SISTEMAS CONCURRENTES Y DISTRIBUIDOS

Álvaro Fernández García

-Documentación Productor-consumidor con buffer limitado-

-Cambios realizados:

1º – Se ha sustituido la clase Buffer por un monitor de tipo Hoare proporcionado por el paquete “monitor”, llamado “MonitorPC”.

2º – Los atributos del monitor no han sufrido cambios conservandose los mismos que presentaba la clase Buffer: numSlots, cont y Buffer[ ].

3º – Se han añadido las correspondientes variables de condición (Se explicarán en el siguiente apartado).

4º – El constructor del Monitor no ha sufrido modificaciones con respecto al constructor de la clase Buffer.

5º – Se han sustituido los Synchronized de los métodos de la clase Buffer por los enter() y leave() correspondientes del monitor para controlar su cerrojo.

6º – Se han sustituido los wait( ) y notifyAll( ) por los await( ) y los signal( ) correspondientes a las variables de condición necesarias para cada caso.

7º – Se han sustituido los bucles while para comprobar las condiciones de bloqueo por condicionales if, puesto que al colocar las colas de condición ya no es necesario reevaluar la condicion al salir del bloqueo.

8º – Se ha modificado el parámetro de tipo Buffer que presentan las hebras por un parámetro de tipo MonitorPC y se han modificado los correspondientes constructores y llamadas a los métodos.

9º – El main permanece sin modificaciones, salvo el cambio del objeto Buffer por uno de tipo MonitorPC.

-Variables de condicion empleadas:

Se han empleado dos variables de condición:

- Producir: cola que sirve para bloquear al productor en caso de que no pueda continuar produciendo y se vea obligado a esperar a que el consumidor consuma algún dato.

- Consumir: cola que sirve para bloquear al consumidor en caso de que no haya ningún valor para ser consumido y este deba de esperar a que el productor produzca algún dato.

-Salida del programa para 2 productores, 2 consumidores, un buffer de tamaño 3 y 5 iteraciones por hebra.

productor 0, produciendo 0.0
productor 1, produciendo 100.0
consumidor 0, consumiendo 0.0
productor 0, produciendo 1.0
productor 1, produciendo 101.0
consumidor 1, consumiendo 100.0
productor 1, produciendo 102.0
consumidor 0, consumiendo 1.0
consumidor 0, consumiendo 102.0
productor 0, produciendo 2.0
productor 1, produciendo 103.0
consumidor 1, consumiendo 101.0
productor 0, produciendo 3.0
consumidor 0, consumiendo 2.0
consumidor 1, consumiendo 103.0
consumidor 0, consumiendo 3.0
productor 1, produciendo 104.0
productor 0, produciendo 4.0
consumidor 1, consumiendo 104.0
consumidor 1, consumiendo 4.0

-Documentación Fumadores-

-Variables de condicion empleadas:

Se han empleado 4 variables de condición:

- necesarioProducir: Esta variable de condición tiene como principal finalidad controlar al Estanquero, es decir, bloqueará al estanquero cuando el mostrador aún este ocupado. Cuando este quede libre porque el fumador correspondiente ha recogido su recurso, se debloqueará de esta cola al estanquero. Adicionalmente para comprobar esta condición se utiliza una variable booleana que le indica al estanquero el estado del mostrador (true = ocupado, false = libre).

- 3 variables de condición, (una por cada fumador) almacenadas en un vector llamado puedeRecoger: Cada una de estas colas de condición sirven para poder bloquear a cada uno de los fumadores en una cola independiente, para tener un mayor control a la hora de desbloquearlos cuando se produce el recurso que necesitan. Los fumadores se bloquearán cuando el recurso que el Estanquero haya producido no sea el suyo (El caso de que el mostrador esté vacío, este contendrá un valor especial (-1), que no coincide con ningún recurso de fumadores y todos se bloquearían) y se desbloquearán cuando el Estanquero produzca su ingrediente. Adicionalmente para comprobar esta condición, se utiliza una variable entera que contiene el ingrediente producido por el estanquero.

-Salida del programa-

Estanquero ha producido 2
Fumador 2 recoge 2 y comienza a fumar
Estanquero ha producido 1
Fumador 1 recoge 1 y comienza a fumar
Estanquero ha producido 1
Fumador 1 termina de fumar
Fumador 1 recoge 1 y comienza a fumar
Estanquero ha producido 0
Fumador 0 recoge 0 y comienza a fumar
Estanquero ha producido 0
Fumador 0 termina de fumar
Fumador 0 recoge 0 y comienza a fumar
Estanquero ha producido 0
Fumador 2 termina de fumar
Fumador 1 termina de fumar

Fumador 0 termina de fumar
Fumador 0 recoge 0 y comienza a fumar
Estanquero ha producido 0
Fumador 0 termina de fumar
Fumador 0 recoge 0 y comienza a fumar
Estanquero ha producido 1
Fumador 1 recoge 1 y comienza a fumar
Estanquero ha producido 2
Fumador 2 recoge 2 y comienza a fumar
Estanquero ha producido 2
Fumador 2 termina de fumar
Fumador 2 recoge 2 y comienza a fumar

-Documentación El Barbero Durmiente-

-Variables de condicion empleadas:

Se han empleado 3 variables de condición:

- pelándose: Esta variable de condición sirve para que los clientes se bloqueen mientras el barbero les está cortando el pelo. Por tanto presentará un await en el procedimiento del monitor cortarPelo y un signal realizado por el barbero en el procedimiento finCliente, indicándole que ya ha finalizado.

- colaClientes: Variable de condición que representa la cola de clientes que se encuentran en la sala de espera. Su finalidad es bloquear a los clientes que van llegando a la barbería mientras que el barbero se encuentra ocupado. Cuando el barbero termina, si hay clientes esperando desbloquea al primero, si no hay ninguno, el barbero duerme. Adicionalmente contamos con una variable entera llamada numClientesSala que controla el número de clientes esperando y otra de tipo booleano (sillaOcupada) que indica si el barbero está con un cliente o no (true = ocupado, false = libre).

- duerme: Variable que sirve para bloquear al barbero (dormir) en el caso de que no haya ningún cliente bloqueado en colaClientes. Cuando llega un nuevo cliente, este manda un signal para despertar al barbero (si este no estaba dormido si no ocupado, el signal no realiza nada).

-Salida del programa-

El cliente despierta al barbero.
El barbero comienza a cortarle el pelo a un cliente. Clientes esperando: 0
Llega un cliente a la sala de espera. Clientes esperando 1
Llega un cliente a la sala de espera. Clientes esperando 2
El barbero ha terminado de pelar a un cliente.
El barbero le dice a un cliente que puede pasar.
El barbero comienza a cortarle el pelo a un cliente. Clientes esperando: 1
El barbero ha terminado de pelar a un cliente.
El barbero le dice a un cliente que puede pasar.
El barbero comienza a cortarle el pelo a un cliente. Clientes esperando: 0
Llega un cliente a la sala de espera. Clientes esperando 1
Llega un cliente a la sala de espera. Clientes esperando 2
El barbero ha terminado de pelar a un cliente.
El barbero le dice a un cliente que puede pasar.
El barbero comienza a cortarle el pelo a un cliente. Clientes esperando: 1
Llega un cliente a la sala de espera. Clientes esperando 2
El barbero ha terminado de pelar a un cliente.

El barbero le dice a un cliente que puede pasar.
El barbero comienza a cortarle el pelo a un cliente. Clientes esperando: 1
Llega un cliente a la sala de espera. Clientes esperando 2
El barbero ha terminado de pelar a un cliente.
El barbero le dice a un cliente que puede pasar.
El barbero comienza a cortarle el pelo a un cliente. Clientes esperando: 1