

Concurrencia de procesos: Interbloqueo e inanición

Definición

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Cena de los
filósofos

Eloy Anguiano

Rosa M^a Carro

Ana González

Escuela Politécnica Superior
Universidad Autónoma de Madrid

Definición

Ejemplos
Estados seguros e
inseguros

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Cena de los
filósofos

- **Bloqueo permanente** de un conjunto de procesos que compiten por los recursos o bien se comunican unos con otros.
- Suponen necesidades conflictivas de recursos por parte de dos o más procesos.
- El bloqueo ocurre cuando un proceso monopoliza el acceso a un recurso y requiere otro recurso que ha sido ya asignado a un segundo proceso, que a su vez necesita el recurso monopolizado por el primer proceso.
- El interbloqueo es permanente porque ninguno de los eventos esperados puede producirse.
- No existe una solución eficiente.

Definición

Ejemplos

Concurrencia de
procesos:
Interbloqueo e
iniciación

- Cena de filósofos sin coordinación.
- Ley del estado de Kansas: *“Si dos trenes se aproximan uno al otro en un cruce, ambos harán un alto total y ninguno arrancará de nuevo hasta que el otro haya salido del cruce”*.

Definición

Ejemplos

Estados seguros e
inseguros

Recursos
reutilizables

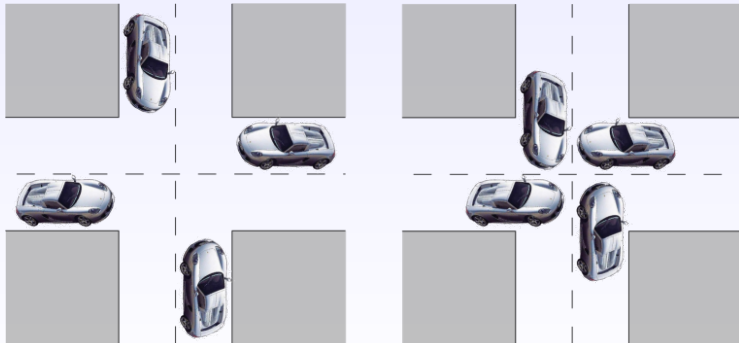
Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Cena de los
filósofos



Posible Interbloqueo

Interbloqueo

Definición

Estados seguros e inseguros

Concurrencia de
procesos:
Interbloqueo e
inanición

Definición

Ejemplos
Estados seguros e
inseguros

Recursos
reutilizables

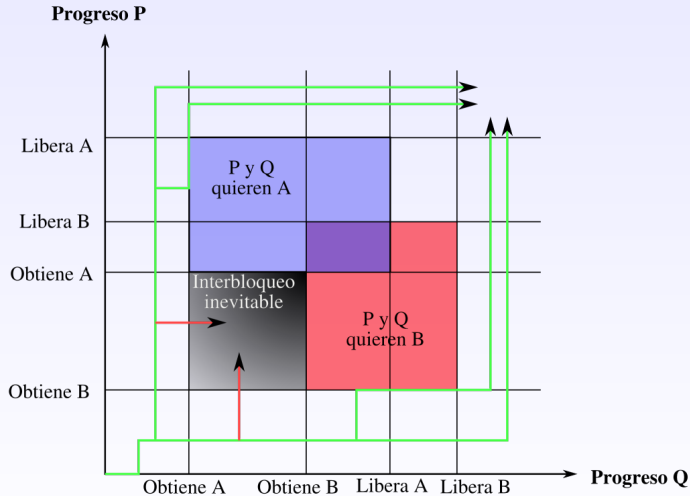
Recursos
consumibles

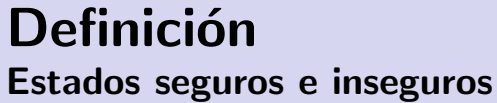
Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Cena de los
filósofos



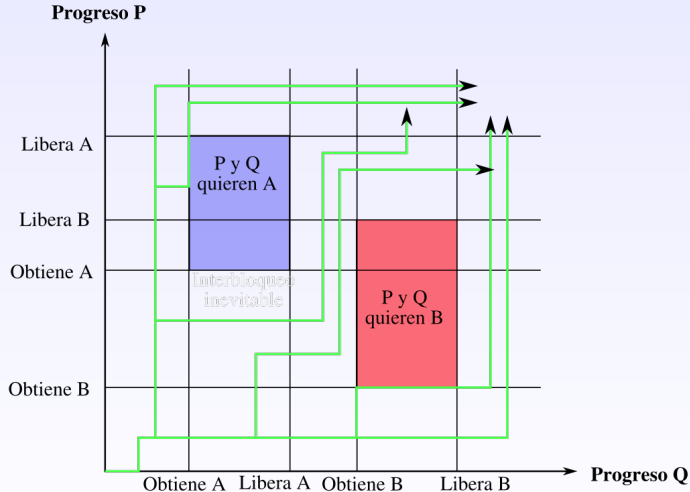


Definición

Recursos consumibles

Condiciones de interbloqueo

Gestión de interbloques





Recursos reutilizables

Características

Concurrencia de
procesos:
Interbloqueo e
iniciación

Definición

Recursos
reutilizables

Características
Ejemplos

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Cena de los
filósofos

- Pueden ser usados por un proceso y no se agotan con el uso.
- Los procesos obtienen unidades de recursos que liberan posteriormente para que otros procesos las reutilicen.
- Procesadores, canales de E/S, memoria principal y secundaria, archivos, bases de datos y semáforos.
- El interbloqueo se produce si cada proceso retiene un recurso y solicita el otro.

Recursos reutilizables

Ejemplos

Concurrencia de
procesos:
Interbloqueo e
inanición

Definición

Recursos
reutilizables

Características
Ejemplos

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Cena de los
filósofos

Acceso a zonas de memoria compartida utilizando semáforos

Memoria Compartida

Semáforo Protege_Var1=1

Semáforo Protege_Var2=1

int Var1

int Var2

Proceso 1

...

down(Protege_Var1);

down (Protege_Var2);

Var1 = Var2+1;

up (Protege_Var2);

up (Protege_Var1);

Proceso 2

...

down(Protege_Var2);

down (Protege_Var1);

Var2 = 2*Var1;

up (Protege_Var1);

up (Protege_Var2);

Recursos reutilizables

Ejemplos

Concurrencia de
procesos:
Interbloqueo e
inanición

Definición

Recursos
reutilizables

Características
Ejemplos

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

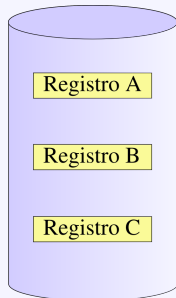
Gestión de
interbloqueos

Cena de los
filósofos

Acceso a registros de bases de datos

Instrucciones

- 1 Proceso 1: Copia Reg. A a Reg. B
- 2 Proceso 2: Copia Reg. B a Reg. C
- 3 Proceso 3: Copia Reg. C a Reg. A



Proceso

- 1 Pr.1: Solicita Reg. A
- 2 Pr.1: Cierra Reg. A
- 3 Pr.2: Solicita Reg. B
- 4 Pr.2: Cierra Reg. B
- 5 Pr.3: Solicita Reg. C
- 6 Pr.3: Cierra Reg. C
- 7 Pr.1: Solicita Reg. B
- 8 Pr.2: Solicita Reg. C
- 9 Pr.3: Solicita Reg. A
- 10 Pr.1: Espera
- 11 Pr.2: Espera
- 12 Pr.3: Espera

Recursos reutilizables

Ejemplos

Concurrencia de
procesos:
Interbloqueo e
iniciación

Definición

Recursos
reutilizables

Características
Ejemplos

Recursos
consumibles

Grafos de
asignación de
recursos

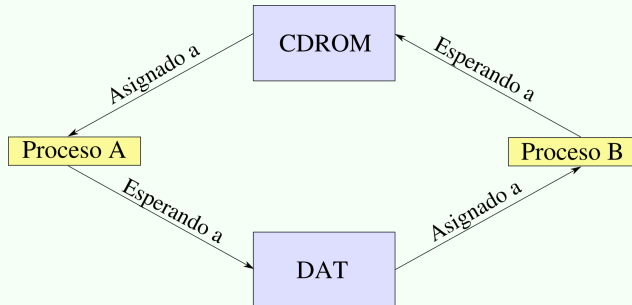
Condiciones de
interbloqueo

Gestión de
interbloqueos

Cena de los
filósofos

Acceso a dispositivos

- 1 Proceso A copia un fichero del CD-ROM a una cinta de backup (DAT).
- 2 Proceso B copia un fichero de la cinta de backup (DAT) al CD-ROM.



Recursos reutilizables

Ejemplos

Concurrencia de
procesos:
Interbloqueo e
inanición

Definición

Recursos
reutilizables

Características
Ejemplos

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Cena de los
filósofos

Uso de memoria en sistemas multiprogramados

Si el espacio disponible es de 200 KB y se origina la siguiente secuencia de peticiones:

- 1 Proceso 1: solicita 80 KB
- 2 Proceso 2: solicita 70 KB
- 3 Proceso 1: solicita 60 KB
- 4 Proceso 2: solicita 80 KB

se produce un interbloqueo si ambos procesos avanzan hasta su segunda petición.

Recursos consumibles

Características

- Puede ser creado (producido) y destruido (consumido) por un proceso.
- Interrupciones, señales, mensajes e información en buffers de E/S.
- El interbloqueo se produce si el **receive** es bloqueante.
- Puede darse una combinación de sucesos poco habitual que origine el interbloqueo.

Por transmisión de mensajes

Si el **receive** es bloqueante estos procesos pueden producir interbloqueo:

Proceso 1

```
...  
Receive(P2);  
...  
Send(P2,M1);  
...
```

Proceso 2

```
...  
Receive(P1);  
...  
Send(P1,M2);  
...
```

Grafos de asignación de recursos

Definición

Permiten visualizar la situación (asignación y solicitud) de recursos.

Proceso i :



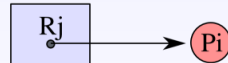
Recurso tipo j (un único recurso):



Recurso tipo j (tres recursos):



Recurso j asignado al proceso i :



Proceso i a la espera del recurso j :



Grafos de asignación de recursos

Ejemplo de grafo con bloqueo

Concurrencia de
procesos:
Interbloqueo e
inanición

Definición

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

Definición
Ejemplo de grafo con
bloqueo

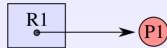
Ejemplo de grafo sin
bloqueo

Condiciones de
interbloqueo

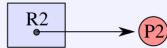
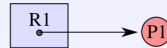
Gestión de
interbloqueos

Cena de los

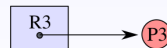
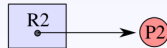
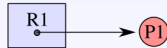
1.- Proceso 1: solicita recurso 1 y se asigna



2.- Proceso 2: solicita recurso 2 y se asigna

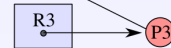
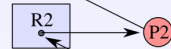
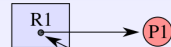


3.- Proceso 3: solicita recurso 3 y se asigna

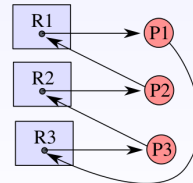


4.- Proceso 2: solicita recurso 1

5.- Proceso 3: solicita recurso 2



6.- Proceso 1: solicita recurso 3



Grafos de asignación de recursos

Ejemplo de grafo sin bloqueo

Concurrencia de
procesos:
Interbloqueo e
iniciación

Definición

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

Definición
Ejemplo de grafo con
bloqueo

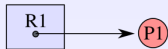
Ejemplo de grafo sin
bloqueo

Condiciones de
interbloqueo

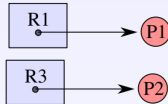
Gestión de
interbloqueos

Cena de los

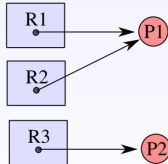
1.- Proceso 1: solicita recurso 1 y se asigna



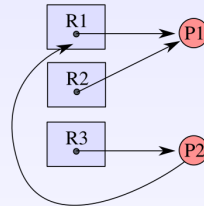
2.- Proceso 2: solicita recurso 3 y se asigna



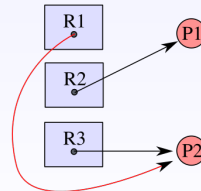
3.- Proceso 1: solicita recurso 2 y se asigna



4.- Proceso 2: solicita recurso 1



5.- Proceso 1: libera recurso 1



Condiciones de interbloqueo

Concurrencia de
procesos:
Interbloqueo e
inanición

Definición

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Cena de los
filósofos

1 Exclusión mutua:

- Sólo un proceso puede usar un recurso cada vez.

2 Retención y espera:

- Hay al menos un proceso que tiene asignado un recurso y se encuentra en espera de que otro proceso libere otro recurso.

3 No apropiación / Sin expropiación:

- No se puede forzar la expropiación de un recurso al proceso que lo tiene.

4 Espera circular:

- Si se dibuja el grafo de asignación y solicitud de recursos, tiene forma circular (ciclo).

Las tres primeras condiciones son necesarias pero no suficientes para que exista interbloqueo. La cuarta condición es una consecuencia potencial de las tres primeras.

Gestión de interbloqueos

Concurrencia de
procesos:
Interbloqueo e
iniciación

Definición

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Prevención de
interbloqueos

Detección de
interbloqueos

Recuperación de
interbloqueos

Evasión de
interbloqueos

- 1 **No hacer nada** (algoritmo del avestruz) Ej. UNIX. Decisión de diseño, por motivos de flexibilidad. Ej. N° de archivos abiertos.
- 2 Uso de **protocolos** que aseguren que el sistema no se **bloqueará**:
 - Mediante **prevención**: garantizar que una (o más) de las condiciones necesarias para la formación de interbloqueos no se cumpla.
 - Mediante **evasión (basada en predicción)**: proporcionar al sistema información anticipada sobre las necesidades de recursos de los procesos, para que pueda predecir qué ocurrirá y encontrar secuencias de asignación de recursos que eviten los interbloqueos.
- 3 Permitir que el sistema se bloquee y proporcionar mecanismos de **detección periódica** y **recuperación** de interbloqueos.

Gestión de interbloqueos

Prevención de interbloqueos

Concurrencia de
procesos:
Interbloqueo e
inanición

Definición

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Prevención de
interbloqueos

Detección de
interbloqueos
Recuperación de
interbloqueos
Evasión de
interbloqueos

La forma de garantizar que no se producirán interbloqueos se basa en garantizar que no se cumpla una de las condiciones necesarias para la formación de los mismos

- ① Prevención de la condición de **exclusión mutua**:
 - Problema: hay recursos que necesariamente requieren la exclusión mutua:
 - Archivos con permiso de escritura.
 - La tabla de procesos.
 - La tabla de nodo-i activos.
 - La memoria.
 - El espacio en disco.
 - ...
 - En general, no se puede impedir la exclusión mutua. No es una solución.

Gestión de interbloqueos

Prevención de interbloqueos

Concurrencia de
procesos:
Interbloqueo e
inanición

Definición

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Prevención de
interbloqueos

Detección de
interbloqueos

Recuperación de
interbloqueos

Evasión de
interbloqueos

2 Prevención de la condición de **Retención y Espera**:

- Se exige a los procesos que soliciten todos los recursos al comenzar su ejecución.
- Un proceso queda bloqueado hasta que se le conceden simultáneamente todas sus solicitudes.
- Problemas:
 - Un proceso no 'sabe' a priori que recursos va a utilizar.
 - Uso muy ineficiente de los recursos: los recursos se acaparan desde el inicio, aunque no se usen hasta el final del proceso.
 - Se puede producir la inanición: procesos no bloqueados pueden no tener nunca acceso a un recurso acaparado por otro.

Gestión de interbloqueos

Prevención de interbloqueos

Concurrencia de
procesos:
Interbloqueo e
inanición

Definición

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Prevención de
interbloqueos

Detección de
interbloqueos

Recuperación de
interbloqueos

Evasión de
interbloqueos

- 3 Prevención contra la condición de **no apropiación / sin expropiación**:
- Si un proceso solicita un recurso no disponible, se interrumpe y se le expropia de todos sus recursos.
 - El proceso se reinicia cuando se le pueden proporcionar todos los recursos que necesita.
 - Es práctico y realizable con recursos cuyo estado puede ser fácilmente almacenable/recuperable:
 - Registros de CPU.
 - Memoria.
 - No es trivial expropiar sin causar efectos secundarios nocivos con otro tipo de recursos:
 - Ficheros inconsistentes.

Gestión de interbloqueos

Prevención de interbloqueos

Concurrencia de
procesos:
Interbloqueo e
iniciación

Definición

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Prevención de
interbloqueos

Detección de
interbloqueos

Recuperación de
interbloqueos

Evasión de
interbloqueos

④ Prevención contra la condición de **espera circular**:

- ① Hacer que los recursos sólo puedan utilizarse uno a uno. Esto no es posible en muchos casos: ej. copia de un fichero de cinta a disco.
 - ② Numerar los recursos, de modo que sólo se puedan solicitar en un orden determinado.
 - ③ Variante: Sólo se pueden solicitar recursos que tengan un número mayor del último que se haya solicitado.
- **Problema**: cuando hay un número alto de recursos, no es fácil diseñar un orden que satisfaga a todos los procesos.

Gestión de interbloqueos

Detección de interbloqueos

Concurrencia de
procesos:
Interbloqueo e
iniciación

Definición

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Prevención de
interbloqueos

Detección de
interbloqueos

Recuperación de
interbloqueos

Evasión de
interbloqueos

Bloqueos con un solo recurso de cada tipo

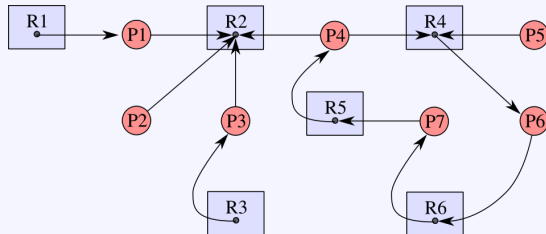
Para cada nodo de la gráfica de procesos/recursos se crea una lista de nodos vacía **L** que se rellena ejecutando el siguiente algoritmo:

- 1 Se añade el nodo activo a la lista **L** y se comprueba si el nodo está dos veces en la lista. Si está dos veces la gráfica tiene un ciclo (y por tanto hay un bloqueo).
- 2 Desde el nodo activo se buscan arcos que salgan del mismo. Si los hay se pasa a **3**, si no a **4**.
- 3 Se elige un arco que no haya sido recorrido, se marca como recorrido y se marca el nodo apuntado por el arco como activo. Se vuelve a **1**.
- 4 Se ha llegado a un punto donde no se puede avanzar. Se vuelve al nodo anterior y se marca como activo. Se pasa al paso. **1**. Si el nodo al que se vuelve es el inicial se acaba. No hay bloqueo.

Gestión de interbloqueos

Detección de interbloqueos

Ejemplo



Se van comprobando todos los caminos:

- 1 $L(R_1) = \langle R_1 \rangle \langle R_1, P_1 \rangle \langle R_1, P_1, R_2 \rangle \langle R_1, P_1 \rangle \langle R_1 \rangle$ **No hay bloqueo**
- 2 $L(P_1) = \langle P_1 \rangle \langle P_1, R_2 \rangle \langle P_1 \rangle$ **No hay bloqueo**
- 3 $L(R_2) = \langle R_2 \rangle$ **No hay bloqueo**
- 4 $L(P_4) = \langle P_4 \rangle \langle P_4, R_2 \rangle \langle P_4 \rangle \langle P_4, R_4 \rangle \langle P_4, R_4, P_6 \rangle \langle P_4, R_4, P_6, R_6 \rangle \langle P_4, R_4, P_6, R_6, P_7 \rangle \langle P_4, R_4, P_6, R_6, P_7, R_5 \rangle \langle P_4, R_4, P_6, R_6, P_7, R_5, P_4 \rangle$ Hay un ciclo luego **hay bloqueo**

Gestión de interbloqueos

Detección de interbloqueos

Concurrencia de
procesos:
Interbloqueo e
iniciación

Definición

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Prevención de
interbloqueos

Detección de
interbloqueos

Recuperación de
interbloqueos

Evasión de
interbloqueos

Bloqueos con varios recursos de cada tipo

- Vector **Recursos existentes** – *Exist* $E = \{E_1, E_2, E_3, \dots, E_m\}$
- Vector **Recursos disponibles** – *Available* $A = \{A_1, A_2, A_3, \dots, A_m\}$
- Matriz **Asignación actual** – *Current* $C = \begin{Bmatrix} C_{11} & C_{12} & \cdots & C_{1m} \\ C_{21} & C_{22} & \cdots & C_{2m} \\ \vdots & \vdots & & \vdots \\ C_{n1} & C_{n2} & \cdots & C_{nm} \end{Bmatrix}$
- Matriz **Solicitudes** – *Required* $R = \begin{Bmatrix} R_{11} & R_{12} & \cdots & R_{1m} \\ R_{21} & R_{22} & \cdots & R_{2m} \\ \vdots & \vdots & & \vdots \\ R_{n1} & R_{n2} & \cdots & R_{nm} \end{Bmatrix}$

Gestión de interbloqueos

Detección de interbloqueos

Concurrencia de
procesos:
Interbloqueo e
inanición

Definición

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Prevención de
interbloqueos

Detección de
interbloqueos

Recuperación de
interbloqueos

Evasión de
interbloqueos

Bloqueos con varios recursos de cada tipo

- $\sum_i C_{ij} + A_j = E_j$

Algoritmo del banquero:

- 1 Se busca un proceso no marcado P_i que cumpla que $\forall x, R_{ix} \leq A_x$. Si no hay procesos sin marcar, se acaba el algoritmo (**no hay bloqueo**).
- 2 Si se encuentra un proceso que cumpla esa condición, se marca, se suma la línea C_{ix} a A y se vuelve a 1.
- 3 Si no existe ese proceso, se acaba el algoritmo (**hay bloqueo**).

Gestión de interbloqueos

Detección de interbloqueos

Concurrencia de
procesos:
Interbloqueo e
inanición

Definición

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Prevención de
interbloqueos

Detección de
interbloqueos

Recuperación de
interbloqueos

Evasión de
interbloqueos

Ejemplo

Recursos existentes

$$E = \{4, 2, 3, 1\}$$

Recursos disponibles

$$A = \{2, 1, 0, 0\}$$

Matriz de solicitudes (aún por asignar)

$$R = \begin{Bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{Bmatrix}$$

Matriz de asignación actual

$$C = \begin{Bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{Bmatrix}$$

Algoritmo

$$1.- R_1 \not\leq A, R_2 \not\leq A, R_3 \leq A$$

$$2.- A = \{2, 2, 2, 0\}$$

$$1.- R_1 \not\leq A, R_2 \leq A$$

$$2.- A = \{4, 2, 2, 1\}$$

$$1.- R_1 \leq A$$

$$2.- A = \{4, 2, 3, 1\}$$

1.- No hay procesos sin marcar. **Sin bloqueo**

Gestión de interbloqueos

Recuperación de interbloqueos

Concurrencia de
procesos:
Interbloqueo e
inanición

Definición

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Prevención de
interbloqueos

Detección de
interbloqueos

Recuperación de
interbloqueos

Evasión de
interbloqueos

Cuando se detecta un interbloqueo, hay varias formas posibles de recuperación:

1 Recuperación mediante apropiación:

- Se selecciona un proceso (o varios) y se le 'requisan' los recursos para cederlos a otros procesos bloqueados.

2 Recuperación mediante Rollback:

- Se almacena en un fichero periódicamente el estado del proceso (imagen de memoria) y estado de los recursos utilizados. Al producirse un bloqueo se detecta los recursos que son necesarios y un proceso que tenga alguno de esos recursos se interrumpe y se retrasa hasta el punto de verificación anterior a la solicitud del recurso (ojo: puede repetirse el bloqueo).

Gestión de interbloqueos

Recuperación de interbloqueos

Concurrencia de
procesos:
Interbloqueo e
inanición

Definición

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Prevención de
interbloqueos

Detección de
interbloqueos

Recuperación de
interbloqueos

Evasión de
interbloqueos

- ③ Recuperación mediante eliminación de procesos. Hay dos posibilidades de actuación:
- Se abortan todos los procesos bloqueados.
 - Se selecciona un procesos bloqueado o uno no bloqueado que tenga recursos necesarios. Elegir procesos que se puedan reiniciar (si existen). Se termina de eliminar procesos cuando se desbloquee el sistema.
 - Criterios para la selección de procesos a terminar:
 - Menor prioridad.
 - Menor líneas de salida producidas al terminar.
 - Menor tiempo de CPU consumido.
 - Mayor tiempo restante.
 - Menor número de recursos asignados.
 - Tipo de recursos asignados.
 - Mayor número de recursos necesarios para terminación.
 - Problemas:
 - Si el proceso está actualizando un archivo, su eliminación puede causar inconsistencias.
 - Si el procesos está imprimiendo, se reinicia la impresora.

Gestión de interbloqueos

Evación de interbloqueos (predicción)

Concurrencia de
procesos:
Interbloqueo e
inanición

Definición

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Prevención de
interbloqueos

Detección de
interbloqueos

Recuperación de
interbloqueos

Evación de
interbloqueos

- Para poder **evadir** interbloqueos, antes de asignar un recurso hay que comprobar que al asignarlo no se conduce al interbloqueo. Se trata de **predecir** qué ocurriría si se asigna.
- Algoritmo: se simula que se concede el recurso (se 'anota' como asignado y se aplica el algoritmo del banquero para comprobar si existiría interbloqueo en caso de producirse esa asignación).
- Si se encuentra una secuencia de asignación de recursos factible que no conduzca al interbloqueo (es decir, si tras la asignación del recurso al proceso el estado es seguro), el recurso se concede.
- Es necesario que todos los procesos declaren sus necesidades totales de recursos al principio.

Cena de los filósofos

Concurrencia de
procesos:
Interbloqueo e
inanición

Definición

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

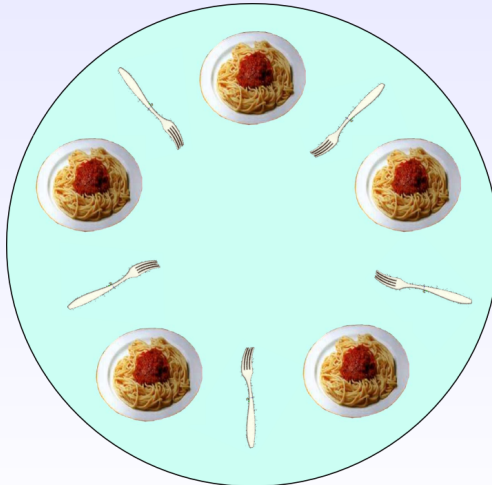
Condiciones de
interbloqueo

Gestión de
interbloqueos

**Cena de los
filósofos**

Primer intento de
solución

Segundo intento de
solución



Cena de los filósofos

Primer intento de solución

Concurrencia de
procesos:
Interbloqueo e
inanición

Definición

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Cena de los
filósofos

Primer intento de
solución

Segundo intento de
solución

```
1  #define N 5
2
3  void filosofo (void) {
4      int i;
5      while (1) {
6          piensa();
7          coge_palillo(i);
8          coge_palillo((i+1) % N);
9          come();
10         deja_palillo(i);
11         deja_palillo((i+1) % N);
12     }
13 }
```

Cada uno coge primero el palillo izquierdo y luego el derecho.

Posible interbloqueo:

Todos cogen primero el palillo de su izquierda y no pueden coger el de la derecha porque está asignado.

Cena de los filósofos

Segundo intento de solución

Concurrencia de
procesos:
Interbloqueo e
inanición

Definición

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Cena de los
filósofos

Primer intento de
solución

Segundo intento de
solución

```
1  #define N 5
2
3  void filosofo (void) {
4      int i;
5      while (1) {
6          piensa();
7          coge_palillo(i);
8          if(libre((i+1) % N)) {
9              coge_palillo((i+1) % N);
10             come();
11             deja_palillo(i);
12             deja_palillo((i+1) % N);
13         }
14         else deja_palillo(i);
15     }
16 }
```

Coger el palillo izquierdo y, si no puede coger el palillo derecho, soltar el izquierdo.

Posible círculo vicioso de espera.

Posible inanición si los filósofos actúan simultáneamente (poco probable pero posible).

Cena de los filósofos

Solución 1

Concurrencia de
procesos:
Interbloqueo e
inanición

Definición

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Cena de los
filósofos

Primer intento de
solución

Segundo intento de
solución

```
1  typedef int semaforo;
2
3  #define N 5
4  semaforo comen = N-1;
5
6  void filosofo(int i)
7  {
8      int i;
9      while (1) {
10         piensa();
11         down (comen);
12         coge_palillo(i);
13         coge_palillo((i+1) %N);
14         come();
15         deja_palillo(i);
16         deja_palillo((i+1) %N);
17         up (comen);
18     }
19 }
```

Se limita a $N-1$ el número de filósofos que pueden comer a la vez.

Por tanto, alguno podrá comer y cuando termine soltará sus palillos, dejándoselos a otros, que podrán estar bloqueados esperando.

No hay interbloqueo ni inanición.

Cena de los filósofos

Solución 2

Concurrencia de
procesos:
Interbloqueo e
iniciación

Definición

Recursos
reutilizables

Recursos
consumibles

Grafos de
asignación de
recursos

Condiciones de
interbloqueo

Gestión de
interbloqueos

Cena de los
filósofos

Primer intento de
solución
Segundo intento de
solución

```
1 #define N 5
2 #define IZQD (i-1)%N
3 #define DCHA (i+1)%N
4 #define PENSANDO 0
5 #define HAMBRIENTO 1
6 #define COMIENDO 2
```

```
1 typedef int semaforo;
2 int estado[N];
3 semaforo mutex=1;
4 semaforo S[N]={0};
```

```
1 void filosofo(int i)
2 {
3     while (1) {
4         piensa();
5         coge_palillos(i);
6         come();
7         deja_palillos(i);
8     }
9 }
10
11 void coge_palillos(int i)
12 {
13     down(mutex);
14     estado[i]= HAMBRIENTO;
15     intento(i);
16     up(mutex);
17     down(S[i]);
18 }
```

```
1 void deja_palillos(int i)
2 {
3     down(mutex);
4     estado[i]= PENSANDO;
5     intento(IZQD);
6     intento(DCHA);
7     up(mutex);
8 }
9
10 void intento (int i)
11 {
12     if(estado[i] == HAMBRIENTO &&
13        estado[IZQD] != COMIENDO &&
14        estado[DCHA] != COMIENDO) {
15         estado[i] = COMIENDO;
16         up(S[i]);
17     }
18 }
```