

# Bloqueo mutuo

En sistemas operativos, el **bloqueo mutuo** (también conocido como interbloqueo, traba mortal, *deadlock*, abrazo mortal) es el bloqueo permanente de un conjunto de procesos o hilos de ejecución en un sistema concurrente que compiten por recursos del sistema o bien se comunican entre ellos. A diferencia de otros problemas de concurrencia de procesos, no existe una solución general para los interbloqueos.

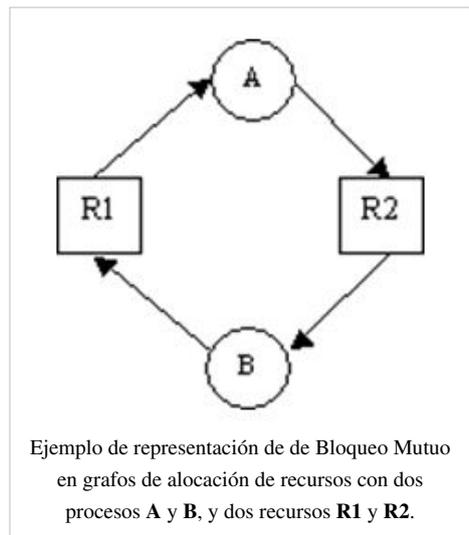
Todos los interbloqueos surgen de necesidades que no pueden ser satisfechas, por parte de dos o más procesos. En la vida real, un ejemplo puede ser el de dos niños que intentan jugar al arco y flecha, uno toma el arco, el otro la flecha. Ninguno puede jugar hasta que alguno libere lo que tomó.

En el siguiente ejemplo, dos procesos compiten por dos recursos que necesitan para funcionar, que sólo pueden ser utilizados por un proceso a la vez. El primer proceso obtiene el permiso de utilizar uno de los recursos (adquiere el lock sobre ese recurso). El segundo proceso toma el lock del otro recurso, y luego intenta utilizar el recurso ya utilizado por el primer proceso, por lo tanto queda en espera. Cuando el primer proceso a su vez intenta utilizar el otro recurso, se produce un interbloqueo, donde los dos procesos esperan la liberación del recurso que utiliza el otro proceso.

## Representación de *Bloqueos Mutuos* usando grafos

El Bloqueo mutuo también puede ser representado usando grafos dirigidos, donde el proceso es representado por un círculo y el recurso, por un cuadrado. Cuando un proceso solicita un recurso, una flecha es dirigida del círculo al cuadrado. Cuando un recurso es asignado a un proceso, una flecha es dirigida del cuadrado al círculo.

En la figura del ejemplo, se pueden ver dos procesos diferentes (**A** y **B**), cada uno con un recurso diferente asignado (**R1** y **R2**). En este ejemplo clásico de bloqueo mutuo, es fácilmente visible la condición de **espera circular** en la que los procesos se encuentran, donde cada uno solicita un recurso que está asignado a otro proceso.



## Condiciones necesarias

También conocidas como *condiciones de Coffman* por su primera descripción en 1971 en un artículo escrito por E.G.Coffman.

Estas condiciones deben cumplirse simultáneamente y no son totalmente independientes entre ellas.

Sean los procesos  $P_0, P_1, \dots, P_n$  y los recursos  $R_0, R_1, \dots, R_m$ :

- Condición de exclusión mutua: Existencia al menos de un recurso compartido por los procesos, al cual sólo puede acceder uno simultáneamente.
- Condición de Posesión y espera: Al menos un proceso  $P_i$  ha adquirido un recurso  $R_i$ , y lo mantiene mientras espera al menos un recurso  $R_j$  que ya ha sido asignado a otro proceso.
- Condición de no expropiación: Los recursos no pueden ser apropiados por los procesos, es decir, los recursos sólo podrán ser liberados voluntariamente por sus propietarios.
- Condición de espera circular: Dado el conjunto de procesos  $P_0 \dots P_n$ ,  $P_0$  está esperando un recurso adquirido por  $P_1$ , que está esperando un recurso adquirido por  $P_2$ , que..., que está esperando un recurso adquirido por  $P_n$ , que está esperando un recurso adquirido por  $P_0$ . Esta condición implica la condición de retención y espera.

## Evitando bloqueos mutuos

Los bloqueos mutuos pueden ser evitados si se sabe cierta información sobre los procesos antes de la asignación de recursos. Para cada petición de recursos, el sistema controla si satisfaciendo el pedido entra en un estado inseguro, donde puede producirse un bloqueo mutuo. De esta forma, el sistema satisface los pedidos de recursos solamente si se asegura que quedará en un estado seguro. Para que el sistema sea capaz de decidir si el siguiente estado será seguro o inseguro, debe saber por adelantado y en cualquier momento el número y tipo de todos los recursos en existencia, disponibles y requeridos. Existen varios algoritmos para evitar bloqueos mutuos:

- Algoritmo del banquero, introducido por Dijkstra.
- Algoritmo de grafo de asignación de recursos.
- Algoritmo de Seguridad.
- Algoritmo de solicitud de recursos.

## Prevención

Los bloqueos mutuos pueden prevenirse asegurando que no suceda alguna de las condiciones necesarias vistas anteriormente.

- Eliminando la exclusión mutua: ningún proceso puede tener acceso exclusivo a un recurso. Esto es imposible para procesos que no pueden ser encolados (puestos en un *pool*), e incluso con colas también pueden ocurrir interbloqueos.
- La condición de retención y espera puede ser eliminada haciendo que los procesos pidan todos los recursos que van a necesitar antes de empezar. Este conocimiento por adelantado muchas veces es imposible nuevamente. Otra forma es requerir a los procesos liberar todos sus recursos antes de pedir todos los recursos que necesitan. Esto también es poco práctico en general.
- La condición de no expropiación puede ser también imposible de eliminar dado que un proceso debe poder tener un recurso por un cierto tiempo o el procesamiento puede quedar inconsistente.
- La condición de espera circular es la más fácil de atacar. Se le permite a un proceso poseer sólo un recurso en un determinado momento, o una jerarquía puede ser impuesta de modo tal que los ciclos de espera no sean posibles.

## Livelock

Un *livelock* es similar a un *deadlock*, excepto que el estado de los dos procesos envueltos en el *livelock* constantemente cambia con respecto al otro. *Livelock* es una forma de inanición y la definición general sólo dice que un proceso específico no está procesando.

En un ejemplo del mundo real, un *livelock* ocurre por ejemplo cuando dos personas, al encontrarse en un pasillo angosto avanzando en sentidos opuestos, y cada una trata de ser amable moviéndose a un lado para dejar a la otra persona pasar, pero terminan moviéndose de lado a lado sin tener ningún progreso, pues ambos se mueven hacia el mismo lado, al mismo tiempo.

*Livelock* es un riesgo con algunos algoritmos que detectan y recuperan los interbloqueos, pues si más de uno toma cartas en el asunto, la detección del interbloqueo puede ser disparada continuamente; pudiendo ser arreglado asegurándose que sólo un proceso (escogido al azar o por prioridad) tome acción.

# Fuentes y contribuyentes del artículo

**Bloqueo mutuo** *Fuente:* <http://es.wikipedia.org/w/index.php?oldid=29896127> *Contribuyentes:* Dodo, Dogor, GermanX, Gothmog, Hari Seldon, Martincarr, Matdrodes, Moriel, NaSz, Sauron, Superzerocool, Tigerfenix, Vanbasten 23, Yucon, 14 ediciones anónimas

# Fuentes de imagen, Licencias y contribuyentes

**Archivo:GrafoDeadlock.jpg** *Fuente:* <http://es.wikipedia.org/w/index.php?title=Archivo:GrafoDeadlock.jpg> *Licencia:* desconocido *Contribuyentes:* -

# Licencia

---

Creative Commons Attribution-Share Alike 3.0 Unported  
<http://creativecommons.org/licenses/by-sa/3.0/>

---