Sistemas Operativos

Detección de interbloqueos

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

Detección de interbloqueo

Si un sistema no emplea un algoritmo de prevención o evitación de bloqueo mutuo, entonces deberá:

- Determinar si ha ocurrido un interbloqueo
- Recuperarse del interbloqueo

Esto requiere tiempo de procesamiento adicional e involucra pérdidas potenciales de procesos y recursos.

Detección de interbloqueo

Si un sistema no emplea un algoritmo de prevención o evitación de bloqueo mutuo, entonces deberá:

- Determinar si ha ocurrido un interbloqueo
- Recuperarse del interbloqueo

Esto requiere tiempo de procesamiento adicional e involucra pérdidas potenciales de procesos y recursos.

Detección de interbloqueo

Si un sistema no emplea un algoritmo de prevención o evitación de bloqueo mutuo, entonces deberá:

- Determinar si ha ocurrido un interbloqueo
- Recuperarse del interbloqueo

Esto requiere tiempo de procesamiento adicional e involucra pérdidas potenciales de procesos y recursos.

Detección de interbloqueo

Si un sistema no emplea un algoritmo de prevención o evitación de bloqueo mutuo, entonces deberá:

- Determinar si ha ocurrido un interbloqueo
- Recuperarse del interbloqueo

Esto requiere tiempo de procesamiento adicional e involucra pérdidas potenciales de procesos y recursos.

Algoritmo de detección

Estructuras necesarias:

- D-Disponibles: Un vector de longitud m que indica el número de recursos disponibles de cada tipo.
- A-Asignación: Una matriz de $n \times m$ que define el número de recursos de cada tipo actualmente asignados a cada proceso.
- S-Solicitud: Una matriz de $n \times m$ que indica la solicitud actual de cada proceso. Si `Solicitud[i,j] = k`, entonces el proceso p_i solicita k ejemplares del tipo R_j.

Algoritmo de detección

Estructuras necesarias:

- D-Disponibles: Un vector de longitud m que indica el número de recursos disponibles de cada tipo.
- A-Asignación: Una matriz de $n \times m$ que define el número de recursos de cada tipo actualmente asignados a cada proceso.
- S-Solicitud: Una matriz de $n \times m$ que indica la solicitud actual de cada proceso. Si `Solicitud[i,j] = k`, entonces el proceso p_i solicita k ejemplares del tipo R_j.

Algoritmo de detección

Estructuras necesarias:

- D-Disponibles: Un vector de longitud m que indica el número de recursos disponibles de cada tipo.
- A-Asignación: Una matriz de $n \times m$ que define el número de recursos de cada tipo actualmente asignados a cada proceso.
- S-Solicitud: Una matriz de $n \times m$ que indica la solicitud actual de cada proceso. Si `Solicitud[i,j] = k`, entonces el proceso p_i solicita k ejemplares del tipo R_j.

Algoritmo de detección

Estructuras necesarias:

- D-Disponibles: Un vector de longitud m que indica el número de recursos disponibles de cada tipo.
- A-Asignación: Una matriz de $n \times m$ que define el número de recursos de cada tipo actualmente asignados a cada proceso.
- S-Solicitud: Una matriz de $n \times m$ que indica la solicitud actual de cada proceso. Si `Solicitud[i,j] = k`, entonces el proceso p_i solicita k ejemplares del tipo R_j.

Algoritmo de detección

Paso 1

- Considere los vectores T (Trabajo) y F (Fin) de longitud m y n respectivamente.
- Asignar $T = D$
- Revisar $A[i]$ para $i = 1, 2, ..., n$:
  - Si $A[i] \neq 0$, $F[i] = falso$
  - Sino, $F[i] = verdadero$

Algoritmo de detección

Paso 1

- Considere los vectores T (Trabajo) y F (Fin) de longitud m y n respectivamente.
- Asignar $T = D$
- Revisar $A[i]$ para $i = 1, 2, ..., n$:
  - Si $A[i] \neq 0$, $F[i] = falso$
  - Sino, $F[i] = verdadero$

Algoritmo de detección

Paso 1

- Considere los vectores T (Trabajo) y F (Fin) de longitud m y n respectivamente.
- Asignar $T = D$
- Revisar $A[i]$ para $i = 1, 2, ..., n$:
  - Si $A[i] \neq 0$, $F[i] = falso$
  - Sino, $F[i] = verdadero$

Algoritmo de detección

Paso 1

- Considere los vectores T (Trabajo) y F (Fin) de longitud m y n respectivamente.
- Asignar $T = D$
- Revisar $A[i]$ para $i = 1, 2, ..., n$:
  - Si $A[i] \neq 0$, $F[i] = falso$
  - Sino, $F[i] = verdadero$

Algoritmo de detección

Paso 2

- Encontrar un proceso i para el cual $F[i] = \textit{falso}$ y $S[i] \leq T$
- Para ese proceso, $F[i] = \textit{verdadero}$, y $T = T + D$

Algoritmo de detección

Paso 2

- Encontrar un proceso i para el cual $F[i] = \textit{falso}$ y $S[i] \leq T$
- Para ese proceso, $F[i] = \textit{verdadero}$, y $T = T + D$

Algoritmo de detección

Paso 3

- Si $F[i] = \mathit{falso}$, para algún $1 \leq i \leq n$, entonces el sistema está en interbloqueo.
- P_i está en bloqueo mutuo.

Ejercicio

Sea un sistema formado por tres procesos: P1, P2 y P3; y los recursos siguientes: una impresora R1, dos unidades de disco R2 y una cinta R3.

Dada la siguiente situación:

- El proceso P1 posee uno de los recursos R2 y solicita R1
- El proceso P2 posee uno de los recursos R2 y un recurso R1 y solicita el recurso R3.
- El proceso P3 posee el recurso R3 y solicita el recurso R2.

Se pide:

a) Ilustrar dicha situación mediante un grafo de asignación de recursos.
b) Determinar si existe interbloqueo

Recuperación

- Violar la exclusión mutua y asignar el recurso a varios procesos
- Cancelar los procesos suficientes para romper la espera circular
- Desalojar recursos de los procesos en deadlock.

Para eliminar el deadlock matando procesos pueden usarse dos métodos:

- Matar todos los procesos en estado de deadlock. Elimina el deadlock 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 reejecutar el algoritmo de detección para verificar si el deadlock efectivamente desapareció o no.

1) **Selección de Víctimas**

Habría que seleccionar aquellos procesos que rompen el ciclo y tienen mínimo costo. Para ello debemos tener en cuenta:

- Prioridad.
- Tiempo de proceso efectuado, faltante.
- Recursos a ser liberados (cantidad y calidad).
- Cuántos procesos quedan involucrados.
- Si por el tipo de dispositivo es posible volver atrás.

Recuperación

2) **Rollback**

- Volver todo el proceso hacia atrás (cancelarlo).

- Volver hacia atrás hasta un punto en el cual se haya guardado toda la información necesaria (CHECKPOINT)

3) **Inanición (starvation)**

Es un tipo especial de Deadlock en el cual los recursos no llegan nunca a adquirirse.
Ejemplo típico: Procesos largos en una administración del procesador de tipo Mas Corto Primero.

Si el sistema trabajase con prioridades, podría ser que las víctimas fuesen siempre las mismas, luego habría que llevar una cuenta de las veces que se le hizo **Rollback** y usarlo como factor de decisión.