Sincronização por Barreiras

Alcides Calsavara

Caracterização

FORMA GERAL:

rendezvouz
ponto crítico

a. Existem *n* threads com a forma geral
b. Cada thread executa o *ponto crítico* somente depois que as *n* threads tiverem executado o *rendevouz*

Base para solução

```
n := número total de threads

contador := 0 // threads que já executaram o rendezvous

mutex := Semáforo( 1 ) // exclusão mútua no contador

barreira := Semáforo( 0 ) // bloqueada até que as n
                          // threads cheguem ao ponto crítico
```

Solução #1

rendevouz

```
mutex.esperar( )
    contador := contador + 1
mutex.sinalizar( )

se contador == n então barreira.sinalizar( )

barreira.esperar( )
```

ponto crítico

Solução #1

rendevouz

```
mutex.esperar( )
    contador := contador + 1
mutex.sinalizar( )

se contador == n então barreira.sinalizar( )

barreira.esperar( )
```

ponto crítico

DEADLOCK

Solução #2

```
rendevouz

mutex.esperar( )
    contador := contador + 1
mutex.sinalizar( )

se contador == n então barreira.sinalizar( )

barreira.esperar( )

barreira.sinalizar( )

ponto crítico
```

Solução #2

```
rendevouz

mutex.esperar( )
    contador := contador + 1
mutex.sinalizar( )

se contador == n então barreira.sinalizar( )

barreira.esperar( )

barreira.sinalizar( )

ponto crítico
```

CORRETA

Padrão Catraca*

```
barreira.esperar( )

barreira.sinalizar( )
```

**turnstile*

Solução #3

```
rendevouz

mutex.esperar( )
    contador := contador + 1
    se contador == n então barreira.sinalizar( )
    barreira.esperar( )
    barreira.sinalizar( )
mutex.sinalizar( )

ponto crítico
```

Solução #3

```
rendevouz

mutex.esperar( )
    contador := contador + 1
    se contador == n então barreira.sinalizar( )
    barreira.esperar( )
    barreira.sinalizar( )
mutex.sinalizar( )

ponto crítico
```

DEADLOCK