Class Fiber < Object

1.9

A fiber is a lightweight asymetrical coroutine. Code in a fiber is created in a suspended state. It runs when resumed and can suspend itself (passing a value back to the code that resumed it). There is a full description of fibers on page 184.

```
fibs = Fiber.new do
  n1 = n2 = 1
  loop do
    Fiber.yield n1
    n1, n2 = n2, n1+n2
  end
 end
 10.times { print fibs.resume, ' ' }
```

produces:

```
1 1 2 3 5 8 13 21 34 55
```

Class methods

new Fiber.new { *block* } → *fiber*

Uses the block as a new, suspended fiber.

yield Fiber.yield(⟨ val ⟩*) → *obj*

Suspends execution of the current fiber. Any parameters will be returned as the value of the resume call that awoke the fiber. Similarly, any values passed to resume will become the return value of the subsequent yield.

```
f = Fiber.new do
  num = 1
  loop do
    num += Fiber.yield(num)
  end
end
square = 1
10.times do
  square = f.resume(square)
  print square, ' '
end
```

produces:

```
1 2 4 8 16 32 64 128 256 512
```

Instance methods

resume *fiber*.resume(⟨ ( )*val)) → *obj*

Resumes *fiber*. See Fiber.yield for a discussion and example of parameter passing.