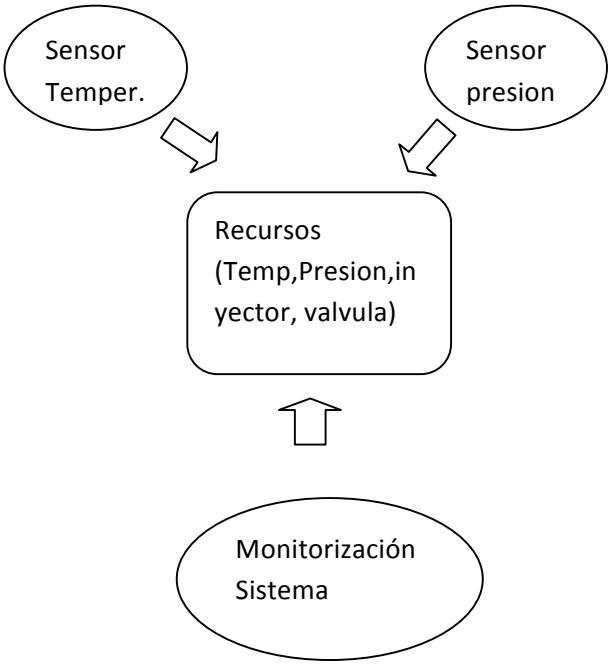
UNIVERSIDAD DE MÁLAGA
Dpto. Lenguajes y Ciencias de la Computación

Programación de Sistemas y Concurrencia Práctica nº6.

1.- Se desea implementar el software de tiempo real de un sistema de control de presión y temperatura. Para ello, periódicamente se realiza la ejecución concurrente de las siguientes tareas:



Sensor de temperatura. Esta tarea tiene una ejecución periódica de 1000 ms. En cada activación se realizan 1000 tomas de temperatura y se toma la media de ellas. Una vez calculada la media, esta tarea revisa que la temperatura no exceda los 100 grados centígrados. En caso de que exceda, se simulará la activación de la inyección de aire frío Cuando la temperatura vuelva a unos valores normales se apagará la inyección de aire frío. Una vez que el aire frio esté conectado no se podrá activar de nuevo hasta que no se desconecte.

Sensor de presión. El periodo de esta tarea es de 2000 ms y puesto que la presión tiene menos variaciones se tomarán 650 muestras. El funcionamiento es el mismo que el control de temperatura pero se realiza la apertura y cierre de una válvula para controlarla presión. La presión que produce la apertura de la válvula es la de 1000 Kpa.

MonitorizacionSistema. Esta tarea accede a los recursos compartidos de presión, temperatura y los estados de los inyectores de aire y válvula y muestra por pantalla la situación del sistema, es decir, los valores de temperatura, presión y los estados de los inyectores y válvulas.

2.- Implementar el problema del productor-consumidor utilizando varios productores y varios consumidores con la particularidad de que el buffer en esta ocasión es NO destructivo. Esto es: cada vez que un productor coloca un elemento en el buffer, el elemento tiene que ser consumido por todos los consumidores (una sola vez por cada consumidor) antes de poder liberar la posición asociada al buffer. El resto del funcionamiento es similar al productor-consumidor normal.

Por ejemplo, si en el buffer tenemos los elementos [1,2,3,4] y 3 consumidores, se podría tener que cada consumidor ha consumido los siguientes elementos:

C1: 1, 2

C2: 1

C3: 1, 2, 3

Hasta que un elemento no ha sido consumido por todos los consumidores, la posición no se libera. En el ejemplo anterior, el elemento "1" ha sido consumido por todos los consumidores y su posición en el buffer podría ser utilizada para colocar otro elemento. El elemento "2", por el contrario, sólo ha sido consumidor por C1 y C3 por lo que su posición todavía no puede ser liberada en el buffer.

3.- Considera un sistema formado por tres hebras **fumadores** que se pasan el día liando cigarros y fumando. Para liar un cigarro necesitan tres ingredientes: **tabaco**, **papel** y **cerillas**. Cada fumador dispone de un surtido suficiente (para el resto de su vida) de uno de los tres ingredientes. Cada fumador tiene un ingrediente diferente, es decir, un fumador tiene una cantidad **infinita de tabaco**, el otro de **papel** y el otro de **cerillas**. Hay también una hebra **agente** que pone dos de los tres ingredientes encima de una mesa. El agente dispone de unas **reservas infinitas de cada uno de los tres ingredientes** y escoge de forma aleatoria cuáles son los ingredientes que pondrá encima de la mesa. Cuando los ha puesto, el fumador que tiene el otro ingrediente puede fumar (los otros dos no). Para ello coge los ingredientes, se lía un cigarro y se lo fuma. Cuando termina de fumar vuelve a repetirse el ciclo. En resumen, el ciclo que debe repetirse es :

"agente pone ingredientes → fumador hace cigarro → fumador fuma → fumador termina de fumar → agente pone ingredientes → …"

Es decir, en cada momento a lo sumo hay un fumador fumando un cigarrillo.

4.- Varios procesos (**N**) compiten por utilizar unos cuantos recursos **Rec** en exclusión mutua. El uso correcto de los recursos es gestionado por una clase Control. Cada proceso pide un numero de recursos llamando al método del monitor **qrecursos(id,num)**, donde **id** es el identificador del proceso que hace la petición y **num** es un número entero que especifica el número de recursos que pide el proceso. Después de usar los recursos, el proceso los libera ejecutando el método **librecursos(id,num)**. Para asignar los recursos Control utiliza la técnica FCFS (First Come, First Serve) que significa que los procesos son servidos siempre en el orden en el que han realizado sus peticiones. Así un proceso que pide **num** recursos debe esperarse si hay otros procesos esperando aún cuando existan **num** recursos disponibles en el sistema. Implementa la clase **Control**, y varias hebras usuarios que realicen las peticiones a **Control**.