Sistemas Operativos

Prevención de interbloqueos

Departamento de Ingeniería en Sistemas y Computación
Universidad Católica del Norte, Antofagasta.

Modelo del sistema

- El interbloqueo es un problema que afecta a procesos concurrentes que utilizan recursos en un sistema.
- Los procesos solicitan recursos al sistema y los liberan cuando ya no los necesitan. Un recurso puede estar disponible o asignado a algún proceso.

Modelo del sistema

- El interbloqueo es un problema que afecta a procesos concurrentes que utilizan recursos en un sistema.
- Los procesos solicitan recursos al sistema y los liberan cuando ya no los necesitan. Un recurso puede estar disponible o asignado a algún proceso.

Modelo del sistema

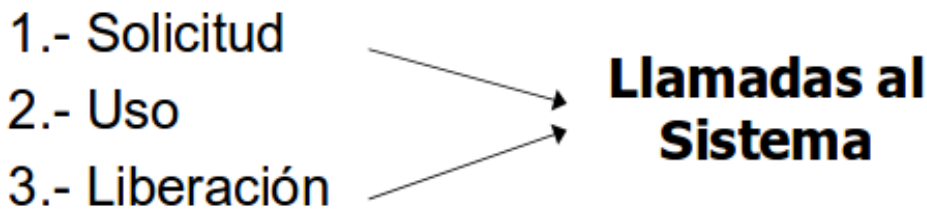
- Ejemplares: Puede haber varios ejemplares de un mismo tipo de recurso (varias impresoras). En este caso, cuando un proceso solicita un recurso, se le concede `cualquiera` de los ejemplares que estén disponibles.
- Si un proceso solicita un recurso que no tiene ejemplares disponibles, el proceso queda bloqueado, esperando hasta que se le asigna un ejemplar.

Modelo del sistema

- Ejemplares: Puede haber varios ejemplares de un mismo tipo de recurso (varias impresoras). En este caso, cuando un proceso solicita un recurso, se le concede `cualquiera` de los ejemplares que estén disponibles.
- Si un proceso solicita un recurso que no tiene ejemplares disponibles, el proceso queda bloqueado, esperando hasta que se le asigna un ejemplar.

Modelo del sistema

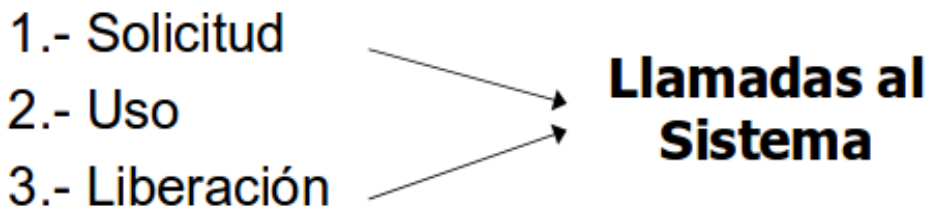
- Esquema de funcionamiento normal



- ¿Cómo saber en qué estado están los recursos?
  - Tabla de sistema podría registrar si cada recurso está libre o asignado, y si un recurso está asignado, a qué proceso se le asignó.

Modelo del sistema

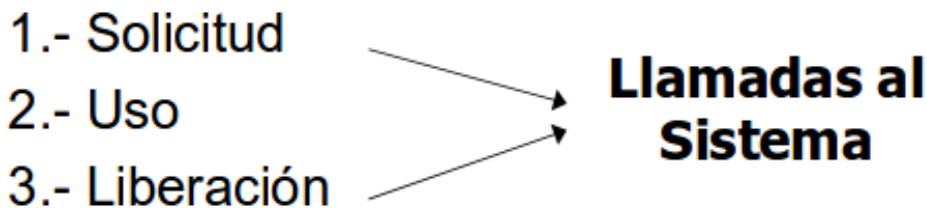
- Esquema de funcionamiento normal



- ¿Cómo saber en qué estado están los recursos?
  - Tabla de sistema podría registrar si cada recurso está libre o asignado, y si un recurso está asignado, a qué proceso se le asignó.

Modelo del sistema

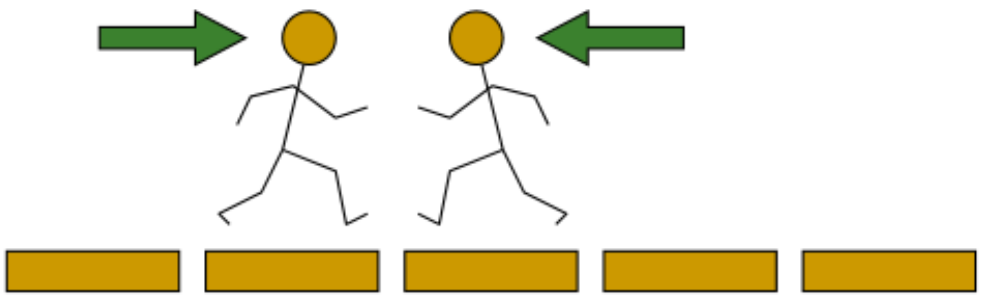
- Esquema de funcionamiento normal



- ¿Cómo saber en qué estado están los recursos?
  - Tabla de sistema podría registrar si cada recurso está libre o asignado, y si un recurso está asignado, a qué proceso se le asignó.

El problema

- Un conjunto de procesos bloqueados, cada uno de ellos esperando por un recurso que retiene otro proceso de ese conjunto.
  - ningún proceso del conjunto puede avanzar
  - interbloqueo, bloqueo mutuo, deadlock, abrazo mortal



De quién es la culpa?

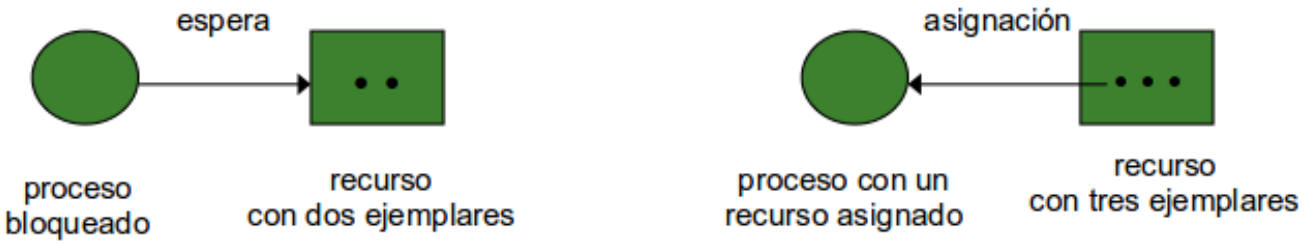
- Muchas veces, el interbloqueo no es responsabilidad de las aplicaciones sino del sistema de gestión de recursos.
- Ejemplo: los procesos A y B se pueden interbloquear, aunque estén escritos correctamente.

Proceso A	**Proceso B**
Pide(escáner)	Pide(impresora)
Pide(impresora)	Pide(escáner)
usa impr. y escáner	usa impr. y escáner
Libera(impresora)	Libera(escáner)
Libera(escáner)	Libera(impresora)

Grafo de asignación de recursos

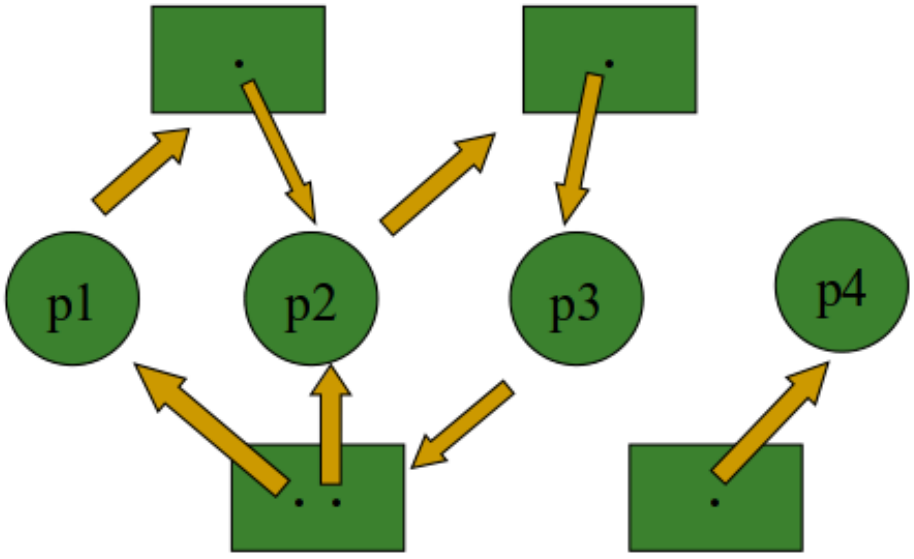
Sirve para representar el estado de un sistema de asignación de recursos. Muestra esta información:

- cuántos ejemplares hay de cada tipo de recurso
- los procesos activos en el sistema
- qué recursos están asignados y a qué proceso
- qué procesos están bloqueados y por cuáles recursos



Ejemplo

P1, P2 y P3 están en interbloqueo



Modelado del interbloqueo

Representación matricial

- Se utiliza una matriz de solicitud S, una matriz de asignación A y un vector E de recursos existentes en el sistema.
- Siendo p el número de procesos y r el número de recursos, entonces:
  - A es de tamaño $p \times r$, donde $A[i, j]$ especifica cuantas unidades del recurso j están asignadas al proceso i
  - S es de tamaño $p \times r$, donde $S[i, j]$ especifica cuantas unidades del recurso j está esperando el proceso i
  - E es de tamaño r, donde $E[i]$ especifica cuantas unidades del recurso i existen

Modelado del interbloqueo

Representación matricial

- Se utiliza una matriz de solicitud S, una matriz de asignación A y un vector E de recursos existentes en el sistema.
- Siendo p el número de procesos y r el número de recursos, entonces:
  - A es de tamaño $p \times r$, donde $A[i, j]$ especifica cuantas unidades del recurso j están asignadas al proceso i
  - S es de tamaño $p \times r$, donde $S[i, j]$ especifica cuantas unidades del recurso j está esperando el proceso i
  - E es de tamaño r, donde $E[i]$ especifica cuantas unidades del recurso i existen

Modelado del interbloqueo

Representación matricial

- Se utiliza una matriz de solicitud S, una matriz de asignación A y un vector E de recursos existentes en el sistema.
- Siendo p el número de procesos y r el número de recursos, entonces:
  - A es de tamaño $p \times r$, donde $A[i, j]$ especifica cuantas unidades del recurso j están asignadas al proceso i
  - S es de tamaño $p \times r$, donde $S[i, j]$ especifica cuantas unidades del recurso j está esperando el proceso i
  - E es de tamaño r, donde $E[i]$ especifica cuantas unidades del recurso i existen

Modelado del interbloqueo

Representación matricial

- Se utiliza una matriz de solicitud S, una matriz de asignación A y un vector E de recursos existentes en el sistema.
- Siendo p el número de procesos y r el número de recursos, entonces:
  - A es de tamaño $p \times r$, donde $A[i, j]$ especifica cuantas unidades del recurso j están asignadas al proceso i
  - S es de tamaño $p \times r$, donde $S[i, j]$ especifica cuantas unidades del recurso j está esperando el proceso i
  - E es de tamaño r, donde $E[i]$ especifica cuantas unidades del recurso i existen

Modelado del interbloqueo

Ejemplo de representación matricial y múltiples unidades por recurso

- Ejecución de 3 procesos con 3 recursos R_1 (2), R_2 (3) y R_3 (2)
  1. P_1: solicita(R_1[2]) → solicita 2 unidades
  2. P_2: solicita(R_2[1])
  3. P_2: solicita(R_1[1]) → se bloquea
  4. P_3: solicita(R_2[1])
  5. P_3: solicita(R_2[1])
  6. P_1: solicita(R_2[1], R_3[2]) → se bloquea

- Matriz resultante:

$$A=\begin{bmatrix}2 & 0 & 0\\ 0 & 1 & 0\\ 0 & 2 & 0\end{bmatrix} \quad S=\begin{bmatrix}0 & 1 & 2\\ 1 & 0 & 0\\ 0 & 0 & 0\end{bmatrix} \quad E=[2\ 3\ 2] \quad D=[0\ 0\ 2]$$

Condiciones para el interbloqueo

Si en un sistema se produce una situación de interbloqueo, entonces se cumplen simultáneamente estas cuatro condiciones:

- **Exclusión mutua**. Los recursos no se pueden compartir.
- **Retención y espera.** Un proceso que retiene uno o varios recursos se encuentra esperando por recursos asignados a otros procesos.
- **No expropiación.** Un recurso sólo puede ser liberado por el proceso que lo retiene, voluntariamente.
- **Espera circular.** Existe una serie de procesos en espera {Po,P1,...Pn} en la que todo Pi espera por un recurso retenido por Pi+1; y Pn espera por un recurso retenido por Po.

Tratamiento del interbloqueo

- Garantizar que en el sistema nunca ocurren interbloqueos
  - **prevención:** diseñar el sistema de manera que nunca se cumpla alguna de las cuatro condiciones del interbloqueo.
  - **evitación:** tratar de no caer nunca en un estado de interbloqueo.
- Permitir la aparición de interbloqueos y recuperarse cuando ocurran
  - necesitamos un sistema de detección y un mecanismo de recuperación
- No tratar el problema
  - si hay interbloqueos, el usuario tiene que intervenir

Prevención del interbloqueo

Se trata de eliminar la aparición de alguna de las cuatro condiciones necesarias para el interbloqueo.

- **Exclusión mutua.** Depende de la naturaleza del recurso, así que esta condición no se puede eliminar.

Prevención del interbloqueo

Retención y espera. Hay que garantizar que un proceso no pueda quedar bloqueado si retiene algún recurso. ¿Cómo conseguirlo?

- el proceso tiene que pedir todos sus recursos de una vez, p.ej. antes de empezar a ejecutarse
  - efecto negativo: muchos recursos retenidos pero no usados,
- un proceso sólo puede solicitar recursos cuando no tiene ninguno asignado
  - Efecto negativo: puede ocurrir que tengamos que liberar un recurso y volver a pedirlo para poder solicitar otros recursos
- En ambos caso puede que un proceso nunca se ejecute (**inanición**)

Prevención del interbloqueo

No expropiación. Permitir que el S.O. desasigne recursos a un proceso bloqueado.

- Si un proceso se bloquea por un recurso, los recursos retenidos quedan a disposición de los procesos activos
- El proceso bloqueado tiene ahora que esperar por *todos* los recursos
- penaliza a los procesos que necesitan muchos recursos
- Es posible seguir este protocolo en recursos cuyo estado se puede guardar fácilmente y después restaurarse (registros de CPU, espacio de memoria, ...). Generalmente no puede aplicarse a recursos tales como impresoras y unidades de cinta

Prevención del interbloqueo

Espera circular. Se puede evitar forzando un orden en la petición de los recursos.

- Cada recurso tiene asignado un número de orden
- Los recursos se deben pedir en orden ascendente
- Aconsejable: el orden de petición de los recursos se establezca según el orden de uso normal de los recursos de un sistema
- Efectos negativo
  - se limita la libertad de escritura de código
  - se puede inducir a una mala utilización de los recursos

Evasión: Algoritmo del banquero

Se trata de conceder los recursos sólo cuando no representen un riesgo futuro de interbloqueo.

Lo procesos han de declarar por anticipado la cantidad máxima de recursos que van a utilizar a lo largo de su vida

Estado seguro: un estado en el cual no hay riesgo inminente de interbloqueo. Un estado es seguro si en él podemos encontrar una **secuencia segura** con todos los procesos del sistema

$\{P_1, P_2, ..., P_N\}$ es una **secuencia segura** si los recursos que Pi puede pedir en el peor caso se pueden atender con lo que hay disponible más los recursos poseídos por todos los procesos $P_{j,j<i}$

Sólo concedemos recursos si el estado resultante tras la petición es seguro

Algoritmo del banquero

¿ Qué significa una secuencia segura ?

- Nos ponemos en el peor caso del sistema: que todos los procesos soliciten al mismo tiempo el máximo de recursos a los que tiene derecho
- El primer proceso de la secuencia es uno que podría finalizar en ese peor caso, con los recursos disponibles en el sistema
- El segundo proceso es uno que puede finalizar con lo que hay disponible más los recursos que liberaría el primer proceso
- De la misma forma, los siguientes procesos pueden finalizar con los recursos que han liberado los anteriores en la secuencia
- Y si todos los procesos pueden terminar, es que no hay interbloqueo

Algoritmo del banquero

Cuando un proceso realiza una petición, el SO calcula si tras conceder los recursos el sistema pasa a un estado seguro

- si el nuevo estado es seguro, se concede la petición
- si el nuevo estado no es seguro, el proceso queda bloqueado (aunque existan recursos suficientes para atender la petición). La petición se concede cuando se observa que no hay riesgo de interbloqueo

Ejemplo

- Dos recursos R1 y R2, con 5 y 6 ejemplares
- En el instante actual quedan libres 1 y 1 ejemplares

	Asignado	Máximo	Necesidades
Pa	1 0	2 2	1 2
Pb	1 3	3 4	2 1
Pc	2 2	3 2	1 0

- El estado es seguro porque existe la secuencia segura {Pc, Pa, Pb}
- ¿ Qué pasa si Pa pide un ejemplar de R1 y se lo damos ?
  - El sistema quedará en un estado inseguro

Ejercicio 1

Existen un sistema con un total de dispositivos del sistema: 12
Existen los siguientes trabajos:

	Dispositivos		
No. Trabajo	Asignados	Máximo requerido	Necesidades Restantes
1	2	4	2
2	3	5	2
3	5	8	3

Total dispositivos asignados: 10
Dispositivos del sistema: 12
Total dispositivos disponibles: 2

Ejercicio 2

Existe un sistema con un total de dispositivos del sistema: 12

Existen los siguientes trabajos:

	Dispositivos		
No. Trabajo	Asignados	Máximo requerido	Necesidades Restantes
1	5	6	1
2	4	7	3
3	2	6	4

Total dispositivos asignados: 11

Dispositivos del sistema: 12

Total dispositivos disponibles: 1

Ejercicio 3

1. Existen un sistema con un total de dispositivos del sistema: 19
2. Existen los Siguientes trabajos:

Trabajo / N°	Dispositivos Asignados	Maximo Requeridos	Necesidades Restantes
1	8	10	2
2	4	8	4
3	2	3	1

Total de Dispositivos En el Sistema: 19
Total de Dispositivos Asigados: 14
Dispositivos Disponibles: 19-14 =5