Problemas de Concurrencia

1. Problema de los lectores y los escritores

Un conjunto de procesos comparten datos. Una parte de estos procesos (lectores) sólo leen los datos, mientras que el resto (escritores) leen y escriben. Varios lectores pueden acceder simultáneamente a los datos compartidos, pero se debe evitar que accedan simultáneamente un proceso escritor y cualquier otro proceso.

2. Problema del productor y el consumidor

Se dispone de un conjunto de tampones (*buffers*), cada uno con capacidad para almacenar un dato. Los procesos que acceden concurrentemente a los tampones son de dos tipos:

- Consumidores: extraen el dato de un tampón y lo consumen.
- Productores: producen un dato y lo almacenan en un tampón.

Los consumidores deben esperar cuando intentan extraer un dato de un tampón vacío, y los productores deben esperar cuando intentan almacenar un dato en un tampón que está lleno.

3. Problema de los filósofos

Cinco filósofos dedican su tiempo a pensar y comer, alternativamente. Los filósofos están sentados en una mesa circular con cinco sitios. En el centro de la mesa hay un cuenco de arroz. En la mesa hay cinco palillos, uno a cada lado de cada filósofo. Para comer, un filósofo necesita usar los dos palillos que tiene a cada lado. Si otro filósofo ha cogido uno de los palillos, deberá esperar a que lo deje en la mesa.

4. Gestión de un recurso con prioridades

Se dispone de un recurso compartido por un conjunto de procesos. Los procesos pueden solicitar el recurso con diferente urgencia: alta, media y baja. Mientras que el recurso está ocupado, los procesos que lo soliciten deben esperar hasta que se libere. Si el recurso está libre y no hay procesos esperando, se concede al primer proceso que lo solicite. Cuando el recurso se libere, si hay procesos esperando se concederá a uno de los procesos en espera con la prioridad más urgente.

5. Problema de los caníbales

Una tribu de caníbales come de una gran marmita común que tiene una capacidad de *n* raciones de estofado. Cuando un caníbal quiere comer, se sirve él mismo de la marmita, a menos que esté vacía. Si la marmita está vacía, el caníbal despierta al cocinero y espera a que el cocinero haya rellenado la marmita. Cuando lo ha hecho, y antes de dejar comer a otro caníbal, come él mismo. Después de alguna otra actividad distinta, vuelve a querer comer. El cocinero, por su parte, vuelve a dormir cuando ha rellenado la marmita.

Se debe desarrollar un monitor (*Marmita*) con dos métodos, uno que invocan los canibales cuando quieren come (*comer*), y otro que invoca el cocinero para rellenar la marmita (*rellenar*). La solución debe evitar interbloqueos, y debe permitir que se despierte al cocinero sólo cuando la marmita esté vacía.

6. Problema del barbero durmiente

Una barbería está compuesta por una sala de espera con n sillas y la habitación del barbero con el sillón correspondiente. Si no hay clientes el barbero se duerme. Si un cliente entra en la sala de espera y todas las sillas están ocupadas, abandona la barbaría. Si el barbero está ocupado y hay sillas disponibles, entonces el cliente se sienta a esperar en una de las sillas. Si el barbero está dormido y llega un cliente, le despierta.

7. Problema de los cigarros y los fumadores

Se tiene un sistema con tres fumadores y un estanqeuro. Cada fumador está continuamente liando un cigarrillo y después se lo fuma. Para liar y fumar un cigarrillo, el fumador necesita tres ingredientes: tabaco, papel y cerillas. Uno de los fumadores tiene tabaco, otro papel y el tercero cerillas. El estanquero tiene una cantidad infinita de los tres materiales. El estanquero deja dos de los ingredientes en una mesa. El fumador que tiene el ingrediente que falta coge esos dos, lía y se fuma un cigarrillo, avisándo al estanquero cuando termina. Éste pone otros dos de los tres ingredientes en la mesa, y el ciclo se repite.

8. Gestión de recursos

Se tiene un sistema concurrente en el que un conjunto de procesos realizan por petición tres tipos de servicios (S1, S2, S3). El sistema dispone de tres recursos (R1, R2, R3), que necesitan los procesos para ejecutar los servicios, según la relación siguiente:

- Para ejecutar S1 se necesita R1 y R2.
- Para ejecutar S2 se necesita R2 y R3.
- Para ejecutar S3 se necesita R3 y R1.

Cuando se solicita un servicio a un proceso, en primer lugar, éste solicita todos los recursos necesarios y cuando los tiene asignados, ejecuta el servicio. Cuando deja de necesitar alguno de los recursos, lo libera. No vuelve a aceptar una petición de servicio (y por tanto solicitar nuevos recursos) hasta que no haya terminado el previo y, por tanto, haya liberado todos los recursos asignados.

Se pide diseñar un monitor o un objeto protegido para gestionar los recursos, que proporcione sólo las dos operaciones siguientes:

- *Solicitar_Recursos(S : T_Servicio)*: Esta operación permite a un proceso solicitar los recursos necesarios para realizar un servicio. Si no está disponible alguno de los recursos necesarios, se bloquea al proceso solicitante hasta que los dos estén disponibles.

- *Liberar_Recurso(R : T_Recurso)*: Esta operación libera uno de los recursos necesarios para realizar un servicio. Se debe comprobar si esta acción puede desbloquear a algún proceso.

9. Puente con limitación de vehículos

En una carretera de dos carriles que discurre de Norte a Sur hay un puente estrecho de un solo carril. Por ese puente solo pueden circular vehículos en un solo sentido al mismo tiempo, con la regla de que, estando los vehículos circulando en un sentido, cuando haya N o más vehículos (N es un parámetro del problema) esperando a cruzar en sentido contrario, se impide que entren nuevos vehículos al puente en el sentido actual de la marcha, para permitir cambiar ésta cuanto antes. En el caso de que haya N o más vehículos esperando a cruzar en cada uno de los dos sentidos, se debe dar prioridad a los vehículos que suben, es decir, que van de Sur a Norte.

Se pide escribir un monitor con cuatro operaciones (Entrar_subiendo, Salir_subiendo, Entrar_bajando, Salir_bajando) que implemente el protocolo anterior.