Universidad de los Andes

Infraestructura computacional

Concurrencia

Universidad de los Andes

Sincronización

- Acceso concurrente a los datos

```
thread 1
…
load R1, maximo
load R2, max
cmp R2, R1
jle continuar
  store maximo, R2
continuar:
```

maximo
4

```
thread 2
…
load R1, maximo
load R2, max
cmp R2, R1
jle continuar
  store maximo, R2
continuar:
```

max
8

max
9

Universidad de los Andes

Sincronización

- Orden en el acceso a los datos

```
thread
load R1, suma
…
add R1, R2
store suma, R1
```

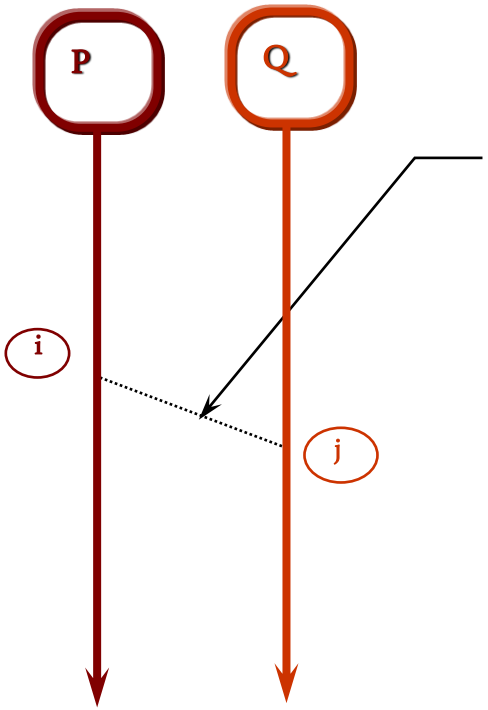
```
main
load R1, suma
load R2, total
add R2, R1
store total, R2
```

suma

total

Universidad de los Andes

Sincronización



Ordenamiento

Exigencia de:

Ti ≠ Tj (exclusión mutua)

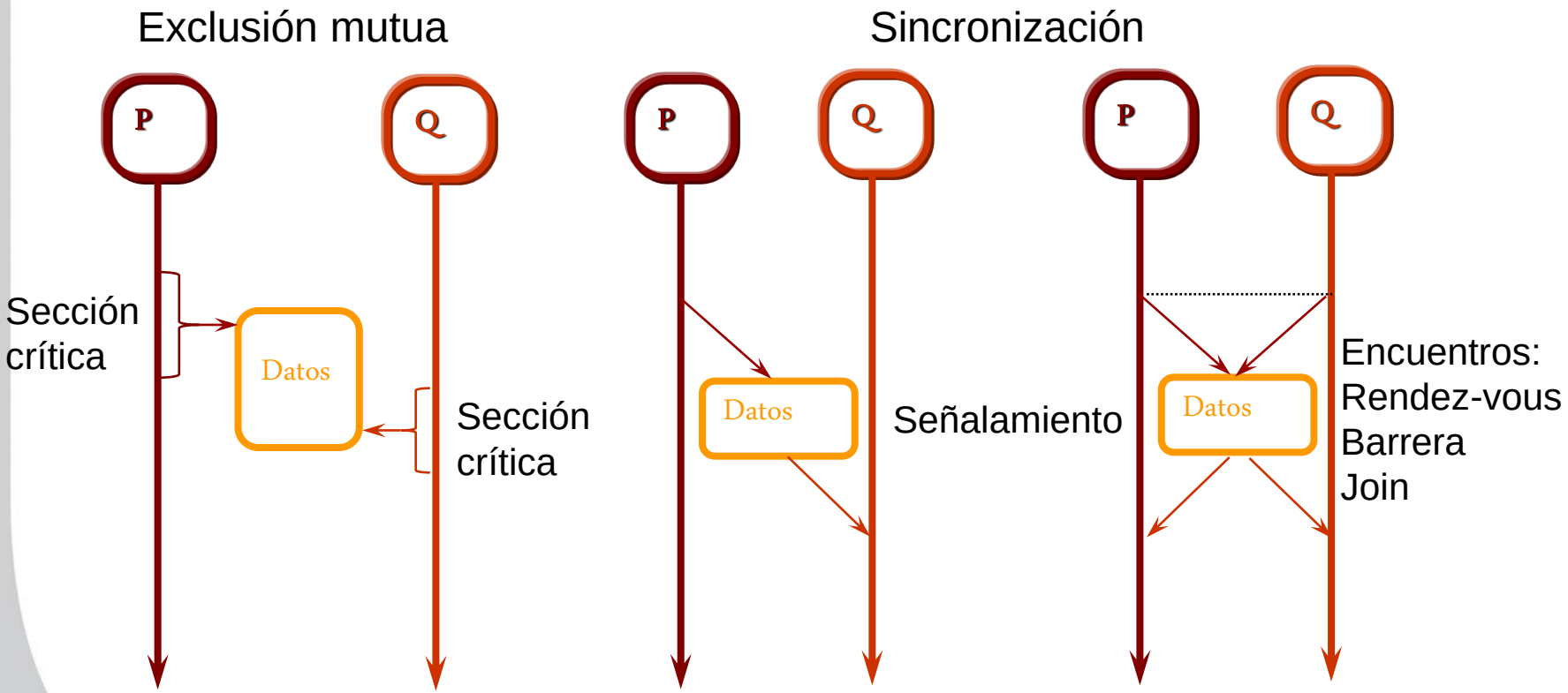
Ti = Tj (sincronización)

Ti < Tj (sincronización)

Ti ? Tj (no determinismo)

Universidad de los Andes

Sincronización



Universidad de los Andes

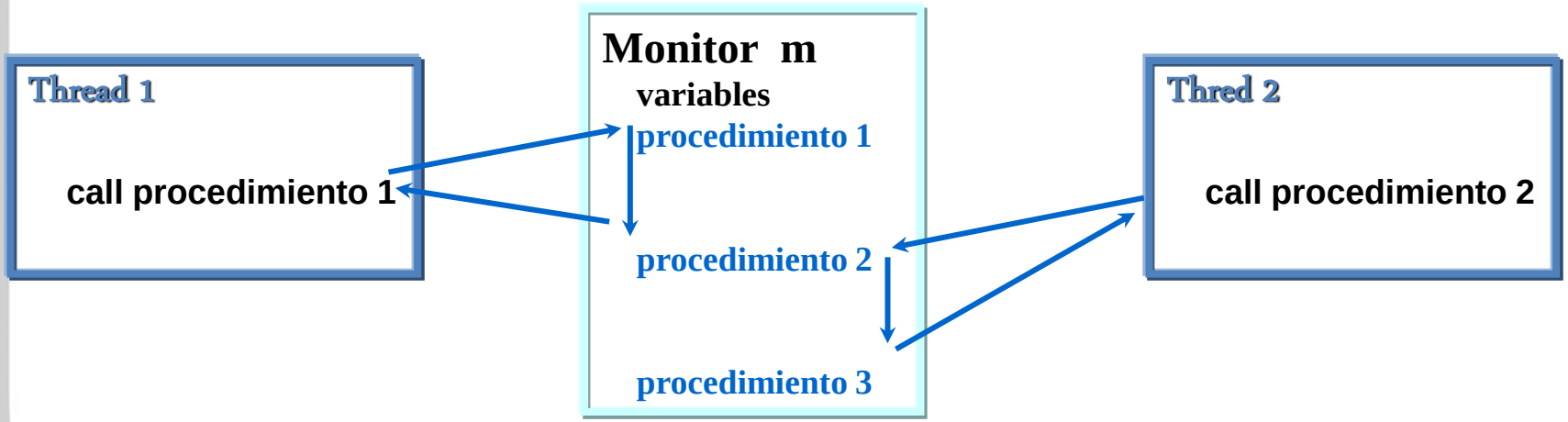
Sincronización

- Monitores
- Eventos, variables de condición
- Mutex
- Semáforos
- Barreras
- Comunicación (especialmente sincrónica)

Universidad de los Andes

Sincronización - exclusión mutua

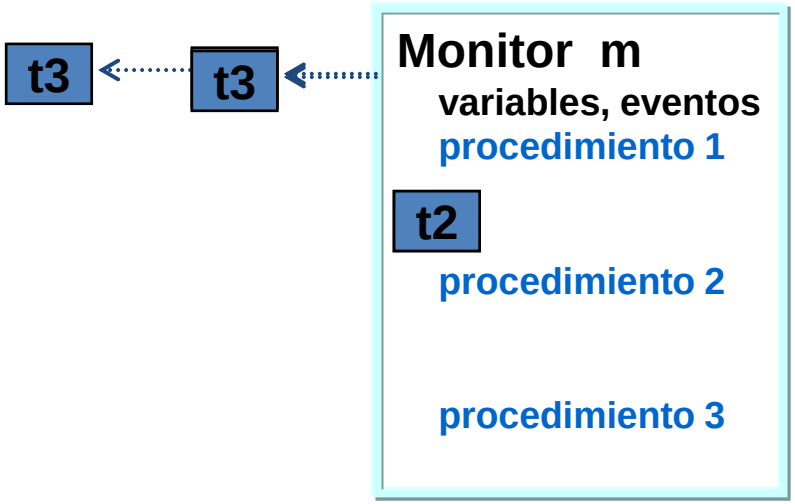
- Monitores:



Universidad de los Andes

Sincronización - exclusión mutua

- Monitores:



Universidad de los Andes

Sincronización - exclusión mutua

- Java: similar a los monitores

```
public class C {
   atributos
   public synchronized void m1( … ) {
      …
   }
   public synchronized void m2( … ) {
      …
   }
   …
   public void mn( … ) {
      …
   }
```

Universidad de los Andes

Sincronización - exclusión mutua

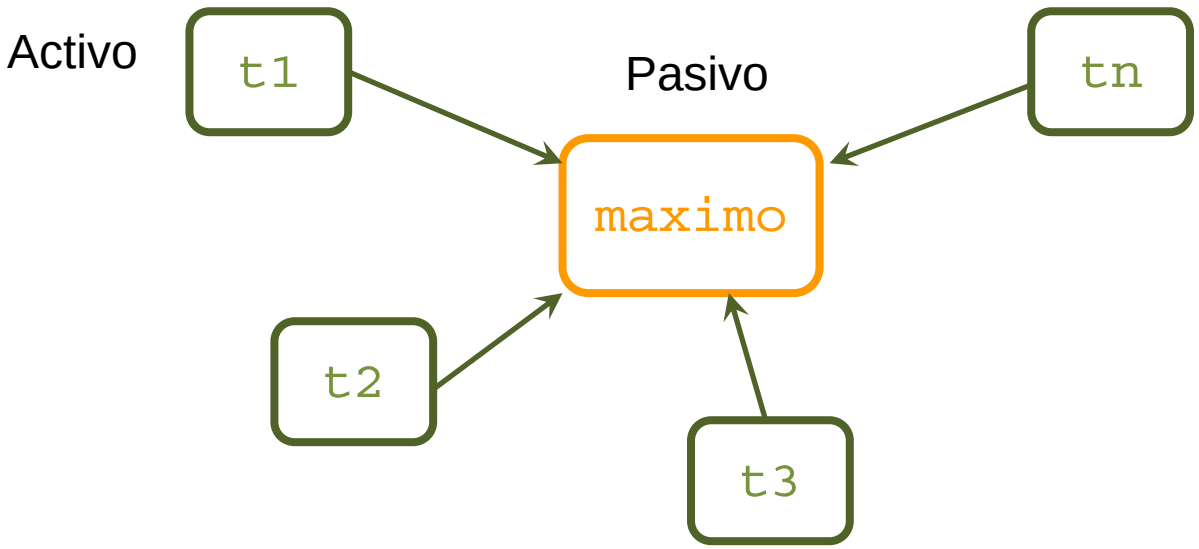
- Java – ejemplo

```
public class Maximo {

  private int maximo;

  public synchronized void anotar( int n ) {
    if ( n > maximo ) maximo = n;
  }
```

Universidad de los Andes

Sincronización - exclusión mutua

- Java – ejemplo



Universidad de los Andes

Sincronización - exclusión mutua

- Java – ejercicio . Encontrar el máximo de la matriz:
  - El último en anotar imprime el resultado
  - `anotar` informa si es el último

```
public class Maximo {

  private int nThreads;
  private int maximo;

  public synchronized boolean anotar( int n ) {
    … desarrollar
  }
  desarrollar run y main de T
```

Universidad de los Andes

Sincronización - exclusión mutua

- Java – ejercicio . Repetir ejercicio anterior, pero ahora el `main` no pasa el identificador entero como parámetro. En lugar de esto:
  - Se tiene un objeto `turno` que reparte turnos. Este objeto tiene un método `darTurno` que retorna los números de 0 a n-1 en secuencia.
  - Cuando el `main` crea los threads les pasa como parámetro una referencia a `turno`.
  - Cada thread invoca `darTurno` para tener su identificador (el número de fila que le toca a él). Después proceden igual que en el caso anterior.

Universidad de los Andes

Sincronización - exclusión mutua

- Java – ejercicio . Resolver de nuevo el ejercicio del máximo, pero con las siguientes características:
  - Se quiere encontrar el máximo pero de un vector de M posiciones .
  - Se quiere generar una cierta cantidad de threads tal que a cada uno le toque procesar N elementos del vector .
  - M no necesariamente es divisible por N, luego a uno de los thread le pueden tocar menos elementos.

Universidad de los Andes

Sincronización - exclusión mutua

- Java: secciones críticas

```
synchronized ( o ) {
    acciones
}
```