Sistemas concurrentes y distribuidos
Práctica 3
Grupo: C2
Alumno: Miguel Torres Alonso

Múltiples productores y consumidores

Para la resolución de este problema he realizado los siguientes cambios con respecto del archivo original proporcionado en PRADO:

- Cada proceso tendrá su ID dentro del rol al que pertenezca.
- El ID del buffer ahora es equivalente en valor al número de productores.
- En la función que realizan los productores, debemos controlar que se produzcan valores dentro de un rango dependiente del ID de cada proceso y el reparto del número de ítems entre todos los procesos productores. Esto se puede llevar a cabo con un contador inicializado de manera diferente según el ID de cada proceso dentro del rol. Debido a esto también dividimos las iteraciones del bucle *for* entre los distintos procesos.
- Los consumidores en su función simplemente envían solicitud al buffer, esperan a recibir un valor y lo consumen, dividiéndose el número de ítems entre todos los procesos consumidores.
- En los *Recv* del proceso buffer se deben de utilizar etiquetas que identifiquen a los productores y a los consumidores dependiendo del número de celdas ocupadas del buffer. Si el buffer está vacío, solo recibe de productores. Si el buffer está lleno solo recibe de consumidores y si no se da ninguna de estas dos condiciones no discrimina ningún rol. A la hora de enviar un valor a un consumidor, el proceso buffer consultará el ID del proceso consumidor el cual le ha enviado la solicitud para enviarle el valor de vuelta.

Problema de los filósofos

Para este problema debemos emplear envíos síncronos de mensajes entre filósofos y tenedores. Cada filósofo envía un mensaje síncrono de solicitud para usar cada tenedor que tiene a su lado, y otro mensaje síncrono a cada tenedor para dejar de utilizarlo y que otro filósofo pueda coger el tenedor adyacente. Para que el tenedor no pueda ser cogido por dos filósofos simultáneamente, el proceso tenedor guarda localmente el ID del filósofo del cual recibe un mensaje de solicitud para cogerlo para posteriormente ejecutar un *Recv* esperando mensaje única y exclusivamente del mismo ID mencionado anteriormente.

La solución inicial puede producir interbloqueo debido a que todos los filósofos intentan coger primero siempre el tenedor de la izquierda, con lo cual una de las posibles soluciones a este problema es que el filósofo con ID igual a 0 coja primero el tenedor de su derecha y luego el de su izquierda.

Problema de los filósofos con camarero

Una nueva solución para el interbloqueo es utilizar un proceso adicional que realice la función de camarero. El camarero en su función lleva la cuenta de los filósofos que hay sentados en la mesa y dispone de un valor de etiqueta aceptable que varía en función del valor del contador.

Si hay menos de 4 filósofos sentados, acepta cualquier valor de etiqueta (levantarse o sentarse). Si hay 4 o más filósofos sentados, sólo aceptará solicitudes de filósofos que quieran levantarse. Una vez haya recibido la solicitud podemos saber si ha recibido solicitud de levantarse o de sentarse, por tanto podemos decrementar o incrementar respectivamente el contador para la siguiente iteración de su bucle infinito.

Como los filósofos utilizan envíos síncronos de mensajes, quedarán bloqueados esperando a sentarse si el camarero solo acepta solicitudes de levantarse (cuando su contador es mayor o igual a 4).