Sistemas Operativos

Deadlocks – II
(Interbloqueos)

Sistemas Operativos

- ☑ Versión: Mayo 2019
- ☑ Palabras Claves: Deadlock, Bloqueo, Procesos, Recursos, Inanición

Algunas diapositivas han sido extraídas de las ofrecidas para docentes desde el libro de Stallings (Sistemas Operativos) , el de Silberschatz (Operating Systems Concepts)

Repaso…

☑ **La Clase pasada vimos que existen distintos enfoques con el fin de llevar adelante el manejo de Deadlocks:**

1. **Usar un protocolo que asegure que NUNCA se entrará en estado de deadlock**
   - 1.1 **Prevenir:** Que no se cumple alguna de las 4 condiciones
   - 1.2 **Evitar:** Tomar decisiones de asignación en base al estado del sistema
2. **Permitir el estado de deadlock y luego recuperar**
3. **Ignorar el problema y esperar que nunca ocurra un deadlock**

Repaso…

- ☑ La Clase pasada vimos que existen distintos enfoques con el fin de llevar adelante el manejo de Deadlocks:

1. Usar un protocolo que asegure que NUNCA se entrará en estado de deadlock
   1.1 **Prevenir:** Que no se cumple alguna de las 4 condiciones
   1.2 **Evitar:** Tomar decisiones de asignación en base al estado del sistema
2. Permitir el estado de deadlock y luego recuperar
3. Ignorar el problema y esperar que nunca ocurra un deadlock

- **Tarea: Investigar cuál de los 3 enfoques utilizan Windows y GNU/Linux:**
  - **Se deberá presentar un informe que explique brevemente los 3 enfoques**
  - **Deberá explicar el/los adoptados por cada SO, si son o no configurables e indicar las fuentes de donde obtuvo la información**

Repaso…

- ***Prevenir**: que por lo menos una de las condiciones no pueda mantenerse. Se imponen restricciones en la forma en que los procesos REQUIEREN los recursos. **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**: En la gráfica de asignación de recursos debe cerrarse un bucle en el grado.

Repaso…

- ☑ Analizamos también algoritmos para evitar el deadlock
  - ☑ Grafo de asignación de recursos
  - ☑ Algoritmo del banquero
- ☑ Recordar: Prevenir <> Evitar
  - ☑ **Prevenir**: que por lo menos una de las condiciones no pueda mantenerse. Se imponen restricciones en la forma en que los procesos REQUIEREN los recursos.
  - ☑ **Evitar**: asignar cuidadosamente los recursos, manteniendo información actualizada sobre requerimiento y uso de recursos

Detección y Recuperación

- En esta clase nos vamos a enfocar en distintas técnicas que nos permitan **determinar** si ocurrió un deadlock, y en tal caso poder **recuperarse** del mismo. Para ello vamos a contar con:
  1. Algoritmo que examine si ocurrió un deadlock
  2. Algoritmo para recuperación del deadlock

1. Detección

- ☑ Con recursos con una sola instancia
  - ✓ Análisis del grafo de asignación
- ☑ Con recursos de varias instancias
  - ✓ Algoritmo del Banquero

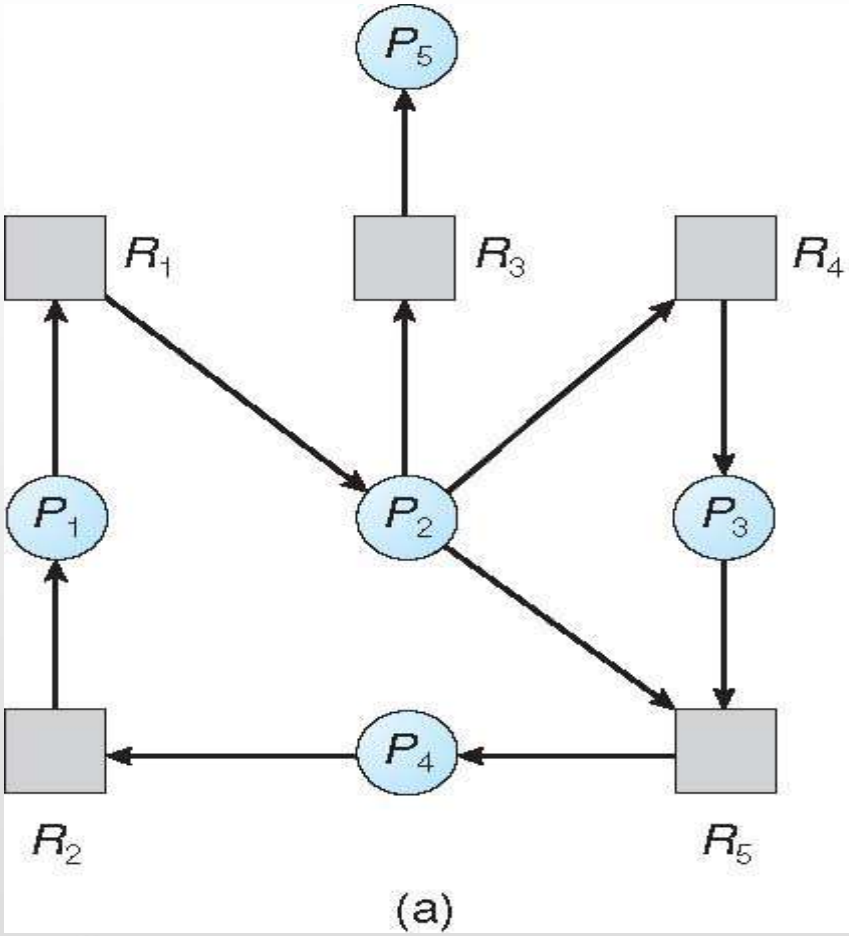
Detección. Instancia única de cada tipo de recurso

- **Mantener** un **grafo wait-for (Grafo de espera)**
  - **Nodos = Procesos**
  - $P_i \rightarrow P_j$, si P_i ***espera que*** P_j **libere recurso Rq**

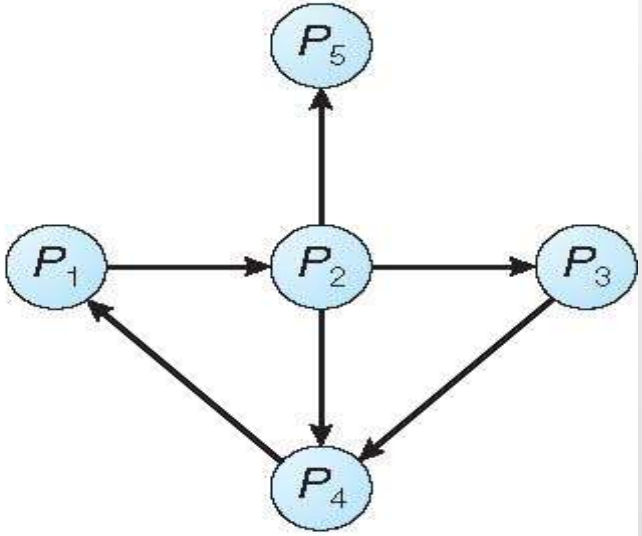
- **El algoritmo se basa en buscar periódicamente** un **ciclo** en el grafo.

Si hay un **ciclo, existe** un **bloqueo**

Detección. Instancia única de cada tipo de recurso



(a)



(b)

Detección. Multiples Instancias de cada tipo de recurso

- Se utiliza el Algoritmo del Banquero
- Se utilizan distintas estructuras para seguir un algoritmo.
- Disponible: Un vector de longitud *m* que indica el número de recursos disponibles de cada tipo
- Asignación: Una matriz *n* x *m* que define el número de recursos de cada tipo asignado actualmente a cada proceso
- Solicitud: Una matriz *n* x *m* indica la petición actual de cada proceso. Si Request [i][j] = *k* , entonces el proceso P_i está solicitando *k* instancias más del tipo de recurso R_j. (El proceso se encuentra en estado de Espera)

Ejemplo. Detección con múltiples instancias

5 procesos P(0)....P(4), 3 recursos A, B, C. A tiene 7 instancias, B tiene 2 y C tiene 6.

	Asignación			Requerimiento			Disponible		
	A	B	C	A	B	C	A	B	C
P(0)	0	1	0	0	0	0	0	0	1
P(1)	2	0	0	2	0	2			
P(2)	3	0	2	0	0	0			
P(3)	2	1	1	1	0	0			
P(4)	0	0	2	0	0	2			

Aprovechemos a repasar…

Decimos que el sistema no está en estado de deadlock, si ejecutamos nuestro algoritmo y encontramos una secuencia segura p(0), p(2), p(3), p(1), p(4).

Ejemplo. Detección con múltiples instancias

5 procesos P(0)....P(4), 3 recursos A, B, C. A tiene 7 instancias, B tiene 2 y C tiene 6.

	Asignación			Requerimiento			Disponible		
	A	B	C	A	B	C	A	B	C
P(0)	0	1	0	0	0	0	0	0	1
P(1)	2	0	0	2	0	2			
P(2)	3	0	2	0	0	**2**			
P(3)	2	1	1	1	0	0			
P(4)	0	0	2	0	0	2			

Aprovechemos a repasar…

Si ahora **modificamos** el **requerimiento de P(2**) para el **recurso C** de modo que pase a pedir 4 instancias (es decir que P(2) pida 2 instancias más de C) el **sistema estará en deadlock.**

A pesar de poder utilizar los recursos asignados a P(0) **no alcanza** para **satisfacer** los **requerimientos** de los **otros procesos**. Luego **existe un deadlock** por los **procesos p(1), p(2), p(3) y p(4).**

Cuando Uso del Algoritmo de Detección

Cuando, y **con qué frecuencia**, invocar **depende** de:

- ¿Con qué **frecuencia** es **probable** que **ocurra** un deadlock?
- ¿**Cuántos procesos tendrán** que **ser revertidos** (rolled back)?
- La **frecuencia** con la que se **debe ejecutar** el **algoritmo** de **detección** de **interbloqueos**, **suele** ser un **parámetro** del **sistema**.
- Los motores de BD también poseen algoritmos de detección de Deadlocks. Por ejemplo SQLserver lo corre cada 5 segundos

Cuando Uso del Algoritmo de Detección

- Una **posibilidad extrema** es **comprobar** el estado **cada vez** que se solicita un recurso y estos **no pueden ser asignados**. **Consume mucha CPU**
- Si el **algoritmo** de **detección** se **invoca arbitrariamente**, puede **haber muchos ciclos** en el **gráfico** de **recursos** y, por lo tanto, **no podríamos** decir cuál de los **muchos procesos bloqueados** "**causó**" el **bloqueo**.
- Hay que encontrar un término medio

Cuando Uso del Algoritmo de Detección

- Otra **alternativa** es **activar** el **algoritmo ocasionalmente**, a **intervalos regulares** o cuando **uno** o **más procesos** queden **bloqueados** durante un **tiempo** sospechosamente **largo**.
- Por ejemplo **1 hora**, o **cuando** la **utilización** del **Procesador desciende** su actividad por **debajo** de un determinado **porcentaje** de **utilización**, pues un '**deadlock**' eventualmente **disminuye** el **uso** de **procesos** en actividad.

Recuperación frente al deadlock

Cuando un algoritmo de detección determina que existe un interbloqueo, existen varias alternativas para tratar de eliminarlo:

☑**Caso 1:** El **Operador** lo puede **resolver manualmente**. (Informar al operador del SO)

☑**Caso 2:** Esperar que el **sistema** se **recupere automáticamente** del **deadlock**. **El Sistema rompe** el deadlock:

☑En cualquiera de los casos, las alternativas son:

- **Abortar** 1 o más **Procesos** para **romper ciclo**
- **Expropiar recursos** a 1 o más **Procesos del ciclo**

Cómo Recuperarse frente al deadlock

- ☑ Violar la exclusión mutua y asignar el recurso a varios procesos → No siempre se puede
- ☑ Cancelar los procesos suficientes para romper la espera circular
- ☑ Expropiar recursos de los procesos que se presume están en estado de deadlock.

Recuperación frente al deadlock

Para eliminar el deadlock matando procesos pueden usarse 2 métodos:

☑ Matar todos los procesos en estado de deadlock. Simple pero a un muy alto costo.

☑ Matar de a un proceso por vez hasta eliminar el ciclo. Este método requiere considerable overhead ya que por cada proceso que vamos eliminando se debe re-ejecutar el Algoritmo de Detección para verificar si el deadlock efectivamente desapareció o no.

Recuperación frente al deadlock

TERMINACIÓN DE PROCESOS

- Puede **no** ser **fácil terminar** un **proceso**
  - (p.e. si se encuentra **actualizando** un **archivo**).
- Se presenta un nuevo **problema** de política o **decisión**: **selección de la víctima**
- Se debe **seleccionar** aquel **proceso** cuya **terminación represente** el **costo mínimo para el sistema**.

Recuperación Por apropiación

- Selección de procesos víctima: ¿a qué proceso le saco los recursos? ¿cuáles recursos? ¿A qué proceso se lo asigno?
- Puede no ser fácil terminar un proceso por ejemplo si se encuentra actualizando un archivo.
- Puede haber intervención "manual" (por operador).
- Se puede aplicar según la naturaleza del recurso (si es RO o RW por ejemplo)

Recuperación: Criterios para elegir proceso "víctima"

¿En qué orden debemos optar por abortar/terminar?

- ☑ **Prioridad más baja**.
- ☑ **Menor cantidad** de **tiempo** de **CPU** hasta el momento.
- ☑ **Mayor tiempo restante estimado para terminar**.
- ☑ **Menor cantidad** de **recursos asignados** hasta ahora.
- ☑ ¿Hay **recursos** del **proceso** que **deben completarse?**
- ☑ ¿**Cuántos procesos tendrán** que ser **terminados**?
- ☑ ¿El **proceso** es **interactivo** o **batch**? Puede **volver atrás**?

Ideal: **elegir** un **proceso** que se **pueda volver** a **ejecutar sin problemas** (ej. una compilación)

Recuperación frente al deadlock

EXPROPIACIÓN DE RECURSOS

Quitar sucesivamente los **recursos** de los **procesos** que los tienen **y asignarlos a otros** que los **solicitan** hasta **conseguir romper** el **interbloqueo**.

Hay que **considerar** tres aspectos:

- **Apropiación/Selección de Víctima - minimizar el costo**
- **Rollback - regresar** a algún **estado seguro**, **reiniciar** el **proceso** para **ese estado**
- **Inanición** - el **mismo proceso siempre** se puede **escoger** como **víctima**. **Asegurar** que se **elige** un **numero finito** de **veces**, **Incluir** el **número** de **retrocesos** como **factor** de **costo**

Retroceso (Rollback)

- ☑ Volver hacia una instancia segura el proceso que le sacamos los recursos.
- ☑ Establecer puntos de comprobación (check points)
- ☑ Información asociada a esos puntos:
  - ➢ Imagen de la memoria
  - ➢ Recursos asignados en ese momento
- ☑ No sobreescribir check points anteriores

Pasos en recuperación con rollback

1. Detectar el interbloqueo
2. Detectar recursos que se solicitan
3. Detectar qué procesos tienen esos recursos
4. Volver atrás antes de la adquisición del recurso (check points anteriores)
5. Asignación del recurso a otro proceso

Inanición

1. El **mismo proceso siempre** se puede **escoger** como **víctima**. (ej. Si se asignan prioridades)

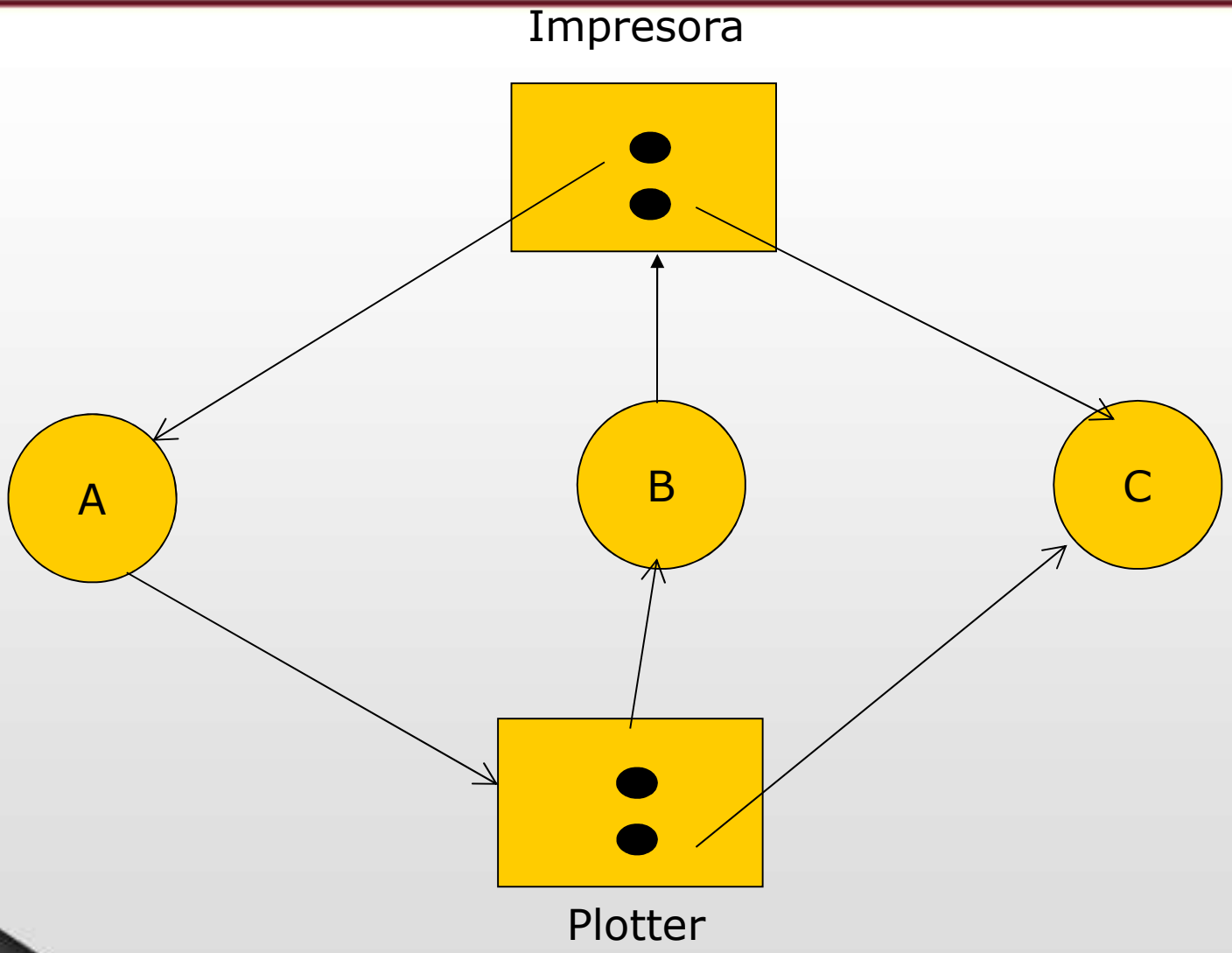
Solución:

1. **Asegurar** que se **elige** un **numero finito** de **veces**
2. **Incluir** el **número** de **retrocesos** como **factor** de **costo**

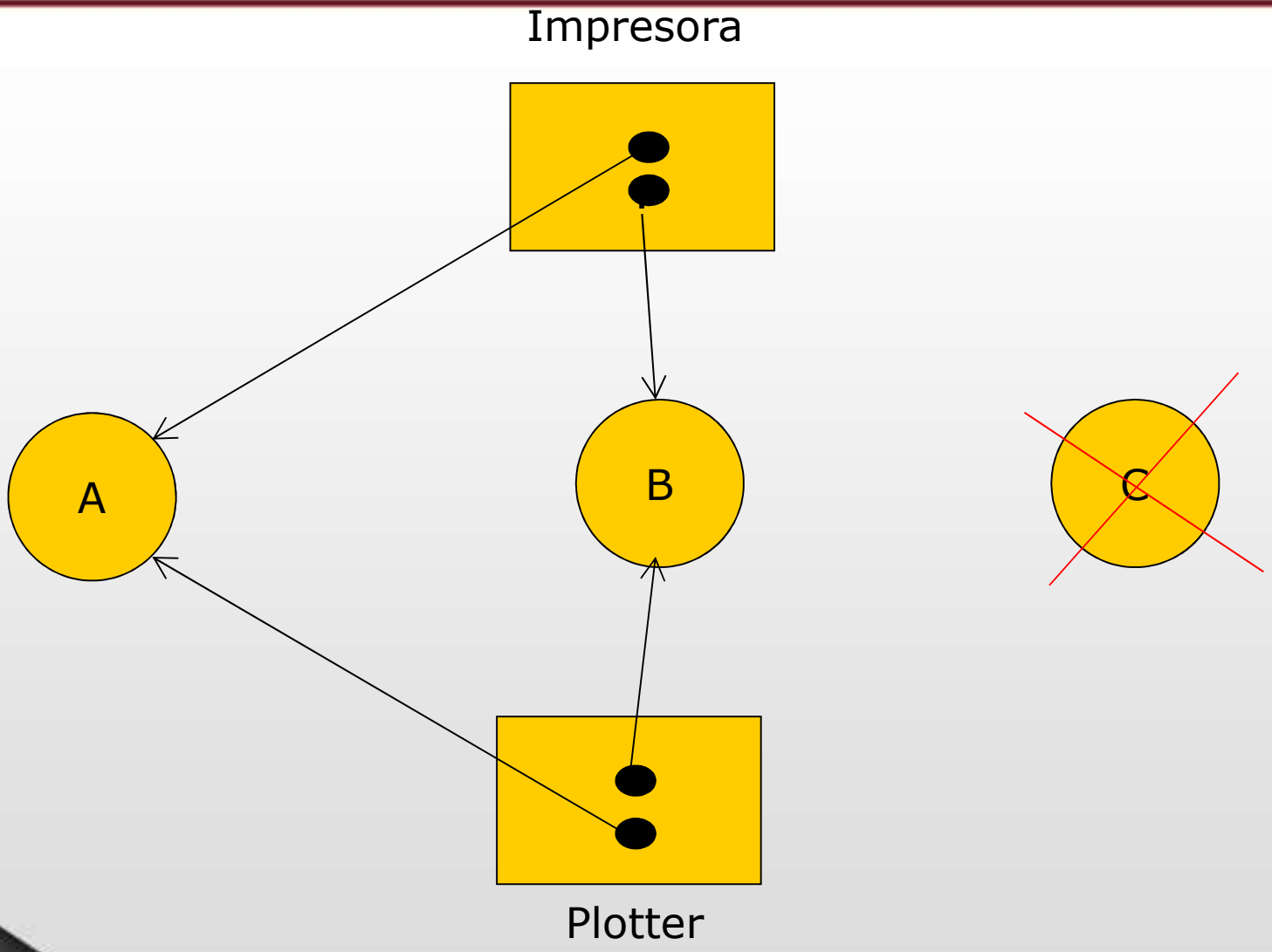
Recuperación por eliminación de procesos

- Alternativas:
  - Abortar todos o algunos procesos o hasta que el ciclo desaparezca → Alto costo, pero garantizo que se elimina el deadlock
  - Se puede elegir un proceso que no esté directamente en el ciclo → Menor costo (a priori), pero puede resultar ineficiente

Ejemplo



Ejemplo (cont.)



Resumen de Detección y Recuperación

- La Detección y Recuperación de interbloqueos proporciona un mayor grado potencial de concurrencia que las técnicas de Prevención o de Evitación.
- Hay sobrecarga en tiempo de ejecución de la Detección
- Hay procesos que son reiniciados o rollback lo que genera demoras en la ejecución.

- La detección y recuperación de interbloqueos puede ser atractiva en sistemas con una baja probabilidad de interbloqueos.
- En sistemas con elevada carga, se sabe que la concesión sin restricciones de peticiones de recursos puede conducir a frecuentes interbloqueos.

Estrategia combinada para el manejo de interbloqueos

- Ninguno de los métodos presentados es 100% adecuado para ser utilizado como estrategia **exclusiva** de manejo de interbloqueos en un Sistema Complejo.
- La Prevención, Evitación y Detección **pueden combinarse** para obtener una máxima efectividad.
- **Se dividen los recursos del sistema en colecciones de clases disjuntas y se aplica el método más adecuado de manejo de interbloqueo a los recursos de cada clase particular.**
- La división puede efectuarse según la jerarquía de diseño de un Sistema Operativo dado, o de acuerdo con las características dominantes de ciertos tipos de recursos, tales como tolerar apropiaciones o permitir predicciones precisas.

Estrategia combinada para el manejo de interbloqueos

Ejemplo:

Consideremos un **sistema con 4 clases de recursos ordenados** de esta forma:

1. **Recursos internos**: utilizados por el sistema, por ejemplo PCB **(tabla de procesos)**
2. **Memoria Central**: **memoria** usada por el proceso del **usuario**
3. **Recursos del Proceso**: **dispositivos asignables. Ejemplo impresoras o unidades extraíbles (cintas, discos, floppies) y archivos.**
4. **Espacio de swapping**: espacio del proceso del usuario en **almacenamiento secundario**

Estrategia combinada para el manejo de interbloqueos

1.Recursos internos del sistema

•Debido a las **frecuentes peticiones** y **liberaciones** de **recursos** a este nivel y a los **frecuentes cambios** de **estado** resultantes, el **recargo** en **tiempo** de **ejecución** de la **Evitación** e incluso de la **Detección** pueden **difícilmente** ser **tolerados**

•La **Prevención** mediante ordenación de recursos es probablemente la **mejor alternativa**.

•**Numeración** de **recursos** demostró que **nunca entra** en **Deadlock**

Estrategia combinada para el manejo de interbloqueos

1. **Memoria Central**

- La **existencia** de **almacenamiento de intercambio** hace que la **Prevención** mediante a**propiación/desalojo** sea una **elección razonable**. (Memoria Virtual, Swapping)
- La **Evitación** **no** es **deseable** debido a su **recargo en tiempo de ejecución** y a su **tendencia** a **infrautilizar** los **recursos**.
- La **Detección** es **posible**, pero **no deseable debido** al **recargo en tiempo de ejecución** en caso de **detección frecuente** o a la **memoria no utilizada retenida** por los **procesos interbloqueados**.

Estrategia combinada para el manejo de interbloqueos

1. **Recursos del Proceso**

- La **Evitación** se ve facilitada por **pre-reclamar** las **necesidades** de **recursos** que habitualmente se efectúa mediante **instrucciones de control**. **Solicitado y asignado en forma total**
- La **Prevención** también es posible **mediante** la **ordenación** de **recursos**
- La **Detección y Recuperación** **no** es **deseable** por la **posibilidad** de **modificación de archivos** que **pertenecen** a esta **clase** de recursos

Estrategia combinada para el manejo de interbloqueos

1. **Espacio de Swapping**

- Una posibilidad es la **adquisición anticipada** de **todo** el **espacio necesario** en **disco** (**Prevención**), ya que **se conocen** los **requisitos previos** de **almacenamiento**.
- **Preasignación** del **espacio** TOTAL NECESARIO