Examen de Programación Concurrente y Distribuida

3º Curso de Grado en Ingeniería Informática

Septiembre. Curso 2012-13

CUESTIONES

1. Describa brevemente las consideraciones de diseño para los sistemas distribuidos (No extenderse más de un folio por una cara). **(0,5 Puntos)**

2. Justifique si el siguiente algoritmo para el control de la concurrencia cumple las condiciones requeridas. **(0,5 Puntos)**

```
process P0
repeat
 c0 := 1;
 while turno <> 0 do
  begin
   while c1 = 1 do;
   turno := 0;
  end
  Sección Crítica
  c0 := 0;
  Resto0
forever
```

```
process P1
repeat
 c1 := 1;
 while turno <> 1 do
  begin
   while c0 = 1 do;
   turno := 1;
  end
  Sección Crítica
  c1 := 0;
  Resto1
forever
```

Donde inicialmente `turno=1; c0=0 y c1=0.`

3. Dado el siguiente conjunto de instrucciones, utilice las condiciones de Bernstein para establecer el grafo de precedencias que le corresponde. **(0,5 Puntos)**

```
S1: num = 3;
S2: x = pi * y;
S3: acc = num * z;
S4: acu = acc * c;
S5: res = x + 5;
```

4. Usando la instrucción hardware *addc*, garantice la exclusión mutua para los procesos P1 y P2. **(0,5 Puntos)**

```
process P1
repeat

   Sección Crítica

   Resto1
forever
```

```
process P2
repeat

   Sección Crítica

   Resto2
forever
```

5. Dado el siguiente programa, ¿qué valor tendrá la variable `x` a la finalización del mismo?. **(0,5 Puntos)**

```
program cuestion5;
var
  contador: integer;
  mutex: semaphore;
process p1;
begin
    wait(mutex);
    contador:=contador+1;
    signal(mutex);
end;
process p2;
begin
    wait(mutex);
    contador:=contador+1;
    signal(mutex);
end;

process p3;
begin
    contador:=contador+1;
end;

begin
  initial(mutex,1);
  contador := 0;
  cobegin
        p1;
        p2;
        p3;
  coend
end.
```

PROBLEMAS

En un sistema existen 3 tipos distintos de procesos que acceden a un registro de 32 bits. El registro permite el acceso por separado a los 16 bits más bajo y a los 16 bits más altos (mitad inferior y mitad superior).

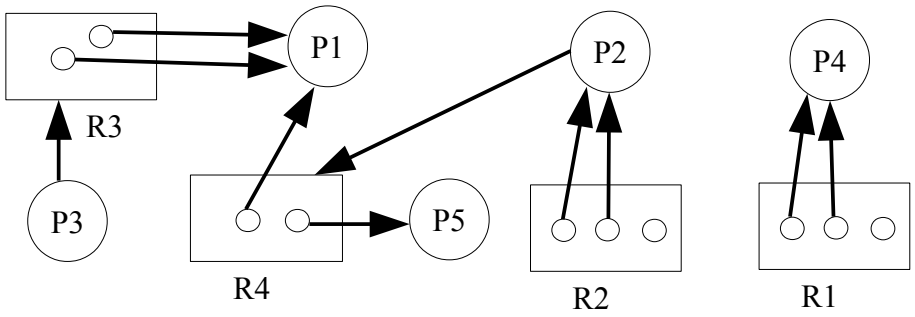
- Los procesos tipo P1 sólo necesitan usar un byte del registro. Dada la forma de acceso permitido será el primer byte de cada una de las dos mitades.
- Los procesos tipo P2 necesitan usar dos bytes, no importa si es la mitad superior o la mitad inferior.
- Los procesos tipo P3 necesitan usar el registro completo.

Cuando algún registro quede libre, la prioridad en el acceso la tendrán los procesos tipo P3 (si es posible), seguido de los de tipo P2 y por último los de tipo P1. No obstante, todo proceso tipo P1 o P2 que al llegar encuentren procesos del tipo P3 esperando deberá esperar a que estos procesos accedan.

6. 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)**

7. Solucionar el problema anterior usando **buzones**. La solución debe ser correcta para un sistema distribuido, donde los procesos P1, P2 y P3 estén en máquinas distintas a la que contiene el registro (no comparten memoria). **(3 Puntos)**

8. **(1,5 Puntos)**. Tenemos un sistema operativo con 5 procesos, que en un momento dado presenta el siguiente grafo de asignación de recursos:



Se sabe que las necesidades máximas de los procesos son:

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

Si estamos usando el algoritmo del banquero para evitar los interbloqueos, ¿debería concederse a P1 un ejemplar del recurso de R2?. Justifique la respuesta.

ANEXO 1. Estructura de los procesos para los problemas 6 y 7

```
program Registro32;

const
     np1=5;
     np2=5;
     np3=5;

process type P1(id:integer);
begin
     repeat
          { PROTOCOLO OCUPACION }
          writeln('Proceso tipo P1 ',id,' accediendo');
          { PROTOCOLO LIBERACION }
     forever
end;

process type P2(id:integer);
begin
     repeat
          { PROTOCOLO OCUPACION }
          writeln('Proceso tipo P2 ',id,' accediendo');
          { PROTOCOLO LIBERACION }
     forever
end;

process type P3(id:integer);
begin
     repeat
          { PROTOCOLO OCUPACION }
          writeln('Proceso tipo P3 ',id,' accediendo');
          { PROTOCOLO LIBERACION }
     forever
end;

var
   i,j,k: integer;
   PR1: array[1..np1] of P1;
   PR2: array[1..np2] of P2;
   PR3: array[1..np3] of P3;

begin
  cobegin
    for i := 1 to np1 do PR1[i](i);
    for j := 1 to np2 do PR2[j](j);
    for k := 1 to np3 do PR3[k](k);
  coend
end.
```