Examen de Programación Concurrente y Distribuida

3º Curso de Grado en Ingeniería Informática

Febrero. Curso 2012-13

CUESTIONES

1. **(0,5 Puntos)** Dado el siguiente conjunto de instrucciones, indique el grafo de precedencia correspondiente.

```
s0;
cobegin
     s1;
     begin
          s2;
          s3;
          cobegin
               s4;
               s5
          coend;
          s6
     end;
coend;
s7
```

2. **(0,5 Puntos)** Dado el siguiente conjunto de instrucciones, utilice las condiciones de Bernstein para establecer el grafo de precedencias que le corresponde.

```
S1: a = 1;
S2: q = 3;
S3: x = a * y;
S4: p = q * q;
S5: b = p * c;
```

3. **(0,5 Puntos)** Dado el siguiente conjunto de instrucciones, indique el grafo de precedencia correspondiente.

```
program P1P2P3;
var
     s: semaphore;

process P1;
begin
     a;
     signal(s);
     b;
end;

process P2;
begin
     c;
     wait(s);
     wait(s);
     d;
end;
```

```
process P3;
begin
     e;
     signal(s);
     f;
end;

begin
     initial(s,0);
     cobegin
          P1;P2;P3;
     coend
end.
```

4. **(0,5 Puntos)** Describa brevemente las consideraciones de diseño para los sistemas distribuidos (No extenderse más de un folio por una cara).

PROBLEMAS

En un sistema existen 3 tipos distintos de procesos que comparten dos recursos de acceso exclusivo R1 y R2. (Ver anexo 1).

- Los procesos tipo P1, sólo necesitan acceder al recurso R1.
- Los procesos tipo P2, sólo necesitan acceder al recurso R2.
- Los procesos tipo P12 necesitan acceder simultáneamente a R1 y R2. Dichos procesos solicitan primero acceso a R1 y una vez que lo han conseguido solicitan el acceso a R2 sin soltar R1. Al finalizar liberan ambos recursos.

En el acceso a R1 tiene la prioridad el tipo de proceso del que haya más esperando. Una vez que un proceso tipo P12 ha conseguido el recurso R1 tiene prioridad en el acceso al recurso R2.

1. **(3 Puntos)**. Solucionar el problema anterior usando **monitores**. Se asume una semántica de la operación `resume` tipo “desbloquear y espera urgente” (la habitual de *Pascal-FC*).

2. **(3 Puntos)**. Solucionar el problema anterior usando **canales**.

3. **(2 Puntos)**. Tenemos un sistema operativo con 5 procesos (P1, P2, P3, P4 y P5). En el sistema tenemos 4 recursos: 2 ejemplares de R1, 3 ejemplares de R2, 2 ejemplares de R3 y 3 ejemplares de R4.

La matriz de necesidades máximas es:

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

Si en un momento dado la matriz de asignados es:

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

a) Se encuentra el sistema en un estado seguro. Justifique la respuesta.

b) Si estamos usando el algoritmo del banquero para evitar los interbloqueos. Debería concederse a p3 un ejemplar del recurso de R2.

c) Si en la situación inicial (antes del apartado 2) se realizan las siguientes acciones:

   - P4 obtiene un recurso de R2
   - P1 obtiene un recurso de R4
   - P2 solicita un ejemplar de R1
   - P3 solicita un ejemplar de R2
   - P1 solicita un ejemplar de R3

   ¿Se produce interbloqueo en el sistema?. Justifique la respuesta.

ANEXO 1. Estructura de los procesos para los problemas 1 y 2

```
program dosrecursos;

const
     np1=5;
     np2=5;
     np12=5;

process type P1(id:integer);
begin

     repeat
          { PROTOCOLO OCUPACION }
          writeln('Proceso tipo P1 ',id,' usando R1');
          { PROTOCOLO LIBERACION }
     forever
end;

process type P2(id:integer);
begin
     repeat
          { PROTOCOLO OCUPACION }
          writeln('Proceso tipo P2 ',id,' usando R2');
          { PROTOCOLO LIBERACION }
     forever
end;

process type P12(id:integer);
begin
     repeat
          { PROTOCOLO OCUPACION }
          writeln('Proceso tipo P12 ',id,' usando R1 y R2');
          { PROTOCOLO LIBERACION }
     forever
end;

var
   i,j,k: integer;
   PR1: array[1..np1] of P1;
   PR2: array[1..np2] of P2;
   PR12: array[1..nP12] of P12;

begin
  cobegin
    for i := 1 to np1 do PR1[i](i);
    for j := 1 to np2 do PR2[j](j);
    for k := 1 to np12 do PR12[k](k);
  coend
end.
```