Capas de software y bloqueo mutuo

- Franco Rayas Ángel Damián
- Gómez López Miguel Ángel
- López Coria Axel Yahir
- Vasco Giraldo Juan Esteba

Contenidos

Modelo de sistemas {

Entorno de multiprogramación donde múltiples procesos compiten por los recursos.

Tipos de recursos finitos: Los recursos son clasificados en diferentes tipos los cuales cuentan con varias instancias, los cuales deben ser compartidos entre todos los procesos. (memoria, CPU, dispositivos E/S).

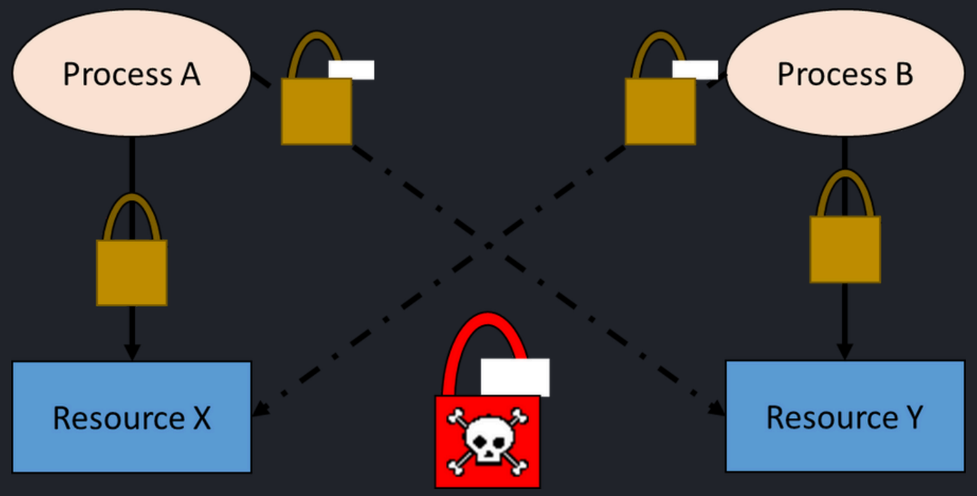
Los procesos deben solicitar los recursos antes de usarlos y liberarlos después de su uso. Pueden solicitar tantos recursos como necesiten, siempre que el número de recursos no exceda el total de recursos disponibles en el sistema.

Interbloqueos

- Definición: Una condición en la que dos o más procesos quedan bloqueados indefinidamente, esperando a que se liberen recursos que están siendo retenidos por otros procesos.
- Causa: Ocurren debido a la competencia por recursos finitos en un entorno de multiprogramación.

}

Caracterización de los interbloqueos {



Para que ocurra un interbloqueo debe cumplirse cuatro condiciones simultáneamente, también cabe destacar que los grafos de asignación de recursos como una herramienta para representar y analizar la situación en el sistema y como detectar posibles interbloqueos.

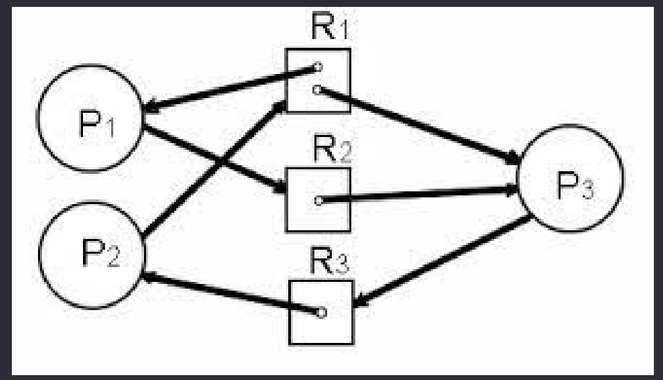
1. Condiciones: Las condiciones necesarias para que se produzca un interbloqueo son la exclusión mutua, retención y espera, sin desalojo y la espera circular.
2. Detección: La existencia de un ciclo en el grafo puede indicar un interbloqueo, esto Si cada tipo de recurso tiene exactamente una instancia. Si hay varios tipos de recursos con más de una instancia, es condición necesaria pero no suficiente para la existencia de un interbloqueo.

}

Grafos de asignación de recursos {

¿Qué es?

El grafo consta de vértices que representan procesos y tipos de recursos, y aristas que indican las relaciones de solicitud y asignación entre procesos y recursos. Siendo una herramienta grafica utilizada para detectar posibles interbloqueos



Detección

Se establece que si el grafo no contiene ningún ciclo, entonces ningún proceso está inter bloqueado.
Si existe un ciclo, puede haber un interbloqueo, especialmente si cada recurso tiene una sola instancia o si el ciclo involucra solo un conjunto de recursos, cada uno con una sola instancia.

Ciclos

Un ciclo en el grafo es condición necesaria para la existencia de un interbloqueo, pero no es condición suficiente si cada recurso tiene varias instancias.

}

Representaciones en los grafos {

NODOS

1. Procesos: Representados por un círculo. Cada círculo representa un proceso activo en el sistema.
2. Recursos: Representados por un rectángulo. Cada rectángulo representa un tipo de recurso en el sistema.

ARISTAS

1. Aristas de Solicitud: Representadas por una línea dirigida desde un proceso hasta el recurso. Indica que el proceso está esperando que se le asigne.
2. Aristas de Asignación: Representadas por una línea dirigida desde un recurso a un proceso. Indica que se ha asignado una instancia del tipo de recurso al proceso.

INSTANCIAS

1. Dentro de cada rectángulo, se pueden incluir puntos para representar las instancias individuales del recurso. Por ejemplo, si un tipo de recurso tiene tres instancias, se representarían tres puntos dentro del rectángulo correspondiente.

}

Métodos para tratar los interbloqueos {

01

Implementar un protocolo para impedir o evitar los interbloqueos.

02

Permitir que el sistema entre en estado de interbloqueo, detectarlo y realizar una recuperacion.

03

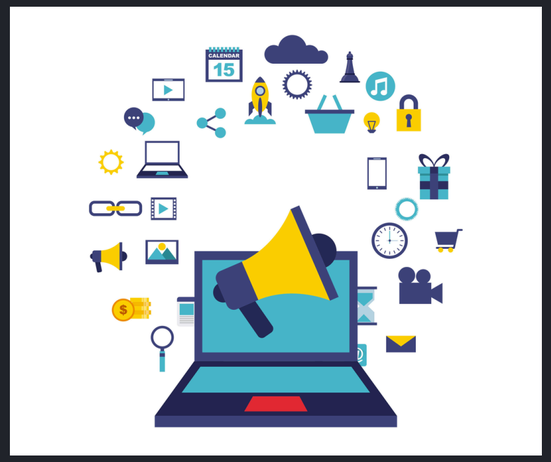
Ignorar el problema y actuar como si nunca se produjeran interbloqueos en el sistema

La tercera solución es la que usan la mayoría de los sistemas operativos, por lo tanto será problema del desarrollador de aplicaciones el escribir programas que resuelvan posibles interbloqueos

}

# Evasion de interbloqueos
{

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



Se proporciona al SO información sobre recursos solicitará y utilizará un proceso durante un tiempo de vida

Con esto se puede decidir, para cada solicitud, si el proceso tiene que esperar o no.

Para decidir si la solicitud actual puede satisfacerse o debe retardarse.

El sistema necesita considerar qué recursos hay disponibles en ese momento, qué recursos están asignados a cada proceso y las futuras solicitudes y liberaciones de cada proceso

> Si un sistema no garantiza que nunca se producirá un interbloqueo, ni proporciona un mecanismo para la detección y recuperación de interbloqueo.

Podría darse la situación de que el sistema esté en estado de interbloqueo y no haya forma de saberlo

> El interbloqueo no detectado dará lugar a un deterioro del rendimiento del sistema, ya que habrá recursos retenidos por el proceso que no pueden ejecutarse y a medida que se realicen nuevas solicitudes de recursos, los procesos entrarán en estado de interbloqueo.

En muchos sistemas, los interbloqueos se producen de forma bastante infrecuente, por lo tanto, este método es más barato que los métodos de prevención, de evasión o de detección y recuperación, que deben utilizarse constantemente.

}

EXCLUSION MUTUA

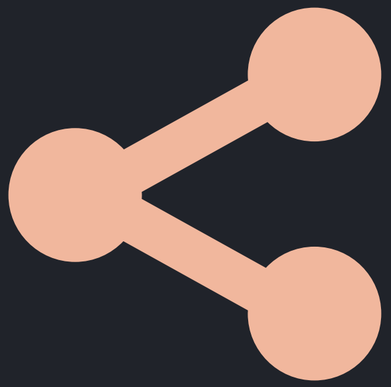
- Esta condición, se aplica a los recursos que no pueden ser compartidos. Varios procesos no pueden compartir simultáneamente una impresora.

- Los recursos que si pueden compartirse no requieren acceso mutuamente excluyente y, por lo tanto, no pueden verse implicados en un interbloqueo.

En los archivos de sólo lectura:

Si varios procesos intentan abrir un archivo de sólo lectura al mismo tiempo, puede concedérseles acceso al archivo de forma simultánea.

Un proceso no necesita esperar nunca para acceder a un recurso compatible, sin embargo, no se puede evitar los interbloqueos negando la condición de exclusión mutua (algunos son intrínsecamente no compatibles)



RETENCIÓN Y ESPERA

- Para que esta condición nunca se produzca en el sistema, se debe garantizar que, cuando un proceso solicite un recurso, el proceso no esté reteniendo ningún otro recurso.
- Un protocolo que permitiera a un proceso solicitar recursos sólo cuando no tenga ninguno retenido podría ser una posible alternativa.

Ambos protocolos tienen dos desventajas importantes:

La tasa de utilización de los recursos puede ser baja, dado que los recursos pueden asignarse pero no utilizarse durante un periodo largo de tiempo.

Puede producirse el fenómeno de inanición: un proceso que necesite varios recursos muy solicitados puede tener que esperar de forma indefinida, debido a que al menos uno de los recursos que necesita está siempre asignado a algún otro proceso.

SIN DESALOJO

- Para impedir que los recursos que ya están asignados sean desalojados, se puede hacer lo siguiente:

  Si un proceso está reteniendo varios recursos y solicita otro recurso que no se le puede asignar de forma inmediata, entonces todos los recursos actualmente retenidos se desalojan. Estos recursos, se añaden a la lista de recursos que el procesos está esperando.

- Por otra parte, si un proceso solicita varios recursos, primero comprobaremos si están disponibles, para asignarse. Si no, se desalojaran los recursos deseados del proceso en espera y se asignaran al proceso que los ha solicitado.

- Un proceso solo puede reiniciarse cuando se le asignan los nuevos recursos que ha solicitado y recupera cualquier recurso que haya sido desalojado mientras esperaba.

ESPERA CIRCULAR

Una forma de garantizar que esta condición nunca se produzca es imponer una ordenación total de todos los tipos de recursos y requerir que cada proceso solicite sus recursos en un orden creciente de enumeración.

Suponiendo se R el conjunto de tipos de recursos.

$$R = \{R_1, R_2, \ldots, R_m\}$$

Se asigna a cada tipo de recurso un número entero unívoco, que permitirá comparar dos recursos y determinar si un precede al otro.

Formalmente se define una función uno-a-uno, donde N es el conjunto de los números naturales.

$$F: R \rightarrow N$$

Por ejemplo, si el conjunto de tipos de recursos R incluye cintas, unidades de disco e impresoras. F se define como:

$$F(\text{unidad de cinta}) = 1$$
$$F(\text{unidad de disco}) = 5$$
$$F(\text{impresora}) = 12$$

Se puede considerar el protocolo, donde cada proceso puede solicitar los recursos sólo en orden creciente de enumeración, es decir, un proceso puede solicitar cualquier número de instancia de un cierto tipo de recurso Rj si y solo si:

$$F(R_j) > F(R_i)$$

En caso de necesitar varias instancias del mismo tipo de recurso, se ejecuta una única solicitud para todos ellos.

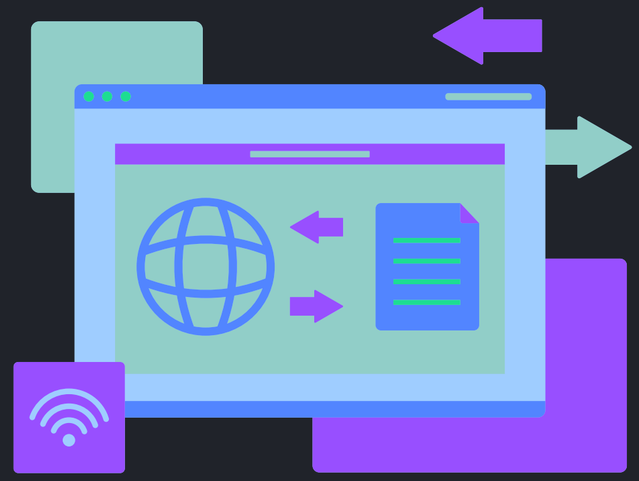
Se puede requerir que, si un proceso solicita una instancia del tipo de recurso R, tiene que liberar primero cualquier recurso Ri, tal que:

$$F(R_i) \geq F(R_j).$$

Definir una ordenación o jerarquía no impide, por si sólo, la aparición de interbloqueos. Es responsabilidad de los desarrolladores, respetar la ordenación.

Además, la función F debería definirse de acuerdo con el orden normal de utilización de los recursos en un sistema.

- Si bien, garantizar que los recursos se adquieran en el orden apropieado es responsabilidad nuestra, puede utilizarse una solución de software para verificar que los bloqueos se adquieren en el orden adecuado y proporcionar las apropidas advertencias.
- Uno de los verificadores existentes del orden de bloqueo, que funciona en las versiones BSD en UNIX, es Witness. Este utiliza bloqueos de exclusión mutua para proteger las secciones criticas. Opera de forma dinámica el orden de los bloqueos en el sistema.



}

Evasión de interbloqueos {

El modelo mas simple y útil requiere que los procesos declaren el numero máximo de recursos

El modelo de evasión de interbloqueos examina dinámicamente el estado de asignación cada recurso para evitar una condición de espera circular

Algoritmos de evasion de interbloqueos

- Estados seguro
- Algoritmo del banquero
  - Algoritmo de seguridad
  - Algoritmo de solicitud de recursos

La prevención de interbloqueos puede tener efectos colaterales (tasa baja de utilización y menor rendimiento).

Un método alternativo, el cual requiere de información adicional sobre como serán utilizados los recursos.
Es decir, necesita conocer la secuencia de solicitudes y liberación de los procesos, con el fin de determinar, si el proceso debe esperar o no, y considerar los recursos disponibles, asignados para las solicitudes y liberaciones futuras.

}

Estado seguro {

¿Qué es?

Un estado es seguro cuando el sistema puede asignar recursos a cada proceso en un orden determinado sin generar un interbloqueo (solo cuando existe una secuencia segura).

Una secuencia segura es aquella donde todos los procesos pueden completar su ejecución sin causar un interbloqueo, es decir, todos los procesos obtienen los recursos necesarios para completar su ejecución incluso si estos están retenidos por otro proceso.

Algoritmo de estado seguro

La definición de este algoritmo asegura que un sistema nunca produzca un interbloqueo, haciendo que la tasa de utilización de los recursos sea menor, pero garantizando siempre el estado seguro.

- **El sistema inicia en estado seguro.**
- **Si un proceso solicita un recurso que este disponible, el sistema decide si asignarlo de forma inmediata o hace esperar al proceso.**
- **La solicitud es concedida solo si la asignación deja al sistema en estado seguro.**

}

Grafo de asignación de recursos

{

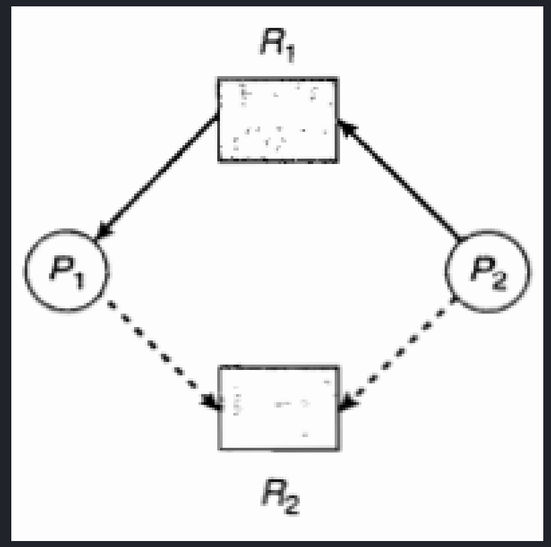
Se utiliza un grafo para representar las relaciones entre procesos y recursos, donde los vértices del grafo son procesos y tipos de recursos, así mismo las aristas son el tipo de relación (solicitud, asignación y declaración).

Aristas de Declaración:

Son “la intención” de un proceso para solicitar un recurso en el futuro, ya que cuando posteriormente se convierten en aristas de solicitud cuando el proceso realmente solicita el recurso y vuelve a ser una arista de declaración cuando el proceso libera el recurso.

Para que el algoritmo pueda funcionar de manera eficiente (estado seguro), debe evitar que las aristas creen un ciclo

No es aplicable para los sistemas de asignación con múltiples instancias



}

Algoritmo del banquero{

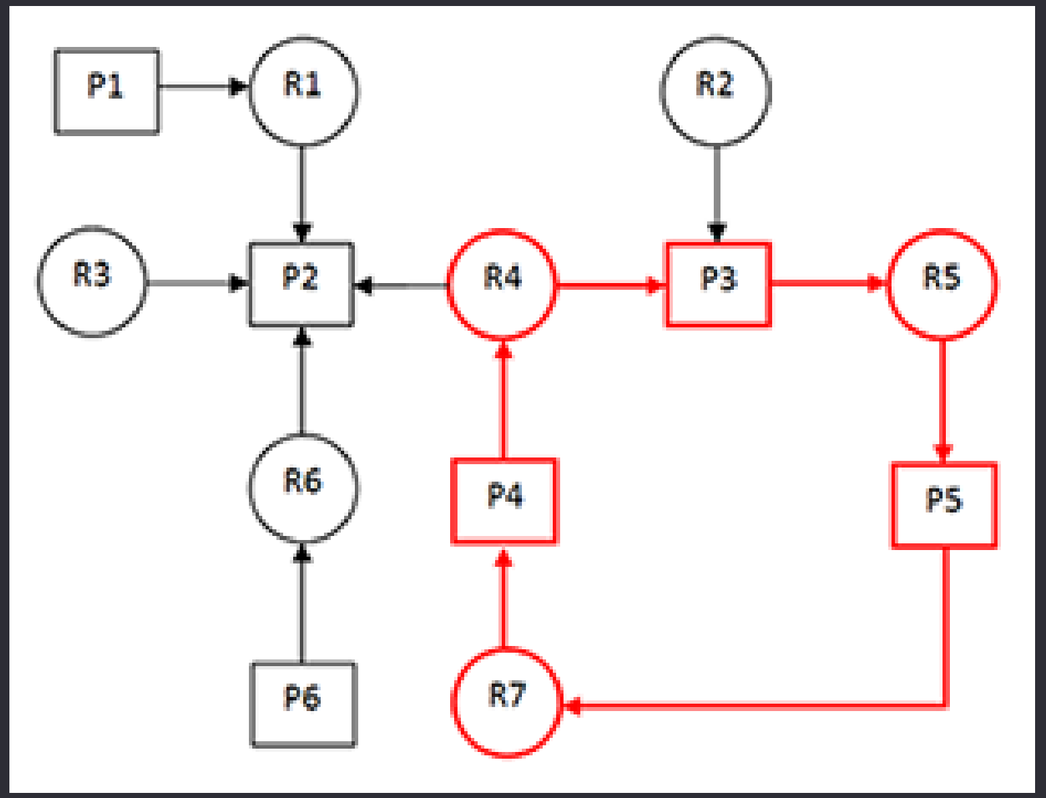
A pesar de ser menos eficiente, requiere que los procesos declaren el numero máximo de recursos que necesita, el cual utiliza para determinar si una asignación propuesta dejaría al sistema en un estado seguro, ya que existe una secuencia segura en la que todos los procesos pueden completar su ejecución, también este nu.

- Availabe: Es un vector de longitud m que indica el número de recursos disponibles de cada tipo.
- Allocation: Es una matriz n x m que define el número de recursos de cada tipo que están asignados actualmente a cada uno de los procesos.
- Request:Es una matriz n x m que especifica la solicitud actual efectuada por cada proceso.

}

Detección de interbloqueos {

En caso de que el sistema no emplee ningún algoritmo de prevención de interbloqueos, entonces puede que se produzca un interbloqueo, en este caso el sistema debe proporcionar un algoritmo que examine el estado del sistema para determinar si se ha producido un interbloqueo y un algoritmo para recuperarse de un interbloqueo.

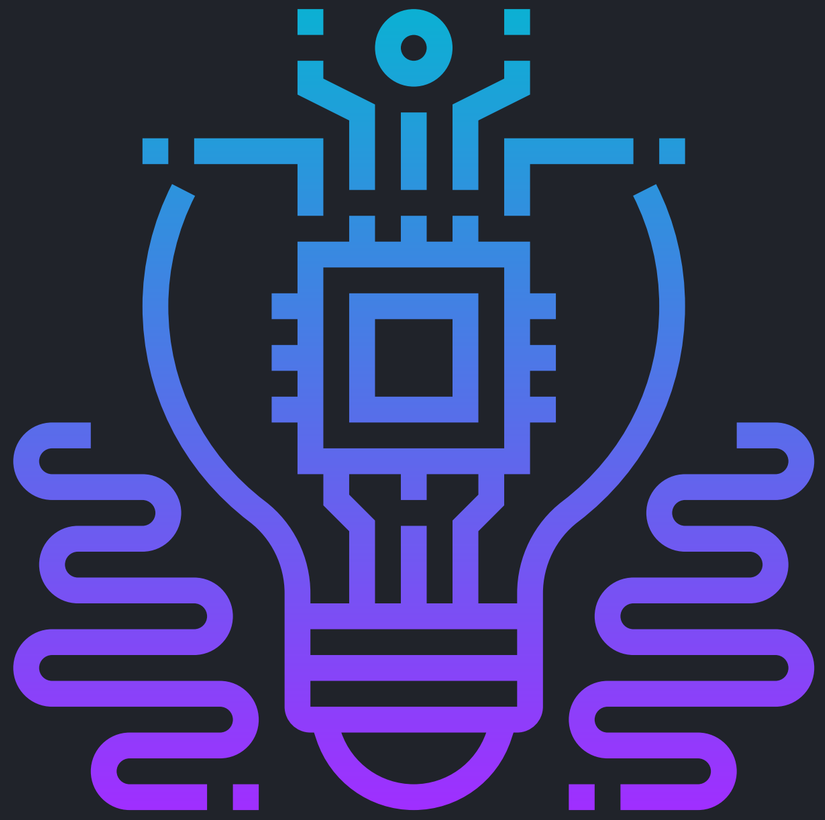


}

Una sola instancia de cada tipo de recurso {

En caso de que todos los recursos tengan una única instancia, entonces puede definirse un algoritmo de detección de interbloqueos que utilice una variante de grafo de asignación de recursos, denominada grafo de espera.

Una arista de Pi a Pj en un grafo de espera implica que el proceso Pi está esperando a que el proceso Pj libere el recurso que necesita el proceso Pi. En un grafo de espera siempre existirá una arista Pi->Pj si y solo si en el grafo de asignación Pi->Rq y Rq->Pj para algún recurso Rq.



}

{

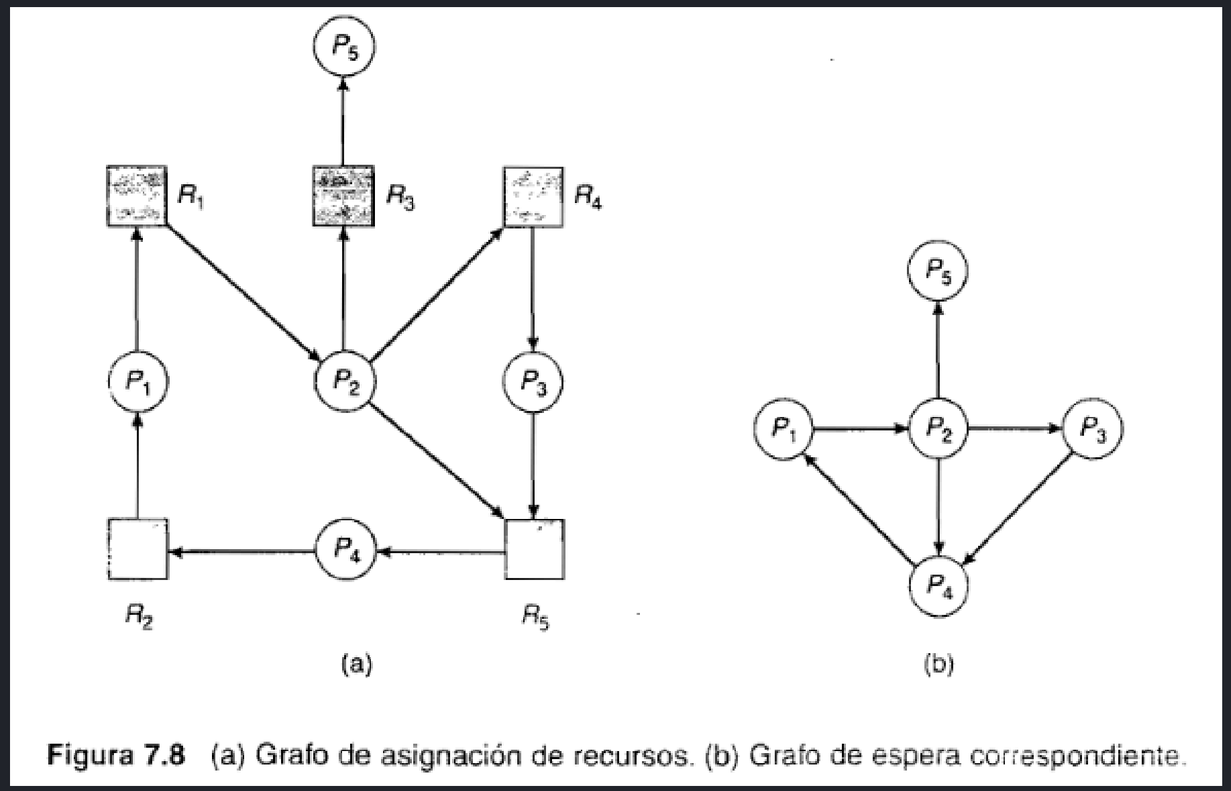


Figura 7.8 (a) Grafo de asignación de recursos. (b) Grafo de espera correspondiente.

Existirá un interbloqueo en el sistema si y solo si el grafo de espera contiene un ciclo. Para detectar los interbloqueos, el sistema debe mantener el grafo de espera e invocar un algoritmo periódicamente que compruebe si existe un ciclo en el grafo.

La complejidad del algoritmo que se usa suele ser de n^2, donde n es el número de vértices del grafo.

}

Varias instancias de cada tipo de recursos{

El esquema del grafo de espera no es aplicable a los sistemas de asignación de recursos con múltiples instancias de un tipo de recurso. En su lugar, se usa un algoritmo que emplea varias estructuras de datos variables con el tiempo, similares a las que se usan en el algoritmo del banquero.

- Availabe: Es un vector de longitud m que indica el número de recursos disponibles de cada tipo.
- Allocation: Es una matriz n x m que define el número de recursos de cada tipo que están asignados actualmente a cada uno de los procesos.
- Request:Es una matriz n x m que especifica la solicitud actual efectuada por cada proceso.

}

{

1. *Work* y *Finish* son vectores de longitud m y n, respectivamente. Inicialmente, *Work* = *Available*. Para $i = 0, 1, \ldots, n-1$, si $Allocation_i \neq 0$, entonces $Finish[i] = false$; en caso contrario, $Finish[i] = true$.
2. Hallar un índice i tal que
   a. $Finish[i] == false$
   b. $Request_i \leq Work$

   Si no existe tal i, ir al paso 4.
3. $Work = Work + Allocation_i$
   $Finish[i] = true$
   Ir al paso 2
4. Si $Finish[i] == false$, para algún i tal que $0 \leq i < n$, entonces el sistema se encuentra en estado de interbloqueo. Además, si $Finish[i] == false$, entonces el proceso P_i está en el interbloqueo.

El algoritmo presentado requiere un orden de m x n^2 operaciones para poder detectar si el sistema se encuentra en un interbloqueo.

}

Utilización de un algoritmo de detección {

El cuándo invocar un algoritmo de detección depende principalmente de dos cosas:

- Con que frecuencia se producirá probablemente un interbloqueo.
- Cuantos procesos se verán afectados por el interbloqueo en caso de que se produzca.

En caso de que se produzcan interbloqueos de manera muy frecuente, el algoritmo de detección debe de invocarse también de forma frecuente. Los recursos asignados a los procesos que se encuentran en un interbloqueo estarán inactivos hasta que este se elimine.

Los interbloqueos se producen 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 casos extremos puede invocarse al algoritmo de detección cada que una solicitud de asignación no es concedida de forma inmediata.

En estos casos, no solamente podemos identificar los procesos que se encuentran en un estado de interbloqueo, sino que también se puede encontrar el proceso que ha causado este estado.

}

{

Cuando existen varios tipos de recursos diferentes, una solicitud puede crear varios ciclos dentro de un grafo.

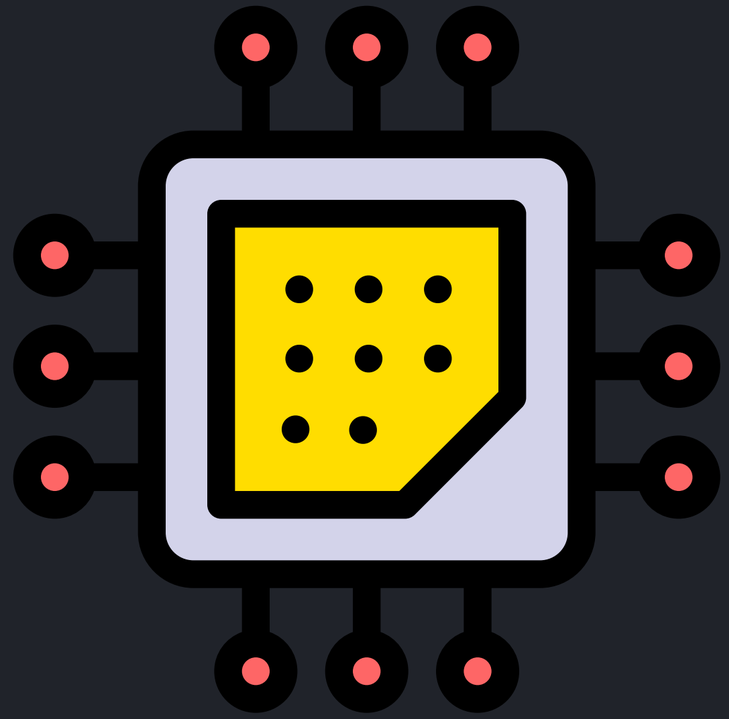
Cuando un algoritmo de detección de interbloqueos se invoca para cada solicitud, se habla también de una sobrecarga al procesador, por lo que es conveniente invocar a este algoritmo en intervalos menos frecuente o en situaciones muy específicas, como, por ejemplo, cuando el rendimiento de la CPU cae por debajo de un dado porcentaje.

}

Recuperación de un interbloqueo {

Cuando el algoritmo de detección determina que existe un interbloqueo, entonces se tiene varias alternativas.

Por lo general existen dos formas para poder romper un interbloqueo, una de ellas es interrumpir uno o más procesos de la cadena de espera circular y la otra es desalojar a alguno de los procesos bloqueados.



}

Terminación de procesos{

Interrumpir todos los procesos bloqueados: Este método claramente interrumpirá el ciclo del interbloqueo, sin embargo, los procesos que se encontraban en este estado puede que hayan consumido un largo periodo de tiempo y probablemente tengan que repetirse más tarde.

Interrumpir un proceso cada vez hasta que el ciclo de interbloqueo se elimine: Este método requiere de una gran cantidad de trabajo adicional, ya que después de haber interrumpido alguno de los procesos, se tiene que llamar nuevamente al algoritmo de detección para verificar si aún existe el interbloqueo.

}

{

La cuestión básicamente es de carácter económico, es decir, debemos interrumpir aquellos procesos que cuya terminación tenga un coste mínimo. Desafortunadamente, el termino de “coste mínimo” no es muy preciso, ya que existen muchos factores para poder seleccionar un proceso:

- La prioridad del proceso.
- Durante cuanto tiempo se ha estado ejecutando y cual es el tiempo que necesita para terminar sus tareas.
- Cuantos y que tipo de recursos ha utilizado.
- Cuantos más recursos necesita para poder completarse.
- Cuantos procesos hará falta terminar.
- Si se trata de un proceso interactivo o de procesamiento por lotes.

}

Apropiación de recursos{

SELECCIÓN DE UNA VÍCTIMA

Al igual que en la terminación de procesos, para determinar el orden de apropiación debemos de minimizar los costes. Los factores de coste pueden incluir parámetros, número de recursos, etc.

ANULACIÓN

En caso de que nos apropiemos del recurso de un proceso, queda una incógnita ¿Qué deberíamos hacer con ese proceso? No se puede dejar que continue con su ejecución ya que no dispone de los recursos necesarios, por lo que debe de devolverse a un estado seguro y reiniciarlo a partir de dicho estado.

Dado que también es difícil determinar un estado seguro, lo más sencillo es realizar una operación de anulación completa y reiniciarlo.

INANICIÓN

¿Cómo se puede asegurar que un proceso no muera por inanición? En un sistema que la selección de la victima se basa en los factores de coste, puede que siempre se elija al mismo proceso como víctima, como resultado, este proceso jamás podrá terminar sus tareas por lo que da lugar a una inanición. Es por esta razón que se debe determina un numero finito para que cada proceso pueda ser tomado como víctima.

}