



INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO
SISTEMAS OPERATIVOS



Unidad 5

Actividad 5

INTEGRANTES DEL EQUIPO:

COLIN RAMIRO JOEL
HERNÁNDEZ REYES JULIO CÉSAR
MALDONADO CERÓN CARLOS
MENDOZA GARCÍA ELIÚ EDUARDO

GRUPO: 4CM1

PROFESOR: CORTÉS GALICIA JORGE

Interbloqueos

- Varios procesos pueden competir por un finito número de recursos.
- Un proceso solicita recursos, si estos recursos no están disponibles, este pasa a un estado de “espera”.
- Es posible que un proceso en este estado, nunca pueda cambiar del mismo, ya que los recursos solicitados, están ocupados por otros procesos, los cuales a su vez, también esperan otros recursos.
- A esta situación se le conoce como **INTERBLOQUEO**.

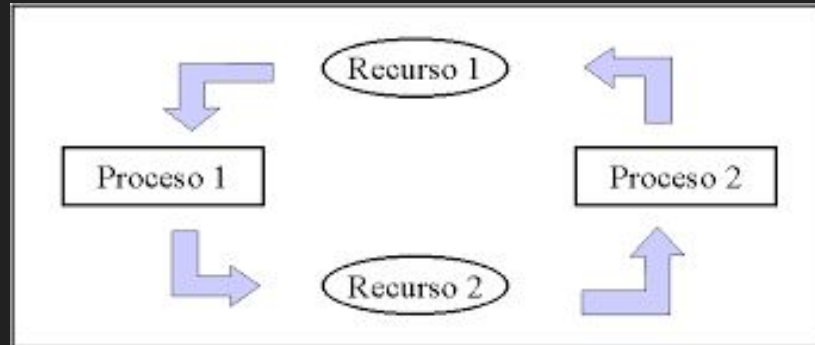
Una buena forma de ilustrar la definición de interbloqueo:



La mayoría de los S.O actuales, no proporcionan facilidades para la prevención de estos bloqueos, sin embargo, es muy probable que en algunos años, se añadan dichos mecanismos.

Estos problemas, son cada vez más habituales, debido a la gran variedad de tendencias que existen actualmente como lo pueden ser:

- La gran cantidad de procesos.
- La utilización de programas “multihebra”.
- Existencia de muchos más recursos dentro de un sistema
- Preferencia por servidores de archivos y bases de datos con transacciones largas.



1. *Modelo de sistema*

Un sistema está constituido por un número finito de recursos. Estos se distribuyen entre una serie de procesos.

Se pueden dividir en varios tipos, cada uno de ellos con un número n de instancias:

- Espacio de memoria.
- Ciclos de CPU
- Archivos de Dispositivos de E/S

Ahora bien, cuando un proceso solicita una instancia de algún tipo de recurso, la asignación de cualquier instancia del mismo tipo satisfará dicha solicitud.

Un proceso debe solicitar cada recurso antes de utilizarlo y debe liberarlo después de usado. Este proceso puede solicitar todos los recursos que necesite para llevar a cabo sus tareas asignadas.

En un modo de operación normal, un proceso puede emplear un recurso, simplemente al seguir la siguiente secuencia:

1. **SOLICITUD** .- Si dicha solicitud no puede ser concedida al momento, el proceso solicitante tendrá que esperar hasta que pueda adquirir el recurso mencionado.



2. **USO**.- El proceso puede operar sobre el recurso.



3. **LIBERACIÓN**.- El proceso libera el recurso.

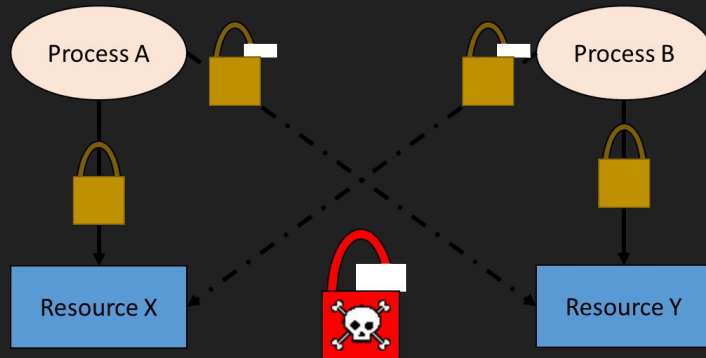
2. Caracterización de Interbloqueos

En un interbloqueo, los procesos nunca concluyen con su ejecución y los recursos del sistema están ocupados, lo que impide que se inicien otros trabajos.

Actualmente existen múltiples métodos para abordar este problema de los interbloqueos:

2.1. Condiciones necesarias

Para que suceda una situación de interbloqueo en un sistema, se deben de dar simultáneamente las siguientes situaciones:



1. **Exclusión Mutua.-** Sólo un proceso puede usarlo una vez. Si otro proceso solicita el mismo recurso, este tendrá que esperar hasta que dicho recurso se libere.
2. **Retención y Espera.-** Un proceso debe estar reteniendo al menos un recurso a la vez y esperando para adquirir otros recursos adicionales que actualmente estén retenidos por otros procesos.
3. **Sin Desalojo.-** Un recurso sólo puede ser liberado voluntariamente por el proceso que le retiene, después de que dicho proceso haya completado su tarea.
4. **Espera Circular.-** Debe existir un conjunto de procesos ($P_0, P_1, P_2, \dots, P_n$) en espera, tal que P_0 esté esperando a un recurso retenido por P_1 , P_1 esté esperando a un recurso retenido por P_2 y así sucesivamente, hasta que P_n espere a un recurso retenido por P_0 .

2.2. Grafo de asignación de recursos

Los interbloqueos, pueden representarse de una forma más clara mediante un grafo dirigido, el cual toma el nombre de ***grafo de asignación de recursos del sistema***.

- Consta de un conjunto de vértices(V) y un conjunto de aristas(E).
- El conjunto de vértices se divide en dos:
 1. Conjunto de todos los procesos activos del sistema ($P_0, P_1, P_2, \dots, P_n$).
 2. Conjunto de todos los tipos($R_0, R_1, R_2, \dots, R_n$).
- Una arista dirigida desde el proceso P_i al recurso R_j se indica: $P_i \rightarrow R_j$. Esto significa que el proceso P_i ha solicitado una instancia del tipo de recurso R_j y que está esperando el mencionado recurso.
- Una arista $P_i \rightarrow R_j$, se llama **arista de solicitud**, mientras que $P_i \leftarrow R_j$, se le denomina **arista de asignación**.

Cuando el proceso P_i solicita una instancia del tipo de recurso R_j , se inserta una arista de solicitud en el grafo de asignación de recursos. Al ser concedida dicha solicitud, la arista de solicitud se convierte en arista de asignación. Cuando el proceso ya no necesita acceso al recurso, se libera y se elimina la arista de asignación.

Conjuntos

$P = \{P_1, P_2, P_3\}$

$R = \{R_1, R_2, R_3, R_4\}$

$E = \{P_1 \rightarrow R_1, P_2 \rightarrow R_3, R_1 \rightarrow P_2, R_2 \rightarrow P_2, R_2 \rightarrow P_1, R_3 \rightarrow P_3\}$

Instancias

Una instancia R_1

Dos instancias R_2

Una instancia R_3

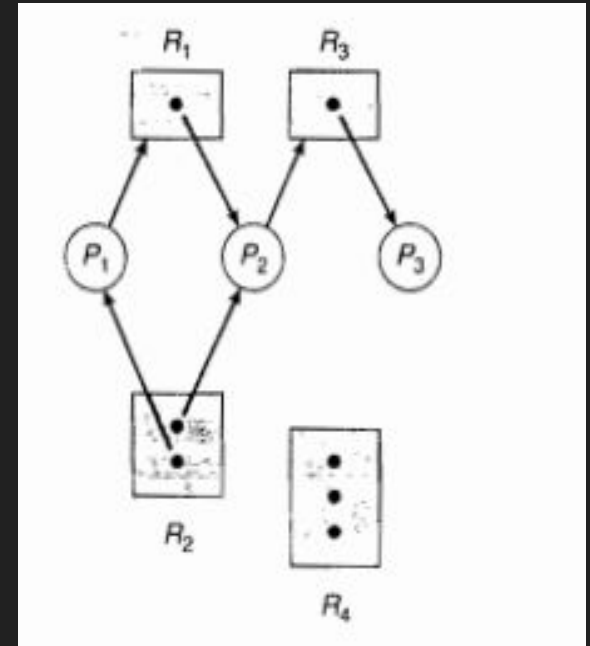
Tres instancias R_4

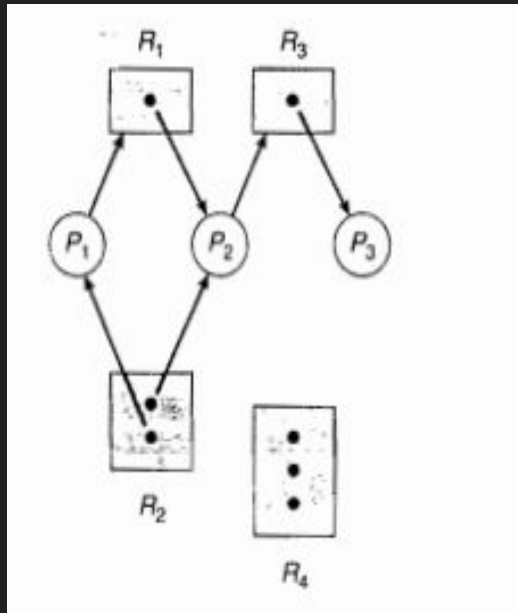
Estados de los Procesos

P_1 retiene una instancia de R_2 y espera una instancia de R_1

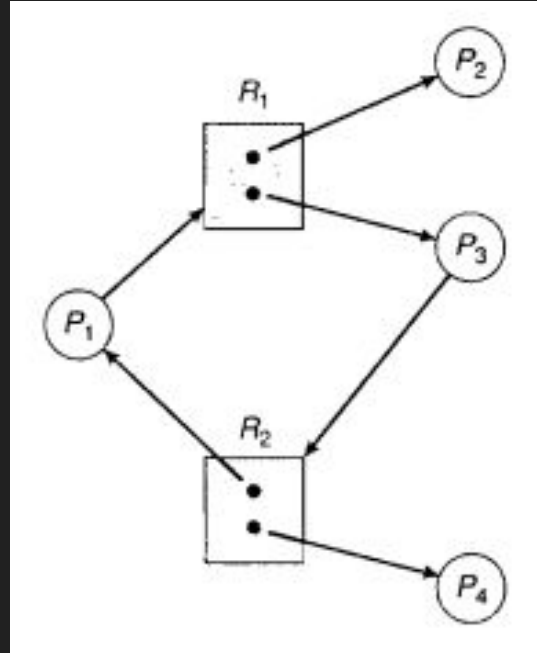
P_2 retiene una instancia de R_2 y espera una instancia de R_3

P_3 retiene una instancia de R_3

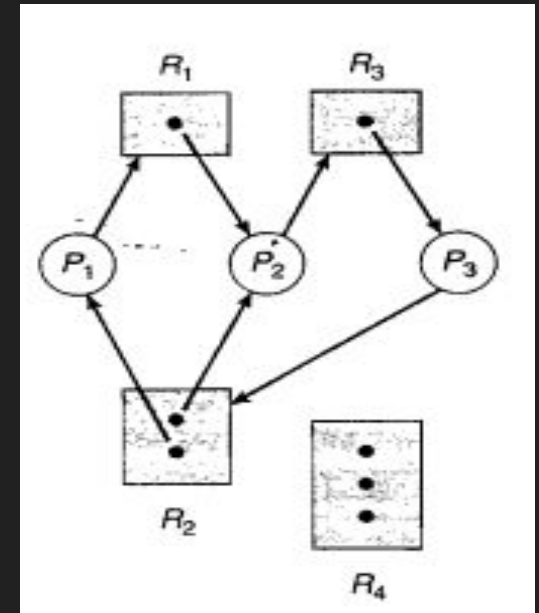




Grafo de asignación de recursos.



Grafo de asignación de recursos con un ciclo pero sin interbloqueo.



Grafo de asignación de recursos con interbloqueo.

3. *Métodos para tratar los Interbloqueos*

Los interbloqueos se pueden abordar de 3 formas:

- 3.1.** Emplear un protocolo para el impedimento de estos mismo, asegurando así que el sistema nunca entre en estado de interbloqueo.
- 3.2.** Permitir que el sistema entre en estado de interbloqueo, detectarlo y realizar una recuperación.
- 3.3.** Ignorar el problema y actuar como si nunca se produjeran interbloqueos en el sistema. Esta es la que la mayoría de los S.O, utilizan.

Estos 3 métodos pueden combinarse para así, permitir la selección de un método óptimo para cada clase de recurso existente en el sistema.

Para garantizar que nunca se produzcan los mencionados interbloqueos, el sistema puede llegar a emplear un:

Esquema de PREVENCIÓN de interbloqueos

ó

Esquema de EVASIÓN de interbloqueos

La **prevención de interbloqueos** proporciona un conjunto de métodos para asegurar que al menos una de las condiciones necesarias no puede cumplirse. Estos métodos evitan los interbloqueos restringiendo el modo en que se pueden realizar las solicitudes.

Mientras que la **evasión de interbloqueos** requiere que se proporcione al S.O. información adicional sobre los recursos a solicitar además de que utilizará un proceso durante su tiempo de vida

4. *Prevención de Interbloqueos*



Como se mencionó en diapositivas anteriores, existen 4 condiciones necesarias para la producción de un interbloqueo. Entonces para prevenir que un interbloqueo, se debe asegurar que una de estas 4 condiciones no se cumpla.

- **Exclusión Mutua.-** Se aplica a los recursos que no pueden ser compartidos. Un proceso no necesita esperar nunca para acceder a un recurso compatible. No obstante no se pueden evitar los interbloqueos negando esta condición, debido a que algunos recursos no son compatibles.
- **Retención y Espera.-** Se debe garantizar que cuando un proceso solicite un recurso, este proceso no se encuentre reteniendo otro recurso. Esto se puede llevar a cabo exigiendo que cada proceso solicite todos sus recursos antes de comenzar su ejecución
- **Sin desalojo.-** Si un proceso está reteniendo varios recursos y solicita otro recurso el cual no se le pueda asignar de forma inmediata, entonces todos los recursos se desalojan.
- **Espera Circular.-** Se trata de imponer una ordenación total de todos los tipos de recursos y requerir que cada proceso solicite sus recursos en un orden creciente de enumeración.

5. Evasión de Interbloqueos

5.1. Estado seguro.- Un estado se dice que es seguro si el sistema puede asignar recursos a cada proceso en determinado orden sin que eso produzca un interbloqueo.

En otras palabras, un sistema es seguro, solo si existe lo que se denomina como **secuencia segura**. Esto se refiere a una secuencia de procesos $\langle P_1, P_2, \dots, P_n \rangle$ si para cada P_i , las solicitudes de recursos que P_i puede todavía hacer, pueden ser satisfechas mediante los recursos disponibles.

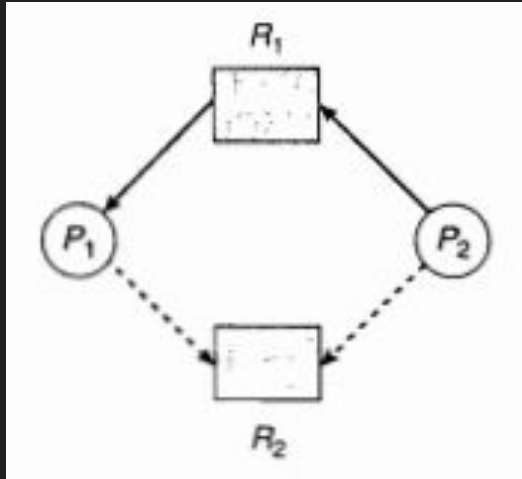
El hecho de que el estado sea seguro implica que no puede producirse el interbloqueo, es decir todo estado de interbloqueo implica que no sea seguro.

No obstante no todos los estados “inseguros” significa que tengan un interbloqueo.

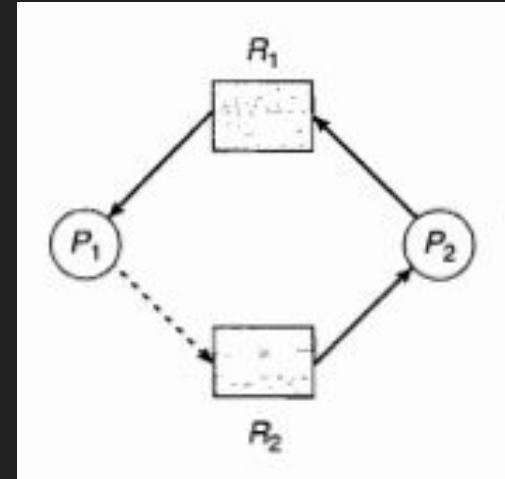
En un estado inseguro el S.O. no puede impedir que los procesos soliciten recursos de tal modo que se produzca el interbloqueo.



5.2 Grafo de Asignación de Recursos



Grafo de asignación de recursos para evitar interbloqueos



Estado inseguro en un Grafo de asignación de recursos

5.3 Algoritmo del Banquero

- Deben utilizarse varias estructuras de datos para su implementación.
- Estas estructuras codifican el estado del sistema de asignación de recursos.

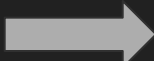
Si consideramos a **n** como el número de procesos y **m** como el número de tipos de recursos, se necesitan las siguientes estructuras:

1. **Available.-** Un vector de longitud **m** que indica el número de recursos disponibles de cada tipo. Si **Available**[*j*] = *k*, significa que existen *k* instancias disponibles del recurso **R_j**.
2. **Max.-** Una matriz **n x m** que indica la demanda máxima de cada proceso. Si **Max**[*i*][*j*] = *k*, significa que el proceso **P_i**, puede solicitar máximo *k* instancias del recurso **R_j**.
3. **Allocation.-** Una matriz **n x m** la cual define el número de recursos de cada uno actualmente asignados a cada proceso. Si **Allocation**[*i*][*j*] = *k*, entonces el proceso **P_i**, tiene asignadas *k* instancias del recurso **R_j**.
4. **Need.-** Una matriz **n x m** que indica la necesidad faltante de recursos de cada proceso. Si **Need**[*i*][*j*] = *k*, entonces el proceso **P_i**, necesita *k* instancias más del recurso **R_j**, para satisfacer sus tareas.

$$\mathbf{Need}[i][j] = \mathbf{Max}[i][j] - \mathbf{Allocation}[i][j]$$

5.4 Algoritmo de seguridad

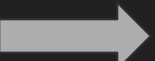
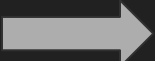
Sirve para averiguar si un sistema se encuentra en estado seguro o no.

1. Sean: **Work[n], Finish[m]**  **2 vectores**

Work = Available y Finish[i] = false para $i = 0, 1, \dots, n-1$

2. Hallar i, tal que:
 - a. Finish[i] = false
 - b. Need(i) \leq Work

Nota: Si no existe i que cumpla las condiciones, se debe saltar al paso 4

3. Work = Work + Allocation(i)  Finish[i] = true  Regresar al paso 2.
4. Si Finish[i] == true para todo i, significa que el sistema está en estado seguro.

Nota: Puede requerir un orden de $m \times n^2$ operaciones para determinar si el estado es seguro.



5.5 Algoritmo de solicitud de recursos

Sea **Request** el vector de solicitud para el proceso P_i , si **Request**[j] == k, entonces el proceso P_i , desea k instancias del tipo de recurso R_j . Se toman las siguientes acciones:

1. Si **Request**(i) <= Need, se debe pasar al 2do paso. Y si no se cumple, se genera una condición de error.
2. Si **Request**(i) <= Available, se debe pasar al 3er paso. En caso contrario, el proceso P_i deberá esperar, dado que los recursos no están disponibles.
3. Se debe suponer que el sistema asignó al proceso P_i los recursos solicitados, modificando el estado de la siguiente forma:

$$\begin{aligned} \text{Available} &= \text{Available} - \text{Request}_i \\ \text{Allocation}_i &= \text{Allocation}_i + \text{Request}_i \\ \text{Need}_i &= \text{Need}_i - \text{Request}_i \end{aligned}$$

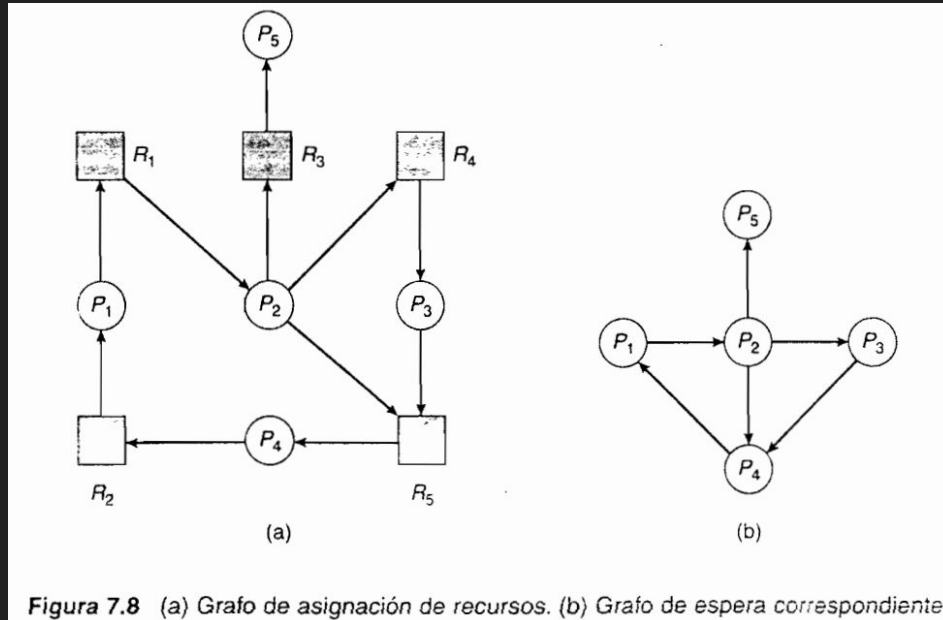
6. *Detección de Interbloqueos*

En caso de producirse interbloqueo en el sistema, el sistema debe proporcionar:

- Un algoritmo que examine el estado del sistema para determinar si se ha producido un interbloqueo
- Un algoritmo para recuperarse del interbloqueo

6.1 Una sola instancia de cada tipo de recurso

GRAFO DE ESPERA



6.2 Varias instancias de cada tipo de recurso

Un algoritmo de detección de interbloqueos emplea varias estructuras de datos variables con el tiempo:

- Available (Disponible)
Un vector de longitud m que indica el número de recursos disponibles de cada tipo
- Allocation (Asignación)
Una matriz $n \times m$ que define el número de recursos de cada tipo que están asignados actualmente a cada proceso
- Request (Solicitud)
Una matriz $n \times m$ que especifica la solicitud actual efectuada por cada proceso. Si $\text{Request}[i][j]$ es igual a k entonces el proceso P_i está solicitando k instancias más del tipo de recurso R_j .

Algoritmo de detección

1. *Work* y *Finish* son vectores de longitud *m* y *n*, respectivamente. Inicialmente, *Work* = *Available*. Para $i=0,1,\dots,n-1$, si $Allocation_i \neq 0$, entonces $Finish[i] = false$; en caso contrario, $Finish[i] = true$.
2. Hallar un índice *i* tal que
 - a. $Finish[i] == false$
 - b. $Request_i \leq Work$Si no existe tal *i*, ir al paso 4
3. $Work = Work + Allocation_i$
 $Finish[i] = true$
Ir al paso 2
4. Si $Finish[i] == false$, para algún *i* tal que $0 \leq i < n$, entonces el sistema se encuentra en estado de interbloqueo. Además, si $Finish[i] == false$, entonces el proceso P_i está en el interbloqueo.

Consideremos un sistema con 5 procesos, de P_0 a P_4 , y 3 tipos de recursos, A, B y C. El tipo de recurso A tiene siete instancias, el tipo de recurso B tiene dos instancias y el tipo de recurso C tiene 6 instancias. Suponga que en el instante T_0 tenemos el siguiente estado de la asignación de recursos.

	<u>Allocation</u>	<u>Request</u>	<u>Available</u>
	A B C	A B C	A B C
P_0	0 1 0	0 0 0	0 0 0
P_1	2 0 0	2 0 2	
P_2	3 0 3	0 0 0	
P_3	2 1 1	1 0 0	
P_4	0 0 2	0 0 2	

La secuencia $\langle P_0, P_2, P_3, P_1, P_4 \rangle$ da como resultado $\text{Finish}[i] == \text{true}$ para todo i

Suponga ahora que el proceso P2 hace una solicitud de una instancia más del tipo de recurso C. La matriz Reques se modifica como sigue:

<u>Request</u>			
A	B	C	
P_0	0	0	0
P_1	2	0	2
P_2	0	0	1
P_3	1	0	0
P_4	0	0	2

Existe un interbloqueo entre los procesos P1, P2, P3 y P4

6.3 Utilización del algoritmo de detección

¿Cuándo debemos invocar el algoritmo de detección? La respuesta depende de dos factores:

1. ¿Con qué frecuencia se producirá probablemente un interbloqueo?
2. ¿Cuántos procesos se verán afectados por el interbloqueo cuando se produzca?

7. Recuperación de un Interbloqueo

Cuando el algoritmo de detección determina que existe un interbloqueo, tenemos varias alternativas. Una posibilidad es informar al operador de que se ha producido un interbloqueo y dejar que lo trate de forma manual. Otra posibilidad es dejar que sea el sistema el que haga la recuperación del interbloqueo de forma automática. Existen dos opciones para romper un interbloqueo. Una de ellas consiste simplemente en interrumpir uno o más procesos para romper la cadena de espera circular. La otra consiste en desalojar algunos recursos de uno o más de los procesos bloqueados.

7.1 Terminación de procesos

- Interrumpir todos los procesos interbloqueados
Este método interrumpirá el ciclo de interbloqueo, pero a un precio muy alto: los procesos en interbloqueo pueden haber consumido un largo periodo de tiempo y los resultados de estos cálculos parciales deben descartarse y, probablemente, tendrán que repetirse más tarde
- Interrumpir un proceso cada vez hasta que el ciclo de interbloqueo se elimine
Este método requiere una gran cantidad de trabajo adicional, ya que después de haber interrumpido cada proceso hay que invocar un algoritmo de detección de interbloqueos para determinar si todavía hay procesos en interbloqueo.

Factores para decidir qué procesos interrumpir

1. La prioridad del proceso
2. Durante cuánto tiempo ha estado el proceso ejecutándose y cuánto tiempo adicional necesita para completar sus tareas
3. Cuántos y qué tipo de recursos ha utilizado el proceso (por ejemplo, si los recursos pueden desalojarse fácilmente)
4. Cuántos más recursos necesita el proceso para completarse
5. Cuántos procesos hará falta terminar
6. Si se trata de un proceso interactivo o de procesamiento por lotes

7.2 Apropiación de recursos

Si se necesita el desalojo de recursos para tratar los interbloqueos, entonces debemos abordar 3 cuestiones:

1. Selección de una víctima

¿De qué recursos hay que apropiarse y de qué procesos?

2. Anulación

Si nos apropiamos de un recurso de un proceso, ¿qué debería hacerse con dicho proceso?

3. Inanición

¿Cómo se puede asegurar que no se produzca la muerte por inanición de un proceso?

8. Conclusiones

- ¿Qué es un estado de interbloqueo?

Se produce cuando dos o más procesos están esperando indefinidamente a que se produzca un suceso que sólo puede producirse como resultado de alguna operación efectuada por otro de los procesos en espera

- 3 métodos principales para tratar los interbloqueos
 - Utilizar algún protocolo de prevención o evasión de los interbloqueos, asegurando que el sistema nunca entrará en un estado de interbloqueo
 - Permitir que el sistema entre en un estado de interbloqueo, detectarlo y luego recuperarse de él
 - Ignorar el problema y actuar como si los interbloqueos nunca fueran a producirse en el sistema