Examen de Programación Concurrente y Distribuida

Septiembre. Curso 2013-14

CUESTIONES

1. Indique como el algoritmo del matón realiza la elección del proceso coordinador para sincronizar sistemas distribuidos. **(0,5 Puntos)**

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

```
S1: c = a + b;
S2: d = a * b + e;
S3: a = x;
S4: f = e / (a + 2);
S5: h = c * 5 + d;
S6: x = f * h;
```

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

```
   process P0
1. repeat
2.  c0 := 1;
3.  turno := 1;
4.  while c1=1 and turno=1 do;
5.  Sección Crítica
6.  c0 := 0;
7.  Resto0
8. forever
```

```
   process P1
1. repeat
2.  c1 := 1;
3.  turno := 0;
4.  while c0=1 and turno=0 do;
5.  Sección Crítica
6.  c1 := 0;
7.  Resto1
8. forever
```

Inicialmente, `c0=c1=0` y `turno=0`. Para referirse a una determinada secuencia de instrucciones, use los número de instrucción, precedidas del nombre del proceso.

4. Usando la instrucción hardware *exchange*, garantice la exclusión mutua para los procesos P1 y P2. No olvide indicar el valor inicial de las variables que use. **(0,75 Puntos)**

```
process P1
repeat

   Sección Crítica

   Resto1
forever
```

```
process P2
repeat

   Sección Crítica

   Resto2
forever
```

PROBLEMAS

5. Solucione el problema de los lectores-escritores con prioridad en lectura usando semáforos. **(2 Puntos)**

6. En un sistema existen dos tipos de procesos PA y PB y dos tipos de recursos R1 y R2. El recurso R1 dispone de dos ejemplares, mientras que el recurso R2 solamente dispone de uno.

   - Los procesos PA necesitan usar simultáneamente los dos ejemplares de R1.
   - Los procesos PB necesitan usar un ejemplar cualquiera de R1 o de R2. Los procesos PB solo podrán usar los ejemplares de R1 si ningún proceso PA está esperando por él.
   - Si un proceso PA quiere usar los ejemplares de R1 deberá esperar a que ambos estén disponibles, es decir, no puede tomar uno y esperar por el otro.

   a) Solucionar el problema anterior usando **monitores**. Se asume una semántica de la operación `resume` tipo "desbloquear y espera urgente" (la habitual de *Pascal-FC*). **(2,25 Puntos)**

   b) Solucionar el problema anterior usando **canales**. Considerar en esta ocasión que solo existen procesos del tipo P1 y que no existe escáner. La solución debe ser correcta para un sistema distribuido, donde los procesos estén en máquinas distintas. **(2,25 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`.**

7. **(1 Puntos)**. Tenemos un sistema operativo con 5 procesos, y cuatro recursos que presentan los siguientes ejemplares: (3,2,3,2).

   Se sabe que las necesidades máximas de los procesos son:

	R1	**R2**	**R3**	**R4**
P1	1	1	2	0
P2	1	1	3	1
P3	1	1	2	1
P4	0	2	1	1
P5	2	0	0	2

   y que en un momento dado, los recursos asignados son:

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

   A partir de ese momento llegan las siguientes solicitudes por parte de los procesos:

   1. `P4 Solicita 1 ejemplar de R2`
   2. `P3 Solicita 1 ejemplar de R3`
   3. `P2 Solicita 1 ejemplar de R1`
   4. `P2 Solicita 1 ejemplar de R3`
   5. `P5 Solicita 1 ejemplar de R1`
   6. `P3 Solicita 1 ejemplar de R3`
   7. `P5 Solicita 1 ejemplar de R4`
   8. `P1 Solicita 1 ejemplar de R2`

   ¿Llega el sistema a interbloquearse?. En caso afirmativo, ¿qué solicitud lo provoca?. Justifique la respuesta usando la técnica adecuada.

ANEXO 1. Estructura de los procesos para la cuestión 5

```
program LectEscr;

process type Lectores(id:integer);
begin




     {LEER RECURSO}




end;

process type Escritores(id:integer);
begin


     {ESCRIBIR RECURSO}


end;


var
   Escritor: array[1..10] of Escritores;
   Lector: array[1..10] of Lectores;
   i,j:integer;




begin
   cobegin
     for i := 1 to 10 do Lector[i](i);
     for j := 1 to 10 do Escritor[j](j)
   coend
end.
```

ANEXO 2. Estructura de los procesos para el problema 6

```
program DosDos;



process type PA(id:integer);
begin


     {ACCESO A LOS DOS EJEMPLARRES DE R1}


end;

process type PB(id:integer);
begin


     { ACCESO A UN EJEMPLAR }


end;


var
   i,j: integer;
   PR1: array[1..10] of PA;
   PR2: array[1..10] of PB;




begin
  cobegin
    for i := 1 to 10 do PR1[i](i);
    for j := 1 to 10 do PR2[j](j);
  coend
end.
```