Filósofos y ProdCons

ProdCons:

Para poder añadir varios productores y consumidores, primero tenemos que añadir una constante con el número de productores y consumidores, al igual que etiquetas para el uso durante el paso de mensajes.

Una vez añadidas las etiquetas, haremos lo siguiente para cada productor

*static int contador = num_prod*porproductor*

Donde *porproductor* es una variable que representa el número de elementos a producir por cada productor (*num_items/nprods*) y *num_prod* es el número del productor (desde 0 hasta *nprods* – 1).

A la hora de mandar el mensaje, especificaremos que se lo estamos mandando al buffer (id_buffer) y que tiene la etiqueta *etiq_prod*.

El consumidor mandará una petición al buffer con la etiqueta *etiq_cons* para esperar la recepción del mensaje.

El buffer recibe mensajes con cualquier etiqueta (o solo la del productor si el buffer está vacío o del consumidor si el buffer está lleno) y de cualquier fuente. Una vez recibido el mensaje, mira si la etiqueta es del consumidor o del productor y actúa en función de la etiqueta.

- Si al etiqueta es del productor, guarda el valor recibido en el buffer y aumenta el contador.
- Si la etiqueta es del consumidor, manda **al mismo consumidor** el valor del buffer (con la correspondiente etiqueta *etiq_cons*) y reduce el contador.

Traza del programa:

Productor 3 va a enviar valor 40
Buffer va a enviar valor 17
Buffer ha recibido valor 40
Consumidor 1 ha consumido valor 8
Consumidor 1 ha recibido valor 17
Productor 1 ha producido valor 20
Productor 1 va a enviar valor 20
Consumidor 4 ha consumido valor 36
Consumidor 4 ha recibido valor 28
Buffer va a enviar valor 28
Buffer ha recibido valor 20
Consumidor 2 ha consumido valor 27
Consumidor 2 ha recibido valor 37
Buffer va a enviar valor 37
Consumidor 0 ha consumido valor 16
Buffer va a enviar valor 10
Consumidor 0 ha recibido valor 10
Consumidor 3 ha consumido valor 9
Consumidor 3 ha recibido valor 18
Buffer va a enviar valor 18

Filósofos:

El interbloqueo se produce cuando todos los filósofos disponen de un tenedor, por lo que no quedan tenedores libres para que puedan coger el segundo

Aquí distinguiremos dos soluciones que evitan el interbloqueo:

- Solución 1 (*filosofos.cpp*):
  Para evitar el interbloqueo en esta solución hemos optado por formar "parejas" de filósofos.
  Según la posición en la mesa, las parejas intentarán coger el mismo tenedor a la vez (de forma que solo uno lo reciba) y el tenedor del lado contrario se quedará libre siempre.
  Traza de esta solución:

  Filósofo 8 solicita ten. izq.9
  Filósofo 8 solicita ten. der.7
  Filósofo 8 comienza a comer
  Filósofo 4 solicita ten. izq.5
  Filósofo 4 solicita ten. der.3
  Filósofo 6 solicita ten. der.5
  Ten. 9 ha sido cogido por filo. 8
  Ten. 5 ha sido cogido por filo. 4
  Filósofo 4 comienza a comer
  Ten. 7 ha sido cogido por filo. 8
  Ten. 1 ha sido cogido por filo. 2
  Filósofo 2 solicita ten. der.1
  Filósofo 2 solicita ten. izq. 3
  Ten. 3 ha sido cogido por filo. 4
  Filósofo 0 solicita ten. izq.1
  Filósofo 8 suelta ten. izq. 9
  Filósofo 8 suelta ten. der. 7
  Filosofo 8 comienza a pensar
  Ten. 9 ha sido liberado por filo. 8

- Solución 2 (*filósofos-cam.cpp*):
  Para evitar el interbloqueo en esta solución lo que hacemos es añadir un camarero que solo permite que haya 4 filósofos sentados a la mesa, de esta forma siempre quedará un tenedor libre, ya que será (máximo) 4 filósofos y 5 tenedores, por lo que un filósofo podrá coger 2 tenedores **siempre**.
  El camarero espera que haya un mensaje disponible, lee los metadatos y en función de la etiqueta del mensaje (sentarse o levantarse) y del número de filósofos a la mesa, levanta o sienta a un filósofo.
  Para el uso de etiquetas, evidentemente es necesario declarar constantes globales con su valor al inicio del programa.
  Traza de esta solución:

  Ten. 1 ha sido liberado por filo. 2
  Filósofo 2 suelta ten. izq. 3
  Filósofo 2 suelta ten. der. 1
  Filósofo 2 solicita levantarse de la mesa.
  Filosofo 2 comienza a pensar
  Levantado filósofo 2
  Sentado filósofo 0
  Filósofo 0 solicita ten. izq.1
  Filósofo 0 solicita ten. der.9
  Ten. 1 ha sido cogido por filo. 0
  Filósofo 4 suelta ten. izq. 5
  Filósofo 4 suelta ten. der. 3
  Filósofo 6 comienza a comer