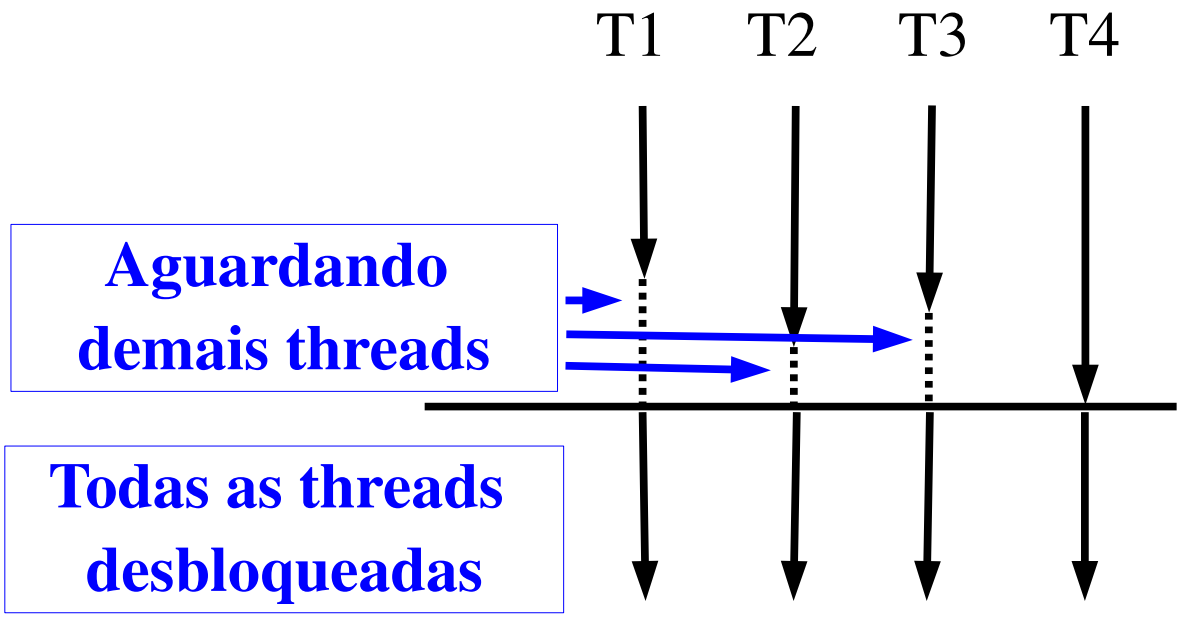
BARREIRAS

- Mecanismo de sincronização que bloqueia as threads em um determinado ponto, aguardando as demais
- Quando todas atingem o mesmo ponto, todas são liberadas]



BARREIRAS EM JAVA

- São implementadas pela classe **CyclicBarrier**
- O **número de threads** que devem ser aguardados deve ser indicado no **construtor** da barreira
- O ponto de espera da barreira deve ser feita uma chamada ao método **await**()

Exemplo:

```
import java.util.concurrent.CyclicBarrier;
import java.util.concurrent.BrokenBarrierException;
public class barreira {
    final CyclicBarrier barr;

    // classe que implementa as threads
    class ImpThread extends Thread {
    int t;
    // construtor
    public ImpThread (String nome, int tempo) {
       super(nome);
       t=tempo; // tempo em milissegundos
    }
```

```
// metodo que executa a thread
public void run () {
   while (true) {
     try {
        Thread.sleep(t);  // bloqueia thread pelo tempo
     } catch ( InterruptedException ie) { System.err.println( ie.toString());
     }
     System.out.print(" Mostra para "+getName());
     try {
       barr.await();
     } catch (InterruptedException ex) {   return;
     } catch (BrokenBarrierException ex) {   return;
     }
     System.out.println(" Acabei com "+getName());
   }
} }
```

Barreira

```
public barreira (){
        ImpThread th1,th2,th3;
         //cria a barreira para 3 threads
         barr= new CyclicBarrier(3);
         // cria as threads
         th1 = new ImpThread("A", 100);
         th2 = new ImpThread("B", 200);
         th3 = new ImpThread("C", 300);
         // inicia a execucao das threads
         th1.start( );    th2.start( );      th3.start( );
    }

 public static void main (String args[]) {
    barreira b = new barreira();
 }
}
```

BARREIRAS EM OPENMP

- Basta usar a diretiva correspondente para cada linguagem:
  - C: #pragma omp barrier
  - Fortran: !$OMP BARRIER