# Ejemplos para sincronización de procesos

a) Un río es compartido por misioneros y caníbales, solo se utiliza un bote de tres espacios para cruzar el río. Con el fin de garantizar la integridad de los misioneros, no se pueden colocar dos caníbales y un misionero juntos en el mismo viaje, las demás combinaciones son válidas. Cada personaje está representado por un hilo, Caníbal y Misionero. Se sugiere la creación de dos métodos **LlegaM()** y **LlegaC()** que llaman los hilos respectivamente, ambos se encargan de realizar la sincronización. Puede existir otro método denominado **Cruzar()** que se llama cuando el bote está listo para cruzar.

b) En una carretera de dos carriles que discurre de Norte a Sur (y de Sur a Norte) 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.

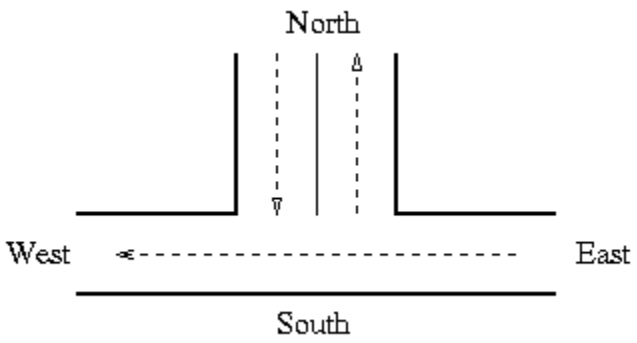
Se pide escribir un objeto protegido dotado de cuatro operaciones (Entrar_subiendo, Salir_subiendo, Entrar_bajando, Salir_bajando) que implemente el protocolo anterior.

c) Se dispone de un sistema de 3 procesos fumadores y uno estanquero. El estanquero dispone de los tres ingredientes que son necesarios para liar un cigarrillo: papel, tabaco y mechero en cantidades infinitas.
El funcionamiento del sistema es como sigue:
o Cada fumador está continuamente liando un cigarrillo y después se lo fuma. Para ello necesita obtener una unidad de cada uno de los tres ingredientes.
o El estanquero pone en una mesa dos unidades cada vez, correspondientes a dos ingredientes aleatorios.
o El fumador que tiene el ingrediente que falta, lía un cigarrillo y se lo fuma. El estanquero repone los ingredientes consumidos inmediatamente y los sustituye por otros dos.

Problema de *los fumadores de cigarrillos*. Considere un sistema con tres procesos *fumador* y un proceso *agente*. Cada fumador pasa todo su tiempo enrollando cigarrillos y fumándolos. Sin embargo, para poder hacerlo, el fumador necesita tres ingredientes: tabaco, papel y cerillas. Uno de los procesos fumadores tiene papel, el otro tiene tabaco y el tercero tiene cerillas. El agente tiene una reserva inagotable de los tres productos. El agente coloca dos de los productos en la mesa. Entonces, el fumador que tiene el ingrediente que falta enrolla y fuma un cigarrillo, enviando una señal al agente cuando termina. A continuación, el agente coloca otros dos de los tres ingredientes en la mesa, y el ciclo se repite. Escribe un programa que sincronize el agente y los fumadores.

d) *El problema del barbero dormilón*. Una barbería tiene una sala de espera con **n** sillas, y una habitación con un sillón donde se atiende a los clientes. Si no hay clientes el barbero se duerme. Si un cliente entra en la barbería pero todas las sillas están ocupadas, entonces se va, dejando la barbería. Si el barbero esta ocupado entonces el cliente se sienta en una de las sillas disponibles. Si el barbero esta dormido el cliente lo despertará. Escribir una aplicación que coordine el barbero y sus clientes de forma adecuada utilizando semáforos.

e) Usted ha sido contratado “ad-honorem” por el MOPT a fin de automatizar el tráfico en una intersección de tres vías, tal y como se muestra en la siguiente figura:




La calle este-oeste es de una sola vía, dirigida hacia el oeste, la otra es de dos vías, una en cada dirección, y está orientada de norte a sur. Las reglas en la intersección para evitar colisiones son las siguientes:

1. Cuando un auto (proceso) llega a la intersección debe llamar a un método denominado Enter(inDir, outDir), donde inDir y outDir son parámetros cuyos valores están definidos por una de las constantes NORTE, SUR, ESTE, u OESTE. El parámetro inDir es la dirección en la que el auto llega a la intersección y outDir es la dirección en la que el auto pretende salir. Este procedimiento regresa únicamente cuando es seguro cruzar la intersección.
2. Cuando el auto deja la intersección debe llamar a otro método Leave(outDir), dónde outDir se define de la misma manera que antes.
3. Los autos pueden continuar en línea recta o hacer giros permitidos, los giros en U son ilegales.
4. Si hay autos procedentes del NORTE y ESTE esperando, ambos pueden continuar al mismo tiempo si no hay colisiones.
5. Si hay autos procedentes del NORTE y ESTE esperando y no pueden continuar con seguridad, entonces deberán alternarse el paso (para evitar inanición).

Se le solicita escribir el código para los métodos Enter() y Leave(). Puede suponer que existe un método denominado CruzarIntersección(), que puede utilizar para simular el paso de los autos por la intersección. No existe ninguna información de que tanto tiempo toma ejecutar este método.

Debe escribir tres versiones de este programa:

1. (15 puntos) El primer programa debe ser escrito utilizando **Semáforos**, suponga que cada auto es representado por un hilo (proceso). Debe escribir el código que emplea cada auto.
2. (15 puntos) El segundo programa debe emplear **monitores**. También debe suponer que cada auto es representado por un hilo (proceso). Debe declarar cuáles son los datos mantenidos por el monitor, los métodos empleados para la sincronización y el código que ejecutaría cada auto para ser sincronizado adecuadamente.
3. (15 puntos) El tercer programa debe emplear “Envio de Mensajes”. Suponga de nuevo que los autos son representados por hilos (procesos) que NO utilizan datos por memoria compartida, únicamente pueden intercambiar mensajes sin tipos a través de buzones.

   Existen dos métodos para realizar la comunicación: Send, Receive, y un constructor Buzon. La operación Send recibe una tira de caracteres como parámetro, no se bloquea y podemos suponer que no ocurren errores en la transmisión, por lo tanto, luego de esta operación el mensaje estaría

en la cola respectiva. Cada buzón posee un identificador único definido por una tira de caracteres que recibe el constructor como parámetro.

La operación Receive es bloqueada hasta que un mensaje arribe. Si un mensaje ya está disponible, o llega luego de que un receptor ha sido bloqueado, entonces esta operación regresa y retorna el mensaje recibido.

f) Santa Claus tiene dos talleres pequeños, en los que sólo pueden trabajar dos duendes a la vez. Nos referiremos a estos talleres como *H* y *L*, donde se fabrican juguetes de alta tecnología (*H*) y de poca tecnología (*L*). Existen cuatro duendes que pueden hacer juguetes para Santa en estos talleres. Llamaremos a los duendes *G*, *D*, *T* y *J*. Los duendes gastan su tiempo trabajando o durmiendo. En cualquier momento, un duende que esté durmiendo puede ser despertado por Santa para asignarle un trabajo particular en un taller determinado. Vamos a escribir esta solicitud como Santa->inicia_trabajo( duende, taller ). Cuando un duende quiere dejar el taller debe solicitarlo a Santa primero. Vamos a escribir esta solicitud como Santa->termina_trabajo( duende, taller ). Para cada tipo de solicitud Santa puede concederla o, en su defecto, hacer que el duende espere hasta que la solicitud pueda ser concedida.

Esta es una lista de las reglas que Santa tiene designadas para asegurar la máxima productividad.

- No pueden estar más de dos duendes en cada taller.
- El duende *G* no puede trabajar con otro en el mismo taller.
- El duende *D* nunca puede estar solo en un taller.
- Los duendes *T* y *J*, por sus constantes pleitos, no pueden estar juntos en el mismo taller.

Muestre cómo utilizar variables de condición y candados (locks) para implantar el taller de Santa.

g) Una alcancía es compartida por varios procesos, ésta posee una propiedad denominada “saldo” y dos métodos **Retirar( monto )** y **Guardar( monto )**, El método Guardar( monto ) simplemente suma el monto indicado como parámetro al saldo, eventualmente despierta a los procesos Retirar( monto ) que se han quedado suspendidos porque no había suficientes fondos al momento de hacer el retiro. El método Retirar( monto ) intenta sacar de la alcancía el monto indicado como parámetro, pero si no hay suficientes fondos el proceso espera hasta que su solicitud pueda ser satisfecha. Su tarea es implantar ambos métodos utilizando regiones críticas condicionales de manera que se sincronicen los procesos que utilizan la alcancía.