Javier Arroyo García
B3

DOCUMENTACIÓN SEMINARIO 2 SCD

(1)

Para esta implementación he añadido tres variables de carácter global, los consumidores, los productores y un vector que actúa como contador de ítems producidos por los productores (que iremos actualizando cada vez que una hebra produzca un valor).
El principal cambio se ha implementado en la función producir_dato, se le ha añadido un parámetro que indica la hebra que llama a dicha función, con lo que podemos asegurarnos de que cada hebra no produce más de num_items / num_productores ítems.
Por otra parte, también se han implementado cambios en la clase ProdCons2SC, se añaden dos variables locales a la clase, primera_ocupada y celdas ocupadas, que se encargarán de (siendo actualizadas correctamente) asegurar que se sigue una organización de tipo FIFO.

(2)

Para esta implementación he utilizado como base el programa descrito en (1), pero esta vez los cambios los he realizado sobre la clase ProdCons, que hereda de HoareMonitor (en scd.h) y que nos permite declarar variables del tipo CondVar, que tienen asignadas las operaciones wait() y signal() entre otras.
Estas variables nos permiten prescindir de las variables globales mutex que veníamos utilizando hasta ahora.
El cambio más significativo es que ahora las funciones de hebras no tienen como parámetro una referencia al monitor de la clase, si no que tienen como parámetro una referencia a la instancia del monitor que van a usar (usamos el tipo Mref), que se crea en el main.
Cabe destacar que en la organización LIFO, a la hora de implementar el método leer(), tendremos que leer de esta manera:
Valor = buffer[primera_libre – 1], ya que si hacemos Valor = buffer[primera_libre] se lee varias veces el primer valor.