Clase 5

[**Problemas con los semáforos 1**](#_8h5a0sly9pkc)

[**Monitores 1**](#_tjd8v0ltq1pt)

[Conceptos básicos 1](#_f6wli7m85jw3)

[Sincronización por condición 2](#_5y5m9840saxf)

[Disciplinas de señalización 2](#_cvuchobbeylt)

[Operaciones adicionales de la teoría: 2](#_zbjfkp59z0uq)

# Problemas con los semáforos

* Variables compartidas globales a los procesos
* Sentencias de acceso a la SC dispersas en el código
* Al agregar procesos se debe verificar acceso correcto a las variables compartidas
* Exclusión mutua y sincronización por condición se hacen usando la misma herramienta y puede darse confusión al usarlos (usar un semáforo de exclusión mutua para otra cosa, etc).

# Monitores

* Módulos de programa
* Eficientes como los semáforos
* Encapsulan recursos, brindan operaciones para manipularlos
* Programas ahora con monitores y procesos, no hay variables compartidas, cada proceso tiene sus variables locales e interactúa con monitores

## Conceptos básicos

* **Exclusión mutua implícita**, una vez llamado un procedimiento de un monitor, hasta que no se duerma o termina otro proceso no podrá ejecutarlo.
* **Sincronización por condición explícita**, con variables condición.
* Programas con procesos activos y monitores pasivos. Procesos activos concurrentes llaman procedimientos del monitor y se abstraen.
* No es un **TAD** por que se usa concurrentemente, poseen interfaz y cuerpo.
  1. Interfaz: Especifica operaciones.
  2. Cuerpo: Variables y procedimientos que implementan la interfaz.
  3. NombreMonitor.opi(args)
  4. No se puede llamar el orden de llamado de los procedimientos y se deben implementar como tal.

monitor NombreMonitor {

declaraciones de variables permanentes;

código de inicialización

procedure op1(par. formales1){

cuerpo de op1

}

.......

procedure opn(par. formalesn){

cuerpo de opn

}

}

## Sincronización por condición

* Programada explícitamente con variables de condición (cond cv) solo en los monitores.
* El valor asociados internamente a cv es una cola de procesos demorados, para interactuar con la cola usamos:
  + wait(cv) -> proceso x llama un procedimiento del monitor, se agrega a la cola de dormidos y luego libera el monitor para que otros puedan seguir
  + signal(cv) -> despierta el primer proceso de la cola y sigue la ejecución del procedimiento
  + signal\_all(cv) -> despierta todos

### Disciplinas de señalización

* Signal and continued => es el utilizado en la materia
* Signal and wait

Determinan quién es el que continúa un proceso luego de levantar otro.

### Operaciones adicionales de la teoría

* Empty(cv) -> te dice si hay alguno dormido.
* wait(cv, rank) -> los duerme según una condición o prioridad
* minrank(cv) -> permite ver la prioridad de un proceso