0x94) [0x560912ff0b4]
/home/samuel/.rubies/ruby-2.7.3/bin/ruby(return_fiber.part.0+0x9) [0x560912f8e39d]
/home/samuel/.rubies/ruby-2.7.3/bin/ruby(rb_fiber_start+0x2f1) [0x560912fd11e1]
/home/samuel/.rubies/ruby-2.7.3/bin/ruby(fiber_entry+0x9) [0x560912fd12c9]