# Monitores

Los semáforos son variables compartidas globales a los procesos, tienen sentencias de control de acceso a la SC dispersas en el código.  
Al agregar procesos se debe verificar que el acceso a variables compartidas sea correcto y aunque la exclusión mutua y sincronización por condición sean conceptos distintos se programan de forma similar.

Teniendo en cuenta estas características aparecen los **monitores.**

Los monitores son módulos de programa con más estructura y que pueden ser implementados tan eficientemente como los semáforos. Son más abstractos que los semáforos. Encapsulan las representaciones de recursos y brindan un conjunto de operaciones que son los únicos medios para manipular esos recursos.

Contiene variables que almacenan el estado del recurso y procedimientos que implementan las operaciones sobre él.

Cada proceso tiene sus variables locales y después tienen las variables que están dentro de los monitores, no hay más variables compartidas entre procesos.

Ejecutan la **exclusión mutua** de forma implícita asegurando que los procesos en el mismo monitor no se ejecutan concurrentemente ( solo puede entrar un proceso a la vez, un único procedure se ejecuta hasta que se duerma o salga del monitor)

La **sincronización por condición** también es explícita con el uso de variables condición. Estas variables nos permiten dormir un proceso para que espere que se cumpla una condición.

Un programa concurrente ahora está formado por procesos activos o monitores pasivos y los procesos interactúan invocando procedures de un monitor.

Tiene como ventaja que un proceso que invoca un procedure puede ignorar como está implementado y el programador del monitor puede ignorar como y donde se usan los procedures.

Diagrama

Descripción generada automáticamente

Un proceso solo puede tocar una variable compartida dentro un monitor, no puede llamarla o usarla en su implementación.

Un monitor agrupa la representación y la implementación de un recurso compartido. Tiene interfaz y cuerpo. La **interfaz** específica operaciones que brinda el recurso y el **cuerpo** tiene variables que representan el estado del recurso y procedures que implementan las operaciones de la interfaz.

Solo los nombres de los procedures son visibles desde afuera. Sintácticamente, los llamados al monitor tienen la forma **NombreMonitor.op (argumentos)**

Los procedures pueden acceder sólo a variables permanentes, sus variables locales y parámetros que le sean pasados en la invocación. Hay que tener en cuenta que quien programa el monitor nunca va a conocer el orden en que son llamados los procesos.

Texto, Carta

Descripción generada automáticamente

Pequeña imagen.

Hasta que el código de inicialización no terminó su ejecución el monitor no acepta llamados de los procedimientos.

Está prohibido hacer busy waiting con monitores. **La sincronización por condición** es programada explícitamente con variables condición 🡪 cond cv;

El valor asociado a cv es una cola de procesos demorados que no es visible directamente al programador. Sobre las variables se pueden hacer las siguientes operaciones:

* Wait(cv): El proceso se demora al final de la cola de cv y deja el acceso exclusivo al monitor.
* Signal(cv): Despierta al proceso que está al frente de la cola ( si hay alguno ) y lo saca de ella. **El procesos despertado recién podrá ejecutar cuando readquiera el acceso exclusivo al monitor**
* Signal\_all(cv): Despierta todos los procesos demorados en cv, quedando vacía la cola asociada a cv.

Hay 2 disciplinas de señalización, la primera es **signal and continued**, donde al despertar un proceso el proceso que lo despertó continúa ejecutándose hasta que libera el monitor o se duerme y el proceso despertado tendrá que competir nuevamente por el acceso al monitor, mientras que en **signal and wait** el proceso que hace el signal pasa a competir por volver a acceder al monitor mientras que el proceso despertado toma control del monitor.

Hay operaciones adicionales que NO SON USADAS EN LA PRÁCTICA:

**Empty(cv):** Retorna true si la cola controlada por cv está vacía.

**Wait(cv, rank):** El proceso se demora en la cola de cv en orden ascendente de acuerdo al parámetro rank y deja el acceso exclusivo al monitor.

**Minrank(cv):** Función que retorna el mínimo ranking de demora.

**Comparación entre wait/signal con P/V**

Tabla

Descripción generada automáticamente

[para más información sobre monitores](https://es.wikipedia.org/wiki/Monitor_de_PC)