Rubyでつくるスレッド

Shugo Maeda

NaCl

2018-06-30

やりたいこと

```
MinThread.start do
  20.times do |i|
    puts "Thread#1: #{i}"
    sleep(0.1)
  end
end

MinThread.start do
  20.times do |i|
    puts "Thread#2: #{i}"
    sleep(0.1)
  end
end
```

継続(Continuation)

- 次に実行される計算を表す
- グローバルgoto
- オブジェクトの状態は戻らない
- Ruby 1.8のスレッドの実装を利用
  - [ruby-dev:4083]
- 継続でスレッドをつくれるのでは？

わかる人にはわかる説明(1)

- Ruby 1.8のスレッドはsetjmp()/longjmp()で切り替える
- スタックは自前で保存して書き戻す
- 継続も同じ仕組み

わかる人にはわかる説明(2)

スレッド	並行宇宙
継続	世界線

継続の例

```
require "continuation"
callcc {|c| $cont = c}
print "Hello, World!\n"
$cont.call
```

実装

スレッドの作成

```
module MinThread
  QUEUE = []

  def self.start(&block)
    QUEUE.push(block)
  end
```

スレッドの実行

```
def self.resume
  proc = QUEUE.shift
  if proc
    proc.call
  end
end

at_exit do
  MinThread.resume
end
```

スレッドの切替

```
def self.pass
  callcc do |c|
    start do
      c.call
    end
    resume
  end
end
```

動いた!

```
MinThread.start do
  20.times do |i|
    puts "Thread#1: #{i}"
    sleep(0.1)
    MinThread.pass
  end
end

MinThread.start do
  20.times do |i|
    puts "Thread#2: #{i}"
    sleep(0.1)
    MinThread.pass
  end
end
```

でも何か違う

```
MinThread.start do
  20.times do |i|
    puts "Thread#1: #{i}"
    sleep(0.1)
    MinThread.pass # これが必要
  end
end

MinThread.start do
  20.times do |i|
    puts "Thread#2: #{i}"
    sleep(0.1)
    MinThread.pass # これが必要
  end
end
```

勝手にスレッドを切り替えたい

TracePoint

- Ruby実行中のイベントをフック
- フックで切り替えればいいのでは？

実装

```
at_exit do
  MinThread.set_next_switch_time
  TracePoint.trace(:line) do |tp|
    MinThread.schedule # 一定時間毎にThread.pass
  end
  MinThread.resume
end
```

１回しか切り替わらない！

理由

- フックの中ではTracePointが無効化される
- フック中で継続を呼ぶと無効化されたまま

じゃあモンキーパッチで

```
at_exit do
  MinThread.set_next_switch_time
  [Integer, String, Array, Hash, IO, File].each do |mod|
    mod.prepend Module.new {
      mod.instance_methods(false).each do |method|
        define_method(method) do |*args, &block|
          MinThread.schedule
          super(*args, &block)
        end
      end
    }
  end
  MinThread.resume
end
```

デモ

課題

- IOなどでブロックすると全部止まる
  - IO#readなどをノンブロッキングIOで再実装すればいい

まとめ

- スレッドはつくれる