Jose Antonio Padial Molina

josepadial@correo.ugr.es



SISTEMAS CONCURRENTES Y DISTRIBUIDOS

PRACTICA 3

Contenido

[PRODUCTOR- CONSUMIDOR PARA MÚLTIPLES PRODUCTORES Y CONSUMIDORES 2](#_Toc532066924)

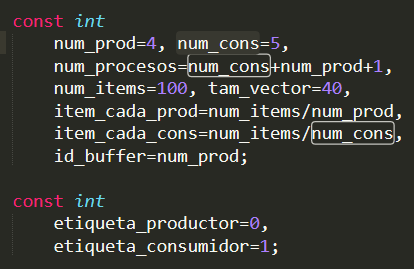
[FILÓSOFOS CON INTERBLOQUEO 4](#_Toc532066925)

[FILÓSOFOS CON SOLUCIÓN INTERBLOQUEO 5](#_Toc532066926)

[FILÓSOFOS CON CAMARERO 6](#_Toc532066927)

# PRODUCTOR- CONSUMIDOR PARA MÚLTIPLES PRODUCTORES Y CONSUMIDORES

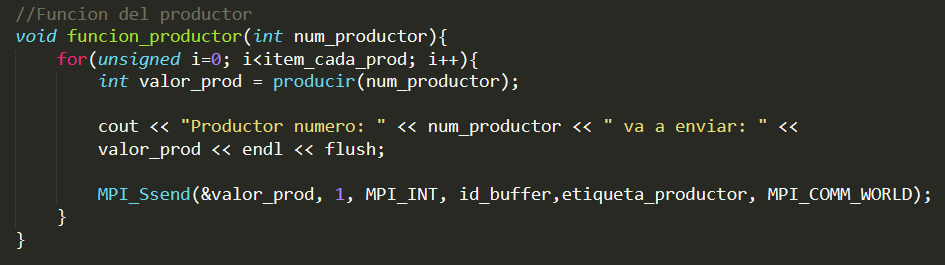
En el nuevo programa, en vez de identificar cada proceso con un identificador único 0,1,2 para el productor, bufer y consumidor respectivamente, se identifica al productor con num\_prod procesos (4 en este caso), num\_cons=5 procesos para consumidores y el buffer con el identificador 4. Como la solución es basada en el uso de etiquetas he creado dos variables constantes enteras para la etiqueda de productor y consumidor.



Como indica el enunciado del problema, la función de los productores y la de los consumidores reciben como parámetro el número de orden del productor o del consumidor, respectivamente (esos números son los números de orden en cada rol, comenzando en 0, no son los identificadores de proceso). Estos números de orden se calculan en el main.

El productor ahora produce num\_items/num\_prod y el consumidor consume num\_items/num\_cons. En el caso de la función del productor en el envío de mensaje Ssend se le pasa los siguientes valores:

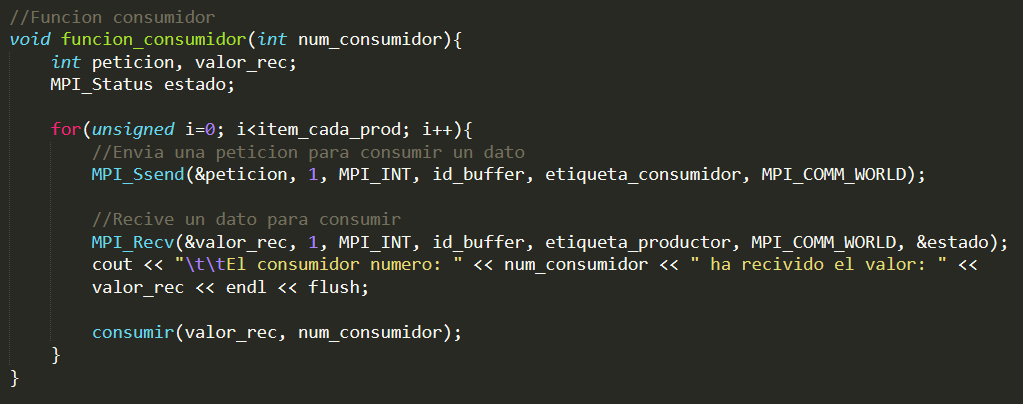
MPI\_Ssend(&valor\_prod, 1, MPI\_INT, id\_buffer,etiqueta\_productor, MPI\_COMM\_WORLD);



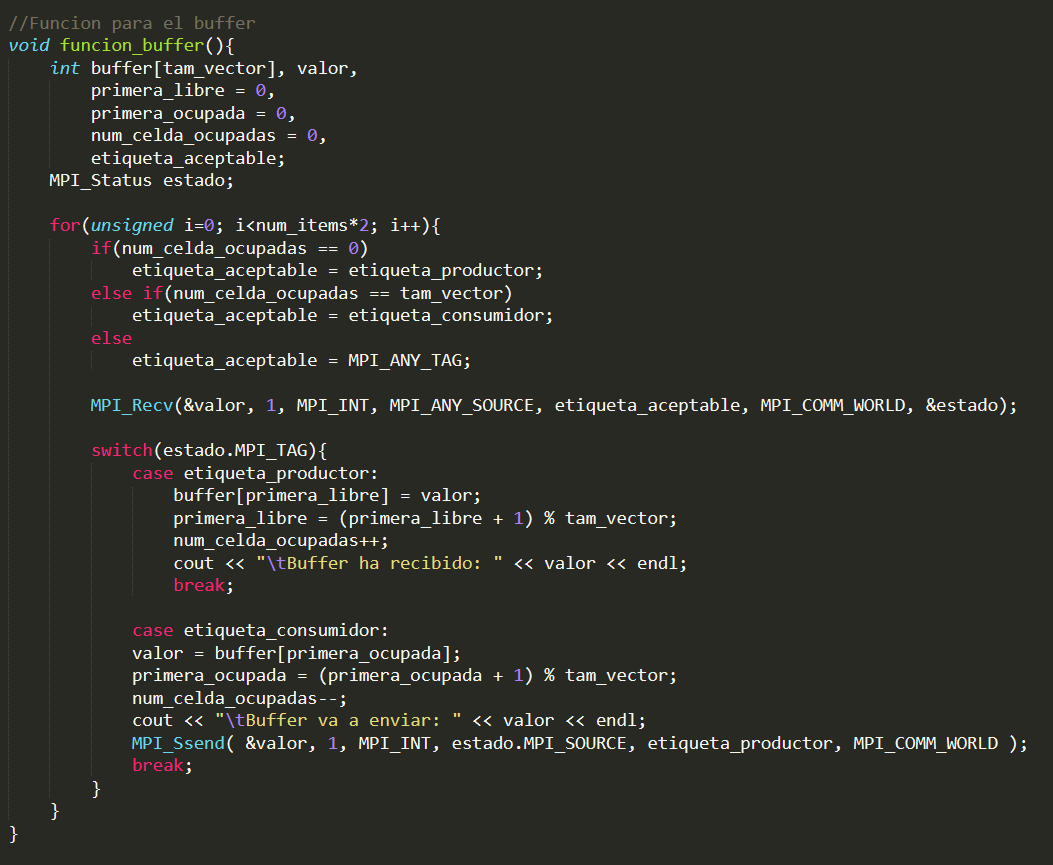
Y para los consumidores, en la función consumidor tanto para el envío Ssend para la petición como Recv para recibir el mensaje se le pasan los siguientes valores:

MPI\_Ssend(&peticion, 1, MPI\_INT, id\_buffer, etiqueta\_consumidor, MPI\_COMM\_WORLD);

MPI\_Recv(&valor\_rec, 1, MPI\_INT, id\_buffer, etiqueta\_productor, MPI\_COMM\_WORLD, &estado);

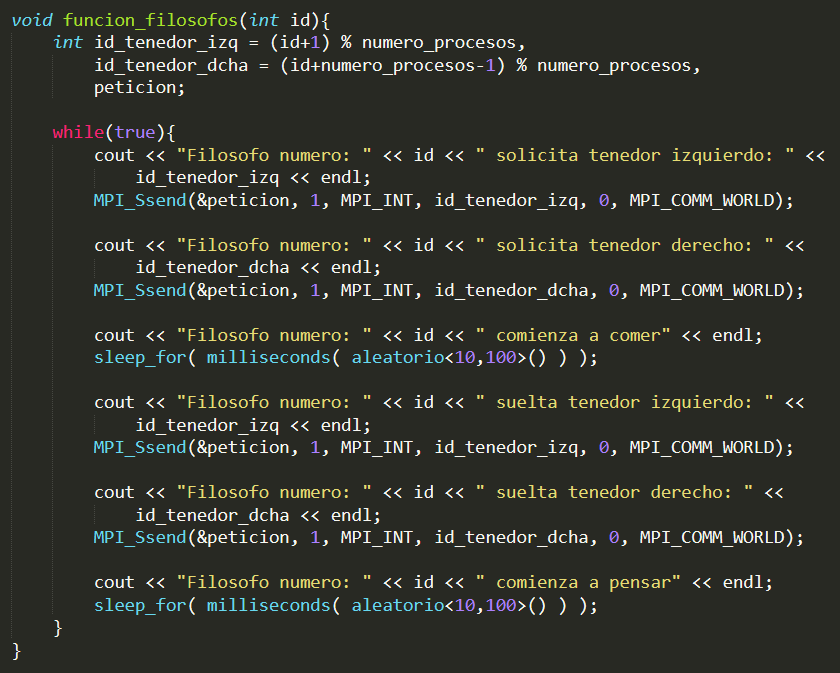


En la función buffer, se calcula en vez de la id\_emisor\_aceptable, como es una solución basada en etiquetas, la etiqueta aceptable según si solo se puede producir o consumir:



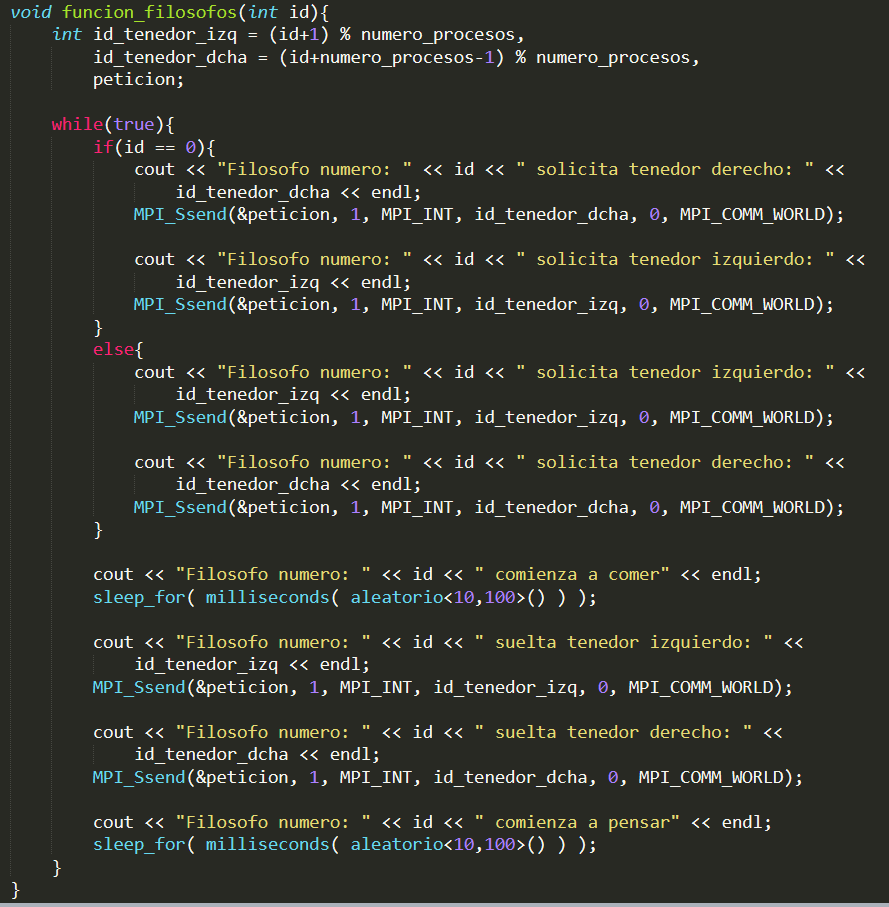
# FILÓSOFOS CON INTERBLOQUEO

Archivo filoso-plantilla.cpp copiado en filósofos-interb.cpp y completado según las directrices del guion de la práctica. Una vez hecho esto, al ejecutarlo compruebo que se da interbloqueo cuando un filósofo solicita un tenedor y este ya lo tiene otro filosofo.



# FILÓSOFOS CON SOLUCIÓN INTERBLOQUEO

La solución ha sido que uno de los filósofos debe comenzar a coger por la derecha, en vez de por la izquierda.



# FILÓSOFOS CON CAMARERO

La solución para el problema de interbloqueo ha sido introducir dos nuevos pasos a los filósofos, que es sentarse a la mesa y levantarse. El nuevo proceso camarero controlará los filósofos que se sientan a la mesa con un contador (filósofos\_sentados) que se incrementara cuando haya menos de 5 filósofos sentadas y decrementará cuando se levante un filósofo.

Para saber si se puede que levantar o sentar un filósofo he creado dos nuevas etiquetas (etiq\_sentarse = 0 y etiq\_levantarse = 1) que utilizo para pedir permiso con el envío síncrono:

MPI\_Ssend(&peticion, 1, MPI\_INT, id\_camarero, etiq\_sentarse, MPI\_COMM\_WORLD);

MPI\_Ssend(&peticion, 1, MPI\_INT, id\_camarero, etiq\_levantarse, MPI\_COMM\_WORLD);

El camarero siempre aceptará las peticiones de un filósofo para levantarse. Otra modificación ha sido el número de procesos, he creado dos variables, una para los procesos efectivos y otra para los procesos esperados (11). Tambien he añadido el identificador del proceso camarero (id\_camarero=num\_procesos\_efectivos).

