# Sincronización de procesos con mutexes y variables de condición - Opcional 2

Pablo Mouriño Lorenzo, Diego Pérez Álvarez

Sistemas Operativos II
Grupo 04

**Resumen**

*En este informe se modificará la solución obtenida anteriormente para el problema del productor-consumidor usando mutexes y variables de condición, con el objetivo resolver el problema usando únicamente mutexes sin la ayuda de variables de condición*

***Palabras clave: carreras críticas, productor-consumidor, hilos, sincronización, mutexes***

## I. Introducción

Partiendo de la solución obtenida en el ejercicio 1, trataremos de obtener una solución funcional sin usar variables de condición para gestionar los casos extemos de llenado del buffer, esto es, cuando un consumidor desea consumir un elemento pero el buffer está vacío, y cuando un productor desea insertar un elemento pero el buffer está lleno.

Para llevar a cabo esto partiremos del código original y, tras eliminar toda la lógica relacionada con el uso de variables de condición, añadiremos la necesaria para suplir su comportamiento en el código.

En este informe se incluirá un apartado sobre el programa, cómo ejecutarlo y su funcionamiento. También se incluirá un apartado de conclusiones en el que desarrollaremos los resultados obtenidos.

## II. Programa

Al igual que el programa principal, este se ha desarrollado en un único programa, escrito en C, de forma que se crean los hilos productores y consumidores a partir del programa. A cada productor se le deberá designar un archivo de texto del cual extraerá únicamente los carácteres alfanuméricos, colocándolos posteriormente en el buffer compartido, el cual tendrá un funcionamiento de cola FIFO. A continuación, los consumidores tomarán los caracteres del array y los colocarán en archivos de texto de salida, creados anteriormente por cada consumidor.

Será imprescindible gestionar aquellas condiciones de las que anteriormente se encargaban las variables de condición, tal y como detallaremos más adelante.

### A. Ejecución

Para la ejecución de este programa es suficiente con compilarlo de la siguiente manera, enlazando la biblioteca `pthread.h` de ser necesario:

```
gcc -Wall -o opcional2 opcional2.c -lpthread
```

Es necesario pasarle 3 parámetros por línea de comandos para que se ejecute correctamente: `P` (número de hilos que actuarán como Productores), `C` (número de hilos que actuarán como Consumidores)y `N` (número de posiciones que tendrá el buffer). Si alguno de estos parámetros se omite o no se introduce correctamente, el programa no se ejecutará.

## B. Funcionamiento

Una vez eliminado el uso de las variables de condición en nuestro código, debemos realizar ciertas modificaciones para controlar el comportamiento del que se encargaban.

En vez de usar las variables de condición para comprobar que hay un hueco libre en el buffer para que inserte el productor, usamos **espera activa**. Además, la comprobación debe realizarse dentro de la región crítica, ya que debemos asegurarnos de que ningún otro proceso la modifica mientras se realiza la comprobación, por lo que solicitamos el mutex del buffer. Deberá realizarse desde el mutex de la región crítica del buffer (y no otro) porque es desde esta región crítica desde la que se modificará `numElementos`, y por lo tanto la región que queremos bloquear para el resto de hilos.

Por lo tanto, se implementa como un `do-while` en el que se solicita el mutex, y una vez concedido comprueba la condición `numElementos != N`. Si se cumple significará que hay al menos una posición libre para inserción en el buffer, por lo que se sale del bucle con el mutex ya adquirido, para realizar la labor de inserción. Si no, se libera el mutex y ejecuta un `sched_yield` para propiciar que otro hilo intente acceder antes de volver a intentarlo.

En el consumidor, el procedimiento es similar. En el bucle `do-while` se solicita el mutex y se sale de la espera activa no solo cuando el buffer no está vacío, sino también en el caso de que se haya llegado al final del buffer, es decir, que se hayan leído todos los asteriscos y el buffer esté vacío. En ese caso se sale del bucle con el mutex ya obtenido, y si no se libera y se vuelve a intentar tras realizar un `sched_yield`.

Una vez fuera de la espera activa, se comprueba otra vez si estamos ante el caso de fin de la lectura, para así salir del bucle principal y finalizar el hilo. Si no, se extrae un elemento y, aún dentro de la región crítica, se comprueba si el caracter es un asterisco y se incrementa la variable. Esto se realiza dentro de la región crítica para evitar que se modifique a la vez que otro hilo lo comprueba dentro del bucle `do-while` de espera activa.

Así, se continúa el proceso y, cuando finaliza el programa, liberamos todos los recursos reservados.

# III. Conclusiones

En este informe se ha abordado la resolución del problema del productor-consumidor mediante mutexes pero sin variables de condición.

Sin embargo, para conseguirlo se ha tenido que ampliar el tamaño de la región crítica, incluyendo no solo el recuento de asteriscos sino que también se usa en la espera activa. Por todo ello, la eficiencia del programa y el grado de paralelismo se pueden ver reducidos.

Este trabajo nos muestra que, aunque no son imprecindibles, el uso de las variables de condición junto con mutexes mejoran los programas y, además, simplifican la labor de programación considerablemente.