

Sistemas Operativos 1

Procesos

Concurrencia, exclusión mutua

Edwin Salvador

11 de mayo de 2017

Clase 4

1 Concurrency

- Condición de carrera
- Interacción de procesos
- Requisitos para la exclusión mutua

2 Interbloqueo

- Fundamentos
- Prevención
- Predicción
- Detección

Principios de la concurrencia

- La concurrencia es un factor a tomar en cuenta dentro de un SO, especialmente en tres áreas relacionadas con la gestión de procesos:
 - **multiprogramación** múltiples procesos en un monoprocesador.
 - **multiprocesamiento** múltiples procesos en un multiprocesador.
 - **procesamiento distribuido**. múltiples procesos que se ejecutan en múltiples sistemas de cómputo.
- La concurrencia influye en aspectos del diseño de un SO como:
 - la comunicación entre procesos
 - la compartición de recursos
 - la competencia por recursos
 - la sincronización de actividades de múltiples procesos
 - la reserva de tiempo de procesador para los procesos.

Términos clave de la concurrencia

- **Sección crítica:** sección de código de un proceso que requiere acceso a recursos compartidos.
- **Interbloqueo (*deadlock*):** cuando dos o más procesos son incapaces de actuar porque cada uno está esperando que alguno de los otros haga algo.
- **Círculo vicioso (*livelock*):** cuando dos o más procesos cambian continuamente su estado en respuesta a cambios en los otros procesos, sin realizar ningún trabajo útil.
- **Exclusión mutua:** requisito de que cuando un proceso esté en una sección crítica que accede a recursos compartidos, ningún otro proceso pueda estar en una sección crítica que acceda a ninguno de esos recursos compartidos.

Términos clave de la concurrencia

- **Condición de Carrera** cuando múltiples hilos o procesos leen y escriben un dato compartido y el resultado final depende de la coordinación relativa de sus ejecuciones.
- **Inanición:** Cuando un proceso preparado para avanzar es puesto en espera indefinidamente por el **planificador**; aunque es capaz de avanzar, nunca se le escoge para ser ejecutado.

Problemas presentes en sistemas multiprogramados y multiprocesador

Principios de la concurrencia

Ambos sistemas presentan los siguientes problemas:

- **La compartición de recursos globales.** Si 2 procesos utilizan la misma variable global y ambos realizan lecturas y escrituras, entonces el orden de las lecturas y escrituras es crítico.
- **Es complicado gestionar la asignación de recursos de manera óptima.**
 - El proceso A puede solicitar el uso de un canal de E/S.
 - Se le concede el control y se suspende **justo antes** de utilizar el canal.
 - Si el SO bloquea el canal e impide su utilización por otros procesos puede causar una condición de interbloqueo.

Ejemplo de problemas de concurrencia

Sistema multiprogramado monoprocesador monousuario

```
void eco(){  
    cent = getchar()  
    ;  
    csal = cent;  
    putchar(csal);  
}
```

- Este programa realiza un eco de un caracter; la entrada se obtiene del teclado, una tecla a la vez. Cada caracter es almacenado en cent, luego es transferido a csal y es enviado a la pantalla. Cualquier programa puede llamar repetidamente a este procedimiento para aceptar la entrada del usuario y mostrarla por pantalla.

Considerando que tenemos un sistema multiprogramado monoprocesador y monousuario

- El usuario puede saltar de una aplicación a otra y cada aplicación utiliza el mismo teclado y la misma pantalla.
- Cada aplicación necesita usar el procedimiento `eco`, este procedimiento será compartido y estará cargado en una porción de memoria global para todas las aplicaciones (solo utilizaremos una copia de este procedimiento.)

Ejemplo de problemas de concurrencia

- La compartición de memoria principal permite interacción eficiente entre procesos.
 - ❶ P1 invoca a `eco` y es interrumpido inmediatamente después de que `getchar` devuelva su valor y sea almacenado en `cent`. En este punto el carácter introducido más recientemente, `x`, está almacenado en `cent`.
 - ❷ P2 se activa, invoca a `eco` y se ejecuta hasta concluir, habiendo leído y mostrado en pantalla un único carácter `y`.
 - ❸ P1 retoma. En este instante, el valor de `x` ha sido sobrescrito en `cent` y por tanto se ha perdido. En su lugar, `cent` contiene `y` y es transferido a `csa1` y mostrado en pantalla.
- ¿Qué problema tenemos aquí? el primer carácter se pierde y el segundo es mostrado 2 veces. Esto se da porque la variable `cent` es global y compartida, múltiples procesos tienen acceso a esta variable.
- ¿Cómo solucionamos este problema?

Ejemplo de problemas de concurrencia

- Si permitimos que solo un proceso pueda estar en `eco`, la secuencia anterior sería:
 - ➊ P1 invoca a `eco` y es interrumpido....
 - ➋ P2 se activa e invoca a `eco`. Pero como P1 aún está en `eco` (suspendido), P2 no puede entrar a `eco` y por lo tanto se suspende hasta que el procedimiento `eco` esté disponible.
 - ➌ P1 retoma y completa la ejecución de `eco` y se muestra el caracter `x`
 - ➍ P1 sale de `eco`, se elimina el bloqueo de P2. Y este puede acceder a `eco`.
- Los problemas de concurrencia se pueden dar incluso en un sistema monoprocesador.

1 Concurrency

- Condición de carrera
- Interacción de procesos
- Requisitos para la exclusión mutua

2 Interbloqueo

- Fundamentos
- Prevención
- Predicción
- Detección

Condición de carrera

- Qué es? Cuando múltiples procesos o hilos leen y escriben datos de manera que el resultado final depende del orden de ejecución de las instrucciones en los múltiples procesos. El proceso que actualiza último (el perdedor) determina el valor de a.
 - P3 y P4 comparten las variables b y c
 - Valores iniciales $b = 1$ y $c = 2$.
 - En algún punto P3 ejecuta la asignación $b = b + c$
 - En otro punto P4 ejecuta la asignación $c = b + c$.
 - Los valores finales dependen del orden en que los procesos ejecuten estas dos asignaciones.
 - Cuáles serían los valores si P3 ejecuta su sentencia primero? $b = 3$ y $c = 5$
 - Cuáles serían los valores si P4 ejecuta su sentencia primero? $b = 4$ y $c = 3$

1 Concurrency

- Condición de carrera
- Interacción de procesos
- Requisitos para la exclusión mutua

2 Interbloqueo

- Fundamentos
- Prevención
- Predicción
- Detección

Interacción de procesos

Podemos clasificar las formas de interacción entre procesos en base al grado en que perciben la existencia de los otros. Así tenemos:

- **Procesos que no se perciben entre sí** Son independientes y no se pretende que trabajen juntos. Multiprogramación de múltiples procesos independientes. **Competencia** por recursos. Dos aplicaciones independientes pueden necesitar el mismo disco, impresora o fichero.
- **Procesos que se perciben indirectamente entre sí** No están necesariamente al tanto de la presencia de los demás mediante sus respectivos ID de proceso, pero comparten accesos a algún objeto como bufer de E/S. Estos **cooperan** en la compartición del objeto.
- **Procesos que se perciben directamente entre sí** Son capaces de comunicarse entre sí vía el ID de proceso y son diseñados para trabajar conjuntamente en cierta actividad. Estos también exhiben **cooperación**.

Competencia entre procesos por recursos

Interacción de procesos

- Los procesos concurrentes entran en conflicto entre ellos cuando compiten por el mismo recurso.
- Cada proceso debe dejar inalterado el estado de los recursos (dispositivos E/S, memoria, tiempo de procesador).
- Si dos procesos desean el mismo recurso, el SO reservará el recurso para uno de ellos y el otro tendrá que esperar, por lo tanto será ralentizado (en ciertos casos el proceso bloqueado puede no conseguir nunca el recurso y por tanto nunca terminar).

Problemas de control en procesos en competencia

Los procesos en competencia deben afrontar tres problemas: exclusión mutua, interbloqueo e inanición.

- **Exclusión mutua:**

- Si dos o más procesos requieren acceso a un recurso no compartible como una impresora (**recurso crítico**), es importante que solo se permita un programa en su **sección crítica**.
- ¿Qué pasaría si se le concede el control de una impresora a dos procesos a la vez?
- La exclusión mutua genera dos problemas: interbloqueo e inanición.

Problemas de control en procesos en competencia

- **Interbloqueo:** Si tenemos dos procesos P1 y P2 y dos recursos R1 y R2. Si los dos procesos necesitan acceder a **ambos** recursos para realizar parte de su función, entonces es posible la siguiente situación:
 - El SO asigna R1 a P2 y R2 a P1.
 - Cada proceso está esperando por uno de los recursos y ninguno liberará el recurso que posee hasta conseguir el otro recurso y realizar la tarea que requiere ambos recursos.
 - Ambos procesos están interbloqueados.

Problemas de control en procesos en competencia

- **Inanición:** Tenemos tres procesos P1, P2 y P3 que requieren todos accesos periódicos al recurso R. Podemos tener la siguiente situación:
 - P1 está en posesión del recurso y P2 y P3 están ambos esperando por ese recurso.
 - Cuando P1 termine su sección crítica, debería permitírsele el acceso a R a P2 o P3.
 - Supongamos que el SO le concede el acceso a P3 y que P1 solicita acceso otra vez antes de completar su sección crítica.
 - Si el SO le concede acceso a P1 después de que P3 haya terminado y luego concede acceso alternativamente a P1 y P3, entonces P2 puede denegársele indefinidamente el acceso al recurso aunque no suceda un interbloqueo.

Cooperación entre procesos vía compartición

Interacción de procesos

- **Cubre procesos que interaccionan con otros procesos sin tener conocimiento explícito de ellos.**
- Por ejemplo, muchos procesos pueden tener acceso a ficheros o bases de datos compartidas.
- Los procesos pueden usar y actualizar los datos compartidos sin referenciar otros procesos pero saben que otros procesos pueden acceder a los mismos datos.
- Así, los procesos deben cooperar para asegurar la integridad de los datos compartidos.
- Los problemas de exclusión mutua, interbloqueo e inanición también están presentes ya que los datos están contenidos en recursos (dispositivos, memoria, archivos). Pero en este caso los datos individuales pueden ser accedidos para lectura y escritura, y solo las operaciones de escritura deben ser mutuamente exclusivas.

Problema de coherencia de datos

Cooperación entre procesos vía compartición

- A más de los problemas anteriores, surge un nuevo problema: **la coherencia de datos**.
- **Ejemplo:** Consideremos una aplicación de contabilidad en la que pueden ser actualizados varios datos individuales. Supongamos que dos datos individuales a y b deben mantener la relación $a = b$. Cualquier programa que actualice un valor debe también actualizar el otro. Si tenemos dos procesos:

P1:

```
a = a + 1;  
b = b + 1;
```

P2:

```
b = 2 * b;  
a = 2 * a;
```

- Si el estado inicial es consistente, cada proceso por separado dejará los datos compartidos en un estado consistente.

Problema de coherencia de datos

Cooperación entre procesos vía compartición

- Ahora consideremos que los procesos se ejecutan **concurrentemente** y los dos procesos respetan la exclusión mutua sobre cada dato a y b.

```
a = a + 1;  
b = 2 * b;  
b = b + 1;  
a = 2 * a;
```

- Que pasará al final de esta secuencia si los valores iniciales son $a = b = 1$? la condición $a = b$ no se mantiene. Tendremos $a = 4$ y $b = 3$.
- Esto puede ser evitado declarando en cada proceso la secuencia completa como una sección crítica.

Cooperación entre procesos vía comunicación

- La comunicación es básicamente el paso de mensajes entre los procesos.
- Cuando los procesos cooperan vía comunicación, los procesos involucrados participan en un esfuerzo común que los vincula a todos ellos. La comunicación proporciona una manera de sincronizar o coordinar actividades varias.
- Ya que en el paso de mensajes los procesos no comparten nada, la exclusión mutua no es un requisito de control en este tipo de cooperación. Pero los problemas de inanición e interbloqueo si están presentes (en este caso con referencia a la comunicación).

1 Concurrency

- Condición de carrera
- Interacción de procesos
- Requisitos para la exclusión mutua

2 Interbloqueo

- Fundamentos
- Prevención
- Predicción
- Detección

Requisitos para la exclusión mutua

Cualquier técnica que vaya a proporcionar exclusión mutua debería cumplir los siguientes requisitos:

- 1 Sólo se permite un proceso dentro de su sección crítica.
- 2 Un proceso que se pare en su sección **no** crítica debe hacerlo sin interferir con otros procesos.
- 3 No debe ser posible que un proceso que solicite acceso a una sección crítica sea postergado indefinidamente: ni interbloqueo ni inanición.
- 4 Si ningún proceso está en una sección crítica, y un proceso solicita entrar en su sección crítica debe permitírsele entrar sin demora.
- 5 No se deben hacer suposiciones de las velocidades de ejecución de los procesos ni sobre el número de procesadores.
- 6 Un proceso debe permanecer en su sección crítica sólo por un tiempo finito.

1 Concurrency

- Condición de carrera
- Interacción de procesos
- Requisitos para la exclusión mutua

2 Interbloqueo

- Fundamentos
- Prevención
- Predicción
- Detección

1 Concurrency

- Condición de carrera
- Interacción de procesos
- Requisitos para la exclusión mutua

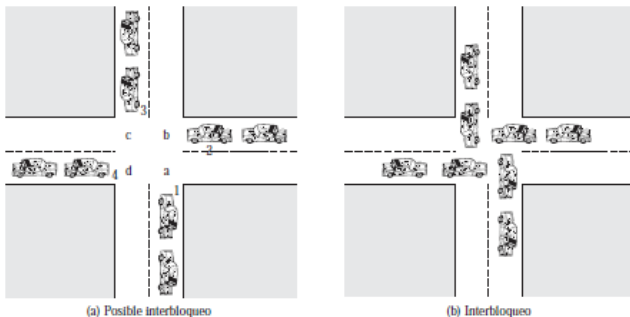
2 Interbloqueo

- Fundamentos
- Prevención
- Predicción
- Detección

Fundamentos del interbloqueo

- Un conjunto de procesos están interbloqueados cuando cada proceso del conjunto está bloqueado esperando un evento (normalmente la liberación de algún recurso requerido) que sólo puede generar otro proceso bloqueado del conjunto.
- El interbloqueo es permanente porque no puede producirse ninguno de los eventos. No existe una solución eficiente.
- Un interbloqueo involucra necesidades de recursos conflictivas de dos o más procesos.

Ilustración de interbloqueo



- Todos los autos llegan al mismo tiempo.
- En un cruce de 4 caminos un auto debe dar paso al autos de la derecha. Esto es efectivo si solo hay 3 autos. En este caso todos los auto dan esperan al de la derecha (interbloqueo).
- Si todos irrespetan la norma anterior y pasan lentamente, cada uno bloqueará a otro auto (interbloqueo).

Interbloqueo en un computador

- Si tenemos dos procesos P y Q ejecutando y requieren el uso exclusivo de recursos durante un tiempo. P (Solicita A, Solicita B, Libera A, Libera B) y Q (Solicita B, Solicita A, Libera B, Libera A).
- Las posibles opciones de ejecución son:
 1. Q adquiere B y, a continuación, A, y, más tarde, libera B y A. Cuando P continúe su ejecución, será capaz de adquirir ambos recursos.
 2. Q adquiere B y, a continuación, A. P ejecuta y se bloquea al solicitar A. Q libera B y A. Cuando P continúe su ejecución, será capaz de adquirir ambos recursos.
 3. Q adquiere B y, a continuación, P adquiere A. El **interbloqueo** es inevitable, puesto que cuando la ejecución continúe, Q se bloqueará a la espera de A y P a la de B.
 4. P adquiere A y, a continuación, Q adquiere B. El **interbloqueo** es inevitable, puesto que cuando la ejecución continúe, Q se bloqueará a la espera de A y P a la de B.
 5. P adquiere A y, a continuación, B. Q ejecuta y se bloquea al solicitar B. P libera A y B. Cuando Q continúe su ejecución, será capaz de adquirir ambos recursos.
 6. P adquiere A y, a continuación, B, y, más tarde, libera A y B. Cuando Q continúe su ejecución, será capaz de adquirir ambos recursos.

Interbloqueo en un computador

- El interbloqueo solo se produce cuando los dos procesos necesitan los dos recursos al mismo tiempo de manera exclusiva.
- Por ejemplo, si P no necesitara a A y B al mismo tiempo (Solicita A , Libera A , Solicita B , Libera B), y Q (Solicita B , Solicita A , Libera B , Libera A), entonces no hay forma de que se produzca un interbloqueo.

Recursos reutilizables

- Un recurso reutilizable es aquél que sólo lo puede utilizar de un proceso en cada momento y que no se destruye después de su uso.
- Ejemplos: procesadores, canales de E/S, memoria principal y secundaria, dispositivos, y estructuras de datos como ficheros, bases de datos y semáforos.

Recursos consumibles

- Los recursos consumibles son los que pueden crearse (producirse) y destruirse (consumirse).
- No hay límite en el número de recursos consumibles de un determinado tipo.
- Un proceso productor puede crear un número ilimitado de estos recursos.
- Cuando un proceso consumidor adquiere un recurso, el recurso deja de existir.
- Ejemplos: las interrupciones, las señales, los mensajes y la información en buffers de E/S.

Condiciones para el interbloqueo

- Se deben presentar 3 condiciones para que se produzca un interbloqueo:
 1. **Exclusión mutua.** Solo un proceso puede utilizar un recurso.
 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.
- Por varios motivos estas tres condiciones son altamente deseables en un sistema. La exclusión mutua asegura la coherencia de resultados. La no expropiación permite que un proceso pueda ser restaurado a un estado previo y repetir sus acciones.
- Si estas tres condiciones están presentes, se puede producir un interbloqueo pero también puede que no. Para que realmente se produzca un interbloqueo se requiere una cuarta condición:
 4. **Espera circular:** Existe una lista de procesos, de tal manera que cada proceso posee al menos un recurso necesitado por el siguiente proceso de la lista.

1 Concurrencia

- Condición de carrera
- Interacción de procesos
- Requisitos para la exclusión mutua

2 Interbloqueo

- Fundamentos
- **Prevención**
- Predicción
- Detección

Técnicas para el tratamiento del interbloqueo

Existen tres estrategias para el tratamiento del interbloqueo:

- Prevenir
- Predecir
- Detectar

Prevención del interbloqueo

- Consiste en diseñar un sistema de manera que se excluya la posibilidad del interbloqueo.
- Existen dos categorías dentro de los métodos de prevención:
 - **Indirecto** previene la aparición de una de las tres primeras condiciones.
 - **Directo** previene que se produzca la cuarta condición.

Técnicas de prevención de las condiciones

- Veamos como se pueden eliminar cada una de las condiciones:
 - **Exclusión mutua:** Por lo general no es posible eliminarla.
 - **Retención y espera:** Se puede eliminar al obligar que un proceso solicite al mismo tiempo todos sus recursos requeridos. El proceso se bloquea hasta que todos los recursos puedan ser asignados. Es ineficiente por 2 razones:
 - Se puede bloquear el proceso por mucho tiempo
 - Los recursos pueden estar inutilizados por mucho tiempo.

También puede ser que el proceso no conozca anticipadamente todos los recursos que necesita.

- **Sin expropiación:** Existen dos maneras:
 - Si un proceso mantiene varios recursos y se le deniega una nueva petición, este proceso debe liberar sus recursos y luego solicitarlos de nuevo si es necesario.
 - Si un proceso solicita un recurso que otro proceso mantiene, el SO puede obligar al proceso a liberar sus recursos.

- **Espera circular:** se puede impedir definiendo un orden lineal entre los distintos tipos de recursos. Se puede asociar un índice a cada tipo de recurso, R_1 y R_2 . Así, si A y B necesitan de R_1 y R_2 y A ha pedido R_1 , B tendrá que esperar a que R_1 esté disponible para luego pedir R_2 , no podrá pedir R_2 antes que R_1 . También puede ser ineficiente, realentizando los procesos y denegando accesos innecesariamente a un recurso.

1 Concurrency

- Condición de carrera
- Interacción de procesos
- Requisitos para la exclusión mutua

2 Interbloqueo

- Fundamentos
- Prevención
- Predicción
- Detección

Predicción del interbloqueo

- A diferencia que en la prevención, la predicción permite las tres condiciones necesarias pero toma decisiones razonables para asegurarse de que nunca se alcanza el punto del interbloqueo.
- Permite más **concurrency** que en la prevención.
- Decide dinámicamente si la petición actual de reserva de un recurso podrá potencialmente causar un interbloqueo. Por lo tanto, requiere el conocimiento de las futuras solicitudes de recursos del proceso.

- Existen dos técnicas para para predecir el interbloqueo:
 - **Denegar el inicio de un proceso** Solo se inicia un proceso si se pueden satisfacer las necesidades máximas de todos los procesos actuales más las del nuevo proceso. No es óptima ya que asume el peor caso en donde todos lo procesos solicitarán sus necesidades máximas simultáneamente.
 - **Denegar la asignación de más recursos:** También se la conoce como *El algoritmo del banquero*. Asegura que el sistema de procesos y recursos está siempre en un **estado seguro**.
 - Un proceso solicita un conjunto de recursos,
 - *Supone* que se concede la petición
 - Actualiza el estado del sistema
 - Determina si el resultado es un estado seguro.
 - En caso afirmativo, se concede la petición. Caso contrario, se bloquea el proceso hasta que sea seguro conceder la petición.sistema.
- Nota:** Un estado seguro es aquél en el que todos los procesos pueden ejecutarse al completo.

1 Concurrencia

- Condición de carrera
- Interacción de procesos
- Requisitos para la exclusión mutua

2 Interbloqueo

- Fundamentos
- Prevención
- Predicción
- Detección

Detección del interbloqueo

- A diferencia de la prevención (muy conservadora), la detección no limita el acceso a los recursos ni restringe las acciones de los procesos.
- Los recursos pedidos se conceden a los procesos siempre que sea posible.
- Periódicamente, el SO realiza un algoritmo que le permite detectar la condición de espera circular.
- Esta comprobación de la existencia de interbloqueo se puede hacer con tanta frecuencia como cada vez que se realiza un petición de recurso, o con menos frecuencia dependiendo de la probabilidad de que ocurra uno.
- La comprobación en cada petición tiene la ventaja de detectar tempranamente los interbloqueos y que el algoritmo es sencillo. Sin embargo, estas comprobaciones consumen tiempo del procesador.

Recuperación de un interbloqueo

Detección del interbloqueo

- Cuando el algoritmo detecta un interbloqueo, debe aplicar una de las siguientes estrategias para recuperarlo:
 1. Abortar todos los procesos involucrados en el interbloqueo. Esta es la solución más utilizada en los SO.
 2. Retroceder cada proceso en interbloqueo a un punto de control establecido y reaniciar todos los procesos. Requiere una implementación de retroceso y reanuncio. El problema es que el interbloqueo puede repetirse aunque gracias al indeterminismo del procesamiento concurrente puede que esto no suceda.
 3. Abortar sucesivamente los procesos en interbloqueo hasta que éste deje de existir. El orden de abortarlos debe basarse en criterios que impliquen el coste mínimo. Después de cada aborto debe invocarse al algoritmo para comprobar que ya no existe el interbloqueo.
 4. Expropiar sucesivamente los recursos hasta que deje de existir el interbloqueo. De igual manera se debe minimizar el coste y se debe verificar que el interbloqueo ha dejado de existir. El proceso expropiado debe retroceder a un punto previo a la adquisición del recurso.