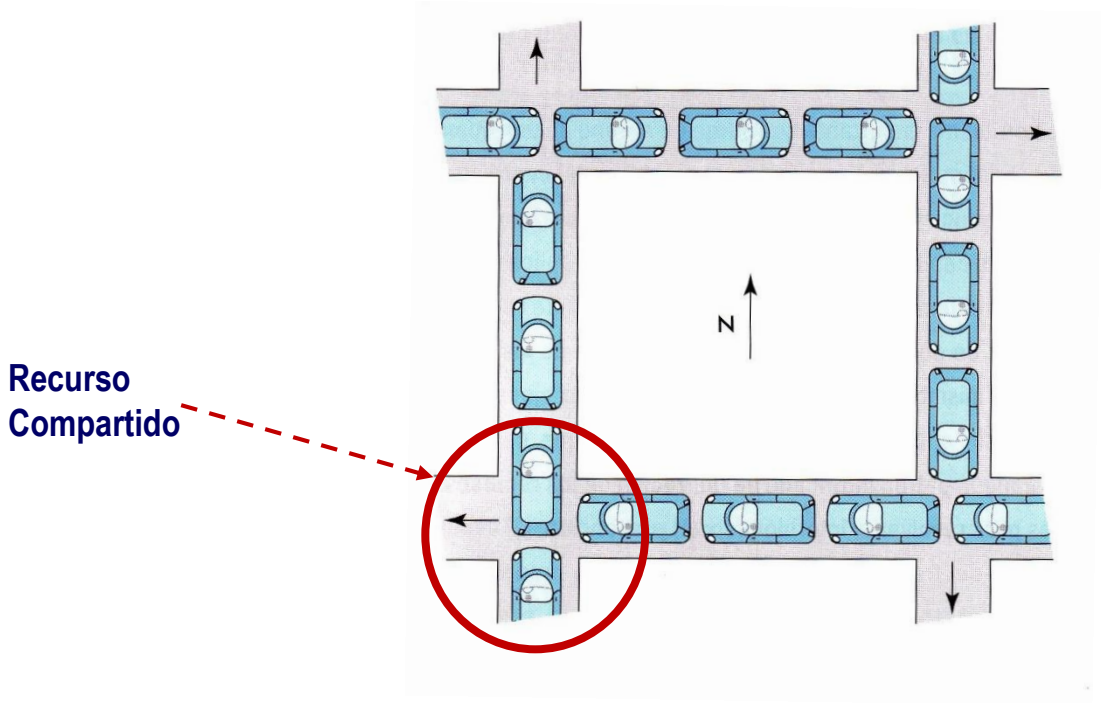
INTERBLOQUEO - DEADLOCK

Karim Guevara Puente de la Vega

2018

Introducción

- Interbloqueo se da en muchas facetas de la vida
- Algunos vistos son la situación de los filósofos
- Embotellamiento de carros



Introducción

- Ocurre en:
  - Una comunidad de procesos concurrentes que comparten recursos.
  - Necesitan acceso exclusivo a uno o varios recursos:
    - Dispositivos físicos → impresora, cinta
    - Dispositivos lógicos → registro de BD
    - Secciones críticas

- Solución: destrucción de hilos/procesos

Abrazo Mortal

- **Recursos**
  - Cualquier cosa que es necesaria para que un proceso avance: consumible/reutilizable.
    - Memoria, unidad de cinta, un mensaje, un valor positivo de un semáforo
  - Proceso se puede bloquear cuando solicite cualquiera de estas entidades
  - **Tipos:**
    - Apropiativos: se puede quitar al proceso que lo posee sin efectos dañinos
    - No apropiativos: no se puede quitar a su propietario actual sin hacer que el cómputo falle

Abrazo Mortal

- Entre hilos dentro de un proceso.
  - AL compartir un recurso local del proceso, pero global a todos los hilos del proceso.
- Entre hilos de distintos procesos.
  - Al compartir recursos del sistema por hilos de distintos procesos
- El problema no está en ningún hilo/proceso concreto, sino en la acción colectiva de los hilos/procesos.

Adquisición de recursos

1. Solicitar el recurso → P(semáforo)
2. Utilizar el recurso
3. liberar el recurso → V(semáforo)

```
typedef int semaforo;
semaforo recurso_1;
void proceso_A(void) {
   down(&recurso_1);
   usar_recurso_1();
   up(&recurso_1);
}
```

Abrazo Mortal

- Dijkstra [1968]: “Como lo que ocurre entre un grupo de procesos cuando cada uno posee un recurso mientras está solicitando otro”.
  - La petición no puede ser satisfecha nunca ya que el recurso solicitado está en poder de otro proceso que está bloqueado esperando por el recurso que tiene el primero.

Proceso 1	Proceso 2	Proceso 3
. . . solicitar (recurso 1); /*tengo rec 1.*/ . . . solicitar (recurso 2);	. . . solicitar (recurso 2); /*tengo rec 2.*/ . . . solicitar (recurso 3);	. . . solicitar (recurso 3); /*tengo rec 3.*/ . . . solicitar (recurso 1);

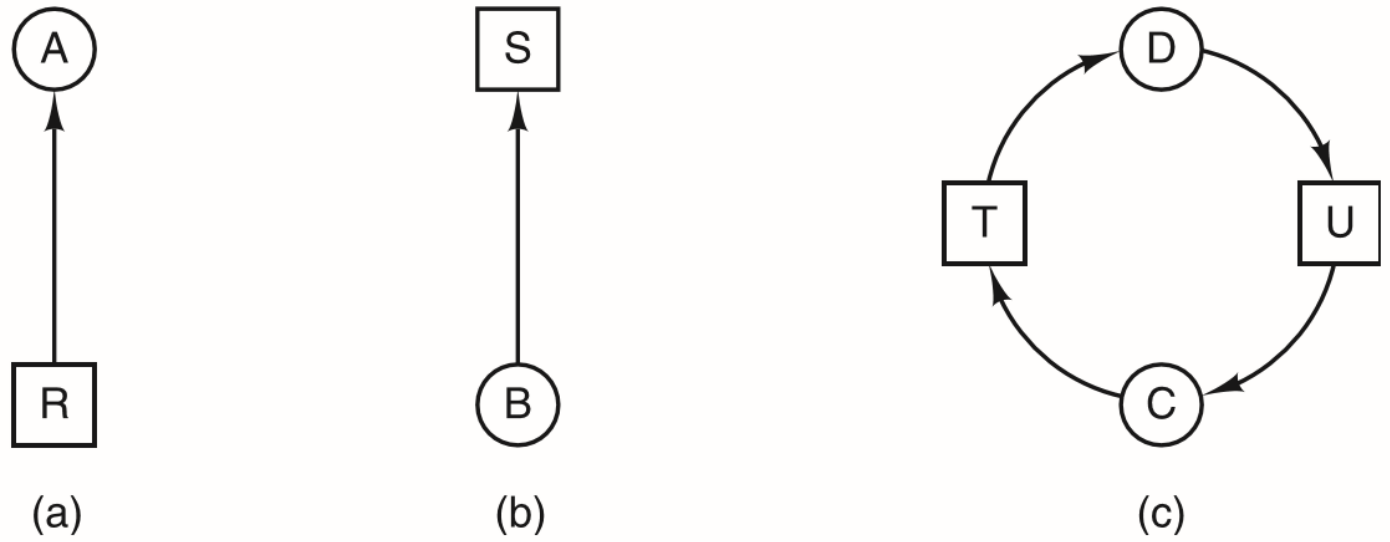
Condiciones para el Interbloqueo

- [Coffman] De forma simultánea:
  1. Exclusión mutua – cada recurso se asigna en un momento dado a sólo un proceso
  2. Poseer y esperar - procesos con recursos asignados, pueden solicitar otros recursos.
  3. No apropiación - recursos no pueden ser apropiados de un proceso, éstos deben ser liberados explícitamente.
  4. Espera circular - cadena circular de procesos, donde cada uno espera un recurso asignado al siguiente proceso.

Modelo de Interbloqueo

- Representa el estado de asignación de los componentes del sistema:
  - Recursos asignados a cada proceso.

- Grafos:



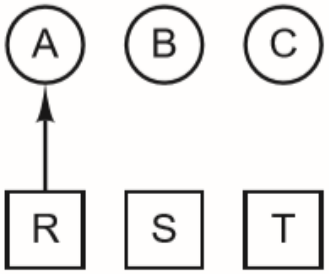
Modelo de interbloqueo

A	B	C
Solicitud R	Solicitud S	Solicitud T
Solicitud S	Solicitud T	Solicitud R
Liberación R	Liberación S	Liberación T
Liberación S	Liberación T	Liberación R

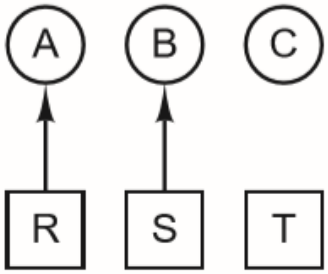
1. A solicita a R
2. B solicita a S
3. C solicita a T
4. A solicita a S
5. B solicita a T
6. C solicita a R

interbloqueo

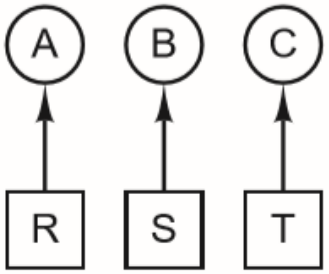
(d)



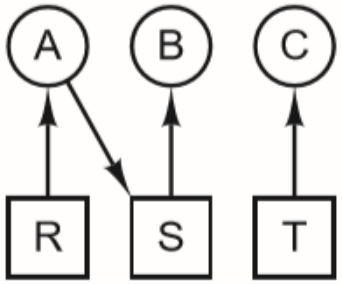
(e)



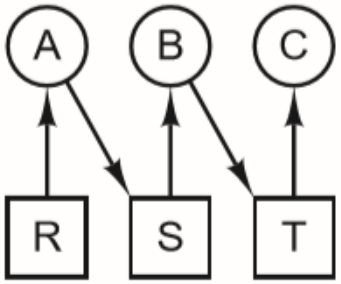
(f)



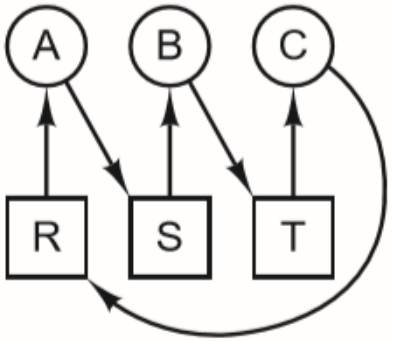
(g)



(h)



(i)

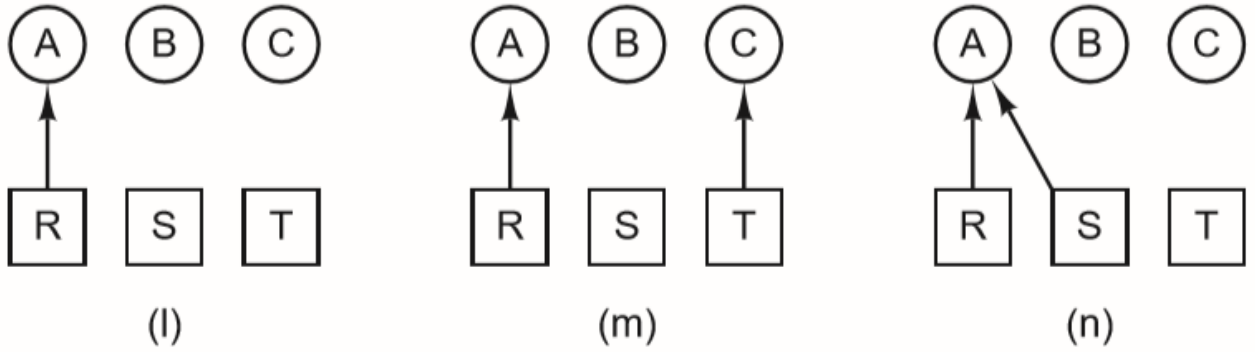


(j)

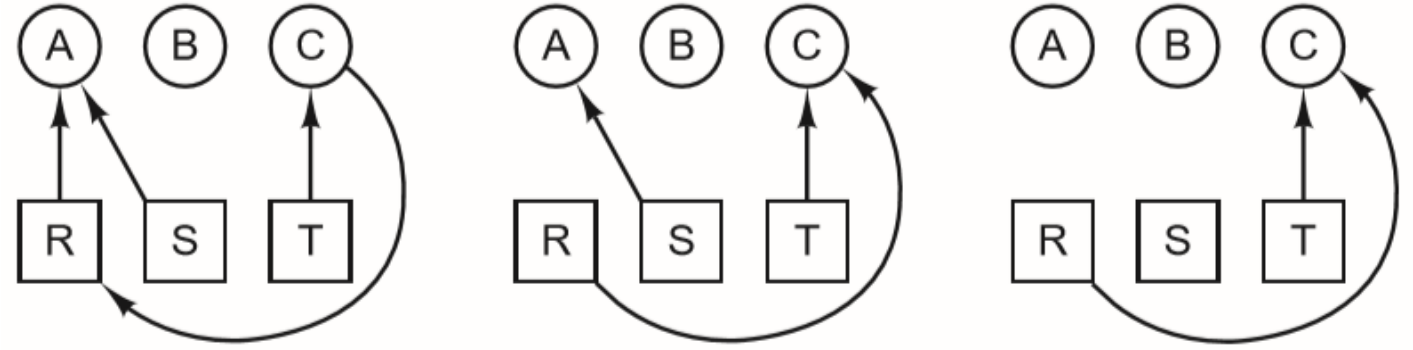
Modelo de interbloqueo

1. A solicita a R
2. C solicita a T
3. A solicita a S
4. C solicita a R
5. A libera a R
6. A libera a S

no hay interbloqueo



(k) (l) (m) (n)



Estrategias de gestión de interbloqueos

- Tres enfoques generales y un Ad-hoc:
  - Ignorar el problema (avestruz)
  - Detección y recuperación
  - Evitación
  - Prevención

Detección y recuperación

- Permite que el gestor de recursos sea más agresivo en la asignación de los mismos.
  - Se asignan los recursos siempre que estén disponibles.
  - Si un proceso está bloqueado por largo tiempo, se ejecuta el algoritmo de detección para ver si el estado actual es de interbloqueo.

Detectar y recuperar

Detección del interbloqueos con un recurso de cada tipo

P.e.:

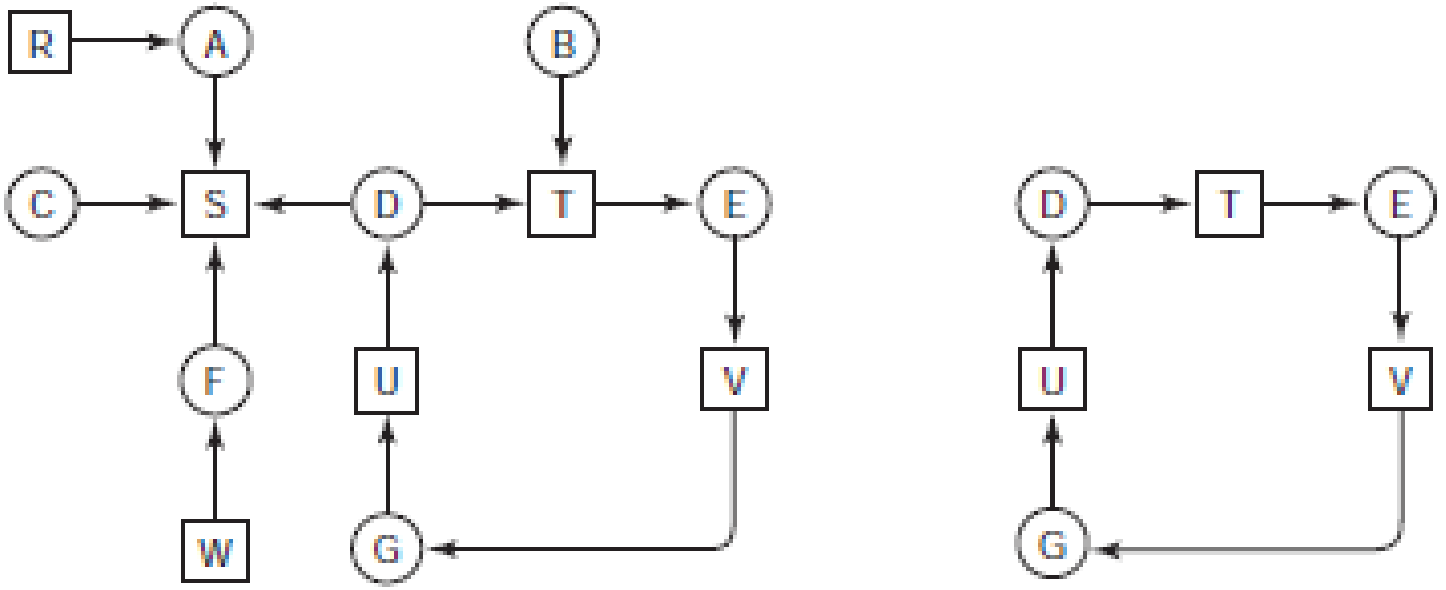
- El proceso *A* contiene a *R* y quiere a *S*.
- 2. El proceso *B* no contiene ningún recurso pero quiere a *T*.
- 3. El proceso *C* no contiene ningún recurso pero quiere a *S*.
- 4. El proceso *D* contiene a *U* y quiere a *S* y a *T*.
- 5. El proceso *E* contiene a *T* y quiere a *V*.
- 6. El proceso *F* contiene a *W* y quiere a *S*.
- 7. El proceso *G* contiene a *V* y quiere a *U*.

¿Está este sistema en interbloqueo y, de ser así, cuáles procesos están involucrados?

- Construir el grafico de recursos

Detectar y recuperar

Detección del interbloqueos con un recurso de cada tipo

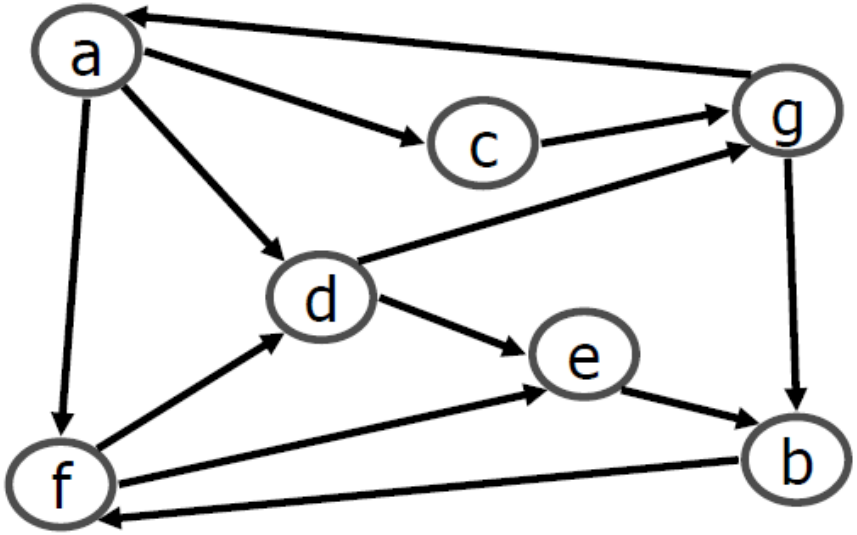


Detectar y recuperar

Detección del interbloqueos con un recurso de cada tipo

- **Algoritmo para detectar ciclos – Depth First Search (DFS)**

  Sea L : estructura dinámica
  lista de nodos y de arcos.



Detección y recuperación

Detección de bloqueos con múltiples recursos por tipo

N procesos

M tipos de recursos

$$\sum_{i=1}^{n} C_{ij} + A_j = E_j$$

Recursos en existencia

$$(E_1, E_2, E_3, \ldots E_m)$$

Recursos disponibles

$$(A_1, A_2, A_3, \ldots A_m)$$

Matriz de asignación actual

$$\begin{pmatrix} C_{11} & C_{12} & C_{13} & \ldots & C_{1m} \\ C_{21} & C_{22} & C_{23} & \ldots & C_{2m} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ C_{n1} & C_{n2} & C_{n3} & \ldots & C_{nm} \end{pmatrix}$$

Matriz de solicitudes

$$\begin{pmatrix} R_{11} & R_{12} & R_{13} & \ldots & R_{1m} \\ R_{21} & R_{22} & R_{23} & \ldots & R_{2m} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ R_{n1} & R_{n2} & R_{n3} & \ldots & R_{nm} \end{pmatrix}$$

Detección y recuperación

Detección de bloqueos con múltiples recursos por tipo

- Inicialmente todos los procesos están sin marcar
- Cuando el algoritmo termina, todos los procesos que no estén marcados habrán caído en un bloqueo mutuo.
  1. Buscar un proceso no marcado, $\mathbf{P_i}$ para el cual la i-ésima fila de **R** sea menor o igual que **A**
     - Proceso que puede ejecutarse hasta terminar
  2. Si se halla tal proceso, añadir la i-ésima fila de **C** a **A**, marcar el proceso y regresar al paso 1.
  3. Si no existe tal proceso, el algoritmo termina

Detección y recuperación

Detección de bloqueos con múltiples recursos por tipo

- Hay un estado de interbloqueo?

$$E = (\underset{r1}{4}, \underset{r2}{2}, \underset{r3}{3}, \underset{r4}{1})$$

$$A = (\underset{r1}{2}, \underset{r2}{1}, \underset{r3}{0}, \underset{r4}{0})$$

Matriz de asignación actual

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

Matriz de solicitudes

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

Interbloqueo?

Detección y recuperación

Detección de bloqueos con múltiples recursos por tipo

- Hay un estado de interbloqueo?

$$E = (\underset{r1}{4}, \underset{r2}{2}, \underset{r3}{3}, \underset{r4}{1})$$

$$A = (\underset{r1}{2}, \underset{r2}{1}, \underset{r3}{0}, \underset{r4}{0})$$

Matriz de asignación actual

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

Matriz de solicitudes

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

Interbloqueo? ……No hay interbloqueo

Detección y recuperación

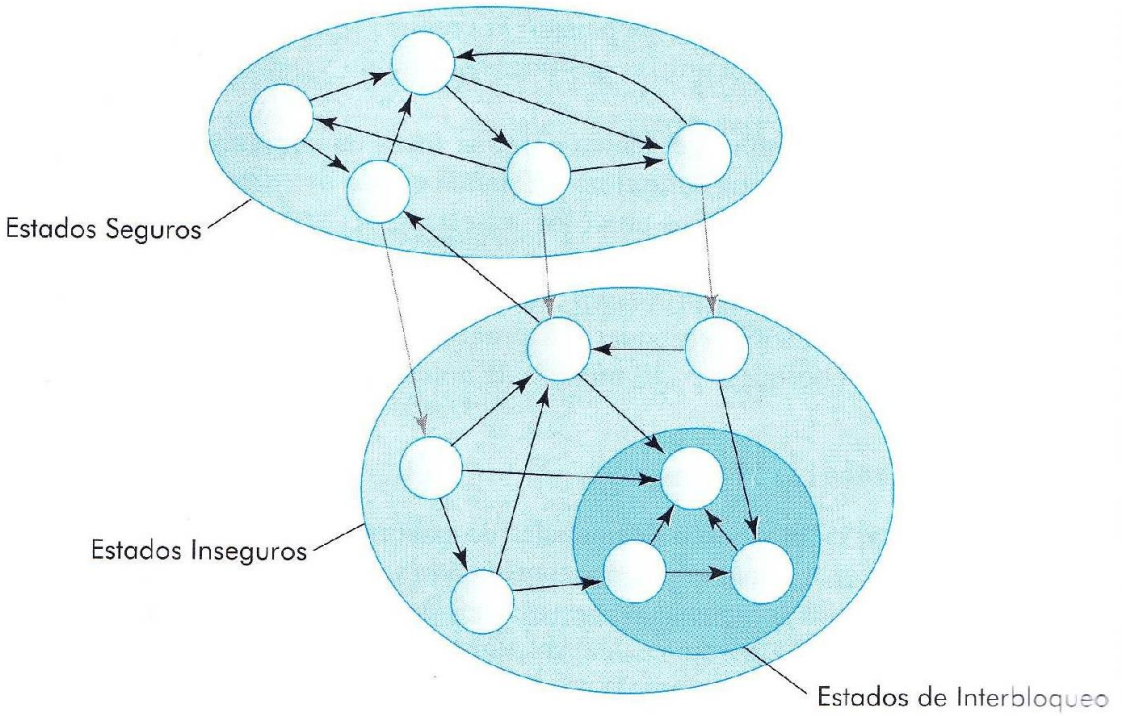
- **Recuperación de interbloqueo**
  - Recuperación mediante expropiación.
    - Depende de la naturaleza del recurso.
    - Puede requerir intervención manual (sistemas en lotes)
  - Recuperación mediante reversión.
    - Si probabilidad de que ocurra bloqueo mutuo es elevada.
    - Procesos pasan por puntos de verificación.
      - El estado del proceso se graba en un archivo: imagen de la memoria, recursos asignados.
  - Recuperación por eliminación de procesos.
    - Eliminar un proceso que pueda volver a ejecutarse desde el principio sin efectos perjudiciales.

Evitación

- Enfoque conservador de la asignación de recursos
- El sistema asigna un recurso sólo cuando hay la certeza de que no producirá un interbloqueo.
- Los procesos solicitan los recursos uno a la vez, estos se entregan siempre que sea seguro asignarlo.
- Se garantiza que el sistema debe de estar **Estado seguro**:
  - Si hay cierto orden de programación en el que se puede ejecutar cada proceso hasta completarse, incluso aunque todos ellos solicitaran de manera repentina su número máximo de recursos de inmediato

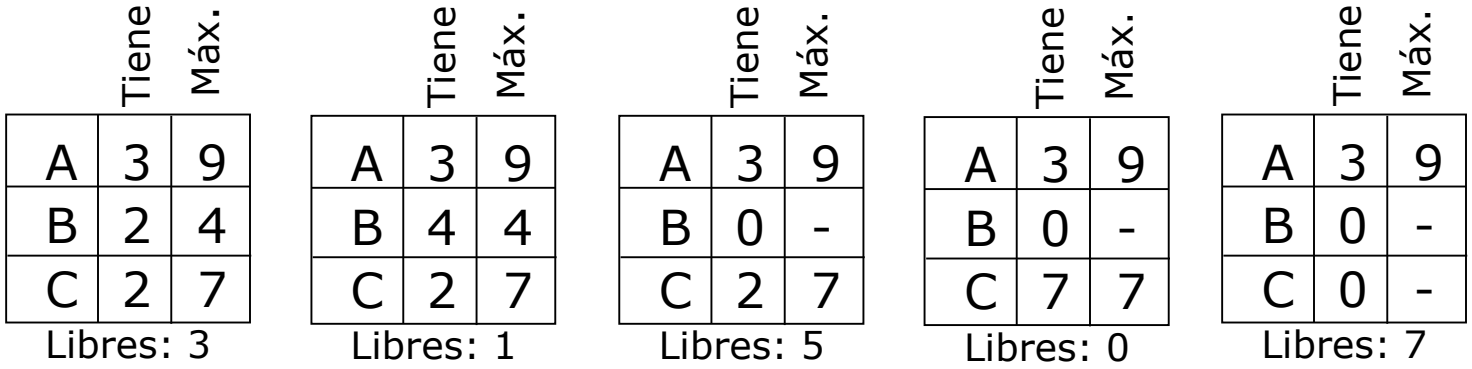
Evitación

- El gestor de recursos puede bloquear algunos procesos mientras otros utilizan su demanda máxima.
- Si un estado es inseguro, no significa que el sistema esté en un interbloqueo o que éste sea inminente

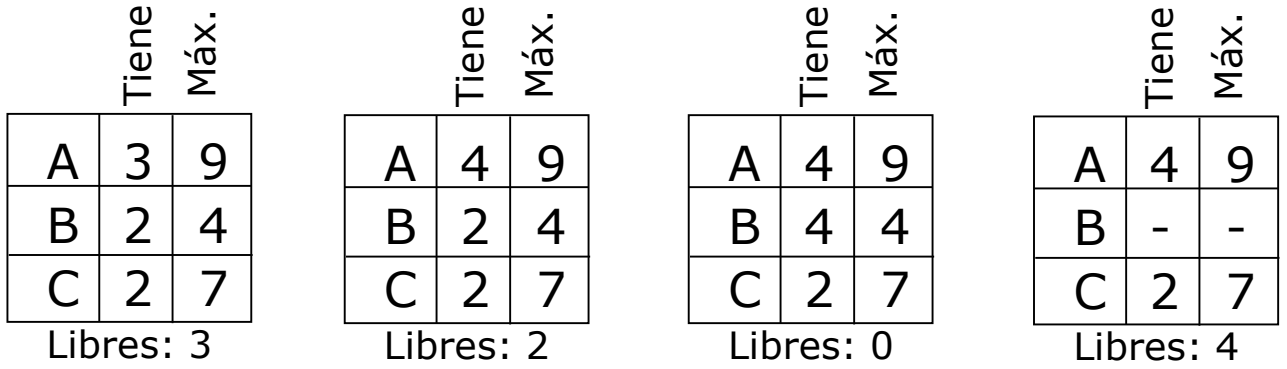


Evitación

- Estado seguro



- Estado inseguro



Evitación

- **Algoritmo del Banquero**
  - **Clientes** (procesos): se les concede líneas (unidades) de crédito.
  - Unidades de crédito: recurso
  - **Banquero** (SO): ente que les concede crédito
  - Algoritmo verifica si al conceder la solicitud conduce a un estado seguro o no.
  - El actual estado del sistema s_k, se puede definir determinando el número de unidades de cada tipo de recurso que tiene cada proceso.

Evitación

- Algoritmo del Banquero
  - **E_j**: cantidad de recursos del tipo j en el sistema
  - **C**: especifica la cantidad de recursos asignados a los procesos:

    C[i,j] → cantidad de recursos de tipo **R_j** asignados al proceso **P_i**
  - **R**: especifica los recursos que aún se necesitan

    R[i,j] → demanda máxima del recurso **R_j** por el proceso **P_i**
  - **A**: cantidad disponible de recursos en el sistema

    $$A[j] = E_j - \sum C[i,j]$$

Evitación

- **Algoritmo del Banquero para un recurso**

A = [10]

Procesos	asig	max
P. Andres	0	6
P. Bárbara	0	5
P. Miguel	0	4
P. Susana	0	7

A = [2]

Procesos	asig	max
P. Andres	1	6
P. Bárbara	1	5
P. Miguel	2	4
P. Susana	4	7

Se atiende la petición de Miguel?

A = [4]

Procesos	asig	max
P. Andres	1	6
P. Bárbara	1	5
P. Miguel	--	--
P. Susana	4	7

Estado Seguro

Evitación

- **Algoritmo del Banquero para un recurso**

disp = [2]

Procesos	asig	dmax
P. Andres	1	6
P. Bárbara	1	5
P. Miguel	2	4
P. Susana	4	7

Se atiende la petición de Barbara?

disp = [0]

Procesos	asig	dmax
P. Andres	1	6
P. Bárbara	2	5
P. Miguel	2	4
P. Susana	4	7

Estado Inseguro

Evitación

- **Algoritmo del Banquero para múltiples recursos**

dmax=

Procesos	Unidades de cinta	Graficadores	Impresoras	CD ROM
A	4	1	1	1
B	0	2	2	0
C	4	2	1	0
D	1	1	1	1
E	2	1	1	0

Asig =

Procesos	Unidades de cinta	Graficadores	Impresoras	CD ROM
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0
	5	3	2	2

Recursos que aún se necesitan

Procesos	Unidades de cinta	Graficadores	Impresoras	CD ROM
A	1	1	0	0
B	0	1	2	0
C	3	1	0	0
D	0	0	1	0
E	2	1	1	0

C = [6,3,4,2]

A = [5,3,2,2]

disp = [1,0,2,0]

Evitación

- **Algoritmo del Banquero para múltiples recursos**

Asig =

Procesos	Unidades de cinta	Graficadores	Impresoras	CD ROM
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

Procesos	Unidades de cinta	Graficadores	Impresoras	CD ROM
A	1	1	0	0
B	0	1	2	0
C	3	1	0	0
D	0	0	1	0
E	2	1	1	0

Recursos que aún se necesitan

C = [6,3,4,2]

A = [5,3,2,2]

disp = [1,0,2,0]

- Si B solicita una impresora? ***estado seguro*** **(D,A o E, etc.)**
- Si después E quiere la última impresora → **(1,0,0,0)→ bloqueo irreversible**

Evitación

- **Algoritmo del Banquero para múltiples recursos**

1. Buscar una fila R, cuyas necesidades de recursos no satisfechas sean menores o iguales que A. Si no existe, se tiene un interbloqueo
2. Suponer que el proceso seleccionado de la fila solicita todos los recursos que necesita y termina. Marcar ese proceso como terminado
3. Repetir los pasos 1 y 2 hasta que todos los procesos se marquen como terminados o hasta que no haya ningún proceso cuyas necesidades de recursos se puedan satisfacer.

Prevención

- Consiste en asegurar que al menos una de las cuatro condiciones (exclusión mutua, poseer y esperar, espera circular y no apropiación) siempre será falsa.
  - Exclusión mutua: evitar asignar un recurso si no es absolutamente necesarios, y asegurarse de que el número de procesos que pueden solicitarlo sea lo más pequeño posible.
    - No es posible hacerlo para todos los recursos

Prevención

- Poseer y esperar
  1. Exigir que los procesos soliciten todos sus recursos al crearse
     - Procesos en lotes
     - Otros procesos
       - No saben que utilizaran apriori
     - Utilización pobre de los recursos
     - Puede haber inanición
  2. Exigir que un proceso libere todos los recursos que posee antes de solicitar otros
     - Se sobrecarga al sistema al salvar el estado de los recursos a cada solicitud.
     - Infrautilización de los recursos.
     - Puede conducir a la inanición

Prevención

- Permitir apropiación
  - Virtualización de dispositivos
    - No es posible en todos los dispositivos
  - SO permita que un proceso se retracte de su solicitud
    - Si el recurso no esta disponible
    - No se garantiza que sea efectiva
    - Puede ocurrir el ***interbloqueo activo (livelock)***

Prevención

- Espera circular
  - Establecer un orden total sobre los recursos del sistema.
  - Solo se concede los recursos en estricto orden:
    - Si un proceso posee el recurso `j` y solicita el recurso `i`, solo se concederá si `i>j.`
    - Donde `j` es el recurso de mayor orden
  - Si no, el proceso debe de liberar los recursos que posee y volver a solicitarlos en orden.
    - Sobrecarga del sistema

1. Impresora
2. Escáner
3. Trazador
4. Unidad de cinta
5. Unidad de CD-ROM