Febrero_2015_SOLUCION.pdf

Javi_Mac

Programación Concurrente y Distribuida

3º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería
Universidad de Huelva

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 *exchage*, 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                          process P1
repeat                              repeat
  c0 := quiereentrar;                 c1 := quiereentrar;
  turno := 1;                         turno := 0;
  while (c1 = quiereentrar)  do;      while (c0 = quiereentrar) do;
  Sección Crítica0                    Sección Crítica1
  c0 := fueraSC;                      c1 := fueraSC;
  Resto0                              Resto1
forever                             forever
```

SOLUCION:

```
process P0                          process P1
repeat                              repeat
  c0 := quiereentrar;                 c1 := quiereentrar;
  turno := 1;                         turno := 0;
  while (c1 = quiereentrar)           while (c0 = quiereentrar)
            and (turno = 1) do;                 and (turno = 0) do;
  Sección Crítica0                    Sección Crítica1
  c0 := fueraSC;                      c1 := fueraSC;
  Resto0                              Resto1
forever                             forever
```

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

MULTI VERSIAL.

MULTIVERSIAL: LA NEWSLETTER

¿Sabes por qué Facebook se cambia de nombre a Meta?

- Cada día el mejor **resumen de negocios digitales** en tu correo
- Infórmate de toda la actualidad en castellano **en 5 minutos**
- **Un nueva visión** sobre tecnología y la estrategia de las mejores empresas y startups

SUSCRÍBETE GRATIS

www.multiversial.es

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

var

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

```
process P2
repeat
      Sección Crítica
forever
```

SOLUCIÓN

```
program alterna;

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

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

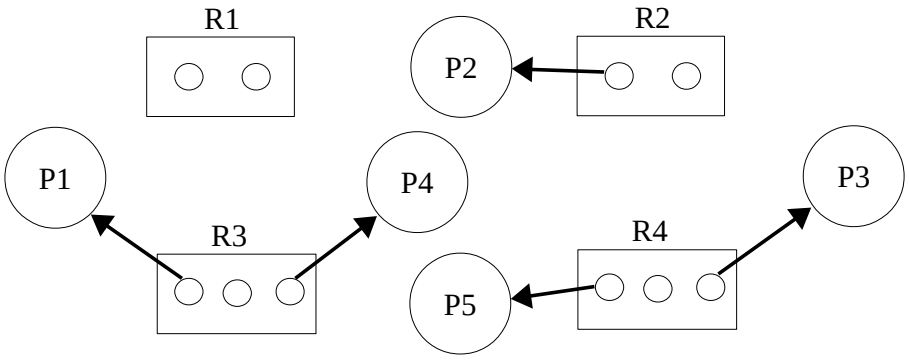
```
process P2
repeat
      wait(s2);
      Sección Crítica
      signal(s1);
forever
```

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`.**

1 solución

```
program Febrero15;

const
        npA=5;
        npB=5;
        NORTE=0;
        SUR=1;
        SINSENTIDO=-1;

var
        entraNorte: array[1..npA] of channel of synchronous;
        entraSur: array[1..npB] of channel of synchronous;
        saleNorte: array[1..npA] of channel of synchronous;
        saleSur: array[1..npB] of channel of synchronous;

process Controlador;
var
         c1, c2, c3, c4, c5:integer;
        dentroNorte, dentroSur:integer;

begin
```

```
        dentroNorte:=0;
        dentroSur:=0;

        repeat
                pri select

                        for c1:=1 to npA replicate
                                when (dentroNorte<2) and (dentroSur=0) =>
                                entraNorte[c1] ? any;
                                dentroNorte:=dentroNorte+1;
                or
                        for c2:=1 to npB replicate
                                when (dentroSur<2) and (dentroNorte=0) =>
                                entraSur[c2] ? any;
                                dentroSur:=dentroSur+1;
                or
                        for c3:=1 to npA replicate
                                saleNorte[c3] ? any;
                                dentroNorte:=dentroNorte-1;
                or
                        for c4:=1 to npB replicate
                                saleSur[c4] ? any;
                                dentroSur:=dentroSur-1;
                or
                        terminate
                end
        forever
end;


process type Tnorte(id:integer);
begin
        repeat
                entraNorte[id] ! any;
                writeln('Vehiculo ',id,' cruzando desde el norte');
                sleep(2);
                saleNorte[id] ! any;
        forever
end;

process type Tsur(id:integer);
```

```
var
 cual: integer;
begin
        repeat
                entraSur[id] ! any;
                writeln('Vehiculo ',id,' cruzando desde el sur');
                sleep(2);
                saleSur[id] ! any;
        forever
end;
```

```
var
  i,j: integer;
  Pnorte: array[1..npA] of Tnorte;
  Psur: array[1..npB] of Tsur;

begin
 cobegin
        Controlador;
   for i := 1 to npA do Pnorte[i](i);
   for j := 1 to npB do Psur[j](j);
 coend
end.
```

2 Solución

```
program Febrero15;

const
        np1=5;
        np2=5;


{Definicion del monitor}
monitor m;

        export
                entranorte, entrasur, salenorte, salesur;
        var

                colanorte, colasur: condition;
```

```
        dentroNorte, dentroSur, cupo:integer;

    procedure entranorte(id:integer);
    begin
        if (dentroNorte = 2) or (dentroSur > 0) or (cupo>4)  then delay(colanorte);
        dentroNorte := dentroNorte + 1;
        if not empty(colasur) then cupo := cupo + 1;
    end;

    procedure salenorte(id:integer);
    begin
        dentroNorte := dentroNorte - 1;;
        if not empty(colanorte) and (cupo < 4) then resume(colanorte)
        else if dentroNorte=0 then begin
                cupo:=0;
                resume(colasur);
                resume(colasur);
        end;
    end;

    procedure entrasur(id:integer);
    begin
        if (dentroSur = 2) or (dentroNorte > 0) or (cupo > 4)   then delay(colasur);
        dentroSur := dentroSur + 1;
        if not empty(colanorte) then cupo := cupo + 1;
    end;

    procedure salesur(id:integer);
    begin
        dentroSur := dentroSur - 1;;
        if not empty(colasur) and (cupo < 4) then resume(colasur)
        else if dentroSur=0 then begin
                cupo:=0;
                resume(colanorte);
                resume(colanorte);
        end;
    end;

    begin
        dentroNorte:=0;
        dentroSur:=0;
        cupo:=0;
    end;
(* Fin del monitor *)

process type Tnorte(id:integer);
begin
    repeat
        m.entranorte(id);
        writeln('Vehiculo ',id,' cruzando desde el norte');
```

```
                m.salenorte(id);
                sleep(random(10));
        forever
end;

process type Tsur(id:integer);
begin
        repeat
                m.entrasur(id);
                writeln('Vehiculo ',id,' cruzando desde el sur');
                m.salesur(id);
                sleep(random(10));
        forever
end;

var
  i,j: integer;
  Pnorte: array[1..np1] of Tnorte;
  Psur: array[1..np2] of Tsur;

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

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.
```