Programación de Sistemas Concurrentes y Distribuidos 1ª Convocatoria curso 14/15

6 de febrero de 2015

Dpto. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza

Ejercicio 1 (2.75 ptos.)

Un sistema concurrente modela el comportamiento de un restaurante de comida rápida. La figura 1 muestra los procesos involucrados (más concretamente, los procesos *Cliente* y el proceso *Cocinero*) y ayuda a entender las restricciones de sincronización existentes entre ellos.

En el restaurante se ha instalado una *máquina expendedora* que es capaz de atender simultáneamente a dos clientes a través de dos pantallas táctiles independientes (para diferenciar estas dos pantallas, las llamaremos *pantalla_1* y *pantalla_2*). La oferta culinaria del restaurante consiste de dos menús diferentes, el *menú A* y el *menú B*. Internamente, la máquina expendedora tiene un número determinado de menús de cada tipo listos para ser servidos. Debido a las limitaciones de espacio, el número máximo de menús de cada tipo que es capaz de almacenar está limitado a 100. Cuando un cliente realiza un pedido a través de una de las pantallas táctiles debe especificar el número de menús A y B que desea comprar. Si la máquina tiene suficientes menús disponibles, atiende el pedido y sirve los correspondientes menús; en caso contrario, el cliente queda a la espera de que los menús solicitados le sean suministrados por la máquina.

Por otro lado, en el sistema intervienen un proceso *Cocinero* y cien procesos *Cliente*. El cocinero está continuamente preparando menús, buscando que siempre haya el máximo número de menús de ambos tipos en la máquina. En cuanto a los procesos de tipo cliente,éstos repiten un número aleatorio de veces el siguiente comportamiento: realiza un pedido de menús de tipo A y B, y se los lleva. Dado que el número de clientes es elevado

y la expendedora sólo tiene capacidad para atender a dos de ellos simultáneamente, se ha instalado un gestor de turnos. Cuando un cliente desea realizar un pedido debe pedir turno en la fila de turnos. El gestor atiende esta fila por orden de llegada, indicando al siguiente cliente a ser atendido si debe dirigirse a la pantalla táctil 1 o 2 para solicitar su pedido.

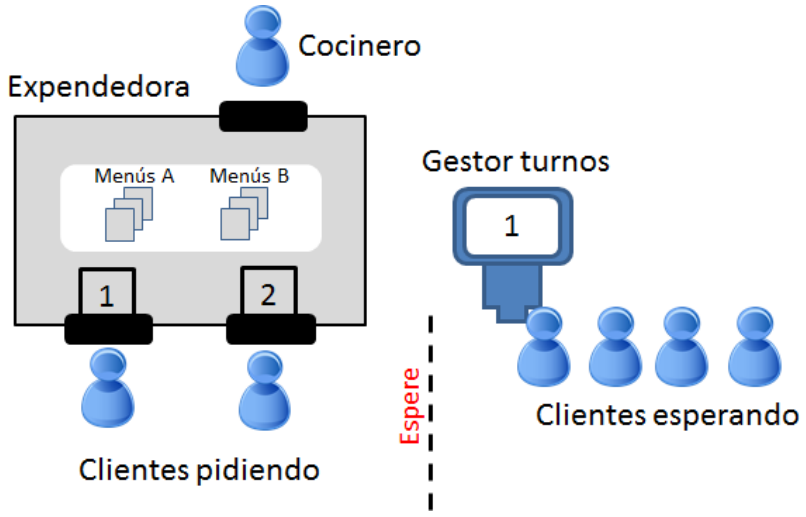

Figura 1: Representación abstracta del sistema concurrente

Se pide implementar el sistema previamente descrito utilizando la notación presentada en clase y explicar de forma breve y justificada todas las decisiones de diseño relevantes que hayan sido adoptadas. En la solución, la máquina expendedora de menús y el gestor de turnos deberán ser implementados como monitores, garantizando las restricciones de sincronización entre procesos descritas en el enunciado.

Ejercicio 2 (2.5 ptos.)

Considérese un sistema de producción compuesto por los siguientes elementos:

- un proceso *pBloques* que constantemente produce bloques de metal en forma de cubo. Cuando un bloque es producido, lo introduce en el almacén AB con capacidad para kB (> 0) bloques, si es que queda algún hueco. Si no lo hay, se queda esperando a que haya algún hueco, deposita entonces el bloque en el almacén y pasa a producir otro bloque
- un proceso *pTuercas* que constantemente produce tuercas y las deposita en el almacén $ATyT$ para guardar tanto tuercas como tornillos (el almacén tiene capacidad para $kTyT$ elementos, entre tuercas y tornillos)
- un proceso *pTornillos* que constantemente produce tornillos y los deposita en el mismo almacén usado para guardar las tuercas. Tanto *pTuercas* como *pTornillos* quedan bloqueados a la espera de espacio en el almacén en caso de que esté lleno.
- un proceso *pEnsambla* que toma un bloque, una tuerca y un tornillo, taladra el bloque e inserta y enrosca la tuerca y el tornillo. Si faltara alguna de estas componentes, quedaría bloqueado hasta disponer de ellas.

El ejercicio pide diseñar los procesos descritos usando la notación algorítmica utilizada en clase, así como el proceso *pControlador*, que controla el acceso a los dos almacenes, de manera que se asegure un buen comportamiento del sistema. Cualquiera de los procesos involucrados debe acceder a los almacenes a través del proceso controlador. Debe usarse paso síncrono de mensajes como mecanismo para la sincronización de los procesos involucrados.

Ejercicio 3 (2.75 ptos.)

Considérese el siguiente algoritmo.

```
              int s := 1, x := 0
              int b1 := 2, b2 := 0

Process P1::                    Process P2::
   loop                            loop
      <await b1>0 AND s>0             <await b2>0 AND s>0
         b1 := b1-1                      b2: = b2-1
         s := s-1                        s := s-1
      >                               >
      <x := x+1>                      <x := x-1>
      < b2 := b2+1                    < b1 := b1+1
        s := s+1                        s := s+1
      >                               >
   end loop                        end loop
```

El ejercicio pide:

- Dar un modelo red de Petri para el algoritmo mostrado
- A partir del modelo anterior, construir su espacio de estados
- Basándonos en el espacio de estados del modelo, responder razonadamente a las siguiente cuestiones:
  - El programa nunca de bloquea, ni total ni parcialmente
  - ¿Es su comportamiento equitativo?
  - ¿Cambiaría el comportamiento del programa si en lugar de $\langle \mathtt{x} := \mathtt{x} + 1 \rangle$ y $\langle \mathtt{x} := \mathtt{x} - 1 \rangle$ se hubiera usado $\mathtt{x} := \mathtt{x} + 1$ y $\mathtt{x} := \mathtt{x} - 1$, respectivamente?

Ejercicio 4 (2.00 ptos.)

Considérese el algoritmo del ejercicio anterior. Ahora se trata de desarrollar una versión con dos procesos distribuidos, que se comporte de manera análoga. Los procesos P1 y P2 sincronizan utilizando Linda.