13: Preguntas Chequeo Capitulo 8

PREGUNTAS DE CHEQUEO – CAPÍTULO 8

1. ¿En qué consiste el modelo de sistema para un abrazo mortal? RPTA: El modelo de sistema para un deadlock describe un entorno donde múltiples hilos compiten por un número finito de recursos, los cuales pueden tener varias instancias y tipos. Un deadlock ocurre cuando un conjunto de hilos queda atrapado en una **espera circular**, donde cada hilo necesita un recurso retenido por otro dentro del mismo conjunto, impidiendo el progreso de todos. Para representar estas situaciones, se usa un **grafo de asignación de recursos**, donde los ciclos indican la posibilidad de un deadlock.
2. ¿Cuáles son las cuatro condiciones necesarias para caracterizar un abrazo mortal?
   Descríbalos brevemente. RPTA:

   Las **cuatro condiciones necesarias** que deben cumplirse simultáneamente para que ocurra un **deadlock**:

   - **Exclusión mutua (Mutual exclusion)**: Al menos un recurso debe estar en **modo no compartible**, es decir, solo un hilo puede utilizarlo a la vez. Si otro hilo solicita el mismo recurso, deberá esperar hasta que sea liberado. Los recursos compartibles, como archivos de solo lectura, no generan deadlocks porque pueden ser utilizados simultáneamente por múltiples hilos.
   - **Espera y retención (Hold and wait)**: Un hilo debe mantener al menos un recurso asignado mientras espera adquirir recursos adicionales que están en posesión de otros hilos. Esta condición crea una situación en la que varios hilos pueden quedar bloqueados esperando indefinidamente.
   - **No apropiación (No preemption)**: Un recurso no puede ser arrebatado forzosamente a un hilo en ejecución. Solo el hilo que posee el recurso puede liberarlo voluntariamente cuando termine su tarea. Esto impide

que el sistema reasigne recursos para resolver bloqueos, aumentando el riesgo de deadlocks.

- **Espera circular (Circular wait)**: Debe existir un conjunto de hilos ${T_0, T_1, ..., T_n}$ en el que **cada hilo está esperando un recurso retenido por el siguiente en la secuencia**, y el último hilo en la cadena espera un recurso que posee el primero, formando un ciclo. Esta condición refuerza la dependencia mutua entre los hilos, impidiendo su avance.

3. ¿Cuáles son las tres estrategias para manejar un abrazo mortal? RPTA:

   Se describen **tres estrategias principales** para manejar un **deadlock**:

   - **Ignorar el problema (Ignore the problem)**: Consiste en **no hacer nada** y asumir que los deadlocks no ocurrirán. Esta es la estrategia adoptada por la mayoría de los sistemas operativos, como Linux y Windows, dejando la responsabilidad de manejar los deadlocks a los desarrolladores del kernel y las aplicaciones.
   - **Prevención o evitación de deadlocks (Prevent or avoid deadlocks)**: Busca garantizar que el sistema nunca entre en un estado de deadlock mediante dos enfoques:
     - La **prevención de deadlocks** impone restricciones para asegurar que al menos una de las **cuatro condiciones necesarias** para un deadlock (exclusión mutua, espera y retención, no apropiación, espera circular) **no se cumpla**.
     - La **evitación de deadlocks** requiere que el sistema conozca **de antemano** los recursos que un hilo necesitará, permitiéndole decidir si una solicitud debe ser concedida o bloqueada para evitar una espera circular. Un ejemplo de esto es el **algoritmo del banquero**.
   - **Detección y recuperación de deadlocks (Detect and recover)**: Permite que el sistema **entre en un estado de deadlock, lo detecte y luego tome acciones para resolverlo**. La detección se realiza mediante el **análisis del grafo de asignación de recursos en busca de ciclos**. Para la recuperación, se pueden tomar dos enfoques:
     - **Terminar uno o más procesos** involucrados en el deadlock.
     - **Forzar la apropiación de recursos**, liberando así los necesarios para romper el ciclo de espera.

4. ¿Cuál es la única condición razonable que se puede emplear para evitar que ocurran
abrazos mortales? RPTA:

   La única condición razonable para evitar **deadlocks** es **eliminar la condición de espera circular**.

   Las otras tres condiciones necesarias para que ocurra un deadlock son difíciles de prevenir en la práctica:

   - **Exclusión mutua**: No se puede evitar porque algunos recursos son **intrínsecamente no compartibles**, como los **mutex locks**.
   - **Espera y retención**: Prevenirla implicaría **asignar todos los recursos antes de la ejecución** o **permitir solicitudes solo cuando no se retienen recursos**, lo cual es **impracticable** debido a la naturaleza dinámica de la asignación de recursos.
   - **No apropiación**: Es difícil de aplicar, especialmente en recursos como **mutex locks** y **semáforos**, donde los **deadlocks ocurren con mayor frecuencia**.

   En cambio, la **condición de espera circular** sí puede prevenirse de manera práctica. Para evitar que se cumpla, se puede **imponer un ordenamiento total de todos los tipos de recursos** y requerir que los hilos **los soliciten en un orden creciente**.

   Por ejemplo, si a cada tipo de recurso se le asigna un número único, un hilo solo podrá solicitar un recurso si su número es mayor que el último recurso solicitado. Otra alternativa es exigir que, antes de solicitar un recurso RjR_j, el hilo haya liberado cualquier recurso RiR_i tal que F(Ri)≥F(Rj)F(R_i) \geq F(R_j), donde FF es la función de ordenamiento.

   Este método **rompe la posibilidad de una cadena circular de esperas**, evitando así los **deadlocks**. Un ejemplo práctico es el uso de **mutexes en Pthreads**, donde se define un orden en la adquisición de locks para garantizar que no se genere una espera circular.

5. ¿Cuál es el estado del sistema en el que los recursos se pueden asignar a todos los
procesos en cierto orden, y aún así evitar los abrazos mortales? RPTA:

   Un **estado seguro (safe state)** es aquel en el que los recursos pueden asignarse a los procesos en cierto orden, evitando así los **deadlocks**.

Definición de estado seguro

- Un estado es **seguro** si el sistema puede **asignar recursos a cada hilo (hasta su máximo requerido) en algún orden y aún así evitar un deadlock**.
- Formalmente, un sistema está en estado seguro **solo si existe una secuencia segura de hilos** para el estado de asignación actual.

  <T1,T2,...,Tn><T_1, T_2, ..., T_n>
- Una **secuencia segura** es aquella en la que, para cada , las solicitudes de recursos que aún puede hacer pueden ser satisfechas por los recursos actualmente disponibles **más los recursos retenidos por los hilos anteriores** .

  TiT_i

  (Tj donde j<i)(T_j \text{ donde } j < i)
- En este escenario, si los recursos que necesita no están disponibles inmediatamente, **puede esperar** hasta que los hilos anteriores terminen su ejecución, liberen sus recursos y le permitan continuar.

  TiT_i
- Si no existe ninguna secuencia segura, el estado del sistema es **inseguro (unsafe)**.

Diferencias entre estado seguro, inseguro y deadlock

- Un **estado seguro** garantiza que todos los procesos podrán completar su ejecución sin caer en un deadlock.
- Un **estado inseguro no implica necesariamente un deadlock**, pero sí indica que el sistema ya no puede garantizar que evitará uno.
- Un **estado de deadlock** es siempre un estado inseguro.

Mientras el sistema permanezca en un **estado seguro**, el sistema operativo puede gestionar la asignación de recursos sin riesgo de entrar en **deadlock**.

6. Describa brevemente el propósito y la filosofía del algoritmo del banquero.
   RPTA:

   El **algoritmo del banquero** tiene como propósito principal **evitar los deadlocks** en un sistema con múltiples instancias de cada tipo de recurso.

Filosofía del Algoritmo del Banquero

El algoritmo se basa en la analogía de un sistema bancario con una cantidad finita de efectivo y varios clientes que han declarado sus necesidades máximas de préstamo. Su funcionamiento sigue estos principios:

- **Declaración de necesidades máximas:**
  - Cada proceso (o "cliente") debe **declarar por adelantado** la cantidad máxima de cada tipo de recurso que podría necesitar.
  - Esta cantidad **no puede exceder** el número total de recursos en el sistema.
- **Análisis de estado seguro:**
  - Cuando un proceso solicita recursos, el sistema (actuando como el "banquero") **verifica si la asignación dejará al sistema en un estado seguro**.
- **Estado seguro:**
  - Un estado es seguro si el sistema puede asignar recursos a cada proceso (hasta su máximo declarado) **en algún orden y evitar un deadlock**.
  - Para ello, debe existir una secuencia en la que cada proceso pueda obtener los recursos que necesita, completar su tarea y liberar los recursos, permitiendo que los siguientes procesos también finalicen.
- **Decisión de asignación:**
  - El banquero **solo concede la solicitud de recursos si el sistema permanece en un estado seguro** después de la asignación.
  - Si la asignación llevaría a un estado **inseguro** (donde podría ocurrir un deadlock), la solicitud se **rechaza**, y el proceso debe esperar hasta que haya suficientes recursos disponibles.

7. ¿Cuáles son las estrategias para recuperar el sistema de un abrazo mortal? RPTA:

Cuando un **algoritmo de detección identifica un deadlock**, existen varias estrategias para **recuperar el sistema** y resolver el problema.

Opciones de Recuperación ante un Deadlock

Notificación al operador

Una opción es simplemente **informar al operador** sobre el deadlock y permitirle **tomar una decisión manual** sobre cómo manejar la situación.

Recuperación automática del sistema

El sistema puede intentar **resolver el deadlock automáticamente** mediante dos enfoques principales:

a) Abortar uno o más procesos

El objetivo es **romper la espera circular** terminando uno o más procesos involucrados en el deadlock. Existen dos estrategias:

- **Abortar todos los procesos involucrados:**
  - Es una solución simple pero costosa, ya que todos los cálculos y avances de estos procesos **se perderán**.
- **Abortar un proceso a la vez:**
  - Se termina un proceso y se vuelve a ejecutar el **algoritmo de detección de deadlocks** para verificar si el problema persiste.
  - Esta estrategia implica decidir **qué proceso debe ser abortado**, considerando factores como:
    - **Prioridad del proceso**
    - **Tiempo de ejecución**
    - **Recursos utilizados**

b) Apropiación de recursos (Preemption)

Si abortar procesos es demasiado costoso, otra alternativa es **retirar ciertos recursos** de los procesos involucrados en el deadlock. Esto plantea tres desafíos:

Selección de la víctima

- Se debe decidir **qué recursos** y de **qué procesos** se van a apropiar.
- Para minimizar el impacto, se consideran factores como:
  - **Cantidad de recursos retenidos por cada proceso**

    - **Tiempo de ejecución del proceso afectado**

Rollback (retroceso de estado)

- Cuando se retira un recurso de un proceso, este **no puede continuar normalmente**.
- Existen dos enfoques para manejar esto:
    - **Rollback total:** Se aborta el proceso y se reinicia desde el principio.
    - **Rollback parcial:** Se retrocede solo hasta un estado seguro, lo que **requiere que el sistema almacene información del estado de ejecución de los procesos**.

Evitar la inanición (Starvation)

- Se debe garantizar que **no siempre se seleccione el mismo proceso** como víctima.
    - Una posible solución es incluir el **número de rollbacks previos** como un factor de costo en la selección de víctimas.