

Variáveis de Condição

Sistemas Distribuídos

Variáveis de condição

- Permitem que threads suspendam/retomem a sua execução dentro de secções críticas, de acordo com uma dada condição
- Métodos:
 - `Condition()` // construtor
 - `await()` // thread atual fica em espera até que seja notificada para retomar execução
 - `signal()` // notifica uma thread para resumir a sua execução
 - `signalAll()` //notifica todas as threads para resumirem a sua execução
- Uma variável de condição está intrinsecamente ligada a um Lock

<https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/locks/Condition.html>



Exemplo:

Object obj; ReentrantLock l; Condition cond = l.newCondition();

T1

```
metodo1() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=1;  
    l.unlock();  
}
```

T2

```
metodo2() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=2;  
    l.unlock();  
}
```

T3

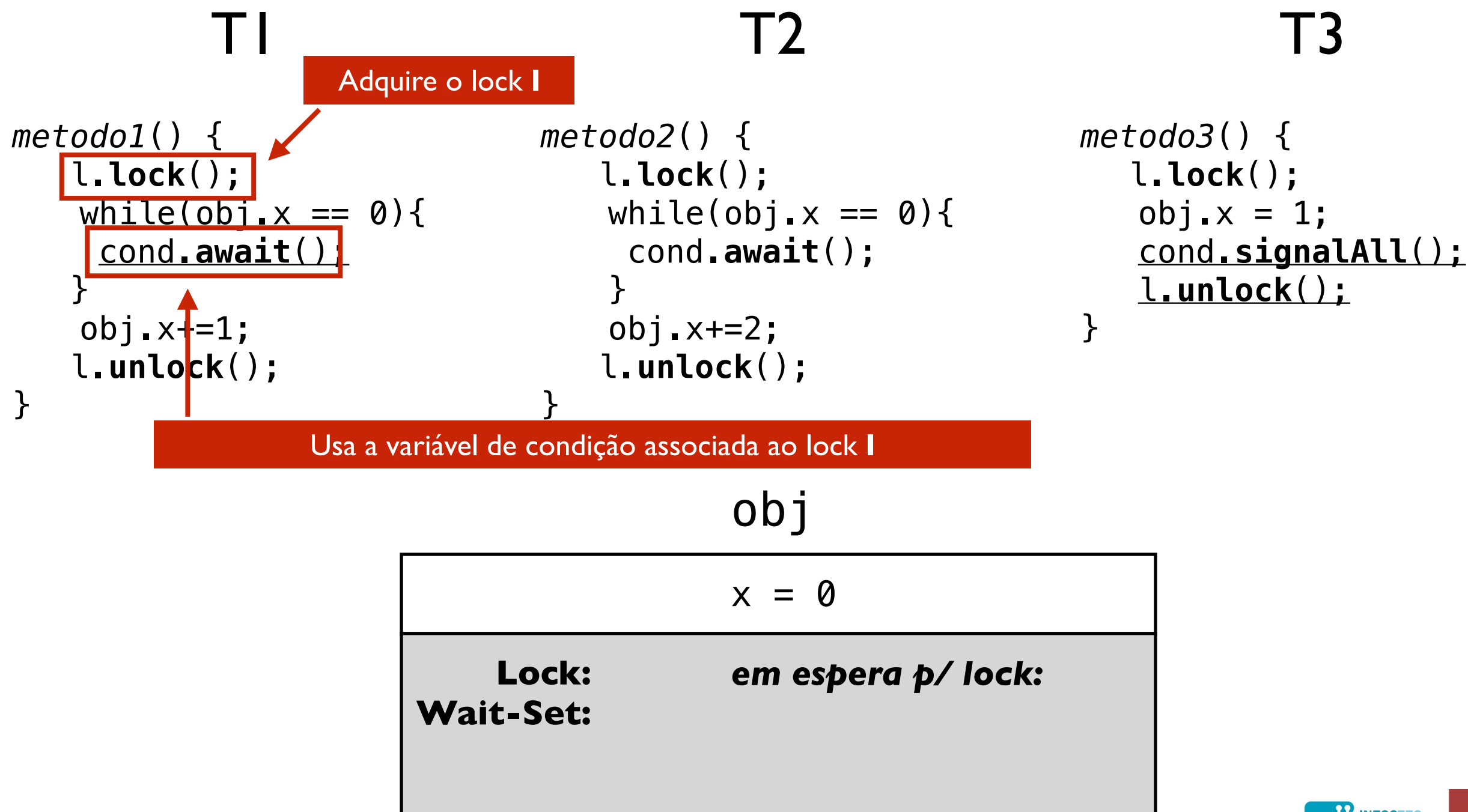
```
metodo3() {  
    l.lock();  
    obj.x = 1;  
    cond.signalAll();  
    l.unlock();  
}
```

obj

x = 0	
Lock: Wait-Set:	em espera p/ lock:

Exemplo:

Object obj; ReentrantLock l; Condition cond = l.newCondition();



Exemplo:

Object obj; ReentrantLock l; Condition cond = l.newCondition();

T1

```
metodo1() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=1;  
    l.unlock();  
}
```

T2

```
metodo2() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=2;  
    l.unlock();  
}
```

T3

```
metodo3() {  
    l.lock();  
    obj.x = 1;  
    cond.signalAll();  
    l.unlock();  
}
```

obj

x = 0	
Lock: T1 Wait-Set:	em espera p/ lock: T2,T3

Exemplo:

Object obj; ReentrantLock l; Condition cond = l.newCondition();

T1

```
metodo1() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=1;  
    l.unlock();  
}
```

T2

```
metodo2() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=2;  
    l.unlock();  
}
```

T3

```
metodo3() {  
    l.lock();  
    obj.x = 1;  
    cond.signalAll();  
    l.unlock();  
}
```

obj

x = 0	
Lock: Wait-Set: T1	em espera p/ lock: T2,T3

Exemplo:

Object obj; ReentrantLock l; Condition cond = l.newCondition();

T1

```
metodo1() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=1;  
    l.unlock();  
}
```

T2

```
metodo2() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=2;  
    l.unlock();  
}
```

T3

```
metodo3() {  
    l.lock();  
    obj.x = 1;  
    cond.signalAll();  
    l.unlock();  
}
```

l.unlock()

Na invocação de await(), o lock é libertado atomicamente e a execução da thread é suspensa

obj

x = 0

lock: **em espera p/ lock: T2,T3**
set: T1

Exemplo:

Object obj; ReentrantLock l; Condition cond = l.newCondition();

T1

```
metodo1() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=1;  
    l.unlock();  
}
```

T2

```
metodo2() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=2;  
    l.unlock();  
}
```

T3

```
metodo3() {  
    l.lock();  
    obj.x = 1;  
    cond.signalAll();  
    l.unlock();  
}
```

obj

x = 0	
Lock: T2 Wait-Set: T1	em espera p/ lock: T3

Exemplo:

Object obj; ReentrantLock l; Condition cond = l.newCondition();

T1

```
metodo1() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=1;  
    l.unlock();  
}
```

T2

```
metodo2() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=2;  
    l.unlock();  
}
```

T3

```
metodo3() {  
    l.lock();  
    obj.x = 1;  
    cond.signalAll();  
    l.unlock();  
}
```

obj

x = 0	
Lock: Wait-Set: T1, T2	em espera p/ lock: T3

Exemplo:

Object obj; ReentrantLock l; Condition cond = l.newCondition();

T1

```
metodo1() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=1;  
    l.unlock();  
}
```

T2

```
metodo2() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=2;  
    l.unlock();  
}
```

T3

```
metodo3() {  
    l.lock();  
    obj.x = 1;  
    cond.signalAll();  
    l.unlock();  
}
```

obj

x = 1	
Lock: T3	em espera p/ lock:
Wait-Set: T1,T2	

Exemplo:

Object obj; ReentrantLock l; Condition cond = l.newCondition();

T1

```
metodo1() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=1;  
    l.unlock();  
}
```

T2

```
metodo2() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=2;  
    l.unlock();  
}
```

T3

```
metodo3() {  
    l.lock();  
    obj.x = 1;  
    cond.signalAll();  
    l.unlock();  
}
```

obj

x = 1	
Lock: T3 Wait-Set: T1,T2	em espera p/ lock:

Exemplo:

Object obj; ReentrantLock l; Condition cond = l.newCondition();

T1

```
metodo1() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=1;  
    l.unlock();  
}
```

T2

```
metodo2() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=2;  
    l.unlock();  
}
```

T3

```
metodo3() {  
    l.lock();  
    obj.x = 1;  
    cond.signalAll();  
    l.unlock();  
}
```

obj

x = 3	
Lock: T2 Wait-Set:	em espera p/ lock: T1

Exemplo:

Object obj; ReentrantLock l; Condition cond = l.newCondition();

T1

```
metodo1() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=1;  
    l.unlock();  
}
```

T2

```
metodo2() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=2;  
    l.unlock();  
}
```

T3

```
metodo3() {  
    l.lock();  
    obj.x = 1;  
    cond.signalAll();  
    l.unlock();  
}
```

l.lock()

Na invocação de signalAll(), todas as threads em espera acordam e competem pelo lock. A thread que o adquire, faz lock e resume a sua execução.

Wait-Set:

Exemplo:

Object obj; ReentrantLock l; Condition cond = l.newCondition();

T1

```
metodo1() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=1;  
    l.unlock();  
}
```

T2

```
metodo2() {  
    l.lock();  
    while(obj.x == 0){  
        cond.await();  
    }  
    obj.x+=2;  
    l.unlock();  
}
```

T3

```
metodo3() {  
    l.lock();  
    obj.x = 1;  
    cond.signalAll();  
    l.unlock();  
}
```

obj

x = 4	
Lock: T1 Wait-Set:	em espera p/ lock:

Pontos-Chave

- Permitem que threads suspendam/retomem a sua execução **dentro de secções críticas**, de acordo com uma dada condição
- **Estão associadas a um lock**; é necessário obter o lock antes de utilizar a variável de condição
- A condição deve ser testada sempre com **while**, para impedir situações inesperadas causadas por interrupções
- Métodos: **await()**, **signal()**, **signalAll()**

Exercício

Implemente uma classe **Barrier** que ofereça um método **await()** cujo objectivo é garantir que cada thread que o invoque bloqueie até que o número de threads nesta situação tenha atingido o valor N.

