



# Computación Concurrente

## Productor – Consumidor



### ▼ Descripción

El problema del **productor-consumidor** es un ejemplo clásico de un problema de sincronización de multiprocesos. Dicho de otra manera, este problema es un ejemplo clásico de lo que pasa cuando dos o más procesos interactúan entre sí bajo un cierto contexto. Un proceso (al igual que un hilo) es un programa de software con una serie de instrucciones definidas a ejecutar. El sistema operativo, que lo podemos pensar como un orquestador, es el encargado de decidir el proceso a ejecutar en un tiempo determinado.

Lo anterior puede llevar al caso en que un proceso esté trabajando en algo y, sin terminar, el sistema operativo le quite su tiempo de ejecución y meta a otro proceso en su lugar. Por tiempo de ejecución nos referimos al tiempo que el sistema operativo otorga a un proceso para ejecutar sus instrucciones y es posible que el tiempo que le asigne no sea el suficiente para terminar de ejecutar todas sus instrucciones. ¿Cuál es el problema de lo anterior? Bueno, pues los procesos no están conscientes de la existencia de otros procesos, por lo que un proceso no se va a detener a pensar si los datos que está leyendo son los correctos, simplemente tomará lo que esté en la memoria y trabajará con ello. Es justamente en este punto donde surge la necesidad de sincronizar procesos para que trabajen coordinadamente y entreguen el resultado correcto. La idea de tener a dos o más procesos trabajando al mismo tiempo se le conoce como *conurrencia* y dentro de ella, hay muchos problemas que pueden surgir; uno de los más simples es justamente nuestro problema del productor y consumidor.

En el nivel más básico del problema, se tienen dos entidades, cada una de éstas desempeñan roles distintos. Un rol es el de generación de datos mientras que el otro rol es el del consumo de datos. Evidentemente, un dato debe ser almacenado hasta ser consumido. Esto último nos conlleva al principal problema, nuestro almacén donde guardaremos los datos producidos es de tamaño finito. Entonces, la entidad que desempeñará el rol de productor de información no podrá producir más datos en el caso en el que nuestro almacén se encuentre lleno; por otro lado, la entidad a la cual le designaremos el rol de consumidor, no podrá consumir datos inexistentes, es decir, no podrá consumir datos en el caso en el que nuestro almacén se encuentre vacío.

Lo ideal es que las entidades no trabajen más de lo necesario. Por lo que, en ciertos momentos, las entidades pueden tomarse descansos. Para lograr este objetivo, la entidad encargada de producir debe ser capaz de informar a la otra que en el almacén hay datos disponibles para su consumo, del otro lado, la entidad que se encarga de consumir debe tener la capacidad de informar a la otra que el almacén dejó de estar lleno.

Por lo último, se puede esperar que, para que este proceso se lleve a cabo de manera satisfactoria, es necesario que exista una correcta sincronización entre las entidades así como una sana comunicación entre ellas.

Finalmente, cabe destacar que este problema tiene la propiedad de que puede ser generalizado. Podemos plantearnos una situación en la que tengamos una cantidad definida de entidades que se dedicarán a producir los datos, y otra, que se dedicará a consumir los datos.

También conocido como *Bounded Buffer Problem*.

#### ▼ Problemática

- Dos o más entidades que tienen dos roles distintos: el primero, producir o generar datos; el segundo, el de consumirlos por medio de un solo canal de comunicación.
- El canal de comunicación debe manejarse con cuidado:
  - Tiene que utilizarse en momentos distintos
  - Tiene un tamaño finito
    - Lo que se va produciendo se almacena hasta ser consumido, pero no dura para siempre ←‘fecha de caducidad’

#### ▼ Roles

### **Productor**

- Ente de software (ya sea un hilo o un proceso) encargado de generar datos relevantes para otro proceso. Ojo que el productor no será encargado de procesar esos datos, simplemente los produce y los coloca en el canal de comunicación con el consumidor.

### **Consumidor**

- Ente de software (principalmente un hilo o proceso) encargado de trabajar con los datos que el productor produzca. El consumidor será encargado de leer y quitar elementos del canal de comunicación para poder trabajar con ellos y realizar algún procesamiento.
- A los hilos y procesos los podemos entender como programas de software con una serie de instrucciones previamente definidas. Es decir, se encargan de realizar una tarea en particular.

## Canal de comunicación

- Formalmente conocido como búfer y tiene un límite de almacenamiento por lo que tenemos que cuidar que el consumidor sea capaz de *consumir* recursos de este buffer antes de que el productor *produzca* más de los que el búfer puede almacenar.
- Entendamos un búfer como una estructura de memoria capaz de almacenar datos. Las operaciones más clásicas que se pueden realizar sobre un búfer es agregar elementos y quitar elementos, similar a como funcionaria una pila o cola en estructuras de datos.
- Cuenta con un espacio de almacenamiento limitado.
  - Por ejemplo, imaginemos una bodega de un mercado. El almacenamiento que tenemos permitirá que se guarden ahí productos que van llegando del campo hasta que no se cuente con más capacidad.
    - El proceso de almacenar y retirar productos lleva un cierto tiempo
    - Los locatarios del mercado no pueden retirar productos al mismo tiempo, ya que podrían confundirse de los productos que son suyos o estorbar a otros
  - Lo almacenado tiene una caducidad
    - Siguiendo la analogía del mercado, los productos que se almacenan en la bodega después de cierta cantidad de tiempo, perecen y se vuelven inutilizables.

### ▼ Referencias

[Sistemas Operativos de Gunnar Wolf](#)

### ▼ ¿Tarea?

- ▼ Descripción de la situación
- ▼ Similitud con un problema cotidiano

- ▼ Problemas en la comunicación entre procesos
- ▼ Soluciones a problemas de comunicación entre procesos

## Mecanismos de sincronización

- ▼ Bloqueos mútuos
- ▼ Semáforos

## Exposición

### Productor-consumidor

Aa	Name
	<a href="#"><u>Relato del problema</u></a>
	<a href="#"><u>Solución al problema</u></a>
	<a href="#"><u>Mecanismo de sincronización</u></a>