Documentación Práctica 3

Por: Arturo Cortés Sánchez

Productor consumidor múltiple - diferencias con prodcons2

Añadidas nuevas constantes: `np` , `nc` , `etiq_consumidor` , `etiq_productor` .Eliminadas las variables `id_consumidor` e `id_productor` . Las constante `id_buffer` ha cambiado su valor al valor de `np` y la constante `num_procesos_esperado` ha cambiado su valor a `np + nc + 1`

Estos cambios se han realizado siguiendo las indicaciones del guion.

La función `producir()` ahora recibe el "numero" del productor como parámetro para facilitar el seguimiento de la ejecución.

La función `funcion_productor()` ahora recibe un parámetro por la razón mencionada anteriormente, además el `MPI_Ssend` ahora usa etiquetas.

La función `consumir()` ahora también recibe el "numero" del consumidor como parámetro para facilitar el seguimiento de la ejecución.

En la función `funcion_consumidor()` ahora hay un parámetro para facilitar el seguimiento de la ejecución. El bucle `for` ahora llega hasta `num_items/nc` en lugar de hasta `num_items` . El `MPI_Ssend` ahora usa etiquetas.

En la funcion `funcion_buffer()` , `id_consumidor` e `id_productor` han sido sustituidos por `etiq_consumidor` y `etiq_productor` . Como ahora usamos etiquetas el `MPI_ANY_SOURCE` ha sido cambiado por un `MPI_ANY_TAG` . Además en el `MPI_Recv` ha sido modificado a `MPI_Recv(&valor, 1, MPI_INT, MPI_ANY_TAG, id_emisor_aceptable, MPI_COMM_WORLD, &estado)` .

El swich-case ahora va en función de `estado.MPI_TAG` en lugar de `estado.MPI_SOURCE` para poder saber quien fue el emisor del mensaje anterior y devolverle un mensaje en el `Ssend` .

En el main las funciones `funcion_productor()` y `funcion_consumidor()` ahora reciben como argumento el "numero" del proceso.

Salida parcial:

```
Productor 0 ha producido valor 1
Productor 0 va a enviar valor 1
Buffer ha recibido valor 1
Buffer va a enviar valor 1
Consumidor ha recibido valor 1
Productor 3 ha producido valor 1
Productor 3 va a enviar valor 1
Buffer ha recibido valor 1
Buffer va a enviar valor 1
Consumidor ha recibido valor 1
Consumidor 5 ha consumido valor 1
Productor 3 ha producido valor 2
Buffer ha recibido valor 2
Buffer va a enviar valor 2
Productor 3 va a enviar valor 2
Consumidor ha recibido valor 2
```

```
Productor 1 ha producido valor 1
Productor 1 va a enviar valor 1
Buffer ha recibido valor 1
Buffer va a enviar valor 1
Consumidor ha recibido valor 1
Consumidor 8 ha consumido valor 1
Productor 2 ha producido valor 1
```

Cena de los Filósofos

En la plantilla de este problema viene la estructura de la solución, por lo que solo hay que poner los `Ssend` y los `Recv`. Cada vez que un filosofo requiera un tenedor hará un `MPI_Ssend`, y cuando quiera soltarlo hará exactamente el mismo `MPI_Ssend` ya que la función tenedor se dedica a esperar un segundo mensaje del filosofo que le envió el primero.

Para la solución sin interbloqueo basta con poner una condición el la `funcion_filosofos()` de forma que el primer filosofo coja los tenedores en el sentido contrario al resto

Filosofos con camarero

Para implementar el camarero he realizado una función `camarero()` que realiza un bucle infinito en el que se comprueba si hay espacio para que un filosofo se siente a la mesa, en caso afirmativo recibe mensajes de sentarse y levantarse, en caso negativo solo de levantarse. Una vez recibido el mensaje, lo procesa y dependiendo del contenido del mensaje, deja que el filósofo se siente o se levante.

Posteriormente en el main hay que llamar a esta nueva función `camarero()`.

Salida parcial:

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