Interbloqueos

Un interbloqueo es una situación en la que dos o más procesos **quedan bloqueados indefinidamente** al entrar en **conflicto sus diferentes necesidades**.

Ocurren porque **los procesos compiten** por el uso de recursos y porque **se comunican** y sincronizan entre ellos.

Puede ocurrir en muchas situaciones.

Recursos

Los recursos son **Objetos** que otorga el Sistema Operativo (dispositivo de hardware, un registro de una base de datos...) otorga a los procesos.

Recursos Apropiativos

Es aquel recurso que se puede quitar al proceso que lo posee sin efectos dañinos (Ej.: la RAM a un proceso).

Recursos Apropiativos

Es aquel que no se puede quitar a su propietario actual sin hacer que el cómputo falle. (Ej.: cuando un proceso ha empezado a quemar un CD-ROM y le quitamos de manera repentina el grabador de CD; ya que ahora el CD estará lleno de basura).

En general, los recursos que producen interbloqueos son los **No Apropiativos**. Para utilizar un recurso es necesario seguir una serie de pasos:

1. Solicitar el Recurso.
2. Usar el Recurso.
3. Liberar el Recurso.

Adquisición de Recursos

Una manera de permitir que los usuarios administren los recursos es asociar un semáforo con cada recurso.

Cuando dos procesos compiten por más de un recurso, se adquieren secuencialmente como se muestra en el código siguiente:

```
typedef int     semaforo;
semaforo        recurso_1;
semaforo        recurso_2;

void proceso_A (void)
{
    down (&recurso_1);
    down (&recurso_2);
    usar_ambos_recursos();
    up (&recurso_2);
```

```
    up (&recurso_1);
}

void proceso_b (void)
{
    down (&recurso_1);
    down (&recurso_2);
    usar_ambos_recursos();
    up (&recurso_2);
    up (&recurso_1);
}
```

Hay que recalcar que en el anterior código es importante que los semáforos se "upeen" y "downeen" en el mismo orden los 2:

Pongamos la situación de que el Proceso B tiene los 'up' en el orden inverso (primero recurso_2 y luego recurso_1): Imaginemos que el Proceso_A adquiere el down del recurso_1 y justo en ese momento el Proceso_B obtiene el quantum y adquiere el down del recurso_2.

En este momento el Proceso_A se queda esperando a que se libere el recurso_2 y el Proceso_B se queda esperando a que se libere el recurso_1 y NO SE AVANZA = INTERBLOQUEO

Interbloqueos

Un interbloqueo es, formalmente... *Un conjunto de procesos se encuentra en un interbloqueo si cada proceso está esperando un evento que sólo puede ser ocasionado por otro proceso del conjunto*. Ninguno de los procesos se puede ejecutar, ninguno de ellos puede liberar recursos y ninguno puede ser despertado.

Para este modelo suponemos que los procesos tienen un sólo hilo y que no hay interrupciones posibles para despertar a un proceso bloqueado.

Condiciones para los Interbloqueos de Recursos

Se deben aplicar 4 condiciones para un interbloqueo de recursos:

1. **Condición de Exclusión Mútua**: Cada recurso se asigna en un momento dado a sólo un proceso, o está disponible.
2. **Condición de Espera**: Los procesos que actualmente contienen recursos que se les otorgaron antes pueden solicitar nuevos recursos.
3. **Condición No Apropiativa**: Los recursos otorgados previamente no se pueden quitar a un proceso por la fuerza. Deben ser liberados explícitamente por el proceso que los contiene.
4. **Condición de Espera Circular**: Debe haber una cadena circular de dos o más procesos, cada uno de los cuales espera un recurso contenido por el siguiente miembro de la cadena.

Modelado de Interbloqueos

Se pueden modelar las cuatro condiciones mediante el uso de Grafos Dirigidos.

```
Proceso = Círculos
Recurso = Cuadrados

Arco Dirigido de un Recurso a un Proceso = el Recurso está asignado a ese Proceso.
Arco Dirigido de un Proceso a un Recurso = el Proceso está bloqueado a la espera
del Recurso.
```

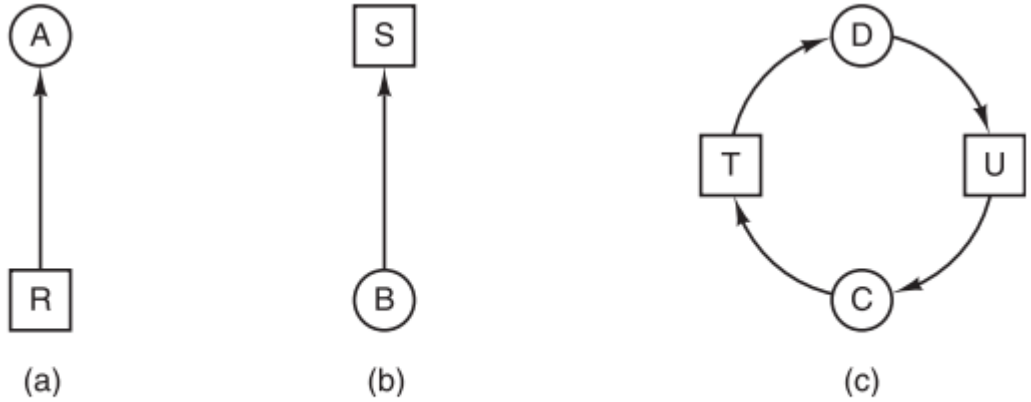


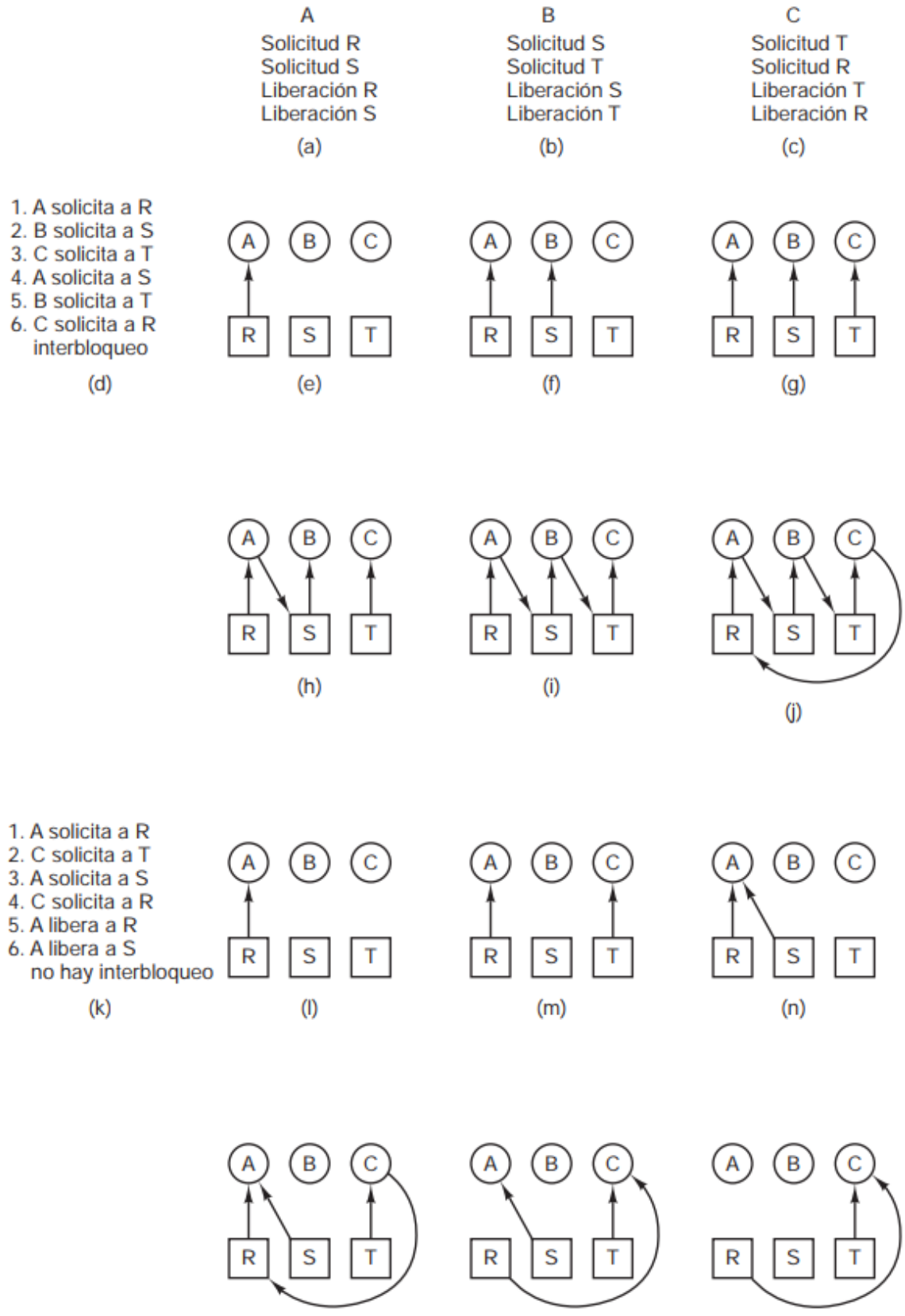
Figura 6-3 Gráficos de asignación de recursos. (a) Contención de un recurso. (b) Petición de un recurso. (c) Interbloqueo.

El Sistema Operativo es libre de ejecutar cualquier proceso desbloqueado en cualquier momento, por lo que podría optar por ejecutar A hasta que terminara todo su trabajo, después B y por último C. Este orden mencionado no produciría ningún tipo de Interbloqueos, pero tampoco Paralelismo.

Una forma que podría tener el Sistema Operativo de asociar los Recursos es ir dando a cada proceso los recursos que vaya necesitando. Sin embargo, esto podría llevarnos a un Interbloqueo como se muestra en el apartado (d) de la siguiente imagen.

La solución: El SO no tiene que ejecutar los procesos en un orden especial, en particular si otorgar un recurso puede producir un Interbloqueo. El SO puede simplemente suspender el proceso siin otorgar la solicitud hasta que sea seguro.

```
Si el SO supiera los interbloqueos que más tarde se podrían suceder, suspendería
el proceso B para evitar Interbloqueos, y más adelante cuando acabe A o C podría
asignarle los recursos (k).
```



Estrategias para Lidiar con Interbloqueos

1. **Ignorar el Problema.**
2. **Detección y Recuperación.**
3. **Evitarlos Dinámicamente mediante la Asignación de Recursos.**
4. **Prevención.**

Ignorar el Problema: ALGORITMO DE LA AVESTRUZ

Los matemáticos la encuentran totalmente inaceptable y dicen que los interbloqueos se deben prevenir a toda costa.

Sin embargo, es el más habitual en Windows y Linux.

Detección: UN SÓLO TIPO DE RECURSO

Cualquier proceso que forme parte de un ciclo está en un Interbloqueo.

A continuación veremos un algoritmo simple que inspecciona un gráfico y termina al haber encontrado un ciclo, o cuando ha demostrado que no existe ninguno.

Utiliza una Estructura Dinámica de Datos (L), una Lista de Nodos (N) y una Lista de Arcos (A).

Para cada nodo N del grafo realizar lo siguiente:

1. Inicializar L y todos los arcos desmarcados.
2. Agregar el nodo actual a L. Si aparece 2 veces en L hay un ciclo -> Finaliza.
3. Si del nodo actual hay arcos salientes desmarcados ir al Paso 4, si no hay arcos salientes ir al Paso 5.
4. Elegir un arco saliente cualquiera y marcarlo. Nos llevará al Nodo Actual ene l que hay que volver al Paso 2.
5. Si el Nodo Actual es el inicial, el grafo no contiene ciclos y finaliza

Detección: VARIOS TIPOS DE RECURSOS

Para realizar este algoritmo necesitamos 4 Estructuras:

- Vector de Recursos Existentes.
- Vector de Recursos Disponibles.
- Matriz de Asignaciones Actuales (cada fila es un proceso con sus recursos).
- Matriz de Peticiones (cada fila es un proceso con sus peticiones).

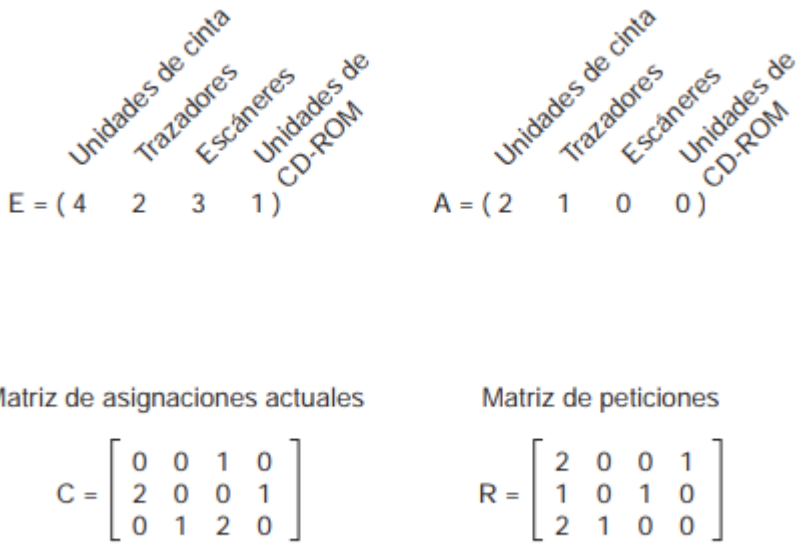


Figura 67 Un ejemplo para el algoritmo de detección de interbloqueos.

Tenemos que ver la Matriz de Peticiones, donde el 1º proceso solicita 2 unidades de cinta y 1 unidad CD-ROM. Pero teniendo en cuenta que no hay unidades de esos recursos disponibles en el Vector de Recursos Disponibles no se ejecuta.

Ahora vemos el siguiente proceso que solicita 1 unidad de cinta y 1 escáner. Pero teniendo en cuenta que no hay unidades de esos recursos disponibles en el Vector de Recursos Disponibles no se ejecuta.

Por último, vemos el 3º proceso que solicita recursos que sí están disponibles. Por lo que se le asignan.

Recuperación de Interbloqueo

- Recuperación por medio de apropiación.
- Recuperación a través del retroceso.
- Recuperación a través de la eliminación de procesos.

Evitar Interbloqueos: TRAYECTORIAS DE RECURSOS

La gráfica siguiente simplemente implementa una forma visual de ver las regiones donde se pueden producir interbloqueos.

El eje horizontal representa la ejecución de un proceso y la linea vertical la de otro proceso.

Entre el tramo de 'p' y 'q' se ejecuta el Proceso 1, entre el tramo 'q' y 'r' se ejecuta el Proceso 2...

El programador es el que elige cuando se ejecuta un proceso y cuánto se ejecuta. Pero tiene que tener cuidado ya que debería elegir una trayectoria por la que pasen la mayor parte de los recursos. Teniendo en cuenta que las regiones donde hay lineas diagonales en ambos sentidos son regiones donde seguramente se producen interbloqueos.

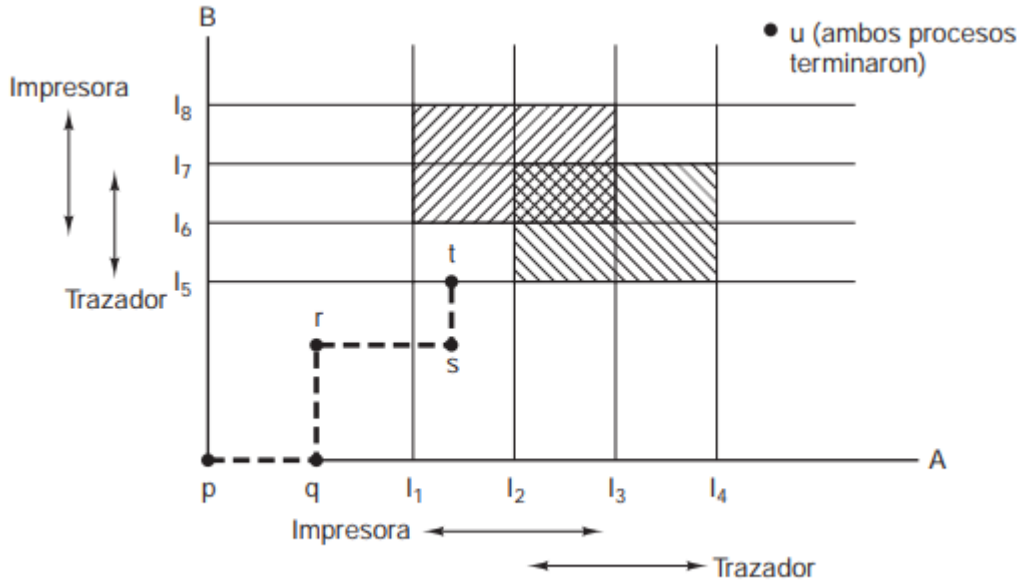


Figura68 Trayectorias de recursos de dos procesos.

Evitar Interbloqueos Un Recurso: ESTADOS SEGUROS E INSEGUROS

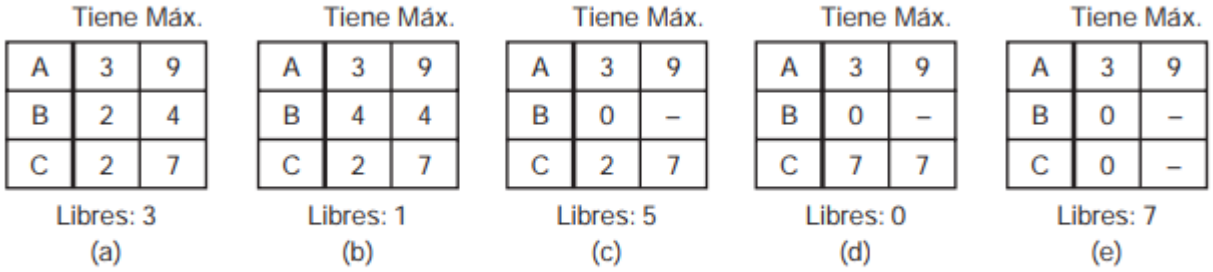
Antes de nada hay que saber lo que es un Estado Seguro: un Estado es Seguro si hay un cierto orden de programación en el que se puede ejecutar cada proceso hasta completarlo.

En la siguiente figura encontramos que cada fila es un proceso (A, B y C). La columna 'Tiene' representa los recursos que actualmente posee un proceso; mientras que la columna 'Max' representa el número máximo de recursos que puede necesitar en un momento dado.

La cuestión es que en la imagen (a) el Proceso_A puede necesitar hasta 6 recursos más, el Proceso_B puede necesitar hasta 2 recursos más y el Proceso_C puede necesitar hasta 5 recursos más.

Si tenemos recursos libres el programador podría darselos todos a Proceso_B y completar el número de recursos máximos. Cuando acabe el trabajo que tenga que hacer el Proceso_B va a dejar un total de 5 recursos libres (2 de ahora + 2 nuevos + 1 que se queda libre). De esta forma podríamos completar después el Proceso_C y después el Proceso_A con los restantes.

Esta secuencia se muestra en las imagenes (b), (c), ...



	Tiene	Máx.
A	3	9
B	2	4
C	2	7

Libres: 3
(a)

	Tiene	Máx.
A	3	9
B	4	4
C	2	7

Libres: 1
(b)

	Tiene	Máx.
A	3	9
B	0	–
C	2	7

Libres: 5
(c)

	Tiene	Máx.
A	3	9
B	0	–
C	7	7

Libres: 0
(d)

	Tiene	Máx.
A	3	9
B	0	–
C	0	–

Libres: 7
(e)

Figura 69 Demostración de que el estado en (a) es seguro.

Algoritmo del Banquero

El Algoritmo del Banquero comprueba si al otorgar la petición se produce un estado inseguro. Si es así, la petición se rechaza. Sin embargo, si al otorgar la petición se produce un estado seguro, se lleva a cabo.

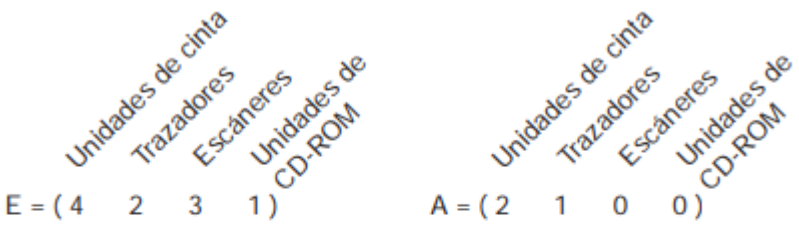
Por ejemplo, si en la imagen anterior (en el estado (a)) se le asocia 2 recursos más al proceso C, se produciría un Estado Inseguro, ya que no habría ninguna forma de completar las necesidades de cada proceso y se produciría un Interbloqueo.

Algoritmo del Banquero para Varios Recursos

El concepto es el mismo, tenemos unos Asociados a cada proceso, unos que Aún Necesitan y los disponibles. Se trata de elegir tuplas completas a las que se le puede dar los recursos que necesiten para ir satisfaciendo necesidades.

```
Primera Matriz = Recursos Actualmente Asignados a cada proceso.
Segunda Matriz = Recursos que Aún Necesitan ser asignados por cada proceso.

E = Recursos totales de los que dispone la computadora.
P = Recursos poseidos en total entre todos los procesos (P_A + P_B + P_C + P_D +
P_E).
A = Recursos Disponibles (E - P)
```



Matriz de asignaciones actuales

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

Matriz de peticiones

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

Figura 67 Un ejemplo para el algoritmo de detección de interbloqueos.