

Tema 5

Interbloqueos

Índice

- Concepto Interbloqueo
- Gráfo de asignación de recursos.
- Condiciones de Coffman.
- Técnicas para el tratamiento del interbloqueo
 - Prevención.
 - Evitación.
 - Detección y recuperación.
- Estrategias integradas.

Concepto interbloqueo

- Tenemos:
 - Un conjunto de procesos ejecutándose en un sistema.
 - Un conjunto de recursos finitos que son utilizados por dichos procesos.
- Entonces, se dice que un conjunto de procesos se encuentran en un estado de **interbloqueo** cuando **todos** los procesos se encuentran **esperando** por un recurso que está retenido por otro proceso del grupo.
- En esta situación:
 - Ningún proceso puede continuar su ejecución.
 - Ningún otro proceso podrá obtener los recursos retenidos, puesto que no pueden ser liberados.

Ejemplo interbloqueo (I)

- Un sistema dispone de dos grabadoras de DVDs y dos procesos P_1 y P_2 :
 - P_1 lee de la primera grabadora y escribe en la segunda.
 - P_2 lee de la segunda grabadora y escribe en la primera.
- Semáforos ($A=1$ y $B=1$)

Proceso 1

```
.....  
P(A);  
....  
P(B);  
.....  
V(A);  
V(B);
```

Proceso 2

```
.....  
P(B);  
....  
P(A);  
.....  
V(A);  
V(B);
```

Ejemplo interbloqueo (II)



Recursos

- Los recursos pueden ser de dos tipos:
 - Apropiable: Se puede retirar el recurso a un proceso sin causarles ningún problema (cpu, memoria virtual).
 - No apropiable: No se puede retirar el recurso una vez asignado (impresora, grabadoras cds, etc...).
- Protocolo de utilización de los recursos por parte de los procesos:
 - Petición del recurso.

Si el recurso no está disponible el proceso queda en espera hasta que el recurso esté disponible.
 - Utilización del recurso.
 - Liberación del recurso.

Grafo de asignación de recursos

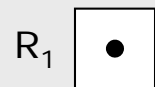
- Conjunto procesos, identificados por P_1, P_2, \dots, P_n



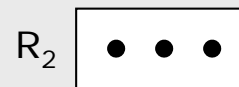
- Conjunto recursos, identificados por R_1, R_2, \dots, R_n



- De cada recurso pueden existir 1 ó más instancias.



Recurso con una instancia



Recurso con tres instancias

Grafo de asignación de recursos

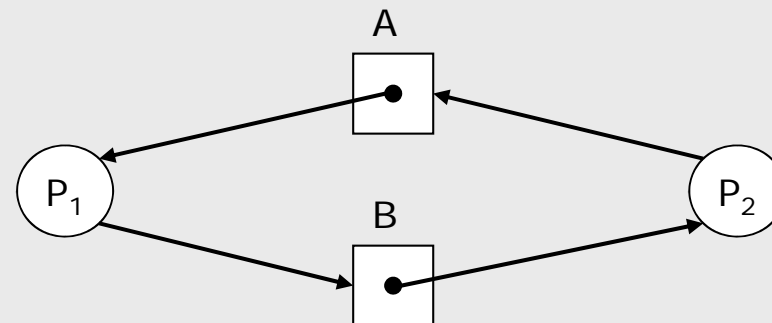
- **La solicitud** de un recurso por parte de un proceso se representa mediante un arco dirigido desde el proceso al recurso:



- La asignación de un recurso a un proceso se marca mediante una flecha desde la instancia del recurso asignado hasta el proceso:



- Ejemplo interbloqueo con semáforos:



Condiciones Coffman

- Para que se produzca un interbloqueo las siguientes cuatro condiciones son necesarias (aunque no suficientes):
 - Exclusión mutua.
Solo un proceso puede utilizar el recurso simultáneamente.
 - Retención y Espera.
El proceso puede retener los recursos asignados mientras espera que le asignen otros.
 - No apropiación.
Ningún proceso puede ser forzado a abandonar un recurso que retiene.
 - Espera circular.
Existe una cadena cerrada de procesos, cada uno de los cuales retiene, al menos, un recurso que necesita el siguiente proceso de la cadena.

Ejemplo Condiciones Coffman



Técnicas tratamiento del interbloqueo

- Ignorar el problema.
- Prevenir el interbloqueo.
- Evitar el interbloqueo.
- Detección y recuperación del interbloqueo.
- Estrategias Integradas.

Técnicas tratamiento del interbloqueo

- Ignorar el problema.
- Prevenir el interbloqueo.
- Evitar el interbloqueo.
- Detección y recuperación del interbloqueo.
- Estrategias Integradas.

Ignorar el problema

- **Método:**

Consiste en no hacer nada para evitar ó corregir las situaciones de interbloqueo.

- **Justificación:**

La probabilidad de que se produzca un interbloqueo es tan baja que no compensa el esfuerzo y los recursos necesarios para su solución.

Este planteamiento no es válido para sistemas que sean críticos (centrales nucleares, sistemas bancarios, ...).

- Unix ha optado por esta política.

Técnicas tratamiento del interbloqueo

- Ignorar el problema.
- Prevenir el interbloqueo.
- Evitar el interbloqueo.
- Detección y recuperación del interbloqueo.
- Estrategias Integradas.

Prevención

- **Objetivo:**

Diseñar un sistema de forma que quede excluida la posibilidad del interbloqueo.

- **Método:**

Lograr que alguna de las 4 condiciones de Coffman no se puedan cumplir nunca.

- **Métodos indirectos de prevención:**

- Exclusión mutua.
- Retención y espera.
- No apropiación.

- **Métodos directos:**

- Impedir la aparición de la espera circular

Prevenir: Exclusión Mutua

- Trata de evitar el acceso a los recursos en modo exclusivo (compartir los recursos).

Ejemplo: Ficheros modo lectura, impresoras.

- No se puede implementar debido a que existen recursos que son intrínsecamente no compartibles

Ejemplo:

- Ficheros modo escritura.
- Regiones críticas.
- Grabadores CDs.

Prevenir: Retención y espera

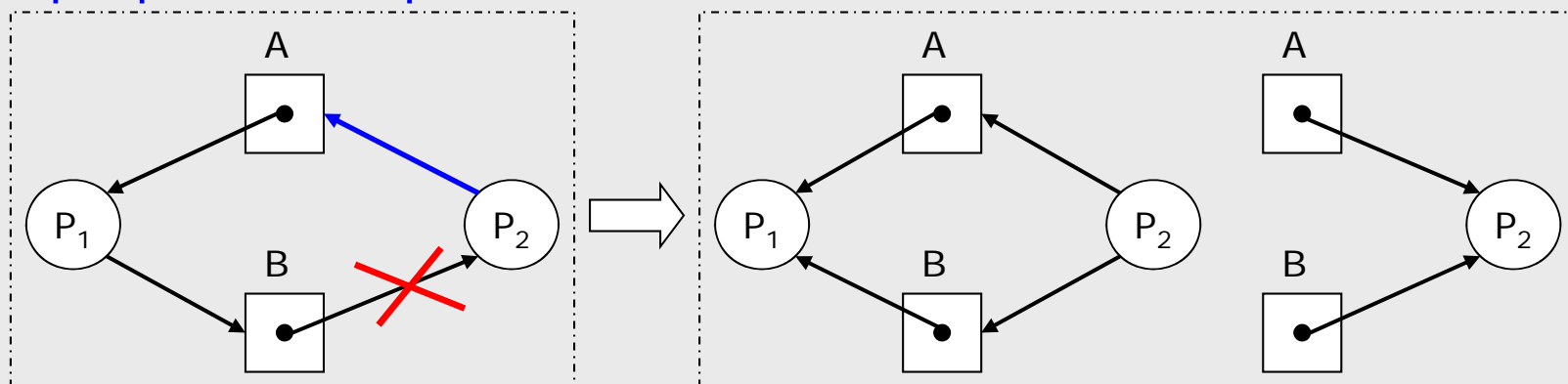
- Para evitar esta condición existen dos posibilidades:
 - Obligar a los procesos a pedir todos los recursos de una vez. Se le conceden todos los que necesita ó ninguno.
 - Utilizar los recursos de uno en uno.
- Inconvenientes:
 - **Baja utilización** de los recursos.
 - **Inanición** de los procesos que necesiten muchos recursos.
 - Se debe conocer las necesidades futuras de recursos.
 - Ejecución ineficiente de los procesos.

Prevenir: No Apropiación

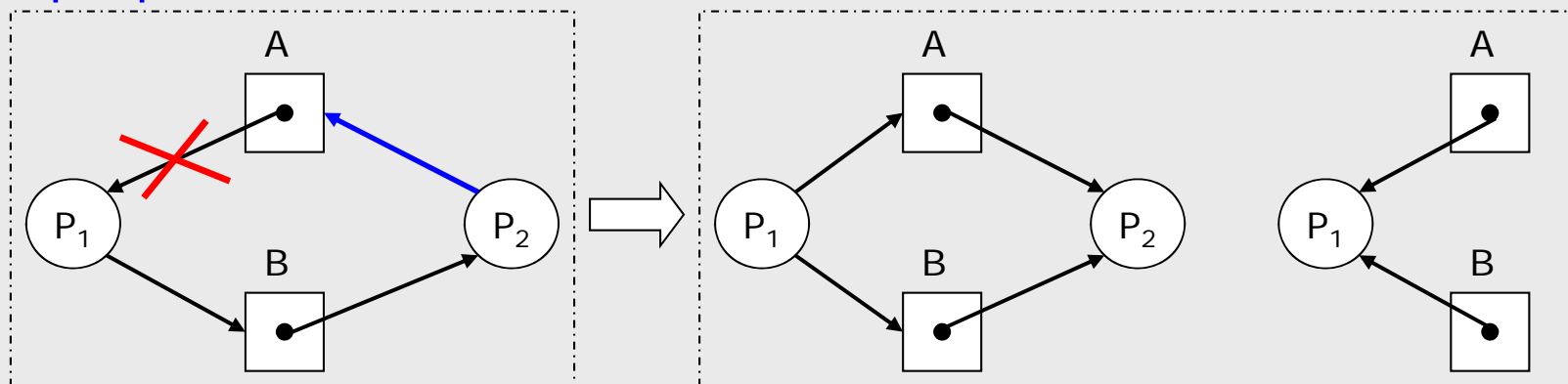
- Se permite la **apropiación** de los recursos.
- Dos alternativas:
 - Apropiación implícita.
 - Apropiación a terceros.
- Esta técnica sólo es práctica cuando se aplica a recursos cuyo estado puede salvarse y restaurarse más tarde (por ejemplo la CPU).

Prevenir: No Apropiación

Apropiación implícita



Apropiación a terceros



Prevenir: Espera circular

- El método directo de prevención consiste en impedir la aparición de la espera circular.
- Se define una ordenación de los tipos de recursos del sistema y se obliga a que los procesos pidan los recursos siguiendo este orden estricto.
 - Un proceso que tenga asignados recursos del tipo i solo podrá realizar peticiones a recursos cuyo tipo sea mayor que i .

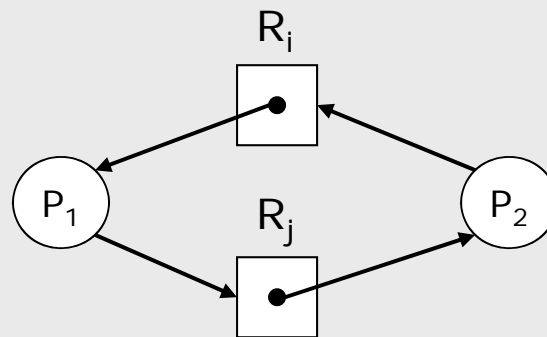
$$\text{Tipo}(r_i) < \text{Tipo}(r_{i+1})$$

- Si un proceso requiera más de una instancia de un recurso deberá pedir las todas al mismo tiempo mediante una única solicitud.

Demostración

■ Ejemplo:

- P_1 ha adquirido recurso R_i y pide el recurso R_j .
- P_2 ha adquirido recurso R_j y pide el recurso R_i .

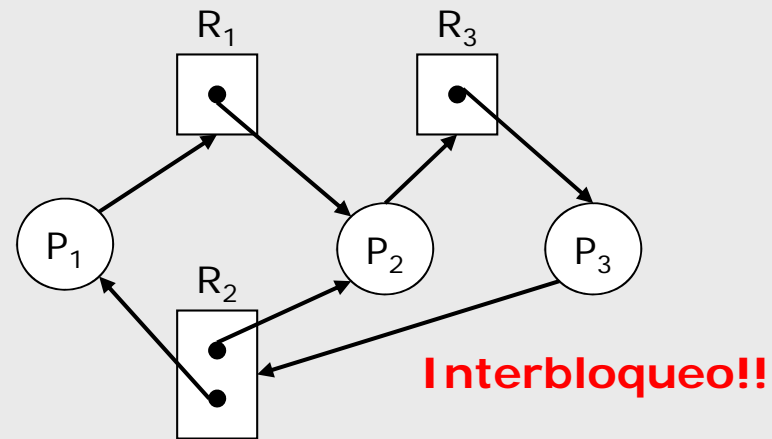


- Si los procesos han realizado las peticiones siguiendo un orden creciente, el grafo de asignación de recursos anterior sólo se puede producir si se cumple al mismo tiempo:
 - $i < j$ (Para que P_1 pueda pedir su segundo recurso)
 - $j < i$ (Para que P_2 pueda pedir su segundo recurso)

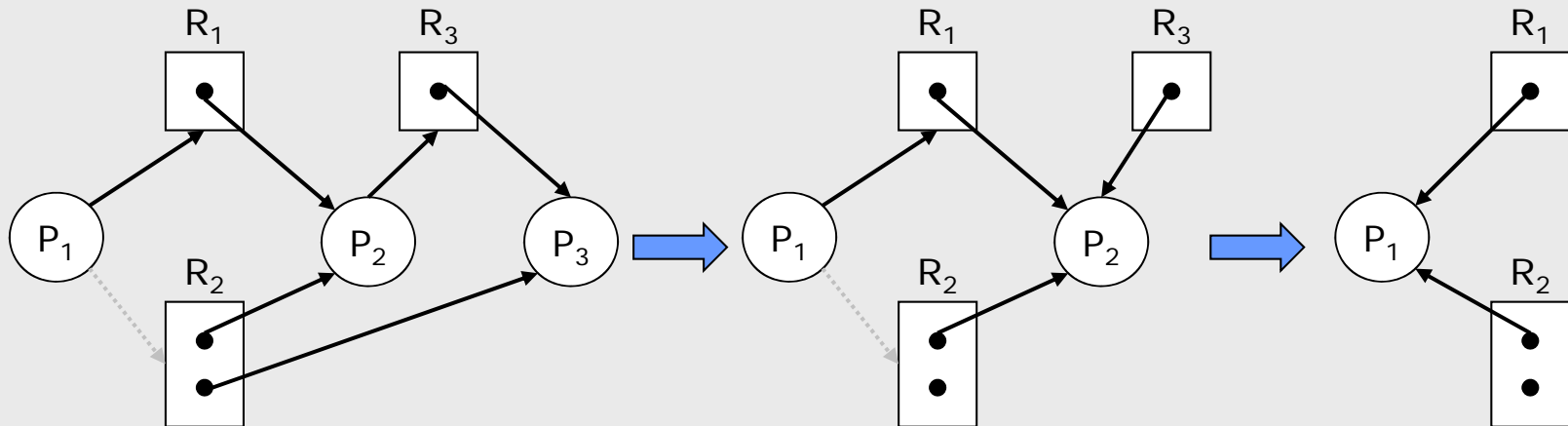
} **Imposible!!**

Ejemplo prevención espera circular

- Grafo de asignación con interbloqueo:



- Solución mediante la prevención de la espera circular:



Prevenir: Espera circular

- Ventajas:

- Aplicación a nivel de la compilación.
- No se requiere procesamiento durante la ejecución.

- Desventajas:

- Poco uso de la apropiación.
- No permite solicitudes incrementales de recursos.

Técnicas tratamiento del interbloqueo

- Ignorar el problema.
- Prevenir el interbloqueo.
- Evitar el interbloqueo.
- Detección y recuperación del interbloqueo.
- Estrategias Integradas.

Evitar el interbloqueo

■ **Objetivo:**

Un método menos restrictivo y más eficiente respecto a la utilización de los recursos para tratar el interbloqueo.

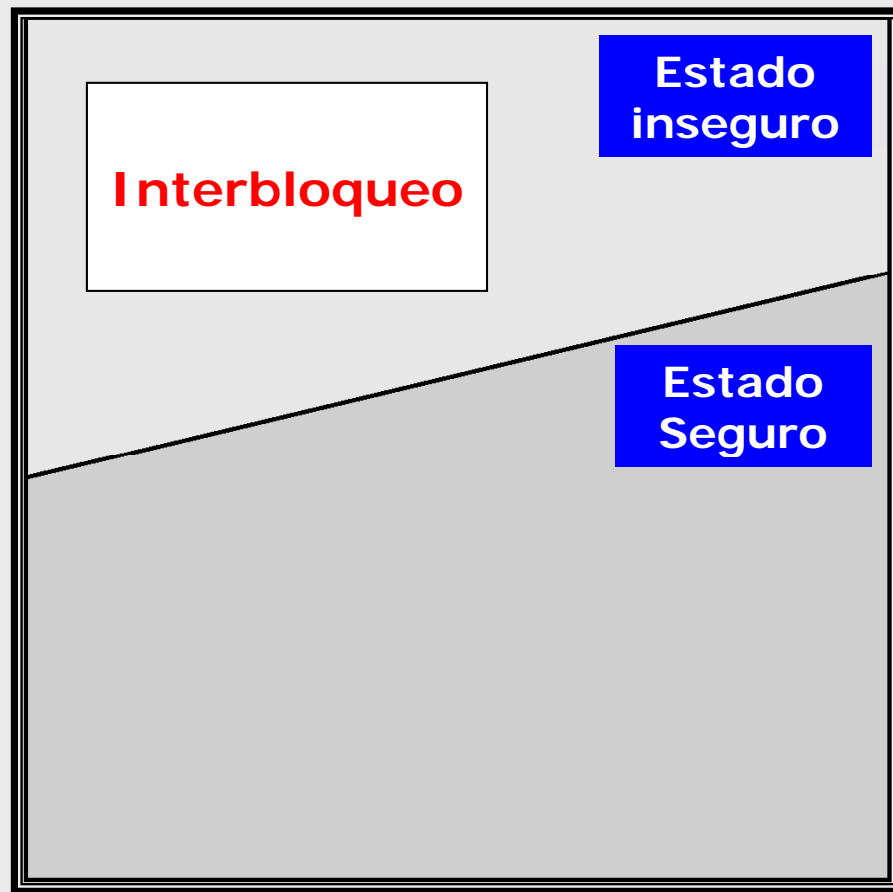
■ **Método:**

- Se basa en pedir información adicional a los procesos sobre como van a solicitarse los recursos:
 - La necesidad máxima de recursos de los procesos en ejecución.
 - La asignación actual de recursos a los procesos.
 - La cantidad actual de instancias disponibles de cada recurso.
- A partir de esta información, el SO sólo concederá un recurso a un proceso cuando su asignación no implique una posibilidad de interbloqueo (estado seguro).

Determinación estado seguro

- Un **estado seguro** es un estado en el cual al menos existe un orden de asignación de los recursos a los procesos (**secuencia segura**) que garantiza que todos los procesos se pueden ejecutar hasta el final sin interbloqueo.
- Una **secuencia segura** es una cierta ordenación de los procesos que garantiza que los recursos que pueda pedir cualquier proceso P_i se puedan satisfacer con los recursos disponible más los recursos retenidos por los procesos P_j ($j < i$).
- Un **estado inseguro** es un estado en el cual no existe un orden de asignación que permita garantizar que los procesos no se interbloqueen.

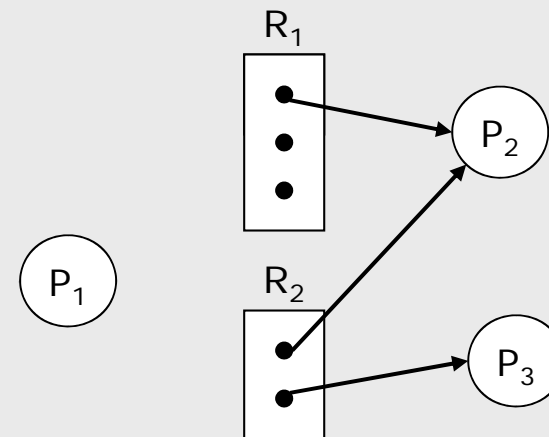
Relación entre los estados y el interbloqueo



Ejemplo estado seguro

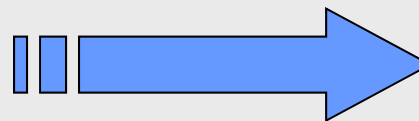
Tabla de necesidades
procesos

	R ₁	R ₂
P ₁	1	2
P ₂	2	0
P ₃	2	0



■ Secuencias seguras:

- P₂, P₃ y P₁
- P₃, P₂ y p₁

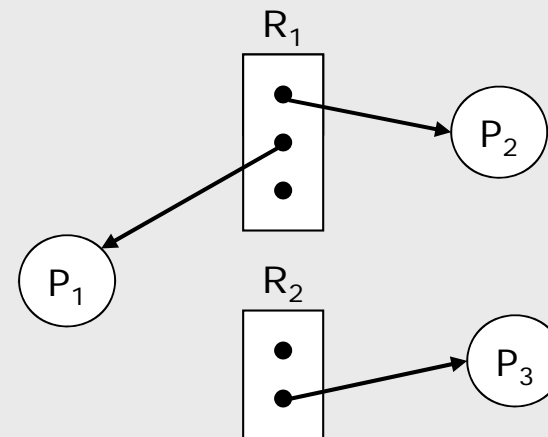


**Estado
seguro**

Ejemplo estado inseguro

Tabla de necesidades
procesos

	R ₁	R ₂
P ₁	0	2
P ₂	2	1
P ₃	2	0



**Estado
inseguro!!**

- Con los recursos disponibles y en función de las necesidades de cada proceso, **no existe ninguna secuencia segura** de asignación que permita evitar el interbloqueo de los procesos.
- La instancia del recurso R₂ no se debería haber asignado al proceso P₃ sino a P₁ ó P₂.

Algoritmo del banquero

- Mediante este algoritmo, el SO decide como se tienen que asignar los recursos a los procesos.
- Características:
 - Funciona con recursos que disponen de **múltiples instancias**.
 - Los procesos deben **anunciar** al arrancarse sus **necesidades** respecto a los recursos del sistema.
 - Cuando un proceso realiza una petición de recursos puede que tenga que esperar, aunque los recursos estén disponibles.
 - Los procesos que obtienen todos los recurso que necesitan los liberarán en un tiempo finito.

Estructuras algoritmo del banquero

- Se necesitan varias estructuras de datos¹:
 - Disponible[m]
Cantidad de instancias actualmente disponibles de cada recurso.
 - Máximo[n,m]
Requisitos máximo para cada proceso (de cada uno de los recursos).
 - Asignado[n,m]
Recursos asignados a cada uno de los procesos.
 - Necesito[n,m]
Instancia de recursos que todavía necesita pedir cada uno de los procesos (para cada recurso).

¹ Asumimos m recurso y n procesos

Métodos algoritmo banquero

- El algoritmo del banquero se descompone en dos funciones:

- **Algoritmo de seguridad.**

Este algoritmo permite averiguar si un sistema se encuentra en estado seguro ó no.

- **Algoritmo de petición de recursos.**

Se utiliza el SO para averiguar si puede satisfacer una determinada petición de recursos.

Algoritmo de seguridad

■ Estructuras auxiliares:

- Trabajo [m]: Acumula los recursos de los procesos que pueden finalizar.
- Acabado[n]: Booleano que indica cuando un proceso ha finalizado.

Función Seguridad **devuelve** Boolean

Trabajo = Disponible

Para todo i

Acabado[n] = Falso

FinPara

Mientras $\exists i$ tal que Acabado[i] == Falso Y Necesito [i] <= Trabajo

Trabajo = Trabajo + Asignado[i]

Acabado[i] = Cierto

FinMientras

Si $\forall i$ Acabado[i] = Cierto **entonces**

return (Cierto)

Sino

return(Falso)

FinSi

FinFunción

Ejemplo algoritmo seguridad

■ ¿Estado seguro?

Disponible

R ₁	R ₂	R ₃
3	3	2

Máximo

	R ₁	R ₂	R ₃
P ₁	7	5	3
P ₂	3	2	2
P ₃	9	0	2
P ₄	2	2	2
P ₅	4	3	3

Necesito

	R ₁	R ₂	R ₃
P ₁	7	4	3
P ₂	0	2	0
P ₃	6	0	0
P ₄	0	1	1
P ₅	4	3	1

Asignado

	R ₁	R ₂	R ₃
P ₁	0	1	0
P ₂	3	0	2
P ₃	3	0	2
P ₄	2	1	1
P ₅	0	0	2

Trabajo

3	3	2
---	---	---

Trabajo

5	4	3
---	---	---

Trabajo

5	4	5
---	---	---

Trabajo

8	4	7
---	---	---

Trabajo

11	4	9
----	---	---

Trabajo

11	5	9
----	---	---

Estado Seguro !!

Secuencia segura: P₄→P₅→P₂→P₃→P₁

Ejemplo algoritmo seguridad

■ ¿Estado seguro?

Disponible

R ₁	R ₂	R ₃
3	1	0

Trabajo

3	1	0
---	---	---

Máximo

	R ₁	R ₂	R ₃
P ₁	7	5	3
P ₂	3	2	2
P ₃	9	0	2
P ₄	2	2	2
P ₅	4	3	3

Necesito

	R ₁	R ₂	R ₃
P ₁	7	2	1
P ₂	0	2	0
P ₃	6	0	0
P ₄	0	1	1
P ₅	4	3	1

Asignado

	R ₁	R ₂	R ₃
P ₁	0	3	2
P ₂	3	0	2
P ₃	3	0	2
P ₄	2	1	1
P ₅	0	0	2

✗
✗
✗
✗
✗

Estado Inseguro !!

No implica que el sistema entre en interbloqueo

Algoritmo de petición de recursos

- Estructuras auxiliares:

Petición[i]: Vector de tamaño m con los recursos que pide el proceso i.

Procedimiento PeticiónRecursos (Petición[i])

Si Petición[i] > Necesito[i] **entonces**

 Error (*"Proceso ha excedido su solicitud máxima"*)

FinSi

Si Petición[i] > Disponible **entonces**

 SuspenderProceso(i)

Sino

 Disponible = Disponible - Petición[i]

 Asignado[i] = Asignado[i] + Petición[i]

 Necesito[i] = Necesito[i] - Petición[i]

Si Seguridad() == Cierto **entonces**

 AsignarRecursos(i)

Sino

 RecuperarEstadoPrevio()

 SuspenderProceso(i)

FinSi

FinSi

Fin Procedimiento

Ejemplo algoritmo petición de recursos (I)

Disponible

R ₁	R ₂	R ₃
3	3	2

Máximo

	R ₁	R ₂	R ₃
P ₁	7	5	3
P ₂	3	2	2
P ₃	9	0	2
P ₄	2	2	2
P ₅	4	3	3

Necesito

	R ₁	R ₂	R ₃
P ₁	7	4	3
P ₂	0	2	0
P ₃	6	0	0
P ₄	0	1	1
P ₅	4	3	1

Asignado

	R ₁	R ₂	R ₃
P ₁	0	1	0
P ₂	3	0	2
P ₃	3	0	2
P ₄	2	1	1
P ₅	0	0	2



1	0	2
---	---	---

?

Disponible

1	0	2
---	---	---

3	3	2
---	---	---

Ok

Necesito P₁

1	0	2
---	---	---

7	4	3
---	---	---

Ok

Disponible

R ₁	R ₂	R ₃
2	3	0

Necesito

	R ₁	R ₂	R ₃
P ₁	6	4	1
P ₂	0	2	0
P ₃	6	0	0
P ₄	0	1	1
P ₅	4	3	1

Asignado

	R ₁	R ₂	R ₃
P ₁	1	1	2
P ₂	3	0	2
P ₃	3	0	2
P ₄	2	1	1
P ₅	0	0	2

Trabajo

2	3	0
---	---	---

Trabajo

5	3	2
---	---	---

Trabajo

7	4	3
---	---	---

Trabajo

10	4	5
----	---	---

Trabajo

11	5	7
----	---	---

Trabajo

11	5	9
----	---	---

Estado Seguro → Se concede la petición de recursos

Ejemplo algoritmo petición de recursos (II)

Disponible

R ₁	R ₂	R ₃
3	3	2

Máximo

	R ₁	R ₂	R ₃
P ₁	7	5	3
P ₂	3	2	2
P ₃	9	0	2
P ₄	2	2	2
P ₅	4	3	3

Necesito

	R ₁	R ₂	R ₃
P ₁	7	4	3
P ₂	0	2	0
P ₃	6	0	0
P ₄	0	1	1
P ₅	4	3	1

Asignado

	R ₁	R ₂	R ₃
P ₁	0	1	0
P ₂	3	0	2
P ₃	3	0	2
P ₄	2	1	1
P ₅	0	0	2

P₁

1	2	2
---	---	---

?

Disponible

1	2	2
---	---	---

≤

3	3	2
---	---	---

Ok

Necesito P₁

1	2	2
---	---	---

≤

7	4	3
---	---	---

Ok

Trabajo

2	1	0
---	---	---

Disponible

R ₁	R ₂	R ₃
2	1	0

Necesito

	R ₁	R ₂	R ₃
P ₁	6	2	1
P ₂	0	2	0
P ₃	6	0	0
P ₄	0	1	1
P ₅	4	3	1

X

X

X

X

X

Asignado

	R ₁	R ₂	R ₃
P ₁	1	3	2
P ₂	3	0	2
P ₃	3	0	2
P ₄	2	1	1
P ₅	0	0	2

Estado Inseguro → No se concede la petición

Evitación interbloqueo:

Conclusiones

■ Ventajas:

- No se requiere expulsar o hacer retroceder la ejecución de los procesos.
- Es menos restrictiva que los métodos de prevención.

■ Inconvenientes:

- Se debe anunciar la máxima demanda de recursos por anticipado.
- Los procesos deben ser independientes entre si.
- Número fijo de recursos y procesos.

Técnicas tratamiento del interbloqueo

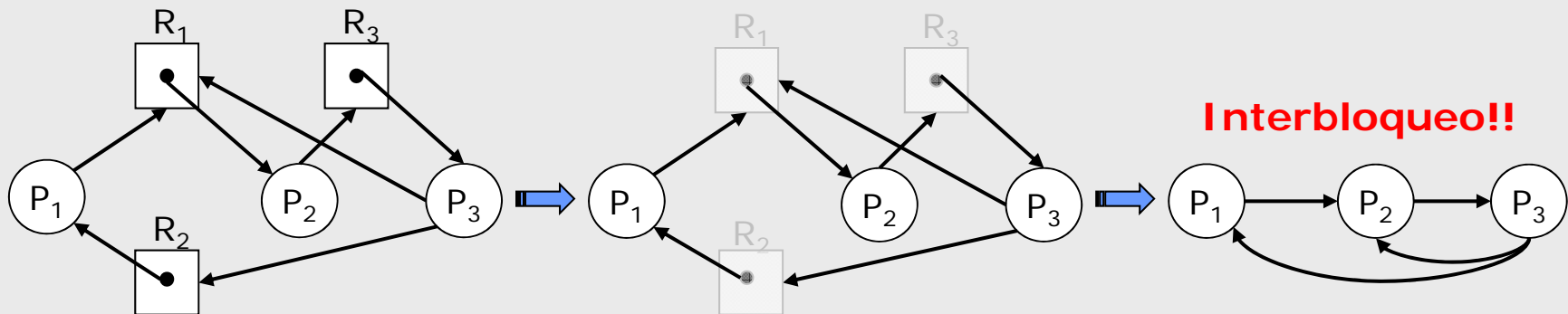
- Ignorar el problema.
- Prevenir el interbloqueo.
- Evitar el interbloqueo.
- **Detección y recuperación del interbloqueo.**
- Estrategias Integradas.

Detección del interbloqueo

- Esta estrategia de tratamiento del interbloqueo se basa en dejar que el interbloqueo se **produzca**, **detectarlo** y realizar un proceso de **recuperación** del interbloqueo.
- Tres aspectos fundamentales para la aplicación de esta estrategia:
 1. Como se detecta el interbloqueo.
 2. Con que frecuencia se utiliza el algoritmo de detección de interbloqueo
 3. Como se recupera el interbloqueo
- Para detectar el bloqueo se pueden utilizar distintos algoritmos en función del número de instancias disponibles de los recursos.

Detección: si solo hay recursos de instancia única

- Para detectar el interbloqueo en este caso, se utiliza el **grafo de espera**.
- El grafo de espera se obtiene a partir del grafo de asignación de recursos:
 - Se eliminan los nodos correspondientes a los recursos.
 - Se ajustan los arcos de forma que habrá un arco del proceso P_i al proceso P_j si P_j posee un recurso que P_i ha solicitado.
- Si aparece un ciclo en el grafo de espera entonces existe interbloqueo.



Detección interbloqueos en los grafos de espera

- A partir del grafo de espera se puede calcular si hay interbloqueo especificando una matriz A de $N \times N$ (siendo N el número de procesos) t.q:
 - Se inicializa toda la matriz a cero.
 - Si existe un arco entre el proceso p_i y p_j entonces $A[i,j]=1$
- Para detectar el interbloqueo se multiplica la matriz por si misma:
$$A^k = A^{k-1} * A \quad \text{siendo } 2 \leq k \leq N$$
- Si aparece un número mayor que 0 en la diagonal de la matriz A^k , existe un interbloqueo de longitud K .
- Ejemplo:

$$A =$$

0	1	0
0	0	1
1	1	0

$$A^2 =$$

0	1	0
0	0	1
1	1	0

$$*$$

0	1	0
0	0	1
1	1	0

$$=$$

0	0	1
1	1	0
0	1	1

Proceso p_2 y p_3 están en interbloqueo de longitud 2 !!

$$A^3 =$$

0	0	1
1	1	0
0	1	1

$$*$$

0	1	0
0	0	1
1	1	0

$$=$$

1	1	0
0	1	1
1	1	1

Proceso p_1 , p_2 y p_3 están en interbloqueo de longitud 3 !!

Si los recursos tienen más de una instancia

- Se utiliza una variante del algoritmo de seguridad.

Función Detección **devuelve** Boolean

Trabajo = Disponible

Para todo i

Si Asignado[n] != 0 entonces

 Acabado[n] = Falso

Sino

 Acabado[n] = Cierto

FinSi

FinPara

Mientras $\exists i$ tal que Acabado[i] == Falso Y Necesito[i] <= Trabajo

 Trabajo = Trabajo + Asignado[i]

 Acabado[i] = Cierto

FinMientras

Si $\forall i$ Acabado[i] = Cierto **entonces**

 return (Cierto)

Sino

 return(Falso)

FinSi

FinFunción

Frecuencia de utilización algoritmo detección

- Alternativas propuestas:
 - Cada vez que una petición de recursos no puede ser concedida inmediatamente.
 - Cada cierto intervalo de tiempo.
- Ventajas / Inconvenientes:
 - La primera alternativa produce **más sobrecarga**, pero es **más fácil detectar** qué procesos provocan el interbloqueo (pues como mucho habrá un interbloqueo).
 - Con la segunda alternativa podemos **limitar** la **sobrecarga** del algoritmo de detección, pero si existen varios ciclos será **difícil determinar** qué **procesos** están **involucrados** en cada uno de los interbloqueos.

Recuperación del interbloqueo (I)

- Existen diversas alternativas par la recuperación del interbloqueo:
 - Avisar al operador y que él decida.
 - Terminación de procesos.
 - Apropiación de los recursos.
- Terminación de procesos:
 - Terminación **de todos** los procesos implicados en el interbloqueo.
 - Terminación **iterativa** de procesos, hasta que el interbloqueo desaparezca.

La elección de los procesos se pueden tener en cuenta diversos factores: la prioridad, tiempo de ejecución, recursos asignados, recursos requeridos, etc...

Recuperación del interbloqueo (II)

- Apropriación de recursos.

El sistema se apropia de recursos de los procesos interbloqueados, hasta que desaparece el interbloqueo.

- Problemas a resolver:

- **Selección de una víctima:** A quién se elige para apropiarse de sus recursos.
- **Vuelta atrás:** El proceso a quien se le quitan los recursos debe ser vuelto a un estado seguro (la solución más fácil es abortar dicho proceso).
- **Inanición:** Hay que considerar que no se debería quitar siempre los recursos al mismo proceso, sobre todo si la vuelta atrás supone abortarlo y obligarle a empezar desde el principio.

Técnicas tratamiento del interbloqueo

- Ignorar el problema.
- Prevenir el interbloqueo.
- Evitar el interbloqueo.
- Detección y recuperación del interbloqueo.
- Estrategias Integradas.

Estrategias Integradas

- Ninguna de las estrategias básicas (prevención, evitación y detección) por si solas es apropiada para todo los problemas de asignación de recursos.
- **Propuesta:**
Combinar las tres enfoques básicos, utilizando la estrategia más idónea según la clase de recurso.
- **Ejemplo:**
 - Dividir los recursos del sistema en clases de recursos, los cuales tienen un ordenamiento jerárquico. Se aplica la técnica de ordenamiento de recursos para prevenir la espera circular.
 - Recursos internos: Recursos utilizados por el sistema (PCB, ...).
 - Memoria principal: Memoria del proceso.
 - Recursos de trabajo: dispositivos asignables y ficheros.
 - Swap.
 - Dentro de cada clase se utiliza la técnica más adecuada al tipo de recurso.

Ejercicio:

- Tenemos un sistema con los siguientes recursos: Cd-Rom 4 instancias (R_1), Impresora 2 instancias (R_2), Plotter 3 instancias (R_3) y Grabadoras 1 instancias (R_4).
- Y las matrices de requisitos y recursos asignados para los 3 procesos del sistema:

Máximo

	R_1	R_2	R_3	R_4
P_1	2	0	1	1
P_2	3	0	1	1
P_3	2	2	2	0

Asignados

	R_1	R_2	R_3	R_4
P_1	0	0	1	0
P_2	2	0	0	1
P_3	0	1	2	0

- Calcular las matrices de solicitudes y disponibles.
- Evaluar si se pueden atender las siguientes peticiones, suponiendo un método de evitación del interbloqueo.
 $P_2(1,0,0,0)$, $P_1(2,0,0,1)$ y $P_3(1,1,0,0)$.

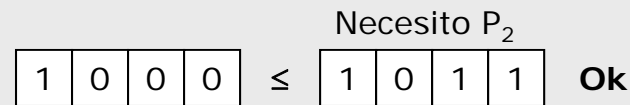
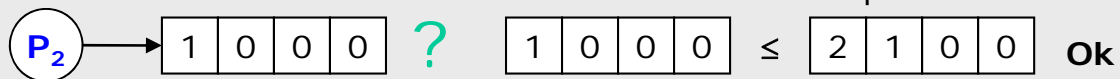
Solución (I)

	R_1	R_2	R_3	R_4
P_1	2	0	1	1
P_2	3	0	1	1
P_3	2	2	2	0

	R_1	R_2	R_3	R_4
P_1	0	0	1	0
P_2	2	0	0	1
P_3	0	1	2	0

R_1	R_2	R_3	R_4
2	1	0	0

	R_1	R_2	R_3	R_4
P_1	2	0	0	1
P_2	1	0	1	0
P_3	2	1	0	0



1	1	0	0
---	---	---	---

R_1	R_2	R_3	R_4
1	1	0	0

	R_1	R_2	R_3	R_4
P_1	2	0	0	1
P_2	0	0	1	0
P_3	2	1	0	0

	R_1	R_2	R_3	R_4
P_1	0	0	1	0
P_2	3	0	0	1
P_3	0	1	2	0

X
X
X

Estado inseguro !!

No se concede la petición del proceso 2, proceso suspendido.

Solución (II)

Máximo

	R ₁	R ₂	R ₃	R ₄
P ₁	2	0	1	1
P ₂	3	0	1	1
P ₃	2	2	2	0

Asignados

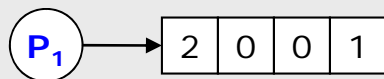
	R ₁	R ₂	R ₃	R ₄
P ₁	0	0	1	0
P ₂	2	0	0	1
P ₃	0	1	2	0

Disponible

R ₁	R ₂	R ₃	R ₄
2	1	0	0

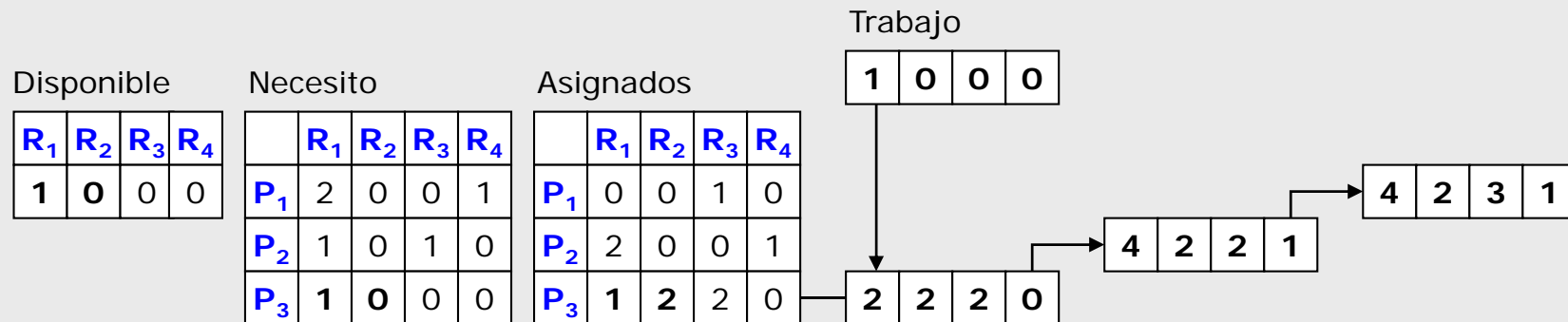
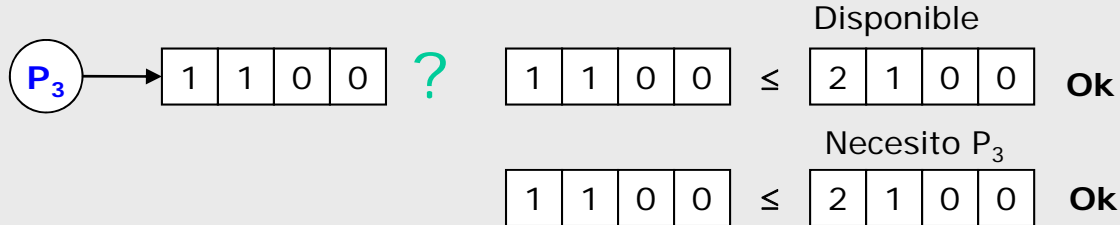
Necesito

	R ₁	R ₂	R ₃	R ₄
P ₁	2	0	0	1
P ₂	1	0	1	0
P ₃	2	1	0	0


 ? [2, 0, 0, 1] ≤ ^{Disponible} [2, 1, 0, 0] NO ⇒ Proceso 1 suspendido

Solución (III)

Máximo					Asignados					Disponible				Necesito					
	R ₁	R ₂	R ₃	R ₄		R ₁	R ₂	R ₃	R ₄		R ₁	R ₂	R ₃	R ₄		R ₁	R ₂	R ₃	R ₄
P ₁	2	0	1	1	P ₁	0	0	1	0	2	1	0	0	P ₁	2	0	0	1	
P ₂	3	0	1	1	P ₂	2	0	0	1					P ₂	1	0	1	0	
P ₃	2	2	2	0	P ₃	0	1	2	0					P ₃	2	1	0	0	



Estado seguro !!

Se concede la petición del proceso 3