

Sesión 5: Diseño de programas concurrentes

- **El problema de lectores/escritores:**

Consideremos un sistema en el que dos tipos de procesos (lectores y escritores) comparten una base de datos.

- Los lectores realizan consultas a la base de datos, pero no la modifican.
- Los escritores consultan y modifican la base de datos. Para evitar las interferencias entre las transacciones, y con el fin de mantener la base de datos coherente pero a su vez, evitar secuenciar el sistema más allá de lo necesario, se han de respetar los siguientes requisitos:

- Requisitos:

- Un proceso escritor debe tener el acceso a la base de datos de manera exclusiva
- Si ningún proceso escritor está accediendo a la base de datos, cualquier número de lectores pueden estar, simultáneamente, accediendo a la base de datos.

```
Process lector(i:1..n)::  
    while true  
        //protocolo de entrada  
        //acceso a la bbdd  
        //protocolo de salida  
    end  
end  
  
Process escritor(i:1..m)::  
    while true  
        //protocolo de entrada  
        //acceso a la bbdd  
        //protocolo de salida  
    end  
end
```

Sesión 5: Diseño de programas concurrentes

- Un club de remo pone a disposición de sus usuarios un servicio de préstamo para que puedan practicar deporte.
- El servicio presta **canoas, remos y chalecos** salvavidas.
- En el caso de las canoas, tras 10 usos se debe realizar una operación de mantenimiento, por lo que se requiere llevar una contabilidad de cuántas veces ha sido usada cada canoa desde la última operación de mantenimiento.

Sesión 5: Diseño de

- Se pide diseñar el programa que simula el comportamiento del sistema, a partir del siguiente esquema:

```
constant integer
```

```
    can := 20, rem := 25,  
    salv := 25, maxUsos := 10
```

```
...
```

```
Process socio(i:1..1000)::
```

```
    ...
```

```
    while true
```

```
        // coger una canoa,
```

```
        // dos remos y dos chalecos
```

```
        // usalos el tiempo deseado
```

```
        // devolver lo prestado
```

```
    end
```

```
end
```

```
Process mantenimiento::
```

```
    ...
```

```
    while true
```

```
        // esperar a que haya
```

```
        // alguna canoa que revisar,
```

```
        // revisarla y repararla si es
```

```
        // necesario
```

```
    end
```

```
end
```

Sesión 6: Diseño de programas concurrentes

- Un centro universitario ha organizado su biblioteca como sigue:
 - Hay 10 salas de estudio de pequeñas dimensiones
 - Cada sala puede ser utilizada por un único alumno, o por dos si ambos están cursando la misma titulación
 - Los dos alumnos que compartan sala no tienen por qué comenzar o finalizar su estudio al mismo tiempo
- El centro tiene actualmente matriculados 100 alumnos y oferta 3 titulaciones diferentes

Sesión 6: Diseño de programas concurrentes

- Comportamiento repetitivo de un alumno:
 - Busca un hueco disponible en cualquier sala de estudio (que respete las condiciones establecidas arriba)
 - Estudia un tiempo (aleatorio), finaliza su estudio y sale de la sala.
 - Luego dedica un tiempo (aleatorio) a alguna actividad de ocio
- Cada alumno tiene un identificador de estudiante único y está matriculado en una sola titulación
- Hay una persona de mantenimiento
 - Las salas se limpian en orden, una detrás de otra, y únicamente cuando están desocupadas
 - Si debe procederse a la limpieza de una sala y está ocupada, la persona encargada tendrá que esperar
 - Mientras que una sala está siendo objeto de limpieza no estará disponible para los alumnos.

Sesión 6: Diseño de programas concurrentes

- Se pide escribir un programa concurrente que simule el funcionamiento del sistema descrito

```
constant integer nAl := 100 //100 alumnos  
constant integer nEst := 3 //3 carreras  
constant integer nSalas := 10 //10 salas  
integer array[1..nAl] tit := (1..nAl, ....)
```

```
Process alumno(id: 1..nAl)::  
    while true  
        ...  
    end  
end
```

```
Process mantenimiento()::  
    while true  
        ...  
    end  
end
```