3º curso / 1º cuatr.
Grado Ing. Inform.
Doble Grado Ing.
Inform. y Mat.

# Sistemas Concurrentes y Distribuidos
Práctica 1

Estudiante: Miguel Lentisco Ballesteros
Grupo de prácticas: A1
Fecha de entrega: TBA

## Productor/Consumidor

**Problema 1.** Se nos pide dar una solución al problema del Productor/Consumidor siguiendo la plantilla proporcionada. Para la solución se usará como estructura para el buffer tanto como una estructura *LIFO* (pila), como *FIFO* (cola).

### LIFO

Con el buffer como una pila, para saber donde estamos escribiendo y leyendo tendremos que usar una variable global que funcione como índice, esta variable será incrementada cuando el Productor escriba en el buffer y será decrementada y luego se lee el dato por el Consumidor, representando así el índice la posición donde tiene que escribir o leer respectivamente.

Por supuesto tenemos que tener en cuenta que pasa cuando vayan a escribir y leer en el buffer el Consumidor y el Productor van a tener que modificar la variable indice pudiendo probar problemas de escritura/lectura; por ello cuando vayamos a usar esta variable tendremos que usar exclusión mutua: o bien un cerrojo o que sea una variable del tipo *atomic* < *int* >, siendo más fácil de usar la versión con *atomic*.

### FIFO

Si el buffer es una cola, tenemos dos posibilidades en este caso:

- En este caso, el bucle del productor es hasta la cantidad de items a producir, luego podemos aprovechar esta ventaja y usar la variable *i* como el índice ya que simplemente se va a escribir en *i %buffer_tam* cuando el semáforo de escritura lo permita (explicado a continuación), e independientemente del productor; el consumidor irá leyendo en el mismo orden *i %buffer_tam* cuando el semáforo de lectura deje hacerlo, por tanto tendremos variables locales que no interfieren entre sí, no necesitamos exclusión mutua.
- Por otro lado, si tuviéramos que llevar la cuenta inicialmente o en cualquier momento saber donde se está escribiendo o leyendo entonces tendríamos que tener dos índices, uno de escritura y otro de lectura, variables globales para saberlo. Igualmente que en el primer caso, estas variables serían independientes entre sí y no habría problemas de carrera, luego de nuevo, no necesitamos exclusión mutua. Se hace igual, sustituyendo *i* por el índice lectura/escritura correspondiente y incrementando cada índice al escribir/leer en el buffer.

### Semáforo

Ahora, para resolver el problema con semáforos, visto ya en clase, solo tendremos que poner dos semáforos, uno que sea para el productor (para escribir) y otro para el consumidor (para leer). Inicializamos el de escritura al tamaño del buffer (puede escribir hasta el máximo) y el de lectura a 0 (tiene que esperar a que el productor escriba), entonces cada vez que se vaya a producir haya un *sem_wait* de escritura, y cuando escriba se produzca un *sem_signal* de lectura para el consumidor que está esperando en *sem_wait* de lectura, pueda leer el dato y dar un *sem_signal* para indicar que hay un hueco vacío al productor.

El semáforo de escritura sirve para parar al consumidor para que no siga escribiendo si el buffer está completo, y el de lectura para el consumidor que se espere a que haya datos que leer. Obviamente al productor le damos un margen de libertad del tamaño del buffer al principio y al de lectura ninguno, ya que se tiene que esperar a que el productor escriba.

### Solución

En los archivos fuentes, se encuentran las soluciones con *FIFO* y *LIFO* a este problema siguiendo la plantilla proporcionada y completando debidamente las funciones de consumidor y productor, añadiendo las variables globales consideradas (índices y buffer) y mostrando por pantalla cada vez que se escribe/lee en buffer así como cuando cada hebra acaba su cometido con un *fin*. Por supuesto, está comprobado con muchas veces que funciona correctamente.

## Fumadores

> **Problema 2.**