Problema del barbero durmiente

En ciencias de la computación, el **problema del barbero durmiente** es un problema de sincronización. El problema consiste en una barbería en la que trabaja un barbero que tiene un único sillón de barbero y varias sillas para esperar. Cuando no hay clientes, el barbero se sienta en una silla y se duerme. Cuando llega un nuevo cliente, éste o bien despierta al barbero o —si el barbero está afeitando a otro cliente— se sienta en una silla (o se va si todas las sillas están ocupadas por clientes esperando). El problema consiste en realizar la actividad del barbero sin que ocurran condiciones de carrera. La solución implica el uso de semáforos y objetos de exclusión mutua para proteger la sección crítica.

El barbero le está cortando el pelo al cliente

Un semáforo es una variable protegida (o tipo abstracto de datos) que constituye el método clásico para restringir o permitir el acceso a recursos compartidos (por ejemplo, un recurso de almacenamiento) en un entorno de multiprocesamiento. Fueron inventados por Edsger Dijkstra y se usaron por primera vez en el sistema operativo THEOS.

En electrónica y en programación concurrente, se conoce como condición de carrera al error que se produce en programas o circuitos lógicos que no se han construido adecuadamente para su ejecución simultánea con otros procesos.

Implementación

- El próximo pseudo-código garantiza la sincronización entre el barbero y el cliente, pero puede llevar a inanición del cliente. *wait()* y *signal()* son funciones provistas por el semáforo.
- Se necesita:

```
Semáforo barberoListo = 0     // (Mutex, sólo 1 ó 0)
Semáforo sillasAccesibles = 1 // (Mutex) Cuando sea 1, el número de sillas libres puede aumentar o disminuir
Semáforo clientes = 0         // Número de clientes en la sala de espera
int sillasLibres = N          // N es el número total de sillas
```

- Función barbero (Proceso/hilo-thread):

```
while(true)                   // Ciclo infinito
{
   wait(clientes)             // Tratar de atender a un cliente. Si no hay, ir a dormir.
   wait(sillasAccesibles)     // (Ya está despierto) Tratar de modificar el nro. de sillas disponibles.
   sillasLibres += 1          // Queda disponible una silla en la sala.
   signal(barberoListo)       // El barbero está listo para cortar.
   signal(sillasAccesibles)   // Ya no se modificará el nro. de sillas libres. Desbloquear.
   // Aquí el barbero corta el pelo de un cliente (zona de código no crítico).
}
```

- Función cliente (Proceso/hilo-thread):

```
wait(sillasAccesibles)       // Trata de acceder a una silla.
if (sillasLibres > 0)        // Si hay sillas disponibles.
{
   sillasLibres =- 1         // Se sienta en una silla.
   signal(clientes)          // Avisar al barbero (el cual está esperando) que haya un cliente.
   signal(sillasAccesibles)  // Ya no se necesita bloquear el nro de sillas libres.
   wait(barberoListo)        // El cliente espera a que el barbero esté listo para atenderlo.
   // Se le corta el pelo al cliente.
}
```

```
else                          // Si no hay sillas libres.
{
   signal(sillasAccesibles)  // Desbloquear la modificación de sillas libres para otros procesos/hebras.
   // El cliente se va de la barbería
}
```

Véase también

- Problema de la cena de los filósofos

Enlaces externos

- Wikilibros tiene una solución a este problema en el manual de Programación en Ada