

# Examen de Programación Concurrente y Distribuida

## 3º Curso de Grado en Ingeniería Informática

### Febrero. Curso 2014-15

#### CUESTIONES

1. Los multicomputadores conmutados están formados por multitud de nodos independientes, formados por procesador más memoria, e interconectados mediante alguna red de estructura regular. Represente gráficamente, tres de sus posibles topologías indicando su nombre correspondiente. **(0,5 Puntos)**
2. Usando la instrucción hardware *exchange*, garantice la exclusión mutua para los procesos P1 y P2. **(0,75 Puntos)**

```
process P1
repeat
```

Sección Crítica

```
Resto1
forever
```

```
process P2
repeat
```

Sección Crítica

```
Resto2
forever
```

#### SOLUCION:

```
process P1
repeat
```

```
  repeat
    exchange(r0,m)
```

```
  until r0 = 1;
```

```
  Sección Crítica
```

```
  exchange(r0,m)
```

```
  Resto1
forever
```

```
process P2
repeat
```

```
  repeat
    exchange(r1,m)
```

```
  until r1 = 1;
```

```
  Sección Crítica1
```

```
  exchange(r1,m)
```

```
  Resto1
forever
```

**m inicialmente vale 1 y los ri valen 0.**

3. A continuación se muestra un intento de implementación del algoritmo de **Peterson** para solucionar el problema de la exclusión mutua. Corríjalo para que sea válido e indique los valores de inicialización de las variables usadas. **(0,75 Puntos)**

```
process P0
repeat
  c0 := quiereentrar;
  turno := 1;
  while (c1 = quiereentrar) do;
  Sección Crítica0
  c0 := fueraSC;
  Resto0
forever
```

```
process P1
repeat
  c1 := quiereentrar;
  turno := 0;
  while (c0 = quiereentrar) do;
  Sección Crítica1
  c1 := fueraSC;
  Resto1
forever
```

## SOLUCION:

```
process P0
repeat
  c0 := quiereentrar;
  turno := 1;
  while (c1 = quiereentrar)
    and (turno = 1) do;
  Sección Crítica0
  c0 := fueraSC;
  Resto0
forever
```

```
process P1
repeat
  c1 := quiereentrar;
  turno := 0;
  while (c0 = quiereentrar)
    and (turno = 0) do;
  Sección Crítica1
  c1 := fueraSC;
  Resto1
forever
```

**c0 y c1 se inicializan a fueraSC. El valor de turno no influye.**

4. Usando semáforos, de la forma más simple posible, haga que estos dos procesos accedan alternativamente a la sección crítica. Deberá comenzar accediendo el proceso P1. **(1,5 Puntos)**

```
program alterna;

process P1
repeat
    Sección Crítica
forever

process P2
repeat
    Sección Crítica
forever

var

begin
    cobegin
        P1; P2;
    coend;
end.
```

## SOLUCIÓN

```
program alterna;

process P1
repeat
    wait(s1);
    Sección Crítica
    signal(s2);
forever

process P2
repeat
    wait(s2);
    Sección Crítica
    signal(s1);
forever

var
    s1, s2: semaphore;
begin
    initial(s1,1);
    initial(s2,0);
    cobegin
        P1; P2;
    coend;
end.
```

## PROBLEMAS

5. (1 Puntos). Tenemos un sistema operativo donde se están ejecutando 5 procesos y se conoce que las necesidades máximas de cada proceso son:

	R1	R2	R3	R4
P1	1	0	2	2
P2	2	2	1	0
P3	1	1	2	1
P4	0	1	1	1
P5	2	2	1	1

y que en un momento dado, el grafo de asignación de recursos es:

Se está usando como algoritmo de evitación del interbloqueo el Algoritmo del Banquero. Si el proceso P5 solicita 1 ejemplar de R1. ¿Debe concederse la petición?

## SOLUCION:

Si se concediese la petición (P5 solicita 1 de R1) el sistema quedaría en el siguiente estado

	ASIGNADOS				NEC. MÁXIMAS				PENDIENTES			
	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
P1	0	0	1	0	1	0	2	2	1	0	1	2
P2	0	1	0	0	2	2	1	0	2	1	1	0
P3	0	0	0	1	1	1	2	1	1	1	2	0
P4	0	0	1	0	0	1	1	1	0	1	0	1
P5	1	0	0	1	2	2	1	1	1	2	1	0

E=(2,2,3,3)

A=(1,1,2,2)

L=(1,1,1,1)

Aplicamos el Banquero:

- Puede finalizar el proceso P4. L=(1,1,2,1)
- Puede finalizar el proceso P3. L=(1,1,2,2)
- Puede finalizar el proceso P1. L=(1,1,3,2)
- Los procesos P2 y P5 no pueden finalizar, por tanto el estado es inseguro.

6. En una carretera comarcal hay un puente en mal estado. El puente es muy estrecho, y sólo dispone de un carril de circulación. Debido a su estado, no soporta tampoco más de

dos vehículos sobre él. Los vehículos llegarán al puente desde el norte o desde el sur. Se deberá diseñar una solución para que los vehículos pasen por el puente evitando el interbloqueo, es decir, asegurando que dos vehículos no quedan enfrentados en el puente.

1. Solucione el problema usando canales **(2,5 Puntos)**
2. Solucione el problema usando monitores con la siguiente regla para evitar la postergación indefinida: “No podrán pasar más de 4 coches en el mismo sentido a partir de que esperen coches en el sentido contrario”. **(3 puntos)**

### **NOTAS:**

- Para simplificar el código se usarán llaves { y } en lugar de las instrucciones `begin` y `end` para marcar los bloques de código.
- Si fuese necesario que un procedimiento devuelva un valor se permite el uso de la instrucción `return`.

**ANEXO 1. Estructura de los procesos para el problema 6**

```
program Febrero15;

const
    np1=10;
    np2=10;

process type Norte(id:integer);
begin
    repeat
        { PROTOCOLO OCUPACION }
        writeln('Vehiculo ',id,' cruzando
desde el norte');
        { PROTOCOLO LIBERACION }
        sleep(random(10));
    forever
end;

process type Sur(id:integer);
begin
    repeat
        { PROTOCOLO OCUPACION }
        writeln('Vehiculo ',id,' cruzando
desde el sur');
        { PROTOCOLO LIBERACION }
        sleep(random(10));
    forever
end;

var
    i,j: integer;
    Pnorte: array[1..np1] of Norte;
```

```
    Psur: array[1..np2] of Sur;  
  
begin  
  cobegin  
    for i := 1 to np1 do Pnorte[i](i);  
    for j := 1 to np2 do Psur[j](j);  
  coend  
end.
```