Concurrencia entre Procesos

Sistemas Operativos

M. en C. Violeta del Rocío Becerra Velázquez

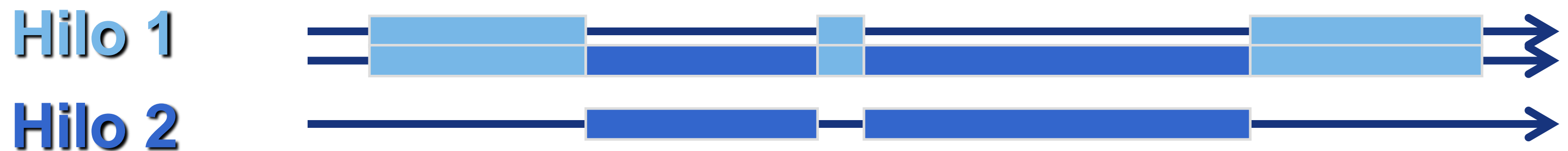
Marzo 2014.

Concurrencia vs. Paralelismo

- Concurrencia: dos o más hilos están en progreso al mismo tiempo:

Hilo 1

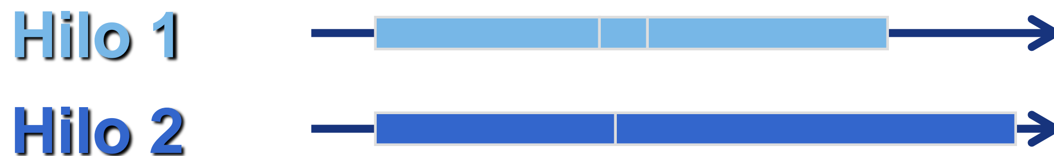
Hilo 2



- Paralelismo: dos o más hilos están ejecutandose al mismo tiempo.

Hilo 1

Hilo 2



- Se requieren varios núcleos.

Tipos de Procesos Concurrentes

- Independientes: Aquel que se ejecuta sin requerir la ayuda o cooperación de otros procesos.

- Cooperantes: Cuando están diseñando para trabajar conjuntamente en alguna actividad, para lo que deben ser capaces de comunicarse e interactuar entre ellos.

Interacciones entre procesos

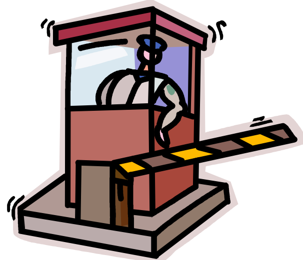
- **Comparten o compiten** por el acceso a recursos físicos o lógicos.
- **Comunicación y sincronización** para alcanzar un objetivo común.

Interacción entre Procesos

Grado de Conocimiento	Relación	Influencia de un proceso en los otros	Posibles Problemas de Control
Los procesos no tienen conocimiento de los demás	Competencia	• Los resultados de un proceso son independientes de las acciones de los otros Los tiempos de los procesos pueden verse afectados	• Exclusión mutua • Interbloqueo (recursos renovables) • Inanición
Los procesos tienen conocimiento indirecto de los otros (por ejemplo, objetos compartidos)	Cooperación por compartición	• Los resultados de un proceso pueden depender de la información obtenida de los otros • Los tiempos de los procesos pueden verse afectados	• Excusión mutua • Interbloqueo (recursos renovables) • Inanición • Coherencia de datos
Los procesos tiene conocimiento directo de los otros (hay disponibles unas primitivas de comunicaciones)	Cooperación por comunicación	• Los resultados de un procesos pueden depender de la información obtenida de los otros • Los tiempos de los procesos pueden verse afectados	• Interbloqueo (recursos consumibles) • Inanición

Exclusión mutua

- Región crítica
  - Porción de código que accesa (lee y escribe) variables compartidas.
- Exclusión mutua
  - Lógica del programa que forza a un hilo para que acceda la sección crítica.
  - Permite corregir estructuras de programación para evitar condiciones de concurso.



- Ejemplo: *Depositar en una caja de seguridad*
  - Quienes la atienden aseguran la exclusión mutua.

Sincronización

- Objetos de sincronización usados para forzar la exclusión mutua
  - Locks, semaforos, secciones críticas, eventos, variables condición, atómicas.
  - Un hilo “retiene” objetos de sincronización; otros hilos deben esperar.
  - Al término, los hilos que retienen los objetos los liberan; el objeto se le entrega a un hilo que esté esperando.

- Ejemplo: *Biblioteca*
  - Una persona tiene prestado un libro.
  - Otros deben esperar a que el libro regrese.

Soluciones por Software

Algoritmo de Dekker

Primer Intento

var turno: 0..1

Proceso 0

.

.

while turno ≠ 0
  do {nada};

<sección crítica>

turno := 1 ;

.

Proceso1

.

.

while turno ≠ 1
  do {nada};

<sección crítica>

turno := 0;

.

Primer Intento

- Los Procesos deben alternarse de forma estricta en el uso de sus secciones críticas; así el ritmo de ejecución viene dictado por el más lento.

- Si un proceso falla, el otro proceso se bloquea permanentemente. Tanto si falla en su sección crítica como fuera de ella.

Segundo Intento

var señal : array[0..1] of booleano;

Proceso 0
.
.
while señal[1] do {nada};
señal[0] := cierto;
<sección crítica>
señal[0] := falso;
.

Proceso 1
.
.
while señal[0] do {nada};
señal[1] := cierto;
<sección crítica>
señal[1] := falso;
.

Fallo 1er. Intento: Tener Información general del Estado de Cada Proceso.

- Si un proceso falla en su sección crítica, el otro proceso quedará bloqueado permanentemente.

- Si ambos procesos activan sus pizarras, los dos procesos tendrán cierto, y entraran a su sección crítica, (puede no cumplirse la exclusión mutua).

Tercer Intento

```
Proceso 0
.
.
.
Señal[0] := cierto;
While señal[1] do {nada};
<sección crítica>
Señal[0] := falso;
.
```

```
Proceso 1
.
.
.
Señal[1]:= cierto;
While señal[0] do {nada};
<sección crítica>
Señal[1] := falso;
.
```

Tercer Intento

Fallo 2do. Intento: Un proceso puede cambiar su estado después de que el otro proceso lo ha comprobado pero antes de que pueda entrar en su sección crítica.

- Si un proceso falla dentro de la sección crítica, incluyendo el código para dar valor a las señales que controlan el acceso a esta, el otro proceso se bloquea y si falla fuera el otro proceso no se bloquea.

- Si ambos procesos ponen sus señales en "cierto", antes del while, tendríamos un caso clásico de interbloqueo.

Cuarto Intento

```
Proceso 0
.
.
señal[0] := cierto;
while señal[1] do
begin
        señal[0] := falso;
        <espera cierto tiempo>
        señal[0] := cierto;
end;
<sección crítica>
señal[0] := falso;
.
```

```
Proceso 1
.
.
señal[1] := cierto;
while señal[0] do
begin
        señal[1] := falso;
        <espera cierto tiempo>
        señal[1] := cierto;
end;
<sección crítica>
señal[1] := falso;
.
```

Cuarto Intento

Fallo 3er. Intento: Un proceso fija su estado sin conocer el estado del otro. El interbloqueo se da por que cada proceso insiste en entrar a su sección crítica.

- Se deben hacer los procesos más educados, deben activar su señal para indicar que desean entrar en la sección crítica, pero deben estar listos para desactivar la señal y ceder la preferencia al otro proceso.

- Excesiva Cortesía.

Algoritmo de Dekker. Solución Correcta.

```
var señal : array[0..1] of booleano;
        turno : 0..1;
procedure P0;
begin
    repeat
    señal[0] := cierto;
    while señal[1] do
        if turno = 1 then
      begin
        señal[0] := falso;
        while turno = 1 do
            {nada};
        señal[0] := cierto;
      end
      <sección crítica>
    turno := 1;
señal[0] := falso;
<resto>
    forever
end;
```

```
procedure P1;
begin
    repeat
        señal[1] := cierto;
        while señal[0] do
          if turno = 0 then
            begin
                señal[1] := falso;
                while turno = 0 do
                 {nada};
                señal[1] := cierto;
            end
        <sección crítica>
        turno := 0;
        señal[1] := falso;
        <resto>
    forever
end;
```

Algoritmo de Dekker. Solución Correcta.

```
begin
      señal[0] := falso;
      señal[1] := falso;
      turno := 1;
      parbegin
         P0 ; P1;
      parend
end
```

Algoritmo de Peterson

```
var señal : array[0..1] of booleano;
turno : 0..1;

procedure P0;
begin
     repeat
    señal[0] := cierto
           turno := 1;
while señal[1] and turno = 1 do {nada};
<sección crítica>
señal[0] := falso;
    <resto>
     forever
end;
```

Algoritmo de Peterson

```
procedure P1;
begin
     repeat
    señal[1] := cierto
           turno := 0;
while señal[0] and turno = 0 do {nada};
<sección crítica>
señal[1] := falso;
    <resto>
     forever
end;
```

```
begin
     señal[0] := falso;
     señal[1] := falso;
     turno := 1;
     parbegin
        P0; P1
     parend
end
```

Semáforos Generales

Un semáforo es un objeto con un valor entero, al que se le puede asignar un valor inicial no negativo y al que sólo se puede acceder utilizando dos operaciones atómicas: wait (down, P) y signal (up, V).

```
wait (s) {
        s = s – 1;
        if ( s  <  0 )
                <Bloquear al proceso>;
}
signal (s) {
        s = s + 1;
        if ( s <= 0 )
                <Desbloquear a un proceso bloqueado
                 en la operación wait>;
}
```

Semáforos

wait(s); /* Entrada en la Sección Crítica */

<Sección Crítica>

signal(s); /* Salida de la Sección Crítica */

Valor del semáforo (s)	P0	P1	P2
1			
0	wait(s)		
-1		wait(s)	
-2			wait(s)
-1	signal(s) — desbloquea		
0		signal(s) — desbloquea	
1			signal(s)

Ejecutando código de la sección crítica

Proceso bloqueado en el semáforo

Semáforos Binarios

```
Type semaforo binario :
record
        valor: (0,1);
        cola: list of proceso;
End

Var s:semaforo binario
waitB(s):
        if s.valor = 1 then s.valor := 0
        else <bloquear proceso poner en s.cola>

signalB(s):
        if s.cola esta vacia then s.valor := 1
        else <quitar un proceso de s.cola>
```

Monitores

Un monitor es una colección de procedimientos, variables y estructuras de datos que se agrupan en un tipo especial de módulo o paquete. Los procesos pueden invocar a los procedimientos de un monitor cuando lo deseen, pero no pueden acceder en forma directa a sus estructuras de datos internas desde procedimientos declarados fuera de dicho monitor.

Solo un proceso puede estar activo en un monitor a la vez.

Bloqueo Mutuo

(Deadlocks,
Interbloqueo,
Abrazo Mortal)

Interbloqueo

- Se caracteriza por la existencia de un conjunto de ***entidades activas*** que usan un conjunto de ***recursos***.

Entidades Activas:

- Procesos y Threads.

Recursos:

- Físicos:
  - UCPs, memoria, dispositivos etc.
- Lógicos:
  - Archivos, semáforos, mensajes, señales, etc.

Tipos de Recursos

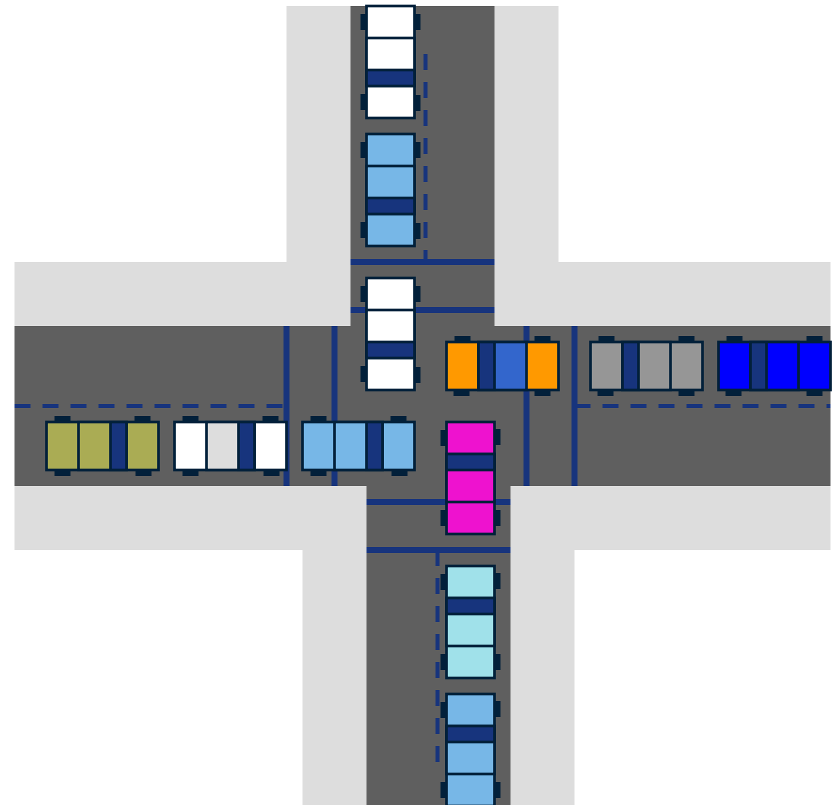
- **Reutilizables o Consumibles**
  ¿Sigue existiendo después de usarse?
- **Uso Dedicado o Compartido**
  ¿Pueden usarlo varios procesos simultáneamente?
- **Con uno o múltiples ejemplares**
  ¿Existen múltiples ejemplares del mismo recurso?
- **Expropiables o no Expropiables**
  ¿Es factible expropiar el recurso cuando se esta usando?

Interbloqueo

- Los hilos esperan por un evento o condición que nunca sucede

Ejemplo:

- Intersección de coches
- Los coches no pueden echarse en reversa



Definición de Interbloqueo

- Conjunto de Procesos tal que cada uno está esperando un recurso que solo puede liberar (generar, si son consumibles) otro proceso del conjunto.

Condiciones de Interbloqueo

- Exclusión mutua:

  Recursos de Uso Exclusivo.

- Retención y espera:

  Mientras los procesos esperan por recursos pedidos, mantiene los ya asignados.

Condiciones de Interbloqueo

- No Apropiación:

  Ningún proceso puede ser forzado a abandonar un recurso que retenga.

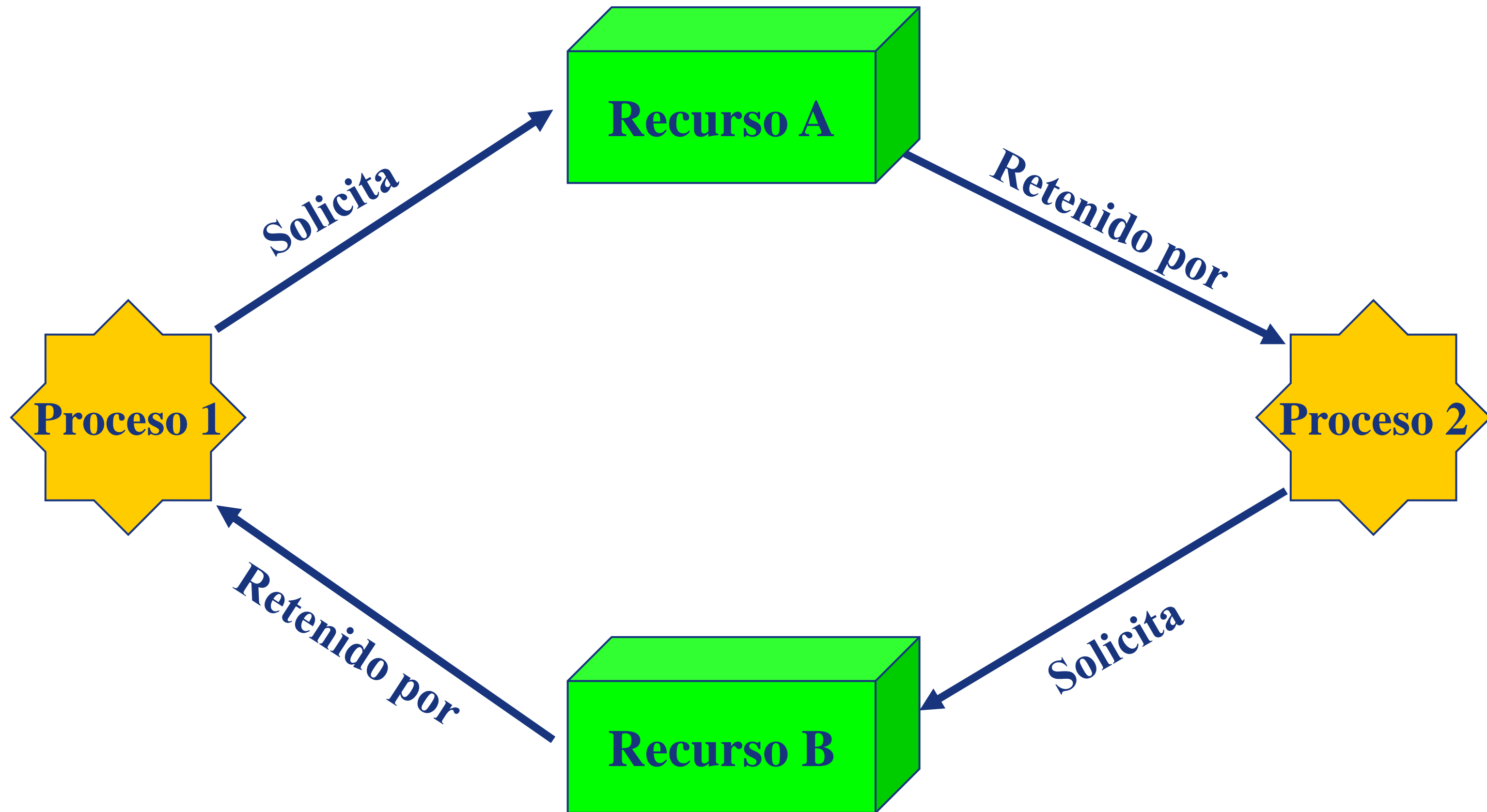
Puede existir interbloqueo con las condiciones anteriores, más no necesariamente. Para que se produzca se necesita una cuarta condición.

Condiciones de Interbloqueo

- Espera Circular / Circulo vicioso de espera:

  Existe una lista de procesos circular tal, que cada proceso espera por recursos que tiene asignados el siguiente proceso de la lista.

Espera Circular



Tratamiento del Interbloqueo

- Prevención:
  - Asegura que no ocurra fijando reglas para pedir recursos.
- Predicción:
  - Asegura que no ocurra basándose en el conocimiento de necesidades futuras de los procesos.

Tratamiento del Interbloqueo

- Detección y recuperación:
  - Permite que se produzca, lo detecta y lo recupera del mismo.
- Ignorar el problema:
  - Consiste en no realizar ninguna acción.