

Un **bloqueo** se produce cuando un proceso está esperando algo que nunca se cumple.

Ejemplo:

Cuando dos procesos P_0 y P_1 quieren tener acceso simultáneamente a dos recursos r_0 y r_1 , es posible que se produzca un bloqueo de ambos procesos. Si P_0 accede con éxito a r_1 y P_1 accede con éxito a r_0 , ambos se quedan atrapados intentando tener acceso al otro recurso.

Se tienen que cumplir cuatro condiciones para que sea posible que se produzca un bloqueo entre procesos:

- 1 los procesos tienen que compartir recursos con exclusión mutua
- 2 los procesos quieren acceder a un recurso más mientras ya tienen acceso exclusivo a otro
- 3 los recursos solo permiten ser usados por menos procesos que lo intentan al mismo tiempo
- 4 existe una cadena circular entre peticiones de procesos y asignaciones de recursos

Un problema adicional con los bloqueos es que es posible que el programa siga funcionando correctamente según la definición, es decir, el resultado obtenido es el resultado deseado, pero algunos de sus procesos están bloqueados durante la ejecución (es decir, se produjo solamente un bloqueo parcial).

Existen algunas técnicas que se pueden usar para que no se produzcan bloqueos:

- Detectar y actuar
- Evitar
- Prevenir

Se implementa un proceso adicional que vigila si los demás forman una cadena circular.

Más preciso, se define el grafo de asignación de recursos:

- Los procesos y los recursos representan los nodos de un grafo.
- Se añade cada vez una arista entre un nodo tipo recurso y un nodo tipo proceso cuando el proceso ha obtenido acceso exclusivo al recurso.
- Se añade cada vez una arista entre un nodo tipo recurso y un nodo tipo proceso cuando el proceso está pidiendo acceso exclusivo al recurso.
- Se eliminan las aristas entre proceso y recurso y al revés cuando el proceso ya no usa el recurso.

Cuando se detecta en el grafo resultante un ciclo, es decir, cuando ya no forma un grafo acíclico, se ha producido una posible situación de un bloqueo.

Se puede reaccionar de dos maneras si se ha encontrado un ciclo:

- No se da permiso al último proceso de obtener el recurso.
- Sí, se da permiso, pero una vez detectado el ciclo se aborta todos o algunos de los procesos involucrados.

Sin embargo, las técnicas pueden dar como resultado que el programa no avance, incluso, el programa se puede quedar atrapado haciendo trabajo inútil: crear situaciones de bloqueo y abortar procesos continuamente.

El sistema no da permiso de acceso a recursos si es posible que el proceso se bloquee en el futuro.

Un método es el algoritmo del banquero (Dijkstra) que es un algoritmo centralizado y por eso posiblemente no muy practicable en muchas situaciones.

Se garantiza que todos los procesos actúan de la siguiente manera en dos fases:

- 1 primero se obtiene todos los cerrojos necesarios para realizar una tarea, eso se realiza solamente si se puede obtener todos a la vez,
- 2 después se realiza la tarea durante la cual posiblemente se liberan recursos que no son necesarias.

Se puede prevenir el bloqueo siempre y cuando se consiga que alguna de las condiciones necesarias para la existencia de un bloqueo no se produzca.

- 1 los procesos tienen que compartir recursos con exclusión mutua
- 2 los procesos quieren acceder a un recurso más mientras ya tienen acceso exclusivo a otro
- 3 los recursos no permiten ser usados por más de un proceso al mismo tiempo
- 4 existe una cadena circular entre peticiones de procesos y asignación de recursos

los procesos tienen que compartir recursos con exclusión mutua:

- No se da a un proceso directamente acceso exclusivo al recurso, si no se usa otro proceso que realiza el trabajo de todos los demás manejando una cola de tareas (por ejemplo, un demonio para imprimir con su cola de documentos por imprimir).

los procesos quieren acceder a un recurso más mientras ya tienen acceso exclusivo a otro:

- Se exige que un proceso pida todos los recursos que va a utilizar al comienzo de su trabajo

los recursos no permiten ser usados por más de un proceso al mismo tiempo:

- Se permite que un proceso aborta a otro proceso con el fin de obtener acceso exclusivo al recurso. Hay que tener cuidado de no caer en *livelock*
- (Separar lectores y escritores alivia este problema también.)

existe una cadena circular entre peticiones de procesos y asignación de recursos:

- Se ordenan los recursos linealmente y se fuerza a los procesos que accedan a los recursos en el orden impuesto. Así es imposible que se produzca un ciclo.

Un ejemplo de un bloqueo en Java muestra el siguiente trozo de código, incluso si se asume que un hilo ya está durmiendo. ¿Por qué?

hilo0:

```
synchronized(A) {  
    ...  
    synchronized(B) {  
        ...  
        A.notify();  
        B.wait();  
    }  
}
```

hilo1:

```
synchronized(B) {  
    ...  
    synchronized(A) {  
        ...  
        B.notify();  
        A.wait();  
    }  
}
```