

# clase 10- Deadlocks

miércoles, 4 de junio de 2025 18:07

Cuando cada uno de los procesos está esperando por un recurso que está usando otro proceso  
Puede darse deadlocks por recursos físicos y lógicos.

Apropiativos: memoria, cpu, se le puede sacar al proceso sin efectos dañinos.

No apropiativos: cd, impresora, se va a romper si le sacas al proceso

Normal:

Proceso solicita, usa y libera dispositivo

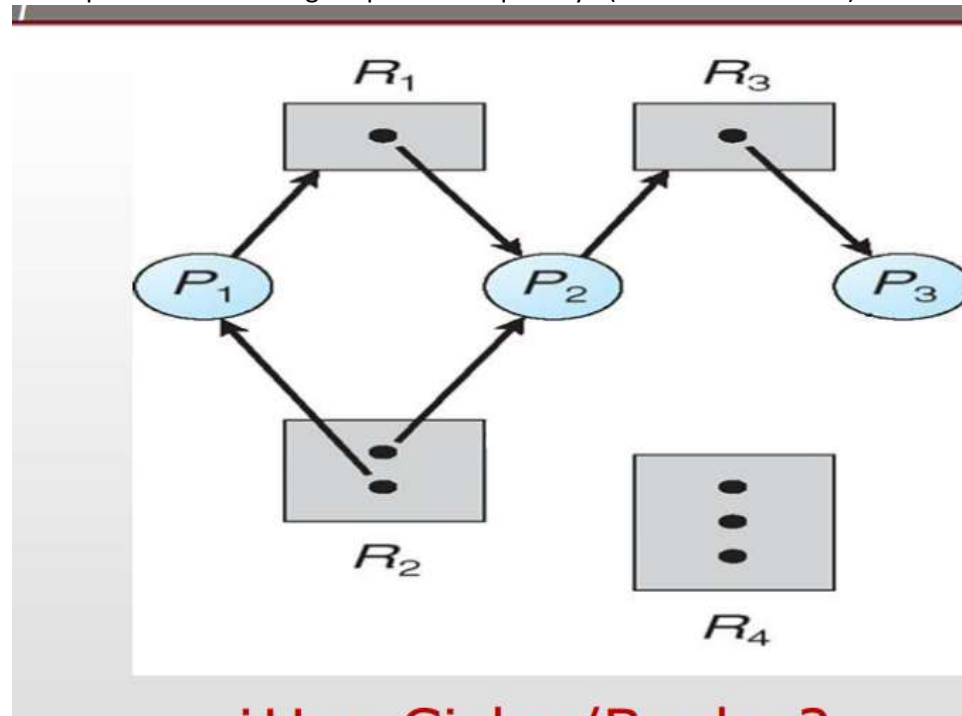
Si está ocupado, se queda esperando

- ✓ Para poder representar la asignación de recursos, se utiliza un **Grafo de Asignación de recursos**.
- ✓ El grafo **permite visualizar el estado** de los recursos del sistema y procesos en un momento determinado.
- ✓ Cada **Proceso o recurso es representado por un nodo**. Un recurso con varias **instancias** posee mas de un "Punto" dentro de nodo.
- ✓ Una **arista representa una relación entre un Proceso y un Recurso**.  
Notar que las aristas **son dirigidas** y dependiendo de la dirección indican distintos estados

(a) Resource is requested

(b) Resource is held

Concepto de estado inseguro puede ser que haya (si no libera nunca 3)



- ❑ Si el grafo no contiene ciclos → NO hay interbloqueo
- ❑ Si el grafo contiene un ciclo:
  - ❑ Si sólo hay una instancia por tipo de recurso → SI hay interbloqueo
  - ❑ Si hay varias instancias por tipo de recurso → hay posibilidad de deadlock.

Error común: espera circular: todos los procesos involucrados estén en una ronda

#### Condiciones:

1. **Exclusión mutua:** En un instante de tiempo dado, solo un proceso puede utilizar una instancia de un recurso
2. **Retención y espera:** Los procesos deben mantener los recursos asignados y esperar por la asignación de los nuevos requeridos
3. **No apropiación:** Los recursos no pueden ser quitados a un proceso que actualmente los posea
4. **Espera circular:** El proceso forma parte de una lista circular en la que cada proceso de la lista está esperando por al menos un recurso asignado a otro proceso de la lista

**Para estar en presencia de un Deadlock se deben cumplir todas!**

Protocolo para asegurar que nunca entra en deadlock:

Prevenir: que alguna de las condiciones no se cumpla

Evitar: tomar decisiones de asignación en base al estado del sistema. Da los recursos de manera tal que evite deadlock

Se ocupa cpu. Es un tema pq tiene que hacerlo por software es uso de recursos. Cpu no haciendo cosas productivas.

Permitir el estado de deadlock, pero si hay hace algoritmo de recuperación.

Ignorar y rezar para que no haya un deadlock

Prevention (prevenir la formación del interbloqueo):

Que por lo menos una de las condiciones no pueda mantenerse.

☑ Avoidance (evitar la formación del interbloqueo): ➤ Asignar cuidadosamente los recursos, manteniendo información actualizada sobre requerimiento y uso de recursos.

**Prevención 1: Exclusión mutua:** En un instante de tiempo dado, solo un proceso puede utilizar una instancia de un recurso

☑ Si ningún recurso se asignara de manera exclusiva (no siempre se puede), no habría interbloqueo.

☑ Considerar que hay recursos compartibles (archivos read only, memoria) y no compartibles (impresora).

Encolamiento para que el recurso no copartible se maneje de una.

**Prevención 2: Retención y espera:** Los procesos deben mantener los recursos asignados y esperar por la asignación de los nuevos requeridos

Si un proceso quiere un recurso no disponible, tiene que liberar otros.

Dos opciones:

O pide y reserva todos los recursos antes de empezar a ejecutar

O hago que solo pueda requerir recursos cuando no tiene ninguno

Usa mal los recursos, es una gaceta

**Prevenir 3: No apropiación:** Los recursos no pueden ser quitados a un proceso que actualmente los

posea

No siempre se puede atacar porque no siempre se le puede sacar un recurso a un proceso  
Accede a un demonio que lo administra. El proceso ve como si lo tuviera

**Prevenir 4: Espera circular:** El proceso forma parte de una lista circular en la que cada proceso de la lista está esperando por al menos un recurso asignado a otro proceso de la lista.

Se ordena los recursos bajo una función

Unas cuantas diapos en el medio importante jiji

Algoritmo del banquero: consume muchísimos recursos, demasiado pesado.

Busca secuencia segura para la asignación

Procesos declaran la max cant de instancias de cada recurso que necesitan

Se decide en que momento asignarlos, garantizando estado seguro

	R1	R2	R3
P1	1	0	0
P2	6	1	2
P3	2	1	1
P4	0	0	2
Asignación			

	R1	R2	R3
P1	3	2	2
P2	6	1	3
P3	3	1	4
P4	4	2	2
Max			

	R1	R2	R3
P1	2	2	2
P2	0	0	1
P3	1	0	3
P4	4	2	0
Necesidad			

R1	R2	R3
0	1	1
Disponible		

Asignación: cuantas instancias del recurso tiene ahora

Max: cuantas necesita para ejecutarse

Necesidad: la resta

Le doy los recursos al proceso que puede ejecutar (al 2 en la imagen).

Una vez que p2 termina, devuelve los recursos (los de max)

Está en estado inseguro pq demanda es mayor a lo que tengo

**Detección y recuperación:**

a