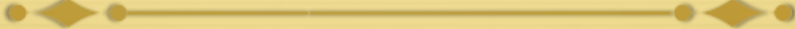


# Alcides Calsavara

# Semáforo



- ✦ Edsger Dijkstra, 1962-1963
  - ✦ Valor inteiro, com um valor inicial atribuído apropriadamente
  - ✦ Operações (atômicas):
    - ✦ *Incrementar*
    - ✦ *Decrementar*
- (não é permitido ler o valor do semáforo)
- ✦ Quando uma thread decrementa o semáforo e o seu valor fica negativo, a thread bloqueia (entra em espera) até que outra thread incremente o semáforo.
  - ✦ Quando uma thread incrementa o semáforo e há outras threads esperando, uma delas é desbloqueada (ou acordada).

# Semáforo - sintaxe

Construtor:

```
s = Semáforo( 10 )
```

Operações:

<code>s.incrementar( )</code>	<code>s.sinalizar( )</code>	<code>s.signal( )</code>
<code>s.decrementar( )</code>	<code>s.esperar( )</code>	<code>s.wait( )</code>
<code>s.up( )</code>	<code>s.release( )</code>	<code>s.V( )</code>
<code>s.down( )</code>	<code>s.acquire( )</code>	<code>s.P( )</code>

# Padrão Mutex

**Thread A:**

```
# região crítica  
x := x + 1
```

**Thread B:**

```
# região crítica  
x := x + 1
```

# Padrão Mutex

**Thread A:**

```
# região crítica  
x := x + 1
```

**Thread B:**

```
# região crítica  
x := x + 1
```

**Thread A:**

```
mutex.esperar()  
# região crítica  
x := x + 1  
mutex.sinalizar()
```

**Thread B:**

```
mutex.esperar()  
# região crítica  
x := x + 1  
mutex.sinalizar()
```



# Padrão Mutex

**Thread A:**

```
# região crítica  
x := x + 1
```

**Thread B:**

```
# região crítica  
x := x + 1
```

**Thread A:**

```
mutex.esperar()  
# região crítica  
x := x + 1  
mutex.sinalizar()
```

**Thread B:**

```
mutex.esperar()  
# região crítica  
x := x + 1  
mutex.sinalizar()
```

Qual é o valor inicial do semáforo **mutex**?

# Padrão Mutex

**Thread A:**

```
# região crítica  
x := x + 1
```

**Thread B:**

```
# região crítica  
x := x + 1
```

Qual é o valor inicial do semáforo **mutex**?

**Thread A:**

```
mutex.esperar()  
# região crítica  
x := x + 1  
mutex.sinalizar()
```

**Thread B:**

```
mutex.esperar()  
# região crítica  
x := x + 1  
mutex.sinalizar()
```

```
mutex := Semáforo( 1 )
```

# Padrão Sinalização



Thread A

1. comando a1
2. s.sinalizar( )

Thread B

1. s.esperar( )
2. comando b1

Assumindo o valor inicial de **s** como zero, tem-se que o comando **a1** executa sempre antes do comando **b1**.



## Padrão Sinalização aplicado ao controle de acesso a variável compartilhada

```
loop:  
  a.aquire()  
  write(v)  
  b.release()
```

```
a = Semaforo( 1 )  
b = Semaforo( 0 )
```

```
loop:  
  b.aquire()  
  read(v)  
  a.release()
```

