

TEMA 3.- INTERBLOQUEOS

1.- Introducción.....	Pág.02
2.- Recursos.....	Pág.02
3.- Modelo del sistema.....	Pág.03
4.- Caracterización.....	Pág.04
5.- Tratamiento del interbloqueo.....	Pág.05
5.1.- Prevención del interbloqueo.....	Pág.05
5.2.- Evitación del interbloqueo.....	Pág.07
5.3.- Detección de interbloqueo.....	Pág.10
5.4.- Recuperación.....	Pág.12
5.5.- Método Combinado.....	Pág.13

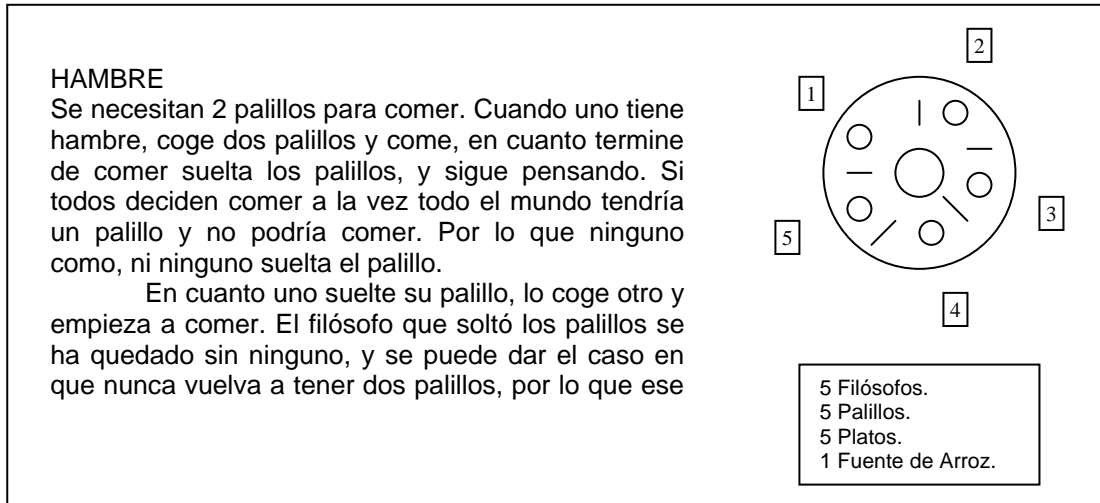
BIBLOGRAFÍA:

[CARRET '01]	Capítulo 6.
[STALLI '01]	Capítulo 6.

1.- INTRODUCCIÓN

Interbloqueo: Un conjunto de procesos han alcanzado una situación de interbloqueo, si cada uno de ellos espera que tenga lugar un suceso que solo puede ser producido por alguno de los procesos de dicho conjunto.

Postergación indefinida (hambre): Se produce si un proceso se ve continuamente adelantado por otros procesos de mayor prioridad o preferencia.



2.- RECURSOS

Recurso: es una entidad que se utiliza para realizar un determinado trabajo en el sistema y que solo puede ser usado por un proceso en un instante dado.

- 1º. En el sistema existen un número finito de recursos del sistema, y los procesos compiten por ellos.
- 2º. Los recursos no son de acceso concurrente

2.1. TIPOS DE RECURSOS:

A) Reutilizables en serie.

Aquellos que son usados por los procesadores de forma secuencial (ej. dispositivos periféricos, estructuras de datos globales y estáticas).

Suele haber un número fijo de recursos de este tipo en el sistema.

Los procesos liberan estos recursos transcurrido un tiempo finito, pero indeterminado.

Tiene dos estados posibles:

- Asignado (a un proceso)
- Libre

B) Consumibles.

Aquellos que una vez utilizados desaparecen (ej. las señales).

Son un número no finito de recursos que varían a lo largo del tiempo.

Pueden existir interbloqueos.

Existirán dos tipos de procesos para estos recursos.

- Procesos productores de recursos.
- Procesos consumibles de recursos.

2.2.- EJEMPLARES DE UN RECURSO:

Cada recurso puede tener varios ejemplares, de modo que:

Ejemplar: Cuando el proceso pide un recurso, y se le puede asignar un ejemplar de este. Si no se le puede asignar, es que el recurso al que quiere acceder el proceso, no es aquel que tiene ejemplares.

2.3.- BLOQUE DE CONTROL DE RECURSOS (RCB)

Es una estructura de datos que permite al sistema operativo controlar los recursos existentes en el sistema.

Para ellos colocamos un nuevo campo en el SCB, justo antes del vector de interrupciones, que contendrá un vector que apuntará al RCB.

1) INFORMACIÓN DEL RCB:

- A. Identificador del recurso.
Número
- B. Tipo del recurso.
- C. Propietario del recurso.
Con un puntero al PCB del proceso
- D. Lista de ejemplares.
Estructura propia para cada ejemplar ICP, donde se guardan los datos específicos para cada ejemplar cada uno tendrá a su vez un propietario).
- E. Gestor del recurso.
Puede denominarse también *driver*, es un programa capaz de manejar el recurso.
- F. Procesos bloqueados.
- G. Estado del recurso.
 - Libre.
 - Ocupado.
- H. Características del recurso.
Es una serie de campos. Entre ellos tenemos:
 - Tipos de acceso.
 - Velocidad de acceso.
 - Códigos de error.
 - Juego de caracteres empleados
 - Etc...

3.- MODELO DEL SISTEMA.

3.1.- PRINCIPIOS DE LOS SISTEMAS INFORMÁTICOS

- A. Número finito de recursos (n).
- B. Número finito de Procesos (m).
- C. Los procesos compiten por los recursos.
- D. Los recursos se dividen en tipos. Donde cada tipo se divide en un número determinado de ejemplares.

3.2.- PASOS PARA QUE UN PROCESO USE UN RECURSO

- PASO 1 – Solicitar el recurso –
Cuando se solicita el recurso puede ser que esté:
 - Ocupado: El proceso se bloquea y pasa también a la lista de procesos bloqueados del RCB del recurso en cuestión.
 - Libre: Se le concede y el proceso sigue con su ejecución.
- PASO 2 – El procesador utiliza el recurso –
El proceso que usa el recurso está bloqueado. Durante el uso del recurso puede solicitar otro.

- PASO 3 – Se libera el recurso –

Recursos Producidos: No está limitada la cantidad de recursos que puede solicitar un proceso, en cambio está limitada la cantidad de recursos en el sistema.

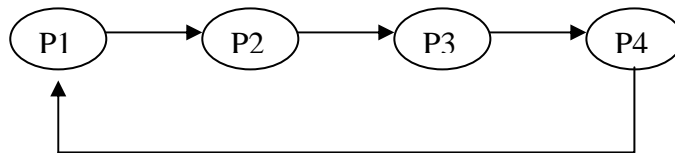
Recursos Consumidos: Son eliminados del sistema. No hay ninguna restricción de cantidad.

4. CARACTERIZACIÓN DEL INTERBLOQUEO.

4.1.- CONDICIONES DEL INTERBLOQUEO.

Las condiciones necesarias pero no suficientes para que se produzca interbloqueo, son:

1. Exclusión mutua (No permite el acceso concurrente a los recursos).
2. Retención y espera (Un proceso puede bloquearse en espera de un recurso, teniendo a su vez en su poder otros recursos).
3. No apropiación (Un proceso no puede usar un recurso que este asignado a otro proceso)
4. Espera circular. Consiste en una serie de procesos, donde cada uno espera que se libere el siguiente recurso de la serie.





P1 espera a R1
 P2 espera a R2
 P3 espera a R3
 P4 espera a R0
 EN esta situación están todos los procesos bloqueados a la espera del recurso. Se solucionará en cuanto algún proceso libere su recurso actual.

4.2.- GRAFO DE ASIGNACIÓN DE RECURSOS.

A. REPRESENTACIÓN

Es la manera de representar la forma en que están asignados los recursos y procesos del sistema. Con este método no representaremos los recursos consumibles.

 : Representa un proceso

R0
 : Representa un recurso. Los círculos interiores representan los ejemplares del recurso.

1. Asignación de recursos a procesos.



2. Petición de recursos por procesos.



B. IMPLEMENTACIÓN

Existen tres formas de implementar la asignación de recursos y procesos.

1. 1ª FORMA – MATRICES –
 Usaremos dos matrices de MxN:

- Matriz de Asignados: Las filas son procesos y las columnas son recursos. En cada celda estará el número de ejemplares del recurso que cada proceso tiene asignado.
- Matriz de Solicitados: Las filas son procesos y las columnas son recursos. En cada celda estará el número de ejemplares que el proceso ha solicitado y no se le han concedido aún.

2. 2ª FORMA – LISTAS ENLAZADAS –

Se necesitan 4 listas enlazadas, de las cuales:

- Cada proceso tendrá dos, las cuales se encuentran en el PCB:
 - Lista de recursos asignados.
 - Lista de recursos solicitados.
- Cada recurso tendrá dos:
 - Lista de procesos que usa el recurso (en uso).
 - Lista de procesos que están en espera de algún ejemplar del recurso.

5.- TRATAMIENTO DE INTERBLOQUEO.

Las formas de tratar los interbloqueos se dividen en 4 grandes técnicas.

. 1ª TÉCNICA - TÉCNICA DEL AVESTRUZ –

Se basa en que: Si no se ve el problema, es que no existe. (se omite).

Para omitir el interbloqueo, debemos tener en cuenta estos factores:

1. Número de veces que ocurre el interbloqueo.
2. Gravedad de la ocurrencia del interbloqueo.
3. Coste de las consecuencias.
4. Coste, en caso de hacer algo.

Linux usa esta técnica.

. 2ª TÉCNICA – PREVENCIÓN –

Consiste en eliminar del sistema al menos, una de las condiciones de interbloqueo.

Ventajas:

- Método limpio, no tiene ni que matar procesos, ni que expropiar recursos.

Desventajas:

- Se aprovechan mal los recursos.

. 3ª TÉCNICA – EVITACIÓN –

Consiste en permitir la posibilidad de interbloqueo. De modo que se chequea el sistema continuamente. Este chequeo se realiza en las asignaciones de recursos a procesos, ya que las solicitudes no son evitables.

Ventajas:

- Método limpio.

Desventajas:

- Poco rendimiento, sobrecarga mucho el sistema.
- Chequear continuamente el sistema.

. 4ª TÉCNICA – DETECCIÓN Y RECUPERACIÓN –

Consiste en permitir a los procesos actuar libremente, incluso permitir que ocurra el interbloqueo. Cada cierto tiempo se realiza un chequeo para comprobar si se ha producido un interbloqueo. En el caso de que lo haya se lanza un mecanismo de recuperación.

Desventajas:

- Método no limpio.

5.1.- PREVENCIÓN DEL INTERBLOQUEO

A) SUPRESIÓN DE LA EXCLUSIÓN MUTUA.

Funcionamiento: Hacemos que se pueda acceder a los recursos de forma paralela. Como, por ejemplo, volcar al disco unos datos como si se hubiesen impreso (spool).

Inconveniente: No soluciona el problema realmente.

B) SUPRESIÓN DE RETORNO Y ESPERA.

Funcionamiento: El retorno y espera se da cuando un proceso espera un recurso, cuando ya tiene asignados otros. Esto ocurre debido a la petición incrementable. Las soluciones son:

1. Hacer que todos los recursos sean pedidos al principio.
 - Si están todos disponibles, se conceden todos.
 - Si no lo están, de volverá a intentar pasado un rato.
 - Inconvenientes:
 - Saber la necesidad que tienen los procesos desde el principio.
2. Pedir los recursos de forma incrementable.
 - Si se encuentra un recurso ocupado, liberamos todos los que tenemos.
 - Inconvenientes:
 - Pueden quedar estados inconsistentes.
 - El grado de multiprogramación.

Solución: que los procesos sean muy pequeños, aunque aun así no siempre es posible.

C) SUPRESIÓN DE LA NO APROPIACIÓN.

Funcionamiento: Hacemos que los procesos puedan quitarse los recursos los unos a los otros.

1. Si el recurso está libre, se le concede al proceso.
 - Inconvenientes:
 - Los estados de los recursos pueden resultar inconsistentes.
2. Si el recurso no está libre, podemos actuar de dos formas:
 - Por proceso activo, deja que otro proceso lo adelante.
 - Por proceso bloqueado, se expropia.
 - Inconvenientes:
 - Los procesos pueden llegar a la postergación indefinida (hambre).

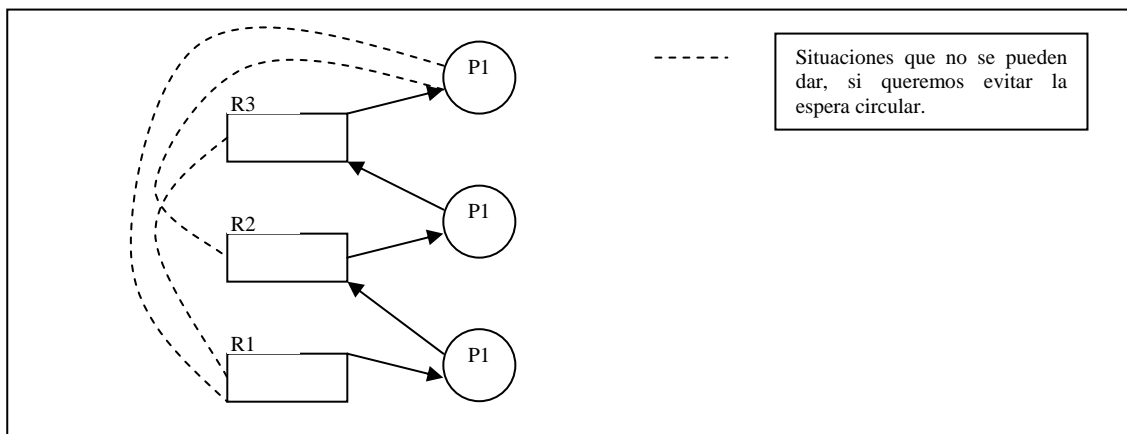
D) SUPRESIÓN DE LA ESPERA CIRCULAR

Funcionamiento: Se intenta evitar que aparezca la espera circular en un grupo de procesos. Para ello imponemos una ordenación lineal de los recursos:

Ordenación lineal: A cada recurso se le asigna un valor.

Después se hace una asignación jerárquica de recursos. Esto consiste en la concesión de los recursos solo en orden numérico creciente.

Un proceso de número J, solo podrá solicitar un recurso I, tal que $J > I$



Inconvenientes:

- Es imposible que satisfaga a todos los procesos.
- Es muy difícil encontrar la ordenación lineal.

5.2. EVITACIÓN DEL INTERBLOQUEO

Funcionamiento: Cuando todas las condiciones necesarias de interbloqueo están presentes, menos la espera circular, seguimos lo siguiente:

- Cada vez que un proceso solicite un recurso se evalúa el sistema.
- En el supuesto de que se le concediera el recurso, apareciera una espera circular, no se le concedería el recurso.
- En el caso de que no apareciera espera circular, se le concedería el recurso.

Inconvenientes:

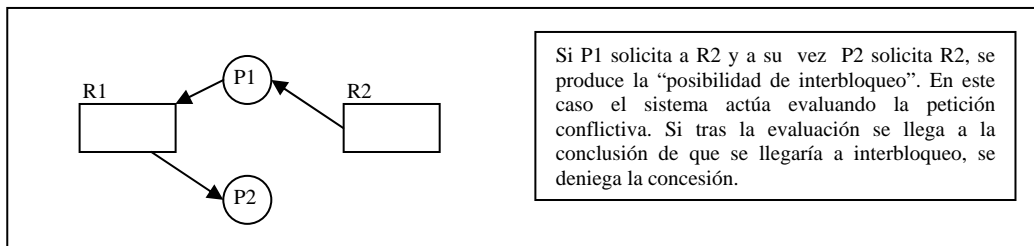
- Necesitamos saber de antemano, la información de los recursos que se van a solicitar. Esto es casi imposible.

OPCIONES DE EVITACIÓN.

A) DIAGRAMA DE EJECUCIÓN DE PROCESOS.

Su utilidad es puramente teórica, ya que explica el "punto de no retorno" y nunca se lleva a la práctica ya que solo es aplicable a dos procesos y dos recursos.

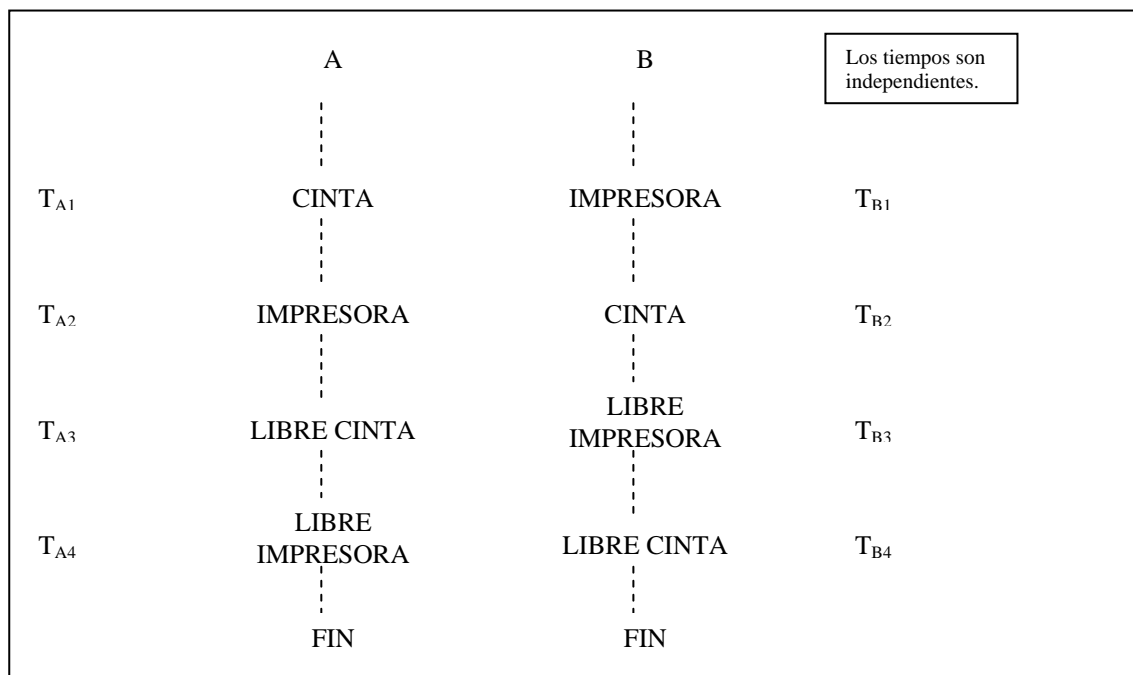
Situación de interbloqueo:

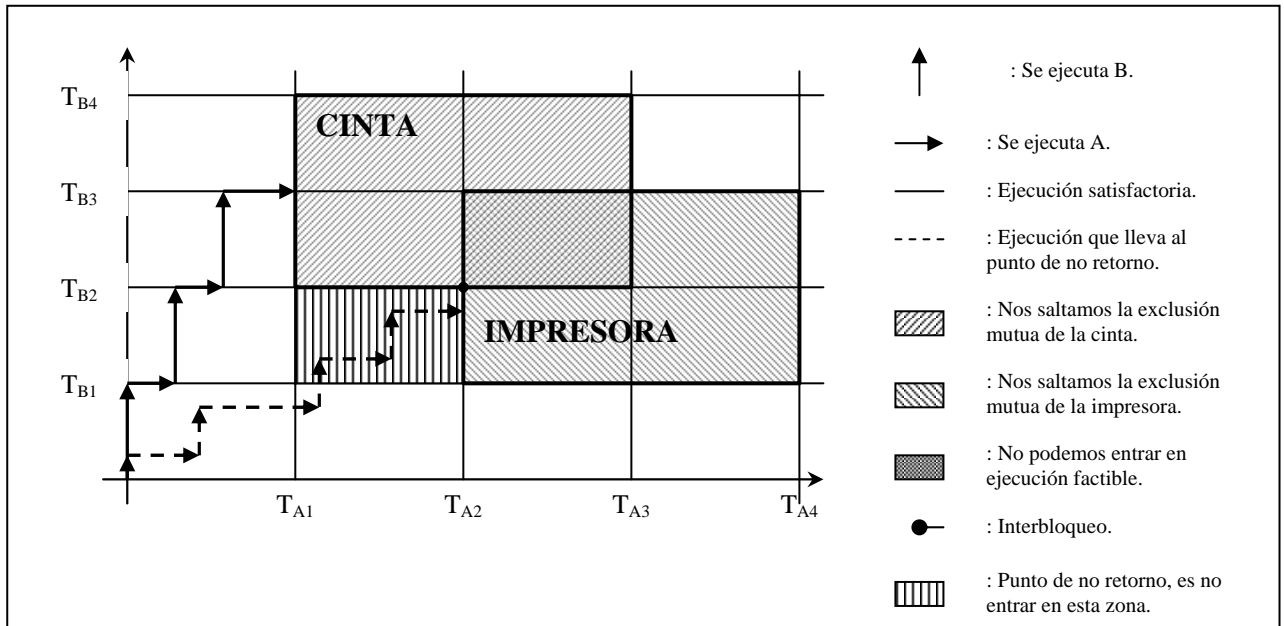


Procesos: A, B

Recursos: Impresora, Cinta.

- El proceso A pide la cinta y luego pide la impresora, finalizando con esta última.
- El proceso B pide la impresora, y luego la cinta, finalizando con esta última.





Para evitar el interbloqueo, solo deberemos evitar el punto de no retorno. De modo que cuando pasemos por T_{A1} o T_{B1} no concederemos los recursos, si se van a sobrepasar. Todo esto lo hace un algoritmo de evitación, uno de ellos es el algoritmo del banquero.

B) EL ALGORITMO DEL BANQUERO.

Funcionamiento:

- 5 procesos y 3 recursos (X, Y, Z):
 - De X habrá 6 ejemplares, X(6).
 - De Y habrá 3 ejemplares, Y(3).
 - De Z habrá 4 ejemplares, Z(4).

MATRIZ DE RECURSOS ASIGNADOS

MATRIZ DE NECESIDADES MÁXIMAS

	X	Y	Z	X	Y	Z
A	3	0	1	4	1	1
B	0	1	0	0	2	1
C	1	1	1	4	2	1
D	1	1	0	1	1	1
E	0	0	0	2	1	1

Calculamos 3 vectores:

- E: Recursos existentes en el sistema.
E = (6, 3, 4)
- A: Recursos asignados (sumar las columnas de la matriz de recursos asignados).
A = (5, 3, 2)
- L: Recursos que no se han concedido aún (libres).
L = (1, 0, 2)

PASO 1

Calculamos la matriz de necesidades pendientes.
Buscamos una fila cuyas necesidades pendientes se pueda satisfacer con el vector L.
Si no encontramos la fila, estaremos en un 'ESTADO INSEGURO'

	MATRIZ DE RECURSOS ASIGNADOS			MATRIZ DE NECESIDADES MÁXIMAS			MATRIZ DE NECESIDADES PENDIENTES		
	X	Y	Z	X	Y	Z	X	Y	Z
A	3	0	1	4	1	1	1	1	0
B	0	1	0	0	2	1	0	1	1
C	1	1	1	4	2	1	3	1	0
D	1	1	0	1	1	1	0	0	1
E	0	0	0	2	1	1	2	1	1

PASO 2

Si se encuentra la fila (la fila D lo cumple). Le concedemos todas las necesidades pendientes a dicho proceso, por lo que éste finaliza y devuelve todos los recursos que tuviese.

PASO 3

Volvemos al paso 1 y si terminamos todos los procesos, podemos determinar que estamos en un 'ESTADO SEGURO'.

Este algoritmo se aplica siempre que se un proceso pida un recurso. Cuando termine el algoritmo nos podemos encontrar en dos situaciones:

- Es seguro, entonces se le concede el recurso.
- No es seguro, se le deniega el recurso.

	MATRIZ DE RECURSOS ASIGNADOS			MATRIZ DE NECESIDADES MÁXIMAS			MATRIZ DE NECESIDADES PENDIENTES		
	X	Y	Z	X	Y	Z	X	Y	Z
A	3	0	1	4	1	1	1	1	0
B	0	1	0	0	2	1	0	1	1
C	1	1	1	4	2	1	3	1	0
D	1	1	0	1	1	1	0	0	0
E	0	0	0	2	1	1	2	1	1

→ Termina

Una vez finalizado "D" → L = (2, 1, 2)

Asignamos a "A" → L = (1, 0, 2)

	MATRIZ DE RECURSOS ASIGNADOS			MATRIZ DE NECESIDADES MÁXIMAS			MATRIZ DE NECESIDADES PENDIENTES		
	X	Y	Z	X	Y	Z	X	Y	Z
A	3	0	1	4	1	1	0	0	0
B	0	1	0	0	2	1	0	1	1
C	1	1	1	4	2	1	3	1	0
D	1	1	0	1	1	1	0	0	0
E	0	0	0	2	1	1	2	1	1

→ Termina

Una vez finalizado "A" → L = (5, 1, 3)

Asignamos a "B" → L = (5, 0, 2)

	MATRIZ DE RECURSOS ASIGNADOS			MATRIZ DE NECESIDADES MÁXIMAS			MATRIZ DE NECESIDADES PENDIENTES		
	X	Y	Z	X	Y	Z	X	Y	Z
A	3	0	1	4	1	1	0	0	0
B	0	1	0	0	2	1	0	0	0
C	1	1	1	4	2	1	3	1	0
D	1	1	0	1	1	1	0	0	0
E	0	0	0	2	1	1	2	1	1

→ Termina

Una vez que finaliza "B" → L = (5, 2, 3)
 Asignamos a "C" → L = (2, 1, 3)

	MATRIZ DE RECURSOS ASIGNADOS			MATRIZ DE NECESIDADES MÁXIMAS			MATRIZ DE NECESIDADES PENDIENTES		
	X	Y	Z	X	Y	Z	X	Y	Z
A	3	0	1	4	1	1	0	0	0
B	0	1	0	0	2	1	0	0	0
C	1	1	1	4	2	1	0	0	0
D	1	1	0	1	1	1	0	0	0
E	0	0	0	2	1	1	2	1	1

→ Termina

Una vez finalizado "C" → L = (6, 3, 4) (ahora tenemos todos los recursos libres)
 Asignamos a "E" → L = (4, 2, 3) (como es le único que queda, los que usa, vuelven, y L no se modifica).

	MATRIZ DE RECURSOS ASIGNADOS			MATRIZ DE NECESIDADES MÁXIMAS			MATRIZ DE NECESIDADES PENDIENTES		
	X	Y	Z	X	Y	Z	X	Y	Z
A	3	0	1	4	1	1	0	0	0
B	0	1	0	0	2	1	0	0	0
C	1	1	1	4	2	1	0	0	0
D	1	1	0	1	1	1	0	0	0
E	0	0	0	2	1	1	0	0	0

→ Termina

L = (6, 3, 4) COMO HEMOS LOGRADO TERMINAR CONCLUIMOS QUE ESTAMOS EN UN "ESTADO CONCLUSO"

Inconvenientes:

- La cantidad de recursos debe ser fija.
- El número de procesos debe ser fijo.
- Necesitamos saber de antemano las necesidades de los procesos.

5.3. DETECCIÓN DE INTERBLOQUEO.

· Funcionamiento: si el sistema cumple las condiciones para poderse bloquear, y comprobamos que se ha producido interbloqueo, pondremos en marcha un algoritmo de recuperación. Estos algoritmos son costosos porque consumen mucho tiempo de la CPU.

Técnicas de detección de interbloqueos: (la usaremos cuando nos pregunten si existe interbloqueo)

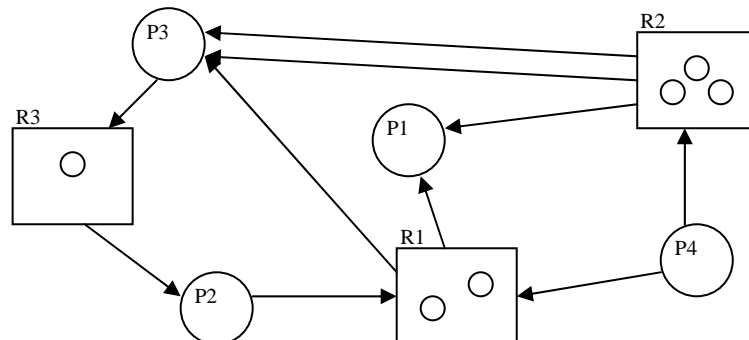
➤ **REDUCCIÓN DEL GRAFO DE ASIGNACIÓN DE RECURSOS.**

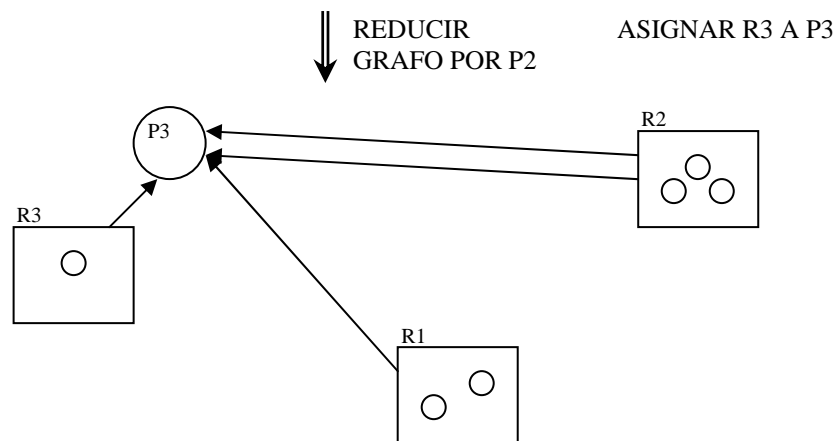
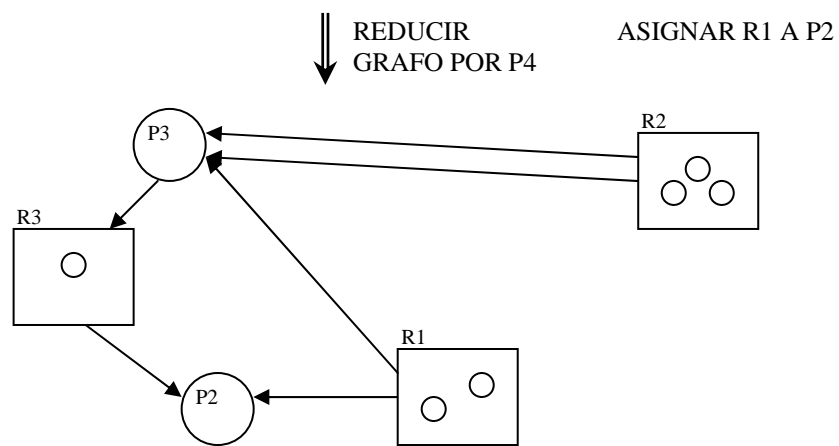
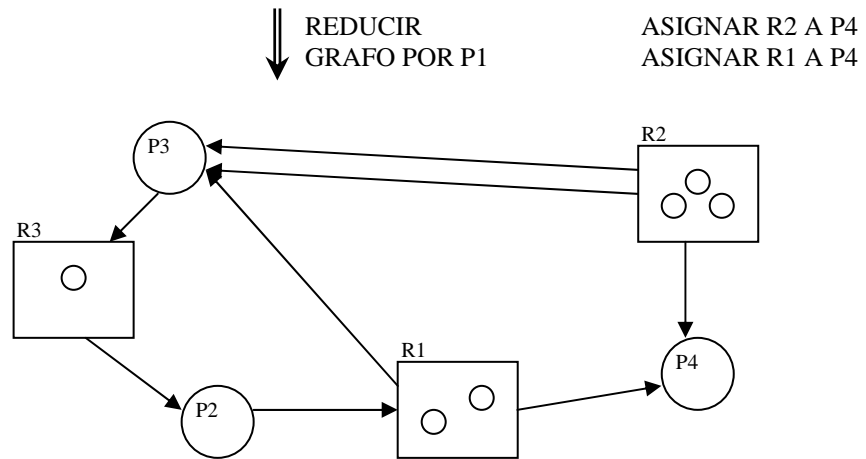
Intentamos ir terminando los procesos hasta que se terminen todos.

1. Si todos pueden ser terminados, concluimos que "NO HAY INTERBLOQUEO".
2. Si no se pueden terminar todos los procesos, concluimos que los que no se han podido terminar.

Finalizar procesos: Buscamos un proceso que tenga todos los recursos que necesita.
 Reducir: Implica la finalización del proceso y la liberación de todos sus recursos.

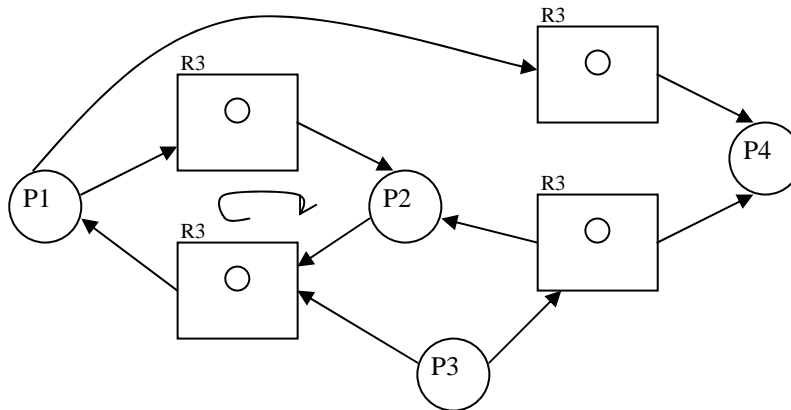
➤ **Ejemplo de una situación sin interbloqueo.**



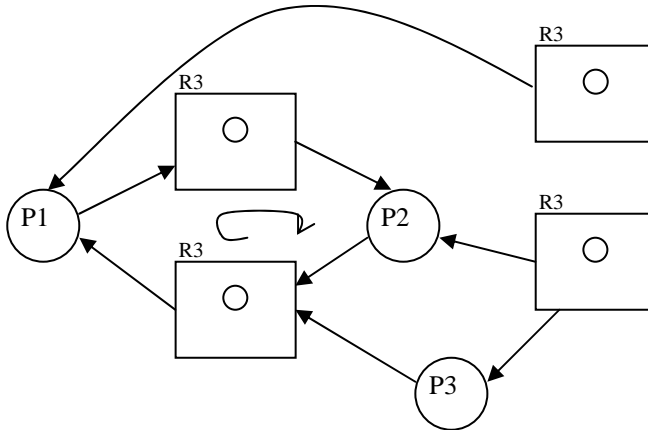


YA NO SE PUEDE PRODUCIR INTERBLOQUEO. FIN

➤ **Ejemplo de una situación donde con interbloqueo.**



REDUCIR
GRAFO POR P4



NO SE PUEDE REDUCIR MÁS, POR LO QUE EXISTE INTERBLOQUEO

➤ **Características de algoritmo de detección:**

Momentos de detección:

1. A intervalos regulares: dependiendo de la frecuencia del interbloqueo, se establece un intervalo de detección, y dependiendo a su vez de este intervalo, se establece la gravedad del interbloqueo.
2. Cuando se puede producir el interbloqueo: Es decir, cuando hay espera circular o cuando no se le concede un recurso a un proceso. Tiene un coste muy grande.
3. Cuando el rendimiento del sistema baja de un determinado umbral. Sobrecarga la memoria.

5.4 ALGORITMOS DE RECUPERACIÓN DE INTERBLOQUEO.

Ante un interbloqueo podemos actuar de dos formas:

- Dar un mensaje y esperar que el usuario solucione el problema.
- Arreglar el interbloqueo. Para arreglar el interbloqueo tenemos dos opciones:

A. **TERMINACIÓN DE PROCESOS.**

· Las diferentes formas de terminar procesos, son:

1. Eliminar todos los procesos interbloqueados (tiene una eficiencia del 100%)

Inconvenientes:

- Se pierde mucho trabajo, ya que los procesos eliminados, se tendrán que volver a lanzar.
 - Pueden aparecer estados inconsistentes.
2. Abortar solo un proceso.

Ventajas:

- Se pierde menos trabajo.

Inconvenientes:

- No estamos seguros de haber solucionado el interbloqueo.

Inconvenientes:

- Pueden aparecer inconsistencias en los procesos.
- Puede producir hambre.

· Factores para matar un proceso:

1. Prioridad, se matará al menos prioritario.
2. Tiempo de procesador, se matará al que menos tiempo de procesador tenga.
3. Tipo y número de recursos usados.
4. Recursos adicionales, se elige al que más vaya a perder.
5. Facilidad de relanzar el proceso.

B. EXPROPIACIÓN DE RECURSOS.

Los factores a tener en cuenta a la hora de expropiar un recurso, son:

1. Seleccionar el proceso a expropiar. La expropiación se hará según los factores para matar a un proceso.
2. Los puntos de recuperación. Una vez seleccionado el proceso, podemos llevarlo a un estado consistente anterior, para ellos iremos marcando unos puntos de recuperación.

Punto de recuperación: Instantes consistentes en los que se guarda el estado del proceso.

3. Evitar la postergación indefinida. Para evitarlo debemos controlar el número de veces que se expropia un mismo proceso. Ya que si produce un interbloqueo y expropiamos el menos prioritario, se volverá a expropiar el mismo una y otra vez.

5.5 MÉTODO COMBINADO.

Se basa en una combinación de los métodos vistos anteriormente.

Seleccionamos los recursos por grupos atendiendo a sus características.

Usamos la técnica que mejor se adecue al grupo del que se trata, de esta forma conseguimos un sistema que no elimina los interbloqueos, ya que puede haber problemas de recursos entre grupos.

Solucionaremos esto:

- poniendo una ordenación lineal de los recursos por grupo.
- Asignando jerárquicamente los recursos por grupos.

