Sesión 13: Comunicación síncrona (canales)

Considérese un sistema distribuido compuesto por N procesos que se comunican mediante canales síncronos. Se pide completar el siguiente esquema con el código necesario para asegurar que ningún proceso P ejecute dos iteraciones consecutivas sin que cada uno de los otros N−1 procesos ejecute la suya.

```
Process P(i: 1..N)::
   loop
       //mis tareas
       //sincronizar con Barrera
   end
end

Process Barrera::
   loop
       //implementa una barrera para los P(i: 1..N)
   end
end
```

Sesión 13: Comunicación síncrona (canales)

Se pide programar un *pipeline* de procesos que ordene de mayor a menor un vector de N enteros. Cada proceso recibe como entrada el vector parcialmente ordenado, como una secuencia de números enteros que corresponde con el valor de cada una de las componentes de la estructura. Después, determina el número de componentes que va a ordenar, un valor aleatorio K comprendido entre 1 y el número de componentes aún no ordenadas, y ordena las primeras K componentes de la parte desordenada. Finalmente, envía el vector resultado al siguiente proceso del *pipeline*.

```
//Pre:   1<=i<=f<=N
//Post:  reordena las componentes v[i],v[i+1],…,v[N]
//       de manera que se colocan en v[i],…,v[f] los valores menores,
//       ordenados en sentido no creciente, dejando inalterados los datos
//       v[1],…,v[i-1]

operation ordena(int REF array[1..N] datos, int i, int f)
```

Sesión 13: Comunicación síncrona (canales)

Además de los procesos responsables de la ordenación, el *pipeline* también tendrá un proceso *Init* que generará el vector a ordenar y lo enviará al primer proceso de la cadena, y un proceso *Pantalla* que recibirá el vector ordenado y lo imprimirá por la pantalla. La comunicación entre todos los procesos involucrados en la solución será a través de canales síncronos. La solución debe garantizar que todos los procesos finalicen correctamente.

Se pide:

- Explicar de manera justificada el tamaño del *pipeline*, es decir, el número de procesos de ordenación que integran la cadena, y aquellas decisiones de diseño que hayáis tomado en cuanto a su comportamiento
- Diseñar y justificar la estructura de canales de comunicación entre los procesos involucrados
- Programar el sistema descrito utilizando la notación de clase

Sesión 14: Comunicación síncrona (canales)

Un sistema consta de 10 procesos de tipo *P* que acceden y usan recursos de tipo R. Inicialmente, el sistema dispone de 10 unidades del recurso *R*. El comportamiento de los procesos consiste en reservar un número aleatorio de unidades del recurso R (entre 1 y 4), usar los recursos obtenidos, y liberarlos una vez acabada la tarea. Este comportamiento se repite de forma indefinida.

El ejercicio pide desarrollar un programa, con la notación utilizada en clase, que implemente el sistema descrito. La coordinación entre los procesos se hará a través de un proceso servidor con comunicación síncrona. La atención a los procesos solicitantes se debe hacer en orden de llegada (política FIFO), siempre que existan los recursos disponibles.

Sesión 14: Comunicación síncrona (canales)

Un sistema consta de 10 procesos de tipo *P* que usan repetidas veces recursos de tipo R. Inicialmente, el sistema dispone de 10 unidades del recurso *R*. El código que describe cómo se comportan los procesos se esboza en la siguiente imagen.

El ejercicio pide desarrollar un programa, con la notación utilizada en clase, que implemente el sistema descrito. La coordinación entre los procesos se hará a través
síncrona. La atención a los
llegada (política FIFO), sien

```
Process P (i:1..10)

  while true
     // reserva un número aleatorio de unidades de R
     // usa los recursos
     // libera los recursos
  end
end
```