Contenidos

- El problema de la sección crítica
- Semáforos
- Regiones críticas
- **Monitores**

Bibliografía

- Programación Concurrente
  - J. Palma, C. Garrido, F. Sánchez, A. Quesada, 2003
  - Capítulo 6
- Concurrent Programming
  - A. Burns, G. Davies. Addison-Wesley, 1993
  - Capítulo 7
- Principles of Concurrent and Distributed Programming
  - M. Ben-Ari. Prentice Hall, 1990
  - Capítulo 5
- Sistemas Operativos
  - A. Silberschatz, P. Galvin. Addison-Wesley, 1999
  - Capítulo 6

Monitor

- Tipo de datos
  - Estructuras de datos
  - Conjunto de operaciones asociadas a tales estructuras

  +
  - Exclusión mutua
  - Sincronización (variables condición)

Monitor

```
monitor nombre_monitor;
   var variables locales;
   export procedimientos exportados;

   procedure P1(parametros)
   var variables locales;
   begin
         {código del procedimiento}
   end;
   ...
   procedure PN(parametros)
   var variables locales;
   begin
      {código del procedimiento}
   end;
begin
   {código de inicialización}
end;
```


Variables condición

- Declaración
  - var x: condition;
- Operaciones
  - delay(x);
  - resume(x);
  - empty(x);

Variables condición

- ¿ Qué ocurre cuando un proceso P realiza una operación ***resume*** sobre una variable condición x y existe un proceso suspendido Q asociado a dicha variable?

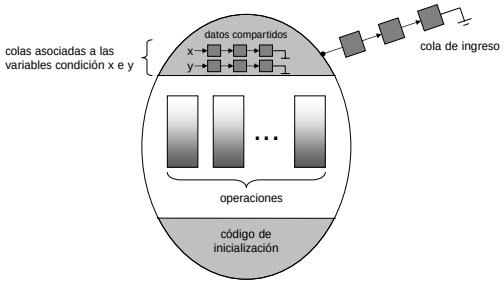
Semántica de la operación resume

- Desbloquear y continuar
  - Lenguaje Mesa
- Retorno forzado
  - Concurrent Pascal
- Desbloquear y esperar
  - Modula-2, Concurrent Euclid
- Desbloquear y espera urgente
  - Pascal-FC, Pascal Plus

Variables condición

- Si varios procesos están suspendidos por la condición x y algún proceso ejecuta x.signal, ¿ qué proceso se reanuda?

Monitor con variables condición



Ejercicios

- En un sistema concurrente existen dos tipos de procesos, A y B, que compiten por utilizar cierto recurso. Al recurso se accede mediante la rutina de solicitarlo, esperar hasta que se pueda usar, usarlo y luego liberarlo. En cualquier momento puede haber un máximo de N procesos de cualquier tipo usando el recurso (N es constante). Por otro lado, para que un proceso de tipo A pueda entrar a emplear el recurso, debe haber al menos el doble de procesos de tipo B que procesos de tipo A dentro del recurso. Diseñe un algoritmo que cumpla con estas reglas de funcionamiento empleando regiones críticas

Ejercicios

- Supongamos dos operaciones: pedir_memoria(cantidad) y dejar_memoria(cantidad). Cuando un proceso pide memoria, se bloquea hasta que haya la cantidad pedida de páginas libres, una vez que ocurre las toma. Un proceso devuelve las páginas llamando a dejar_memoria. Implementar dichas operaciones usando RCC en los siguientes supuestos:
  - Implementar la sincronización sin establecer ninguna política de quién recibe primero.
  - Modificar lo anterior para que el trabajo que pide menos memoria sea el primero.
  - Cambiar la política anterior para que el primero en llegar sea el primero en salir o ser atendido.
  - Suponer que pedir y dejar devuelven páginas contiguas. Es decir, si un proceso reclama dos páginas, se retrasa hasta que haya dos páginas adyacentes disponibles. Desarrollar implementaciones de esa versión de pedir_memira y dejar_memoria. Elegir una representación del estado de páginas de memoria.