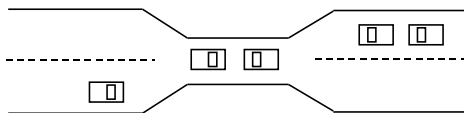


7 BLOQUEOS MUTUOS

- 1 MODELO DE SISTEMA
- 2 CARACTERIZACIÓN DEL BLOQUEO MUTUO
- 3 MÉTODOS PARA MANEJAR BLOQUEOS MUTUOS
- 4 PREVENCIÓN DE BLOQUEOS MUTUOS
- 5 EVITACIÓN DE BLOQUEOS MUTUOS
- 6 DETECCIÓN DE BLOQUEOS MUTUOS
- 7 RECUPERACIÓN DESPUÉS DE UN BLOQUEO MUTUO
- 8 COMBINACIÓN DE ESTRATEGIAS

7.1 MODELO DE SISTEMA



- Tráfico en una sola dirección.
- Cada sección del puente puede ser vista como un recurso.
- Si ocurre deadlock, puede resolverse cuando uno o más autos retroceden (apropiación y rollback).
- Puede haber hambre.

7.1 MODELO DE SISTEMA

- Tipos de Recurso $R_1, R_2, \dots, R_m$ (datos de CPU, espacio de memoria, dispositivos de E/S)
- Cada tipo de recurso R_i tiene varias instancias W_i
- Un proceso, respecto de un recurso, debe:
 - Requerirlo (Request).
 - Usarlo.
 - Liberarlo (Release).
- Request y Release son llamadas al sistema.

7.1 MODELO DE SISTEMA

- Un conjunto de procesos está en estado de bloqueo mutuo si cada proceso del conjunto está esperando un evento que sólo puede ser causado por otro proceso del mismo conjunto.
- El evento puede ser Request o Release.

7.2 CARACTERIZACIÓN DEL DEADLOCK

- En un Bloqueo Mutuo, los procesos nunca terminan su ejecución.
- Los recursos del sistema quedan *amarrados*, impidiendo a los otros procesos comenzar.
- Hay dos características a describir de un bloqueo mutuo:
 - Condiciones necesarias
 - Grafo de Asignación de Recursos

7.2.1 Condiciones Necesarias

- El Bloqueo Mutuo ocurre cuando se dan cuatro condiciones simultáneamente.
 - **Exclusión mutua:** sólo un proceso a la vez puede usar un recurso.
 - **Ocupar y Esperar:** un proceso que ocupa al menos un recurso, espera adquirir recursos adicionales, ocupados por otros procesos.
 - **No Apropiación:** un recurso sólo puede ser liberado voluntariamente por el proceso que lo ocupa, luego que ha completado su tarea.
 - **Espera Circular:** hay un conjunto $\{P_0, \dots, P_n\}$ de procesos en espera, donde P_i espera un recurso que ocupa P_{i+1} , y P_n espera un recurso que ocupa P_0 .

7.2.1 Condiciones Necesarias

- Las condiciones no son del todo independientes.
- Por ejemplo, la condición de Espera Circular implica Ocupar y Esperar.

7.2.2 Grafo de Asignación de Recursos

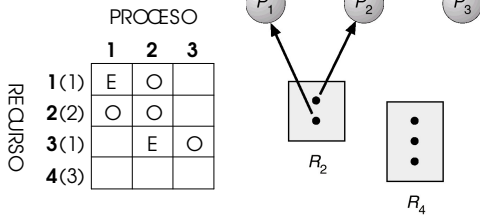
- El Bloqueo Mutuo puede ser descrito más precisamente mediante un grafo dirigido llamado *Grafo de Asignación de Recursos del Sistema*, conformado por dos conjuntos de elementos:
 - V (vértices), que es particionado en dos tipos:
 - $P = \{P_1, P_2, \dots, P_n\}$, conjunto de todos los procesos en el sistema.
 - $R = \{R_1, R_2, \dots, R_m\}$, conjunto de todos los tipos de recurso en el sistema.
 - E (bordes) que es particionado en dos tipos:
 - de Requerimiento: $P_i \rightarrow R_j$
 - de Asignación: $R_j \rightarrow P_i$

7.2.2 Grafo de Asignación de Recursos

- Proceso 
- Tipo de Recurso con 4 instancias 
- P_i requiere instancia de R_j 
- P_i tiene una instancia de R_j 

7.2.2 Grafo de Asignación de Recursos

Ejemplo de Grafo de Asignación de Recursos



7.2.2 Grafo de Asignación de Recursos

V: $P = \{P_1, P_2, P_3\}$, $R = \{R_1, R_2, R_3, R_4\}$.

E: $Req = \{P_1 \rightarrow R_1, P_2 \rightarrow R_3\}$

Asig = $\{R_1 \rightarrow P_2, R_2 \rightarrow P_2, R_2 \rightarrow P_1, R_3 \rightarrow P_3\}$

R_i : $R_1 = 1, R_2 = 2, R_3 = 1, R_4 = 3$.

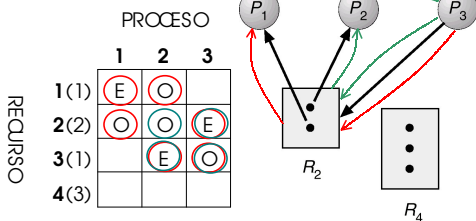
Estados:

- P_1 tiene R_2 y espera R_1 .
- P_2 tiene R_1 y R_2 y espera R_3 .
- P_3 tiene R_3 .

No hay deadlock, pues no hay espera circular.

7.2.2 Grafo de Asignación de Recursos

Grafo de Asignación de Recursos con Deadlock



7.2.2 Grafo de Asignación de Recursos

V: $\mathbf{P} = \{P_1, P_2, P_3\}$, $\mathbf{R} = \{R_1, R_2, R_3, R_4\}$.

E: $\text{Req} = \{P_1 \rightarrow R_1, P_2 \rightarrow R_3, P_3 \rightarrow R_2\}$

Asig = $\{R_1 \rightarrow P_2, R_2 \rightarrow P_2, R_2 \rightarrow P_1, R_3 \rightarrow P_3\}$

R_i : $R_1 = 1, R_2 = 2, R_3 = 1, R_4 = 3$.

Estados:

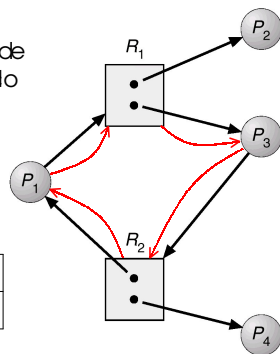
- P_1 tiene R_2 y espera R_1 .
- P_2 tiene R_1 y R_2 y espera R_3 .
- P_3 tiene R_3 y espera R_2 .

Hay deadlock, pues hay 2 ciclos, se dan las 4 condiciones y R_1 y R_3 tienen sólo 1 instancia.

7.2.2 Grafo de Asignación de Recursos

Grafo de Asignación de Recursos con un Ciclo pero sin Deadlock

PROCESO				
	1	2	3	4
REC	1(2)	E	O	O
	2(2)	O	E	O



7.2.2 Grafo de Asignación de Recursos

V: $\mathbf{P} = \{P_1, P_2, P_3, P_4\}$, $\mathbf{R} = \{R_1, R_2\}$.

E: $\text{Req} = \{P_1 \rightarrow R_1, P_3 \rightarrow R_2\}$

Asig = $\{R_1 \rightarrow P_2, R_1 \rightarrow P_3, R_2 \rightarrow P_1, R_2 \rightarrow P_4\}$

R_i : $R_1 = 2, R_2 = 2$.

Estados:

- P_1 tiene R_2 y espera R_1 .
- P_2 tiene R_1 .
- P_3 tiene R_1 y espera R_2 .
- P_4 tiene R_2 .

No hay deadlock, hay más de 1 instancia/Recurso.

7.2.2 Grafo de Asignación de Recursos

Hechos Básicos

- Si el grafo no contiene ciclos $\Rightarrow$ no hay deadlock.
- Si el grafo contiene ciclos $\Rightarrow$
 - Si hay sólo una instancia por tipo de recurso, $\Rightarrow$ hay deadlock.
 - Si hay varias instancias por tipo de recurso, $\Rightarrow$ hay posibilidad de deadlock.

7.3 MÉTODOS PARA MANEJAR DEADLOCK

- Se puede tratar el problema del bloqueo mutuo de tres maneras:
 - Asegurar que el sistema nunca entrará en estado de deadlock, previniendo o evitando su ocurrencia.
 - Permitir al sistema entrar en deadlock, detectarlo y luego recuperarse.
 - Ignorar el problema y pretender que los deadlocks nunca ocurren en el sistema; esto es usado por la mayoría de los sistemas operativos.

7.4 PREVENCIÓN DE DEADLOCKS

Limita las maneras en que puede hacerse un requerimiento, previniendo la ocurrencia de, al menos, una de las 4 condiciones:

- 1 Exclusión Mutua
- 2 Retener y Esperar
- 3 No Apropiación
- 4 Espera Circular

7.4.1 Exclusión Mutua

- No requerida para recursos compartibles.
- Requerida para recursos no compartibles.

7.4.2 Retener y Esperar

- Debe garantizar que cada vez que un proceso requiere un recurso, no tiene otro:
 - El proceso solicita y se le asignan todos sus recursos antes que comience su ejecución, o se le permite solicitarlos sólo cuando los requiere y no los tiene.
 - Hay una baja utilización de recursos.
 - Puede haber hambruna.

7.4.3 No Apropiación

- Si un proceso que tiene algunos recursos requiere otro que no le puede ser asignado de inmediato, entonces todos los recursos actualmente retenidos son liberados.
- Los recursos apropiados se agregan a la lista de recursos por los cuales está esperando el proceso.
- El proceso se reiniciará sólo cuando pueda recuperar sus antiguos recursos y obtener los nuevos requeridos.

7.4.4 Espera Circular

- Impone un ordenamiento total de todos los tipos de recurso y requiere que cada proceso solicite los recursos en orden ascendente de enumeración.

7.5 EVITACIÓN DE DEADLOCK

- Requiere que el sistema tenga disponible información adicional *a priori*.
- El modelo más simple y útil requiere que cada proceso declare el *número máximo* de recursos de cada tipo que puede necesitar.
- El algoritmo de evitación de deadlock examina dinámicamente el estado de asignación de los recursos para asegurar que nunca exista una condición de espera circular.

7.5 EVITACIÓN DE DEADLOCK

- El estado de asignación de los recursos es definido por el número de recursos disponibles y asignados, y la demanda máxima de los procesos.

7.5.1 Estado Seguro

- Cuando un proceso solicita un recurso disponible, el sistema debe decidir si una asignación inmediata dejaría al sistema en estado seguro o no.
- Un sistema está en estado seguro si existe una secuencia segura de todos los procesos.

7.5.1 Estado Seguro

- $\langle P_1, \dots, P_n \rangle$ es segura si, para cada P_i , los recursos que éste aún puede solicitar se satisfarán con aquellos actualmente disponibles + los retenidos por P_j , con $j < i$.
 - Si las necesidades de P_i no están disponibles de inmediato, P_i puede esperar hasta que todos los P_j hayan terminado.
 - Cuando P_j termina, P_i puede obtener los recursos necesitados, ejecutarse, devolver los recursos y terminar.
 - Cuando P_j termina, P_{j+1} puede obtener los recursos necesitados y así sucesivamente.

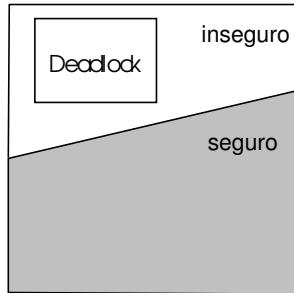
7.5.1 Estado Seguro

Hechos Básicos:

- Si un sistema está en estado seguro $\Rightarrow$ no hay deadlock.
- Si un sistema no está en estado seguro $\Rightarrow$ hay posibilidad de deadlock.
- Evitación $\Rightarrow$ asegura que un sistema nunca entrará en estado no seguro.

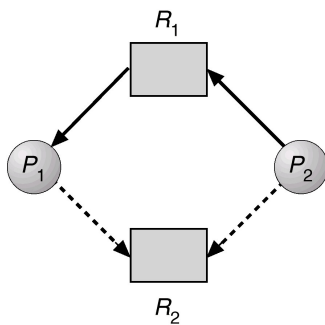
7.5.1 Estado Seguro

Estados seguro e inseguro en la asignación de recursos (ver ejemplo)



7.5.2 Grafo de Asignación de Recursos

Grafo de asignación de recursos para evitación de deadlock (aristas de reserva)



7.6 DETECCIÓN DE DEADLOCK

- Si el sistema no emplea mecanismos de prevención o evitación de bloqueo mutuo, entonces entrará en estado de deadlock.
- En esta situación, el sistema debe proporcionar:
 - Algoritmo de Detección. (recursos con un sólo ejemplar: grafo de espera, múltiples ejemplares: en general es costoso cuando baja la utilización de la CPU y no se puede determinar qué proceso causó el bloqueo)
 - Esquema de Recuperación.

7.7 RECUPERACIÓN DEL DEADLOCK

- Después de la detección de un deadlock viene la recuperación de éste.
- Se tienen varias alternativas, manuales y automáticas.
- La recuperación conlleva romper la espera circular. Hay dos alternativas:
 - Abortar uno o más procesos bloqueados (o todos).
 - Expropiar (apropiar) todos los recursos de cada proceso bloqueado.

7.7 RECUPERACIÓN DEL DEADLOCK

- Abortar procesos:
 - Abortar todos los procesos bloqueados : costo elevado podría tener que recalcularse
 - Abortar un proceso a la vez hasta eliminar el ciclo de bloqueo mutuo: costo considerable, después de abortar cada proceso se invoca nuevamente el algoritmo de detección.

7.7 RECUPERACIÓN DEL DEADLOCK

- Expropiación de recursos: Se expropián algunos recursos sucesivamente de unos a otros procesos hasta romper el bloqueo.
- Se debe considerar:
 - Selección de cuáles recursos de cuáles procesos se deben expropiar: Se determina un orden para minimizar el costo.
 - Retroceso: Abortar el proceso hasta un estado seguro y reiniciar, por lo cual se debe mantener más información de los estados de todos los procesos en ejecución.
 - Inanidón: Se selecciona siempre el mismo proceso para expropiarse. Solución: Asegurar un número limitado de veces.

7.8 Estrategia combinada para el manejo de bloqueos mutuos

- Estrategias básicas vistas: Prevención, evitación y detección, por sí solas no solucionan el problema
- Solución: Combinación de estrategias
- Idea: Los recursos pueden dividirse en clases que tienen un ordenamiento jerárquico.

7.8 Estrategia combinada para el manejo de bloqueos mutuos

Clases de recursos	Definición	Estrategias por clase
Recursos internos	Recursos utilizados por el sistema: PCB	Prevención mediante ordenamiento de recursos
Memoria central	Memoria utilizada por un trabajo de usuario	Prevención mediante expropiación, ya que se pueden intercambiar a disco
Recursos de trabajos	Dispositivos asignables	Evitación, por la información requerida para asignar
Espacio intercambiable	Espacio para cada trabajo de usuario en el almacenamiento auxiliar	Preasignación, pues se conocen las necesidades de almacenamiento máximas
