Sesión 10: Sincronización mediante monitores

- Consideremos un puente estrecho, que cruza un río de norte a sur
- En cada momento, solo pueden estar pasando vehículos en una dirección
  - En caso contrario habría accidentes
- Cada coche lleva un sistema que, conectándose con el puente, puede saber cuántos coches hay en el puente y en qué dirección van
- Un coche, al llegar al puente, actualiza la información, y debe parar el coche si hay coches en dirección contraria
- También actualiza la información al dejar el puente

Sesión 10: Sincronización mediante monitores

- Se pide diseñar el programa que simula el comportamiento del sistema, asegurando que no nunca hay dos coches en sentido contrario simultáneamente
  - Olvidamos, de momento, posibles problemas de inanición

```
...

Process cocheNS::
   //llega puente, actualiza infor-
   //mación y espera si necesario

   //atraviesa puente

   //actualiza info
end

Process cocheSN::
   . . .
end
```

Sesión 10: Sincronización mediante monitores

- Se trata de diseñar una segunda solución que evite problemas de inanición
- Para ello, el sistema conoce, además, cuántos coches están esperando en cada lado del puente
- Así a la hora de permitir entrar un nuevo coche, no solo mirará que no haya ninguno en dirección contraria, sino que podrá usar la información de cuántos coches hay esperando para entrar

Sesión 11: Sincronización mediante monitores

- Dos tipos de procesos, A y B, entran y salen de una habitación.
- Para abandonar la habitación se han de cumplir ciertos requisitos:
  - Un proceso A que entra no puede salir hasta que se encuentre en la habitación con dos procesos B
  - Un proceso B sólo puede salir si se ha encontrado con un proceso A
- Hay que diseñar un monitor para la sincronización de los procesos
- Hay que escribir el código de los procesos de tipo A y B
  - **Nota**: se pide hacer una versión para cada una de las dos posibles interpretaciones

Sesión 12: Sincronización mediante monitores

- Se pide diseñar un programa concurrente correspondiente a un sistema con:
  - un proceso productor de mensajes
  - "n" procesos consumidores de mensajes
  - un buffer compartido con capacidad para un mensaje
- El productor: produce mensajes y los deposita en el buffer
- Un consumidor: debe leer (una vez) cada mensaje
- Requisitos:
  - no se deben perder mensajes
  - todos los consumidores leen todos los mensajes
- Nota: simulamos el buffer con un string, y lo mismo los mensajes

Sesión 12: Sincronización mediante monitores

- Esquema:

```
Process productor::
   while true
      //produce mensaje m
      //lo inserta en el buffer cuando haya hueco
      //espera a que los "n" clientes lo hayan leído
   end
end

Process cliente(i:1..n)::
   while true
      //espera que haya un mensaje nuevo
      //leo el mensaje
   end
end
```