Examen de Programación Concurrente y Distribuida

3º Curso de Grado en Ingeniería Informática

Febrero. Curso 2015-16

CUESTIONES

1. Explique brevemente el problema que plantea el paso de parámetros en RPC y que posibilidades hay para solucionarlo. **(0,5 Puntos)**

2. Justifique si el siguiente algoritmo para el control de la concurrencia cumple las condiciones requeridas. **(0,75 Puntos)**

```
process P0
repeat
 c0 := 1;
 while c1=1 do;
 Sección Crítica
 c0 := 0;
 Resto0
forever
```

```
process P1
repeat
 c1 := 1;
 while c0=1 do;
 Sección Crítica
 c1 := 0;
 Resto1
forever
```

Donde inicialmente `c0=0 y c1=0.`

3. Indique el grafo de precedencias que correspondería al siguiente programa. **(1,5 Puntos)**

```
Program P
var
    s1: semaphore;
    s2: semaphore;
    s3: semaphore;
    s4: semaphore;
```

```
process P1
begin
    a;
    signal(s1);
    signal(s1);
    wait(s2);
    d;
    signal(s3);
    signal(s3);
end
```

```
process P2
begin
    wait(s1);
    c;
    signal(s2);
    wait(s3);
    f;
    wait(s4);
    g;
end
```

```
process P3
begin
    wait(s1);
    b;
    signal(s2);
    e;
    signal(s4);
end
```

```
begin
    initial(s1,2);
    initial(s2,1);
    initial(s3,0);
    initial(s4,0);
    conbegin
        P1; P2; P3;
    coend
end
```

4. Usando la instrucción hardware *testset*, 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
```

PROBLEMAS

En una piscina de saltos de trampolín entrenan 5 saltadores y 5 saltadoras todos los días. La piscina dispone de dos trampolines, el trampolín de un metro y el trampolín de tres metros. Todos usan los trampolines de forma indistinta, de forma que el saltador/a que quiera saltar usará el trampolín que esté libre. De estar libre los dos prefieren hacer uso del trampolín de tres metros. Una vez que el saltador/a acceda al trampolín, deberá esperar a que la piscina quede libre para realizar su salto. En el caso de que haya varios saltadores/as esperando, se gestionará el acceso de forma paritaria, es decir, se dará acceso al género que tenga más miembros esperando (saltadores o saltadoras). Una vez que el saltador/a ha realizado el salto, deberá indicar mediante un mensaje el trampolín usado y descansará un rato antes de intentar un nuevo salto.

5. Solucionar el problema anterior usando **monitores**. Se asume una semántica de la operación `resume` tipo "desbloquear y espera urgente" (la habitual de *Pascal-FC*). **(3 Puntos)**

6. Solucionar el problema anterior usando **buzones**. En esta solución no se tendrá en cuenta la paridad, es decir, cuando un trampolín quede libre lo podrá usar cualquiera de los que esperan **(2,5 Puntos)**

7. Tenemos un sistema operativo con 5 procesos, que en un momento dado presenta el siguiente estado:

   E=(2,3,3,2)

   A=(2,2,2,2)

   L=(0,1,1,0)

	N. Máximas					R. Asignados			
	R1	**R2**	**R3**	**R4**		**R1**	**R2**	**R3**	**R4**
P1	1	0	1	0		0	0	0	0
P2	0	2	0	1		0	2	0	0
P3	2	1	0	1		2	0	0	1
P4	0	1	2	0		0	0	2	0
P5	1	1	0	1		0	0	0	1

Si estamos usando el algoritmo del banquero para evitar los interbloqueos, ¿debería concederse a P3 un ejemplar del recurso de R2?. Justifique la respuesta. **(1 Punto)**.

ANEXO 1. Estructura de los procesos para los problemas 5 y 6

```
program Trampolines;

const
     np1=5;
     np2=5;

process type TSaltador(id:integer);
begin
     repeat
          { PROTOCOLO OCUPACION }
          writeln('He saltado desde el trampolín de ..');
          { PROTOCOLO LIBERACION }
          sleep(random(4));

     forever
end;

process type TSaltadora(id:integer);
begin
     repeat
          { PROTOCOLO OCUPACION }
          writeln('He saltado desde el trampolín de ..');
          { PROTOCOLO LIBERACION }
          sleep(random(4));
     forever
end;

var
   i,j: integer;
   Saltador: array[1..np1] of Tsaltador;
   Saltadora: array[1..np2] of Tsaltadora;

begin
  cobegin
    for i := 1 to np1 do Saltador[i](i);
    for j := 1 to np2 do Saltadora[j](j);

  coend
end.
```