SISTEMAS OPERATIVOS – PARCIAL 2

CONCURRENCIA

INTERBLOQUEO E INANICIÓN

FUNDAMENTOS DEL INTERBLOQUEO

Definimos **interbloqueo** como el bloqueo permanente de un conjunto de procesos que o bien compiten por recursos del sistema o se comunican entre sí.

Decimos que un conjunto de procesos está interbloqueado cuando cada proceso del conjunto está bloqueado a la espera de un evento (generalmente la liberación de un recurso requerido) que solo puede generar otro proceso del conjunto, que también está bloqueado.

En un **diagrama de progreso conjunto**, si una trayectoria de ejecución entra en una **región fatal**, el interbloqueo es inevitable.

La existencia de una región fatal depende de la lógica de los procesos, pero el interbloqueo es solo inevitable si el progreso conjunto de los procesos crea una trayectoria que entra en la región fatal.

RECURSOS REUTILIZABLES

Podemos distinguir dos categorías de recursos: reutilizables y consumibles. Un **recurso reutilizable** es aquél que sólo lo puede utilizar de forma segura un proceso en cada momento y que no se destruye después de su uso.

Los procesos obtienen unidades de este recurso que más tarde liberarán para que puedan ser usadas por otros procesos.

Algunos ejemplos de recursos reutilizables son: procesadores, canales de E/S, memoria principal y secundaria, dispositivos, y estructuras de datos como ficheros, bases de datos y semáforos.

El interbloqueo no es un error de programación como tal. Hemos observado que el diseño de programas concurrentes es complejo. Estos interbloqueos se pueden producir, estando su causa frecuentemente empotrada en la compleja lógica del programa, haciendo difícil su detección. Una estrategia para tratar con este interbloqueo es imponer restricciones en el diseño del sistema con respecto al orden en que se pueden solicitar los recursos.

RECURSOS CONSUMIBLES

Un recurso consumible es aquél que puede crearse (producirse) y destruirse (consumirse). Normalmente, no hay límite en el número de recursos consumibles de un determinado tipo. Un proceso productor desbloqueado puede crear un número ilimitado de estos recursos.

Cuando un proceso consumidor adquiere un recurso, este deja de existir.

Algunos ejemplos de recursos consumibles son: las interrupciones, las señales, los mensajes y la información en buffers de E/S.

GRAFOS DE ASIGNACIÓN DE RECURSOS

Los grafos de asignación de recursos constituyen una herramienta muy útil para la caracterización de la asignación de recursos a los procesos.

Los grafos de asignación de recursos son grafos dirigidos, donde procesos y recursos son representados mediante nodos.

Una arista dirigida de un proceso a un recurso significa que el proceso ha solicitado el recurso, pero aún no se le ha asignado.

Una arista dirigida de un recurso a un proceso implica que el recurso ha sido asignado al proceso.

LAS CONDICIONES PARA EL INTERBLOQUEO

Deben presentarse tres condiciones de gestión para que sea posible un interbloqueo:

1. **Exclusión mutua**: sólo un proceso puede utilizar un recurso en cada momento. Ningún proceso puede acceder a una unidad de un recurso que se ha asignado a otro proceso.
2. **Retención y espera**: un proceso puede mantener los recursos asignados mientras espera la asignación de otros recursos.
3. **Sin expropiación**: no se puede forzar la expropiación de un recurso a un proceso que lo posee.

Si se cumplen estas tres condiciones, se puede producir un interbloqueo, pero es posible que no lo haya. Para que realmente se produzca el interbloqueo es necesaria una cuarta condición:

4. **Espera circular**: existe una lista cerrada de procesos, de tal manera que cada proceso posee al menos un recurso necesitado pro el siguiente proceso de la lista.

La espera circular, enumerada como cuarta condición, es irresoluble debido a que se cumplen las tres primeras condiciones. Por tanto, las cuatro condiciones de forma conjunta constituyen condiciones necesarias y suficientes para el interbloqueo.

Existen tres estrategias para el tratamiento del interbloqueo. En primer lugar, se puede **prevenir** el interbloqueo adoptando una política que elimine una de las cuatro condiciones enumeradas anteriormente. En segundo lugar, se puede **predecir** el interbloqueo tomando las apropiadas decisiones dinámicas basadas en el estado actual de asignación de recursos. En tercer lugar, se puede intentar **detectar** la presencia del interbloqueo (se cumplen las cuatro condiciones) y realizar las acciones pertinentes para recuperarse del mismo.

PREVENCIÓN DEL INTERBLOQUEO

La estrategia de prevención del interbloqueo consiste, de forma simplificada, en diseñar un sistema de manera que se excluya la posibilidad del interbloqueo.

Podemos clasificar los métodos de prevención del interbloqueo en dos categorías:

- Un método indirecto de prevención es impedir la aparición de una de las tres condiciones necesarias listadas previamente (las tres primeras).
- Un método directo de prevención del interbloqueo impide que se produzca una espera circular (la cuarta condición).

A continuación, se examinan las técnicas relacionadas con las cuatro condiciones:

EXCLUSIÓN MUTUA

Generalmente, la primera de las cuatro condiciones no puede eliminarse. Si el acceso a un recurso requiere de exclusión mutua, el sistema operativo debe proporcionarlo.

RETENCIÓN Y ESPERA

La condición de retención y espera puede eliminarse estableciendo que un proceso debe solicitar al mismo tiempo todos sus recursos requeridos, bloqueándolo hasta que se le puedan conceder simultáneamente todas sus peticiones.

Esta estrategia es insuficiente en dos maneras:

- Un proceso puede quedarse esperando mucho tiempo hasta que todas sus solicitudes de recursos puedan satisfacerse, cuando podría haber continuado con solamente algunos de los recursos.
- Los recursos asignados a un proceso pueden permanecer inutilizados mucho tiempo, durante el cual se impide su uso a otros procesos

Otro problema existente es que un proceso puede no conocer por anticipado todos los recursos que requerirá.

Existe también un último problema práctico creado por el uso de una programación modular o una estructura multihilo en una aplicación. La aplicación necesitará ser consciente de todos los recursos que se solicitarán en todos los niveles o en todos los módulos para hacer una solicitud simultánea.

SIN EXPROPIACIÓN

Esta condición se puede impedir de varias maneras:

- Si a un proceso que mantiene varios recursos se le deniega una petición posterior, ese proceso deberá liberar sus recursos originales y, si es necesario, los solicitará de nuevo junto con el recurso adicional.
- Si un proceso solicita un recurso que otro proceso mantiene actualmente, el sistema operativo puede expropiar al segundo proceso y obligarle a liberar sus recursos. Esto impediría el interbloqueo si no hay dos procesos que posean la misma prioridad.

Esta estrategia es solo práctica cuando se aplica a recursos cuyo estado se pude salvar y restaurar más tarde, como es el caso de un procesador.

ESPERA CIRCULAR

La condición de espera circular se puede impedir definiendo un orden lineal entre los distintos tipos de recursos.

Si a un proceso se le han asignado recursos de tipo “R”, posteriormente solo puede pedir recursos cuyo tipo tenga un orden superior al de “R”.

Como ocurría en el caso de la retención y espera, la prevención de la espera circular puede ser ineficiente, ralentizando los procesos y denegando innecesariamente el acceso a un recurso.

PREDICCIÓN DEL INTERBLOQUEO

En la **prevención del interbloqueo**, hemos visto que se restringen las solicitudes de recurso para impedir al menos una de las cuatro condiciones de interbloqueo, lo que conlleva un uso ineficiente de los recursos y una ejecución ineficiente de los procesos.

Por otro lado, la **predicción del interbloqueo** permite las tres condiciones necesarias, pero toma decisiones razonables para asegurarse de que nunca se alcanza el punto del interbloqueo.

De esta manera, la predicción permite más concurrencia que la prevención.

Con la predicción del interbloqueo, se decide dinámicamente si la petición actual de reserva de un recurso, si se concede, podrá potencialmente causar un interbloqueo. La predicción, por tanto, requiere el conocimiento de las futuras solicitudes de recursos del proceso.

Existen dos técnicas para predecir el interbloqueo:

- No iniciar un proceso si sus demandas podrían llevar a un interbloqueo.
- No conceder una petición adiciona de un recurso por parte de un proceso si esta asignación podría provocar un interbloqueo.

DENEGACIÓN DE LA INICIACIÓN DEL PROCESO

Para que la predicción de interbloqueo funcione, el proceso debe declarar con antelación los requisitos máximos con respecto a cada recurso.

De esta manera, solo iniciamos un proceso si se pueden satisfacer las necesidades máximas de todos los procesos actuales más las del nuevo proceso.

Esta estrategia está lejos de ser óptima, debido a que asume el peor caso: todos los procesos solicitarán sus necesidades máximas simultáneamente.

DENEGACIÓN DE ASIGNACIÓN DE RECURSOS

Esta estrategia se denomina como **el algoritmo del banquero**.

En primer lugar, debemos definir los conceptos de **estado** y de **estado seguro**:

Consideramos un sistema con un número fijo de procesos y recursos. En un determinado momento, un proceso puede tener cero o más recursos asignados. El **estado** del sistema refleja la asignación actual de recursos a procesos.

- Un **estado seguro** es aquél en el que hay, al menos una secuencia de asignación de recursos a los procesos que no implica un interbloqueo (es decir, todos los procesos pueden ejecutarse al completo).
- Un **estado inseguro** es todo aquél estado que no es seguro.

El algoritmo del banquero asegura que el sistema de procesos y recursos estará siempre en un estado seguro y consiste en lo siguiente:

Cuando un proceso solicite un conjunto de recursos, suponemos que le concedemos su petición. En consecuencia, actualizamos el estado del sistema y determinamos si el

resultado es un estado seguro. En caso afirmativo, concedemos la petición. En caso contrario, se bloquea el proceso hasta que sea seguro conceder la petición.

La predicción del interbloqueo tiene como ventaja que no es necesario expropiar a los procesos ni retroceder su ejecución, como ocurre con la detección del interbloqueo, y es menos restrictivo que la prevención. Sin embargo, tiene varias restricciones de uso:

- Deben establecerse por anticipado los requisitos máximos de recursos de cada proceso.
- Los procesos involucrados deben ser independientes, es decir, el orden en el que se ejecutan no debe estar restringido por ningún requisito de sincronización.
- Debe haber un número fijo de recursos que asignar.
- Ningún proceso puede terminar mientras mantenga recursos.

DETECCIÓN DEL INTERBLOQUEO

Las estrategias de prevención del interbloqueo son muy conservadoras: resuelven el problema limitando el acceso a los recursos e imponiendo restricciones a los procesos.

En el extremo contrario, la estrategia de detección del interbloqueo no limita el acceso a los recursos ni restringe las acciones de los procesos.

Con la **detección del interbloqueo** los recursos pedidos se conceden siempre que sea posible. Periódicamente el S.O. realiza un algoritmo que le permite detectar la condición de espera circular.

ALGORITMO DE DETECCIÓN DEL INTERBLOQUEO

La comprobación de si hay interbloqueo se puede hacer con tanta frecuencia como una vez por cada petición de recurso o con menos frecuencia, dependiendo de la probabilidad de que ocurra un interbloqueo.

El desarrollo del algoritmo puede verse en el libro o en las diapositivas.

RECUPERACIÓN

Una vez detectado un interbloqueo, se necesita alguna estrategia para recuperarlo. A continuación, se listan todas las estrategias posibles:

- Abortar todos los procesos involucrados en el interbloqueo. Esta es, se crea o no, una de las soluciones más usuales.
- Retroceder cada proceso a algún punto de control previamente definido y rearrancar todos los procesos. El riesgo de esta técnica es que puede repetirse el interbloqueo original. Sin embargo, el indeterminismo del procesamiento del procesamiento concurrente puede asegurar que probablemente esto no suceda.
- Abortar sucesivamente los procesos en el interbloqueo hasta que éste deje de existir. El orden en que se seleccionan los procesos para abortarlos debería estar basado en algunos criterios que impliquen un coste mínimo. Después de cada aborto, se debe invocar de nuevo el algoritmo de detección para comprobar si todavía existe el interbloqueo.
- Expropiar sucesivamente los recursos hasta que el interbloqueo deje de existir. Como en el tercer punto, se debería utilizar una selección basada en el coste y se requiere una nueva invocación al algoritmo de detección después de cada expropiación. Un proceso al que se le ha expropiado un recurso debe retroceder a un punto anterior a la adquisición de ese recurso.

Para el tercer y cuarto punto, un criterio de selección podría ser alguno de los siguientes:

- La menor cantidad de tiempo de procesador consumida hasta ahora.
- La menor cantidad de salida producida hasta ahora.
- El mayor tiempo restante estimado.
- El menor número total de recursos asignados hasta ahora.
- La menor prioridad.

UNA ESTRATEGIA INTEGRADA DE TRATAMIENTO DEL INTERBLOQUEO

Existen ventajas y desventajas para todas las estrategias de tratamiento de un interbloqueo. En vez de intentar diseñar una solución en el S.O. que utilice una sola de estas estrategias, podría ser más eficiente usar estrategias diferentes en distintas situaciones.

Un ejemplo de este tipo de técnica podría ser el siguiente:

- Agrupar los recursos en diversas clases de recursos diferentes.
- Utilizar la estrategia del orden lineal, definida previamente, para prevenir la espera circular impidiendo los interbloqueos entre clases de recursos.
- Dentro de una clase de recursos, usar el algoritmo que sea más apropiado para esa clase.

EL PROBLEMA DE LOS FILÓSOFOS COMENSALES

El enunciado de este problema es el siguiente:

Cinco filósofos viven en una casa, donde hay una mesa preparada para ellos. Básicamente, la vida de cada filósofo consiste en pensar y comer. Debido a su falta de habilidad manual, cada filósofo necesita tener dos tenedores para poder comer. La disposición para la comida es simple: cinco platos con cinco tenedores. Un filósofo que quiere comer, se dirige a su sitio asignado en la mesa y utilizando los dos tenedores situados a cada lado del plato, come. El problema es diseñar un algoritmo que permita a los filósofos comer.

La solución para este problema puede verse en el libro.