

Febrero19.pdf



mrsergio8



Programación Concurrente y Distribuida



3º Grado en Ingeniería Informática



Escuela Técnica Superior de Ingeniería
Universidad de Huelva

WUOLAH + BBVA

Te regalamos



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

1

Abre tu Cuenta
Online
sin comisiones
ni condiciones

2

Haz una compra
igual o superior
a 15€ con tu
nueva tarjeta

3

BBVA
te devuelve
un máximo de
15€

Te regalamos



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

1
Abre tu Cuenta Online sin comisiones ni condiciones

2
Haz una compra igual o superior a 15€ con tu nueva tarjeta

3
BBVA te devuelve un máximo de 15€

Examen de Programación Concurrente y Distribuida

Febrero de 2019

Examen de Programación Concurrente y Distribuida 3º Curso de Grado en Ingeniería Informática

Febrero. Curso 2018-19

1. Explique de forma escueta los diferentes modelos de comunicación de procesos en los Sistemas Distribuidos. **(0,5 Puntos)**

-Modelo Cliente-Servidor: modelo no orientado a conexión donde una serie de procesos clientes solicitan servicios a otros procesos servidores, los cuales proporcionan dichos servicios o devuelven un error. El núcleo de cada máquina es quien se encarga del intercambio de mensajes.

-Llamada a procedimiento remoto (RPC): Basado en la estructura cliente-servidor, usa llamadas a procedimientos y funciones para establecer la comunicación, dichas funciones no tienen por qué encontrarse en la máquina que las ejecuta.

-Modelo de objetos distribuidos: usa el mecanismo de invocación a objetos RMI, usando módulos a los que se accede de forma remota. No se puede acceder a las variables de forma remota.

-Comunicación de grupos: Se trata de enviar mensajes a diferentes miembros de un grupo de procesos situados en diferentes máquinas del sistema distribuido. Esos grupos son dinámicos (pueden crearse, destruirse, incorporar componentes, excluir procesos pertenecientes...). Para la comunicación se puede usar tanto redes multicasting como broadcasting.

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

```
process P0
repeat
    while c <> 1 do;
        Sección Crítica
        c := c * -1;
        Resto0
    forever
```

```
process P1
repeat
    while c <> -1 do;
        Sección Crítica
        c := c * -1;
        Resto1
    forever
```

Donde inicialmente $c = 1$

Garantiza la exclusión mutua ya que al requerir que la condición tenga un valor en concreto y no modificar dicha condición hasta salir de la sección crítica nos aseguramos que ambos procesos no van a entrar simultáneamente a dicha sección. Aunque si algún proceso sufriera algún error en sus sección crítica, el otro podría esperar indefinidamente.

3. Usando únicamente semáforos, haga que, de forma cíclica, los procesos accedan a la sección crítica en la siguiente secuencia: P1, P2, P2, P3, P1, P2, P2, P3, P1, P2, P2, P3, (1,5 Puntos)

No se considera válida la solución si no se inicializan los semáforos correctamente

```
Program unodosuno
var
process P1          process P2          process P3          begin
begin               begin               begin               cobegin
repeat              repeat              repeat              P1;P2;P3;
                    Sección Crítica      Sección Crítica      coend
                    Resto1
forever             forever             forever
end                 end                 end
                    Resto1
                    Resto2
                    forever
end                 end                 end

Program unodosuno
var
    S1, S2, S3 : semaphore
process P1
    repeat
        wait(S1);
        SC
        signal(S2);
        signal(S2);
        resto
    forever
end

process P2
    repeat
        wait(S2);
        SC
        signal(S3);
        signal(S3);
        resto
    forever
end

process P3
    repeat
        wait(S3);
        wait(S3);
        SC
        signal(S1);
        resto
    forever
end

begin
    initial(S1,1); initial(S2,0); initial(S3,0);
    cobegin
        P1; P2; P3;
    coend
end
```

1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

Te regalamos



1

Abre tu Cuenta
Online
sin comisiones
ni condiciones

2

Haz una compra
igual o superior
a 15€ con tu
nueva tarjeta

3

BBVA
te devuelve
un máximo de
15€



4. En una estación de I.T.V trabajan dos inspectores y un supervisor. A dicha estación acuden dos tipos distintos de vehículos:

- Vehículos tipo A. Son aquellos que quieren pasar una inspección periódica. Estos pueden ser atendidos indistintamente por cualquier inspector o por el supervisor.
- Vehículos tipo B. Son aquellos que quieren validar una modificación técnica. Estos necesitan ser atendidos **simultáneamente** por el supervisor y uno cualquiera de los inspectores.

Los vehículos tipo B tendrán prioridad sobre los del tipo A, es decir, los del tipo A no podrán pasar si hay de tipo B esperando, aunque esté libre alguno de los tres.

Cada vehículo será representado por un hilo.

- 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)**
- Solucionar el problema anterior usando **buzones**. **(3 Puntos)**

a) Monitores

```

1  program ITV;
2
3  const
4      nPA=15;
5      nPB=15;
6
7  monitor mon;
8      export
9          entraA, entraB, saleA, saleB;
10
11 var
12     libreInspector, libreSupervisor : integer;
13     esperaA, esperaB : condition;
14
15 procedure entraA(cual: integer);
16 begin
17     if not empty(esperaB) or (libreInspector = 0 and libreSupervisor = 0) then
18         delay(esperaA);
19         if libreInspector > 0 then
20             begin
21                 libreInspector := libreInspector - 1;
22                 cual := 1;
23             end
24         else if libreSupervisor > 0 then
25             begin
26                 libreSupervisor := libreSupervisor - 1;
27                 cual := 2;
28             end
29     end
30
31 procedure saleA(cual: integer);
32 begin
33     if cual = 1 then
34         libreInspector := libreInspector + 1;
35     else if cual = 2 then
36         libreSupervisor := libreSupervisor + 1;
37     if not empty(esperaB) and libreInspector > 0 and libreSupervisor > 0 then
38         resume(esperaB);
39     else if empty(esperaB)
40         resume(esperaA);
41 end

```

```

41             resume(colaInspeccion);
42         end
43
44     procedure entraModificacion()
45     begin
46         if(inspectores_libres=0 OR supervisor_libre=false) then
47             delay(colaModificacion);
48             inspectores_libres--;
49             supervisor_libre:=false;
50         end
51
52     procedure saleModificacion()
53     begin
54         inspectores_libres++;
55         supervisor_libre:=true;
56         if not empty(colaModificacion) then
57             resume(colaModificacion);
58         else
59             resume(colaInspeccion);
60             resume(colaInspeccion);
61         end
62
63     begin
64         inspectores_libres:=2;
65         supervisor_libre:=true;
66     //fin monitor
67
68 process type vehiculoInspeccionPeriodica(id:integer);
69 var
70     cual : integer; //1 si lo atiende inspector 2 si lo atiende supervisor
71 begin
72     { PROTOCOLO OCUPACION }
73     m.entradaInspeccion(cual);
74     writeln('Entra vehículo ' + id + ' para inspección periodica');
75     m.saleInspeccion(cual);
76     { PROTOCOLO LIBERACION }
77 end;
78
79 process type vehiculoModificacionTecnica(id:integer);
80 begin
81     { PROTOCOLO OCUPACION }
82     m.entradaModificacion();
83     writeln('Entra vehículo ' + id + ' para modificación técnica');
84     m.saleModificacion();
85     { PROTOCOLO LIBERACION }
86 end;
87
88 var
89     i,j: integer;
90     VehiculoA: array[1..nPA] of TIPOA;
91     VehiculoB: array[1..nPB] of TIPOB;
92
93 begin
94     cobegin
95         for i := 1 to nPA do VehiculoA[i](i);
96         for j := 1 to nPB do VehiculoB[j](j);
97     coend
98 end

```

Te regalamos



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

1

Abre tu Cuenta Online sin comisiones ni condiciones

2

Haz una compra igual o superior a 15€ con tu nueva tarjeta

3

BBVA te devuelve un máximo de 15€

b)Buzones

```

1  program ITV;
2  const
3      nPA=15;
4      nPB=15;
5
6  var
7      entraInspeccion: mailbox of integer; //solicita entrar
8      saleInspeccion: mailbox of integer; //avisa de que sale
9      asignado: array[1...nPA] of mailbox of integer; //atendido (devuelve quien lo
10     atiende)
11     atendidoModificacion: array[1...nPB] of mailbox of integer; //atentido
12     entraModificacion: mailbox of integer; //solicita entrar
13     saleModificacion: mailbox of integer; //avisa de que sale
14
15 process controlador;
16     var
17         inspectores_libres : integer;
18         supervisor_libre : boolean;
19         id:integer;
20         begin
21             inspectores_libres:=2;
22             supervisor_libre:=true;
23             repeat
24                 select
25                     //entra inspeccion
26                     when empty (entraModificacion) and (inspectores_libres>0 or
27                     supervisor_libre=true) =>
28                         receive(entraInspeccion, id);
29                         if inspectores_libres>0 then
30                             begin
31                                 inspectores_libres--;
32                                 send(asignado[id], 1); //1 si es inspector
33                             end
34                         else
35                             begin
36                                 supervisor_libre:=false;
37                                 send(asignado[id], 2); //2 si es supervisor
38                             end
39             //entra supervision
40             when not empty (entraModificacion) and inspectores_libres>0 AND
41             supervisor_libre=true =>
42                 receive(entraModificacion, id);
43                 begin
44                     inspectores_libres--;
45                     supervisor_libre:=false;
46                     send(atendidoModificacion, 1); //el numero es irrelevante
47                 end
48             //sale inspeccion
49             or receive(saleInspeccion, cual);
50                 if(cual=1) then
51                     inspectores_libres++;
52                 else if(cual=2) then
53                     supervisor_libre:=true;
54
55             //sale modificacion
56             or receive(saleModificacion, id);
57                 inspectores_libres++;
58                 supervisor_libre:=true;
59
59         end select
60     forever
61 end

```

```

60
61 process type vehiculoInspeccionPeriodica(id:integer);
62 var
63     cual : integer; //1 si lo atiende inspector 2 si lo atiende supervisor
64 begin
65     { PROTOCOLO OCUPACION }
66     send (entraInspeccion,id);
67     receive (asignado[id], cual);
68     writeln('Entra vehiculo ' + id + ' para inspeccion periodica');
69     send saleInspeccion(cual);
70     { PROTOCOLO LIBERACION }
71 end;
72
73 process type vehiculoModificacionTecnica(id:integer);
74 var
75     atendido:integer; //solo se usara para saber que se ha atendido, su valor es
76     irrelevante
77 begin
78     { PROTOCOLO OCUPACION }
79     send (entraModificacion, id);
80     receive (atendidoModificacion[id], atendido)
81     writeln('Entra vehiculo ' + id + ' para modificacion tecnica');
82     send (saleModificacion, id);
83     { PROTOCOLO LIBERACION }
84 end;
85 var
86     i,j: integer;
87     VehiculoA: array[1..nPA] of TIPOA;
88     VehiculoB: array[1..nPB] of TIPOb;
89     controlador: Controlador;
90
91 begin
92     cobegin
93         for i := 1 to nPA do VehiculoA[i](i);
94         for j := 1 to nPB do VehiculoB[j](j);
95     coend
96 end

```

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

Se sabe que las necesidades máximas de los procesos son:

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

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

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

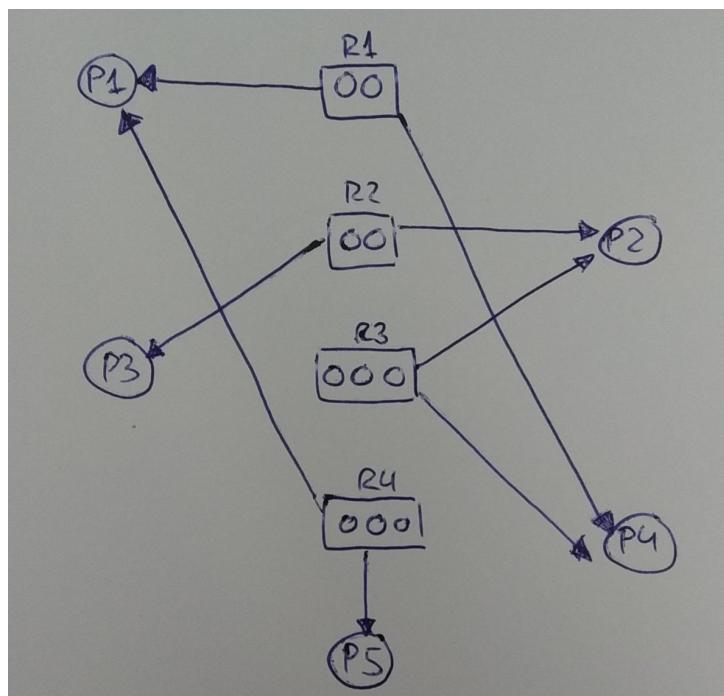
A partir de ese momento llegan las siguientes solicitudes por parte de los procesos:

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

¿Llega el sistema a interbloquearse?. En caso afirmativo, ¿qué solicitud lo provoca?. Justifique la respuesta usando la técnica adecuada. **(1,25 Puntos)**.

Con la ayuda de un grafo estudiaremos una a una las peticiones para ver si alguna provoca un interbloqueo. Analizaremos el estado de las peticiones, si se aceptan, si están esperando a que se les acepte etc.

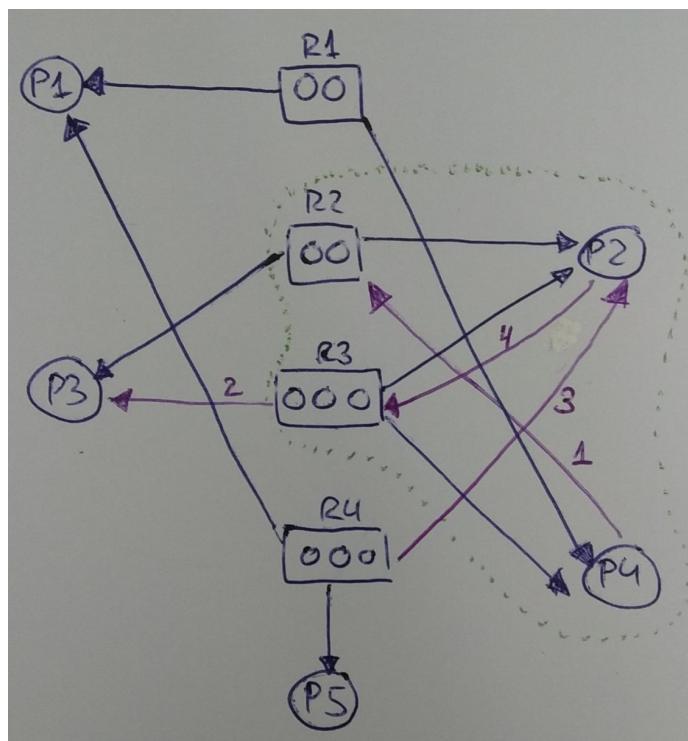
Grafo con los recursos asignados inicialmente:



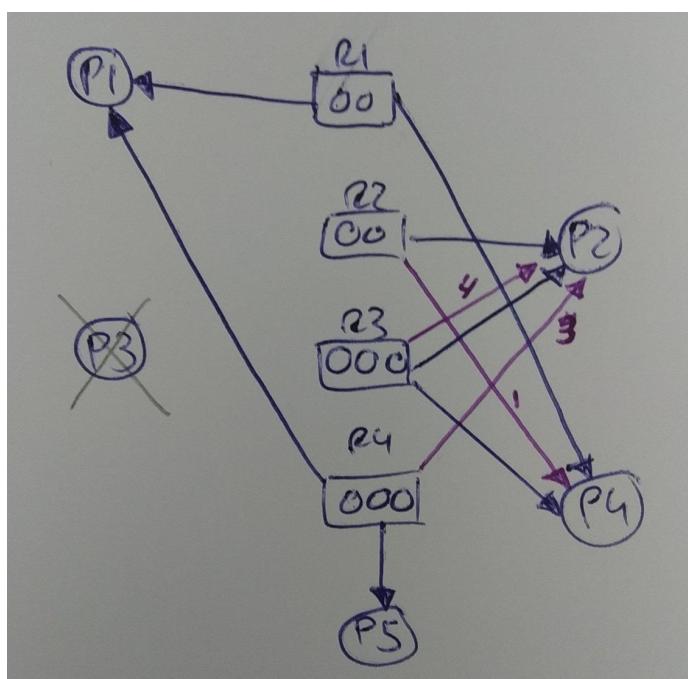
- 1- P4 → R2: En espera.
- 2- P3 → R3: Aceptada.
- 3- P2 → R4: Aceptada.
- 4- P2 → R3: En Espera.

Tras esta petición se produce un ciclo entre P2-R3-P4-R2.

Grafo tras la petición 4:



Se puede resolver reduciendo el grafo en el proceso P3, cuyas peticiones actuales están concedidas. Tras reducir, quedan un recurso R2 y otro de R4 disponibles para que no se produzca el ciclo mencionado anteriormente.



Te regalamos



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

1

Abre tu Cuenta Online sin comisiones ni condiciones

2

Haz una compra igual o superior a 15€ con tu nueva tarjeta

3

BBVA te devuelve un máximo de 15€

5- P1 → R4: En espera

6- P3 → R3: En espera. Esta petición provoca que aparezcan dos ciclos, el primero de ellos es el que apareció tras la petición 4 (P2-R3-P4-R2), ya que al realizar P3 una nueva petición vuelve a incluirse en el grafo y las peticiones que tenía aceptadas anteriormente y que se reasignaron al reducir el grafo vuelven a pertenecer a P3. Por otra parte se forma otro ciclo en P3-R3-P4-R2. Podría reducirse el grafo primero por P5 e inmediatamente después por P1 pero los ciclos seguirían estando por lo que ese grafo sería irreducible por lo que la petición 6 produce un **INTERBLOQUEO**

