Productor-Consumidor

Luis Fernando Flores Tiburcio
José David Vázquez Rojas
Yañez Espindola Jose Marcos
Hugo Carlos Moran Peraza

Planteamiento del problema

El problema lo podemos plantear dado un producto en una tienda/almacén y el límite de oferta de dicho producto con una producción y adquisición de este producto de forma simultánea.

¿ Cómo podemos organizar la creación de nuevos productos de tal forma que no exceda el límite de oferta y al consumidor para que no intente comprar el producto cuando no hay en existencia?.

Esquema del problema

Consumidor

Almacén

Productor

Solución propuesta.

Pensemos que cada unidad de tiempo podemos crear un nueva unidad de este artículo y que el consumidor puede o no adquirir este artículo.

Empezamos con el productor con ciertas unidades enviadas en almacén mientras que el consumidor adquiere estos productos en una cierta unidad de tiempo independiente de la producción. Lo anterior, el productor estará produciendo artículos y enviandolos al almacén mientras el consumidor tomará estos de forma simultánea en que se producen.

Ahora, si el almacén tiene 0 existencia de artículos, este avisará al consumidor para entrar a un estado "dormido" y esperará hasta que el productor envié un artículo a almacen para "despertar" al consumidor.

Por otra parte, si el almacén llegase a tener su capacidad llena, almacén le avisará al productor para ponerlo en estado "dormido" hasta que el consumidor adquiera un artículo en almacén y despertará al productor.

Esquema del problema

Consumidor

No hay producto

Ya hay productos

Almacén

Límite excedido

Cupo disponible

Productor

Escenario 1

Cantidad artículos:
0

Consumidor

No hay producto

Almacén

Cupo disponible

Productor

Escenario 2

Cantidad artículos:
500 (excedido)

Consumidor

Ya hay productos

Almacén

Límite excedido

Productor

Escenario 3

Cantidad artículos:
23

Consumidor

Ya hay productos

Almacén

Cupo disponible

Productor

Monitor

Es un tipo abstracto de datos que encapsula:

- Tanto un conjunto de variables (internas)
- Como una serie de operaciones de acceso

Propiedades

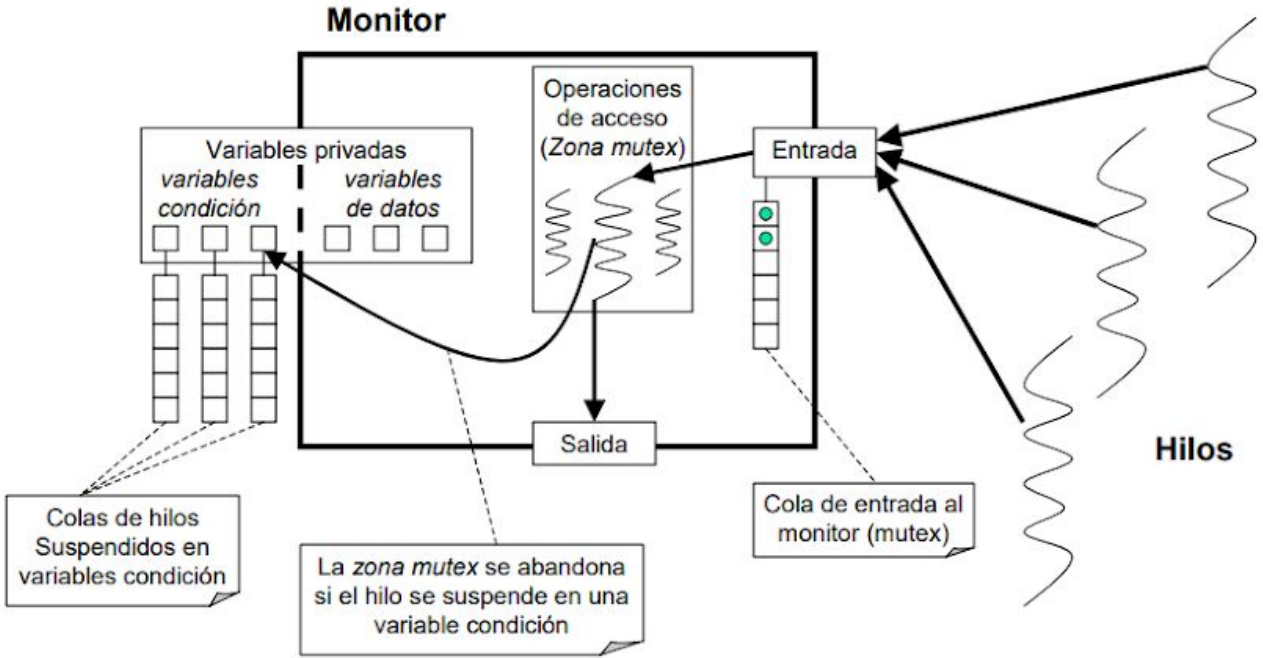
Los procesos/hilos pueden únicamente invocar a las operaciones de acceso.

Las variables internas sólo pueden ser accedidas desde el código de las operaciones de acceso (encapsulamiento)

El código de las operaciones se ejecuta en exclusión mutua de forma automática

Se proporciona además un mecanismo general de suspensión, basado en variables denominadas “condición”

- Los procesos/hilos pueden suspenderse en una variable condición
- Otro proceso/hilo puede despertar a un proceso/hilo suspendido anteriormente


Monitor
Operaciones
de acceso
(Zona mutex)
Variables privadas
variables
condición
variables
de datos
Entrada
Salida
Hilos
Colas de hilos
Suspendidos en
variables condición
La zona mutex se abandona
si el hilo se suspende en una
variable condición
Cola de entrada al
monitor (mutex)

Sintaxis

```
type nombre_monitor = monitor

... Declaración de variables

procedure entry P1 (...);
   begin
      ... Código de P1
   end;

function entry F2 (...);
   begin
      ... Código de F2
   end

...
begin
   ... Código de inicialización
end;
```

Variables internas al monitor:
- variables compartidas
- variables de condición

Métodos de acceso:
- única vía de acceso a las variables compartidas
- Exclusión mutua automática

Inicialización:
- Se invoca automáticamente al instanciar el monitor y antes de ser accedido por ningún proceso
- Se utiliza para la inicialización de las variables del monitor

```
type buffer_acotado = monitor
   const N = 100;
   contador, entrada, salida: integer;
   lleno, vacio: condition;
   buffer: array[0..N-1] of integer;

procedure entry insertar
                  (item: interger);
   begin
      if (contador = N) then
         lleno.wait;
      contador:= contador +1;
      buffer[entrada] := item;
      entrada:= (entrada+1)mod N;
      vacio.signal;
   end;
(...)
```

```
(...)
procedure entry extraer
                  (var item: integer);
   begin
      if (contador = 0) then
         vacio.wait;
      contador:= contador - 1;
      item:= buffer[salida];
      salida:= (salida+1) mod n;
      lleno.signal;
   end;

begin
   entrada:= 0;
   salida:= 0;
   contador:= 0;
end;
```

```
class Monitor:
        contador = 0
        N=100
        buffer = array[N-1]
        def insertar():
                if (contador=N):
                        sleep()
  contador=contador+1    # Sección Crítica
  buffer.push(producto)  # Sección Crítica
```

Esquema con el monitor implementado.