# SISTEMAS OPERATIVOS

## ACTIVIDAD 12 (PROGRAMA 6, PRODUCTOR-CONSUMIDOR)

**Profesor:**
**VIOLETA DEL ROCIO BECERRA VELAZQUEZ**

VERDUZCO ROSALES LUIS ENRIQUE
223992388
Ingenieria en Computación

09/11/25

CUCEI DIVTIC D04

## ÍNDICE



---

## Objetivo de la actividad

El objetivo de esta actividad fue desarrollar una simulación visual del clásico problema de sincronización Productor-Consumidor. El programa busca representar de manera gráfica y comprensible la interacción entre dos procesos que comparten un recurso limitado: un productor que añade elementos a un buffer y un consumidor que los retira. Se implementó una solución que garantiza la exclusión mutua para evitar condiciones de carrera, y se utilizaron temporizadores asíncronos para simular los tiempos de trabajo y descanso de cada proceso, haciendo que la interacción sea dinámica y no determinista.

## Lenguaje Utilizado

Para este proyecto se utilizó la combinación de HTML, CSS y JavaScript. La elección de estas tecnologías fue ideal para crear una experiencia interactiva y visualmente atractiva.

- **HTML** se usó para estructurar los elementos de la simulación, como el contenedor del estante y los paneles de estado.
- **CSS** fue clave para transformar la simulación en una "librería", diseñando el buffer para que pareciera un estante de madera y representando los productos como libros.
- **JavaScript** manejó toda la lógica del programa. Su naturaleza asíncrona, a través de funciones como `setTimeout` y `setInterval`, fue perfecta para simular los ciclos independientes de sueño y trabajo del productor y el consumidor, que son el núcleo de este problema.

## Soluciones Implementadas

La simulación se construyó pensándolo como un modelo de eventos asíncronos para manejar las acciones del productor (bibliotecario) y el consumidor (cliente).

1. Entorno de la Librería y Catálogo de Libros:
   Para hacer la simulación más atractiva, se representó un estante de librería. El "producto" es una serie de imágenes de portadas de libros. Se creó un arreglo en JavaScript llamado librosDir que contiene las rutas a 26 imágenes diferentes, funcionando como un catálogo. Cada vez que el productor añade un libro, selecciona el siguiente de este arreglo, asegurando que los productos sean variados.

2. Gestión del Buffer Circular:
   El estante se implementó con un arreglo de JavaScript de tamaño fijo (cantEstante = 18). Dos variables, productorIndex y consumidorIndex, actúan como punteros para saber dónde colocar o retirar el siguiente libro. Al usar el operador módulo (%), estos índices se reinician a 0 al llegar al final del estante, simulando de manera efectiva el comportamiento de un buffer circular. La función actualizarUI se encarga de reflejar el estado de este arreglo en el HTML, creando o eliminando elementos <img> dinámicamente.

3. Exclusión Mutua (Mutex):
   Para asegurar que solo uno de los procesos pueda acceder al estante a la vez, se utilizó una variable booleana mutex. Antes de intentar producir o consumir, cada

proceso verifica si mutex es false (libre). Si lo es, lo establece en true (ocupado), realiza su trabajo y lo libera (false) al terminar. Si el recurso está ocupado, el proceso entra en un estado de espera y vuelve a intentarlo más tarde.

4. Ciclos Asíncronos de Sueño y Trabajo:
   La simulación no sigue un turno fijo. En su lugar, se usa setTimeout con tiempos aleatorios para programar cuándo "despertará" cada proceso. La función programarProximoDespertar es la encargada de esta lógica. Cuando un proceso despierta, intenta acceder al recurso. Si tiene éxito, realiza su tarea; si no, vuelve a programar un reintento.

5. Producción y Consumo en Ráfagas:
   Para que la acción de añadir o quitar libros sea visible, las funciones producir() y consumir() utilizan setInterval. En lugar de añadir o quitar todos los libros de golpe, lo hacen uno por uno cada 400 milisegundos. La cantidad de libros a procesar en cada turno también es aleatoria, variando entre 3 y 6.

## Conclusión

Esta práctica fue una excelente oportunidad para comprender los desafíos de la sincronización de procesos. Implementar el problema Productor-Consumidor me permitió ver la importancia de mecanismos como la exclusión mutua para prevenir el acceso concurrente a recursos compartidos.

El mayor reto fue manejar el estado de la aplicación de forma asíncrona, asegurando que los temporizadores y la variable `mutex` funcionaran correctamente para evitar bloqueos o condiciones de carrera. La temática de la librería no solo hizo el proyecto más entretenido, sino que también me ayudó a materializar conceptos abstractos: el "estante" como buffer, el "bibliotecario" como productor y el "cliente" como consumidor. Al final, ver la simulación funcionar fluidamente fue muy gratificante y reforzó mi comprensión sobre por qué la sincronización es un pilar fundamental en los sistemas operativos.

**Código Fuente:** https://github.com/UnluckyLuren/SisOperativos.git

**Ejecutable:** https://unluckyluren.github.io/SisOperativos/