Instituto Politécnico Nacional
Escuela Superior de Cómputo

TAREA 12

Sistemas Operativos

Integrantes:
Mora Ayala José Antonio
Ramírez Cotonieto Luis Fernando
Torres Carrillo Josehf Miguel Ángel
Tovar Jacuinde Rodrigo

Profesor:
Cortés Galicia Jorge

Interbloqueos

Modelo de sistema

Un sistema consta de un número finito de recursos, que se distribuyen entre una serie de procesos de competición.

Los recursos se dividen en varios tipos, constando cada uno de ellos en cierto numero de instancias
(Espacio de memoria, ciclos de CPU, archivos, dispositivos ES, etc...)

Si un proceso solicita una instancia de un tipo de recurso, la asignación de cualquier instancia del tipo satisfará la solicitud, si no es así, quiere decir que las instancias no son idénticas, y por lo tanto, los tipos de recursos no se han definido apropiadamente.

En un modo de operación normal, un proceso puede emplear un recurso solo siguiendo esta secuencia:
1. Solicitud
2. Uso
3. Liberación

La solicitud y liberación de los recursos del sistema operativo no gestiona puede hacerse mediante las operaciones wait( ), y signal( )de los semáforos, o a través de la adquisición y liberación de un cerrojo mutex.

Una tabla del sistema a registra al proceso asignado actualmente.

Un conjunto de procesos estará en un estado de Inter bloqueo cuando todos los procesos del conjunto estén esperando a que se produzca un suceso que solo puede producirse como resultado de la actividad de otro procesos del conjunto.

Caracteristicas de los interbloqueos

En un interbloqueo los procesos nunca terminan de ejecutarse y los recursos del sistema están ocupados, lo que impide que se inicien otros trabajos.

Condiciones necesarias:
1. Exclusión mutua: Al menos un recurso debe de estar en modo no compartido.
2. Retención y espera: Un proceso debe estar retenido al menos un recurso y esperando para adquirir otros recursos adicionales.
3. Sin desalojo: Los recursos no pueden ser desalojados
4. Espera circular: Debe de existir un conjunto de procesos en espera tal que uno este esperando a un recurso retenido por otros dos.

Gafo de asignación de recursos
Pueden definirse de forma mas precisa mediante un grado dirigido, que se llama grado de asignación de recursos de sistema. Este grado consta de un conjunto de vertices y de un conjunto de aristas E.

Dada la definición de un grado de asignación de recursos, podemos demostrar que, si el grado no contiene ningún ciclo, entonces ningún proceso del sistema esta Inter bloqueado. Si el grado continúen un ciclo,, entonces puede existir un interbloqueo.

Si cada recurso tiene exactamente una instancia, entonces la existencia de un ciclo implica necesariamente que se ha producido un interbloqueo.

Si cada tipo de recurso tiene varias instancias, entonces la existencia de un ciclo no necesariamente implica que se haya producido un interbloqueo. En este caso, la existencia de un ciclo en el grado es condi-

Metodos para los interbloqueos

En general podemos abordar el problema de los interbloqueos de una de tres formas:
1. Podemos emplear un protocolo para impedir o evitar los interbloqueos, asegurando que el sistema nunca entre en el estado de interbloqueo.
2. Podemos permitir que el sistema entre en estado de interbloqueo, detectarlo y realizar una recuperación
3. Podemos ignorar el problema y actuar como si nunca se produjeran interbloqueos en el sistema.

La tercera solución es la mas ocupada por los sistemas operativos.

PAra garantizar que nunca se produzcan interbloqueos, el sistema puede emplear un esquema de prevención de interbloqueos o un esquema de evacuen de interbloqueos.

La evasion de interbloqueos requiere que se proporcione de antemano al sistema operativo información adicional sobre que recursos solicitara y utilizara un proceso durante su tiempo de vida.

Si un sistema no emplea un algoritmo de prevención de interbloqueos ni un algoritmo de evasion, entonces puede producirse una intuición de interbloqueo.

Prevención de interbloqueos

Al tener cuatro condiciones necesarias para crear un interbloqueo, podemos asegurar que una de estas cuatro condiciones no se cumpla y prevenir la aparición de interbloqueos.

Exclusión mutua: Se aplica a los recursos que no pueden ser compartidos. Un proceso no necesita esperar nunca para acceder a un recurso compatible. Sin embargo, no podemos evitar los bloqueos de exclusión mutua.

Retención y espera: Debemos garantizar que cuando un proceso solicite un recurso, el proceso no esté retenido ningún otro recurso.

Sin desalojo: Los recursos desalojados se añaden a la lista de recursos que el proceso esta esperando. El proceso solo se reiniciara cuando pueda recuperar sus antiguos recursos junto con los nuevos que esta solicitando.

Espera circular: Se impone una ordenación total de todos los tipos de recursos y requerir que cada proceso solicite sus recursos en un orden creciente de enumeració

Evasión de interbloqueos

Un método alternativo consiste en requerir información adicional sobre como ban a ser solicitados los recursos.

El algoritmo de evasión de interbloqueos examina dinámicamente el estado de asignación de cada recurso con el fin de asegurar que nunca se produzca una condición de espera circular

Estado seguro

Puede asignar recursos de cada proceso en determinado orden sin que eso produzca un interbloqueo.

Grafo de asignación de recursos:
Si tenemos un sistema de asignación de recursos con solo una instancia de cada tipo de recurso, puede utilizarse una variante de grafo de asignación de recursos para evitar los interbloqueos.

Si no se crea un ciclo de la asignación del recurso, dejara al sistema en un estado seguro. Si se encuentra un ciclo, entonces la asignación colocaría al sistema en un estado inseguro.

Algoritmo del banquero

El algoritmo del grafo de asignación de recursos no es aplicable a los sistemas de asignación de recursos con múltiples instancias de cada tipo de recursos.

Se eligió este nombre para un algoritmo ya que podría utilizarse en un banco para garantizar que nunca asigne sus fondos disponibles de tal forma que no pueda satisfacer las necesidades de todos sus clientes.

Cuando entra en el sistema un proceso nuevo, debe declarar el numero maximo de instancias de cada tipo de recursos que puede necesitar.
Este numero no puede exceder el numero total de recursos el sistema.

Deben de utilizarse varias estructuras de datos para implementar el algoritmo del banquero, available, max, allocation, need

Algoritmo de seguridad

Ahora podemos presentar el algoritmo para averiguar si un sistema se encuentra en estado seguro o no. Este algoritmo puede describirse del siguiente modo:

1. Sean *Work* y *Finish* sendos vectores de longitud m y n, respectivamente. Inicializamos esos vectores de la forma siguiente: $Work = Available$ y $Finish[i] = false$ para $i = 0, 1, ..., n - 1$.
2. Hallar i tal que
   a. $Finish[i] = = false$
   b. $Need_i \leq Work$

   Si no existe i que cumpla estas condiciones, ir al paso 4.
3. $Work = Work + Allocation_i$
   $Finish[i] = true$
   Ir al paso 2.
4. Si $Finish[i] = = true$ para todo i, entonces el sistema se encuentra en un estado seguro.

Este algoritmo puede requerir del orden de $m \times n^2$ operaciones para determinar si un estado es seguro.

Algoritmo de solicitud de recursos

1. Si $Request_i \leq Need_i$, ir al paso 2. En caso contrario, se genera una condición de error, dado que el proceso ha excedido la cantidad máxima de recursos que había declarado.
2. Si $Request_i \leq Available$, ir al paso 3. En caso contrario, P_i tendrá que esperar, dado que los recursos no están disponibles.
3. Hacer como si el sistema hubiera asignado al proceso P_i los recursos solicitados, modificando el estado del modo siguiente:

$$Available = Available - Request_i;$$
$$Allocation_i = Allocation_i + Request_i;$$
$$Need_i = Need_i - Request_i;$$

Si el estado que resulta de la asignación de recursos es seguro, la transacción se completa y se asignan los recursos al proceso P_i. Sin embargo, si el nuevo estado resulta ser inseguro, entonces P_i debe esperar a que se le asignen los recursos $Request_i$, y se restaura el antiguo estado de asignación de recursos.

Deteccion de interbloqueos

Si un sistema no emplea ni algoritmos de prevención ni de evasion de interbloqueos, entonces puede producirse una situación de interbloqueo en el sistema, en este caso, el sistema debe de proporcionar.
1. Un algoritmo que examine el estado del sistema para determinar si se ha producido un interbloqueo
2. Un algoritmo para recuperarse del interbloqueo

Una sola instancia de cada tipo de recurso

Si todos los recursos tienen una única instancia, entonces podemos definir un algoritmo de detección de interbloqueos que utilice una variante del grafo de asignación de recursos, denominada grafo de espera. Obtenemos este grafo a partir de grados de asignación de recursos, denominados grados de espera. Obtenemos este grafo a partir del grafo de asignación fe recursos eliminando los nodos de recursos y colapsando las correspondientes aristas.

Varias instancias de cada tipo de recurso

El esquema del grafo de espera no es aplicable a los sistemas de asignacion de recursos con multiples instancias de cada tipo de recurso. Veamos ahora un algoritmo de detección de interbloqueos que pueda aplicarse a tales sistemas. El algoritmo emplea varias estructuras de datos variables con el tiempo, que son similares a las utilizadas en el algoritmo del banquero

- **Available**. Un vector de longitud m que indica el número de recursos disponibles de cada tipo.
- **Allocation**. Una matriz $n \times m$ que define el número de recursos de cada tipo que están asignados actualmente a cada proceso.
- **Request**. Una matriz $n \times m$ que especifica la solicitud actual efectuada por cada proceso. Si $Request[i][j]$ es igual a k entonces el proceso P_i está solicitando k instancias más del tipo de recurso R_j.

Utilización de algoritmo de detección

Si los interbloqueos se producen frecuentemente, entonces el algoritmo de detección debe invocarse frecuentemente. Los recursos asignados a los procesos en interbloqueo estarán inactivos hasta que el interbloqueo se elimine. Además, el número de procesos implicados en el ciclo de interbloqueo puede aumentar.

Los interbloqueos se producen sólo cuando algún proceso realiza una solicitud que no se puede conceder de forma inmediata. Esta solicitud puede ser la solicitud final que complete una cadena de procesos en espera. En el caso extremo, podemos invocar el algoritmo de detección de interbloqueos cada vez que una solicitud de asignación no pueda ser concedida inmediatamente. En este caso, podemos no sólo identificar el conjunto de procesos en interbloqueo sino también el proceso concreto que ha "causado" dicho interbloqueo. (En realidad, cada uno de los procesos en interbloqueo es una arista del ciclo que aparece en el grafo de recursos, por lo que todos ellos, conjuntamente, dan lugar al interbloqueo.) Si existen muchos tipos de recursos diferentes, una solicitud puede crear muchos ciclos en el grafo de recursos, siendo completado cada ciclo por la solicitud más reciente y estando "causado" por ese proceso perfectamente identificable.

Por supuesto, si se invoca el algoritmo de detección de interbloqueos para cada solicitud de recursos, esto dará lugar a una considerable sobrecarga en el tiempo de uso del procesador. Una alternativa más barata consiste, simplemente, en invocar el algoritmo a intervalos menos frecuen-