


Métodos Avanzados de Programación Científica y Computación

M^a Luisa Díez Platas

Tema 10. Problemas comunes en Programación Concurrente

¿Cómo estudiar este tema?

IDEAS CLAVE	LO + RECOMENDADO	+ INFORMACIÓN	TEST
<p>¿Cómo estudiar este tema?</p> <p>El problema de los productores consumidores</p> <p>El problema de los lectores y escritores</p> <p>El problema de la cena de los filósofos</p>	<p>No dejes de leer...</p> <p>Problema de agentes y fumadores</p>	<p>A fondo</p> <p>Filósofos comensales</p> <p>Barbero durmiente</p>	

Problema productor - consumidor

Inserción de información

- El productor debe esperar a que el buffer tenga espacio.
- La operación de inserción se debe completar antes de que entre el consumidor
- Operación sincronizada → buffer es un **monitor**

```

public synchronized void insertarElemento (int elemento) {
    while (cima >= maximo - 1) {
        try {
            System.out.println ("No se puede añadir nada");
            wait ();
            System.out.println ("Ya se puede volver a añadir");
        }
        catch (InterruptedException e) {
            System.err.println ("Se ha producido un error: " + e.toString
        ));
    } // catch
} // while

cima++;
info[cima] = elemento;

notifyAll ();
}
    
```

Espera a tener espacio

Activa los hilos que están esperando

Problema productor - consumidor

Extracción de información

- El consumidor debe esperar a que el buffer tenga información
- La operación de extracción se debe completar antes de que entre el productor
- Operación sincronizada → buffer es un **monitor**

```
public synchronized int sacarElemento () {
    int elem;

    while (cima == -1) {
        try {
            System.out.println ("No se puede sacar nada");
            wait ();
            System.out.println ("Ya se puede volver a sacar");
        }
        catch (InterruptedException e) {
            System.err.println ("Se ha producido un error: " + e.toString
());
        } // catch
    } // while
    elem = info[cima];
    cima--;

    notifyAll ();

    return (elem);
} // sacarElement
```

Espera a que haya información

Activa los hilos que están esperando

Problema productor - consumidor

```
buffer= new Buffer (10);
    productorRunnable_ = new Productor
(buffer);
    consumidorRunnable_ = new Consumidor
(buffer);

    Thread productorThread = new Thread
(productorRunnable_);
    Thread consumidorThread = new Thread
(consumidorRunnable_);

    productorThread.start ();
    try {
        Thread.sleep (15);
    }
    catch (InterruptedException e) {
        System.err.println ("Se ha producido
un error: " + e.toString ());
    } // catch

    consumidorThread.start ();
```

Productor

```
public void run () {
    int elementoAInsertar;
    int tiempoEspera;

    for (int i = 0; i < 20; i++) {

        elementoAInsertar = (int) Math.round
(Math.random () * 500);

        tiempoEspera = (int) Math.round (Math.random ()
* 20) + 1;

        buffer.insertarElemento (elementoAInsertar);
        System.out.println ("Se acaba de insertar el
elemento: " + elementoAInsertar);

        try {
            Thread.sleep (tiempoEspera);
        } // try
        catch (InterruptedException e) {
            System.err.println ("Se ha producido una
excepción: " + e.toString ());
        } // catch

    } // for
} // run ()
```

Problema productor - consumidor

Consumidor

```
buffer= new Buffer (10);
productorRunnable_ = new Productor
(buffer);
consumidorRunnable_ = new Consumidor
(buffer);

Thread productorThread = new Thread
(productorRunnable_);
Thread consumidorThread = new Thread
(consumidorRunnable_);

productorThread.start ();
try {
    Thread.sleep (15);
}
catch (InterruptedException e) {
    System.err.println ("Se ha producido
un error: " + e.toString ());
} // catch

consumidorThread.start ();
```

```
public void run () {
    int elementoAExtraer;
    int tiempoEspera;

    for (int i = 0; i < 20; i++) {

        tiempoEspera = (int) Math.round (Math.random
+ 1;

        elementoAExtraer = buffer.sacarElemento ();

        try {
            Thread.sleep (tiempoEspera);
        } // try
        catch (InterruptedException e) {
            System.err.println ("Se ha producido una excepción:
e.toString ());
        } // catch

    } // for
```

Problema de los filósofos

Cinco filósofos dedican sus vidas a pensar y comer. Los filósofos comparten una mesa circular rodeada por cinco sillas, cada una de las cuáles pertenece a un filósofo. En la mesa hay cinco platos de arroz y cinco palillos.



Si todos los filósofos cogen el palillo que está a su derecha al mismo tiempo → todos esperando **interbloqueo**

Nadie libera su palillo hasta comer.

Posibles soluciones:

- No dejar pasar a más de cuatro al comedor (poner un camarero).
- Si no puede coger el otro palillo obligar a que lo suelte

Fuente imagen: <http://webdiis.unizar.es/asignaturas/ISBC/isbc/PracticaLinda/practLinda.html>

Problema de los filósofos

5 hilos filósofos
5 monitores palillos



Filósofo

```
dcha.cogerPalillo(id);
//simula el tiempo que puede tardar en coger el otro palillo
Filosofo.sleep(random.nextInt(1000) + 100);

izda.cogerPalillo(id);

// Si ha conseguido el Tenedor Izquierdo el filósofo come

System.out.println("El Filósofo " + (id+1) + " está comiendo.")

try {
    sleep(random.nextInt(1000) + 500);
} catch (InterruptedException ex) {
    System.out.println("Error. " + ex.toString());
}

izda.soltarPalillo(id);
dcha.soltarPalillo(id);
```

Palillo

```
public synchronized boolean cogerPalillo(int id_f) throws InterruptedException{
    while(!libre)
        this.wait();
    System.out.println("El Filósofo " + (id_f+1) + " coge el palillo " + (id+1));

    libre = false;
    return true;
}

/**
 * Método para dejar el palillo
 *
 */
public synchronized void soltarPalillo(int id_f) throws InterruptedException {
    libre = true;
    System.out.println("El Filósofo " + (id_f+1) + " deja el palillo " + (id+1));

    this.notify();
}
```

Fuente imagen:<http://webdiis.unizar.es/asignaturas/ISBC/isbc/PracticaLinda/practLinda.html>

Problema de los filósofos. Solución I

Si no coge el segundo palillo, libera el primero

Filósofo

```
private boolean cogerPalillos(){
    try{
        dcho.cogerPalillo(id);
        Filosofo.sleep(random.nextInt(1000) + 100);
        if (!izdo.cogerPalillo(id)){
            dcho.soltarPalillo(id);
            return false;
        }
        return true;
    }catch (InterruptedException ex) {
        return false;
    }
}

//intenta coger los dos palillos pero si no lo consigue libera el que tiene y es
while (!cogerPalillos()){
    System.out.println("El Filósofo " + (id+1) + " está esperando para coger el segundo palillo");
    Filosofo.sleep(random.nextInt(1000) + 100);
};

// Si se han conseguido los dos palillos come

System.out.println("El Filósofo " + (id+1) + " está comiendo.");

try {
    sleep(random.nextInt(1000) + 500);
} catch (InterruptedException ex) {
    System.out.println("Error. " + ex.toString());
}

izdo.soltarPalillo(id);
dcho.soltarPalillo(id);
```

5 hilos filósofos
5 monitores palillos



Palillo

```
public synchronized boolean cogerPalillo(int id_f) throws InterruptedException{
    while(!libre)
        this.wait();
    System.out.println("El Filósofo " + (id_f+1) + " coge el palillo " + (id+1));

    libre = false;
    return true;
}

/**
 * Método para dejar el palillo
 */
public synchronized void soltarPalillo(int id_f) throws InterruptedException {
    libre = true;
    System.out.println("El Filósofo " + (id_f+1) + " deja el palillo " + (id+1));
    this.notify();
}
```

Fuente imagen:<http://webdiis.unizar.es/asignaturas/ISBC/isbc/PracticaLinda/practLinda.html>

Problema de los filósofos. Solución II

Un camarero no deja pasar a más de 4

Filósofo

```
FilosofosConCamarero.camarero.darPermisoEntrada(id);  
System.out.println("El Filósofo " + (id+1) + " ha entrado en el comedor");
```

```
dcho.cogerPalillo(id);  
izdo.cogerPalillo(id);
```

```
// Si ha conseguido los dos palillos come
```

```
System.out.println("El Filósofo " + (id+1) + " está comiendo.");
```

```
try {  
    sleep(random.nextInt(1000) + 500);  
} catch (InterruptedException ex) {  
    System.out.println("Error. " + ex.toString());  
}
```

```
izdo.soltarPalillo(id);
```

```
dcho.soltarPalillo(id);
```

```
FilosofosConCamarero.camarero.invitarSalidaFilosofo(id);
```

5 hilos filósofos
5 monitores palillos
Un monitor camarero

Camarero

```
public class Camarero {  
    private int numFilosofos = 0; // Es el número de comensales total de filósofos menos 1  
  
    /**  
     * @param id_f ID del filósofo  
     */  
    /**  
     * public synchronized void darPermisoEntrada(int id_f) throws InterruptedException{  
     *     System.out.println("El Filósofo " + (id_f+1) + " pide permiso para entrar en el comedor " );  
     *     while(numFilosofos==5){ // Si no hay comensales libres espera  
     *         this.wait();  
     *     }  
     *     numFilosofos++;  
     * }  
  
    /**  
     * @param id_f ID del filósofo  
     * @throws InterruptedException Posibles errores  
     */  
    /**  
     * public synchronized void invitarSalidaFilosofo(int id_f) throws InterruptedException{  
     *     numFilosofos--;  
     *     System.out.println("El Filósofo " + (id_f+1) + " ha salido del comedor " );  
     *     this.notify(); //Pone disponible al siguiente  
     * }  
}
```



Fuente imagen:<http://webdiis.unizar.es/asignaturas/ISBC/isbc/PracticaLinda/practLinda.html>

Problema Lectores-Escritores

Acceden a un mismo tablón varios lectores simultáneos o un solo escritor

1 monitor tablón
Hilos lectores y escritores

Lector

```
public class Lector extends Thread {  
    private int id;  
    private String dato;  
    private Tablon tablon;  
    public Lector (int _id, Tablon _tablon) {  
        id=_id;  
        tablon=_tablon;  
    }  
  
    public void run () {  
        int idL=id+1;  
        System.out.println ("El lector "+idL+" quiere leer");  
        tablon.permisoLeer();  
        System.out.println ("El lector "+idL+" esta leyendo");  
        dato=tablon.dejarLeer();  
        System.out.println ("El lector "+idL+" ha leído"+dato);  
    }  
}
```

Escritor

```
public class Escritor extends Thread {  
    private int id;  
    private String dato;  
    private Tablon tablon;  
    public Escritor (int _id, String _dato, Tablon _tablon) {  
        id=_id;  
        dato=_dato;  
        tablon=_tablon;  
    }  
  
    public void run () {  
        System.out.println ("El escritor "+id+" quiere escribir");  
        tablon.permisoEscribir(dato);  
        System.out.println ("El escritor "+id+" escribiendo "+dato);  
        tablon.dejarEscribir();  
        System.out.println ("El escritor "+id+" ha escrito");  
    }  
}
```

Problema Lectores-Escritores

Acceden a un mismo tablón varios lectores simultáneos o un solo escritor

1 monitor tablón
Hilos lectores y escritores

Tablón

```
public synchronized void permisoLeer () {
    try{
        while (numEscritores>0 || dato.equals("")){
            wait();
        }
        numLectores++;
    }catch (InterruptedException e){
    }
}

public synchronized String dejarLeer () {
    numLectores--;
    notifyAll();
    return dato;
}

public synchronized void permisoEscribir (String _dato) {
    try{
        while (numLectores >0 || numEscritores>0 )
            wait ();
        numEscritores++;
        dato=_dato;
    }catch (InterruptedException e){
    }
}

public synchronized void dejarEscribir () {
    numEscritores=0;
    notifyAll();
}
```

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

unir

www.unir.net