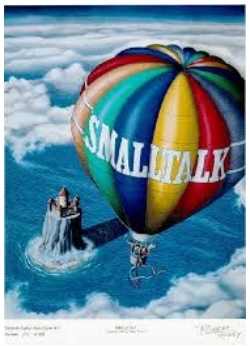
Departamento de Programación
Facultad de Informática
Universidad Nacional del Comahue

Programación Concurrente



Instrumentos de la concurrencia

Temario

- Condiciones para la Concurrencia / Berstein
- Problemas de la PC
- Sincronización
- Propiedades de la PC
  - Propiedades de seguridad (safety)
  - Propiedades de viveza (liveness)
- Problemas clásicos de sincronización
  - Productor/Consumidor
  - Lector/Escritor
  - Barbero dormilón
  - Filósofos cenando

Recordemos...concurrencia

- Proceso: secuencia de acciones que se realizan independientemente de las acciones realizadas por otros procesos.

En general, la prioridad la asigna el SO

- Los procesos que se ejecutan en paralelo, son objetos que poseen una prioridad, la cual puede asignarse en un principio e ir modificándose a lo largo de la ejecución.

- La prioridad de un proceso, describe la importancia que tiene ese proceso por sobre los demás.

Condiciones para la concurrencia

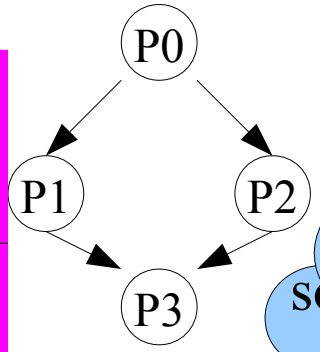
Al compartir datos entre procesos se pueden producir problemas de indeterminismo (resultados diferentes según escenario de prueba).

```
Proceso P0
    x:= 0;
End;
```

```
Proceso P3;
 escribir('fin');
 end;
```

```
proceso P1;
     var i: integer;
begin
   for i:=1 to 100 do x:= x+1;
end;
```

```
proceso P2;
     var j: integer;
begin
   for j:=6 to 150 do x:= x+1;
end;
```



P1 y P2 no son independientes, comparten la variable x

Las condiciones de Berstein!!

Condiciones para la concurrencia

Las ***condiciones de Berstein*** determinan que conjunto de instrucciones (procesos) se pueden ejecutar en forma concurrente y determinista

Debemos considerar:

- El conjunto de instrucciones/procesos

  S1, S2, … Sn

- El conjunto de lectura de cada S_k , k:1..n (variables accedidas)
- El conjunto de escritura de cada S_k (variables modificadas)

Condiciones para la concurrencia

Sea:

- $L(S_k)=\{a_1,a_2,...,a_n\}$ el conjunto formado por todas las variables cuyos valores son referenciados (se leen) durante la ejecución de S_k
- $E(S_k)=\{b_1,b_2,...,b_n\}$ el conjunto formado por todas las variables cuyos valores son actualizados (se escriben) durante la ejecución de S_k
- Para que dos conjuntos de instrucciones S_i y S_j se puedan ejecutar concurrentemente, se debe cumplir que:

$$L(S_i) \cap E(S_j) = \emptyset$$

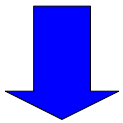
$$E(S_i) \cap E(S_j) = \emptyset$$

$$E(S_i) \cap L(S_j) = \emptyset$$

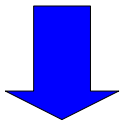
Condiciones para la concurrencia

Entonces …
Veamos analíticamente si P1 y P2 pueden ejecutarse de forma concurrente y determinista

L(P1)= {x, i} , E(P1) = {x, i}
L(P2)= {x, j} , E(P2) = {x, j}



L(P1) ∩ E(P2) = {x}
E(P1) ∩ E(P2) = {x}
E(P1) ∩ L(P2) = {x}



No se puede

```
Proceso P0
    x:= 0;
End;
```

```
Proceso P3;
 escribir('fin´);
 end;
```

```
proceso P1;
     var i: integer;
begin
   for i:=1 to 100 do x:= x+1;
End;
```

```
proceso P2;
     var j: integer;
begin
   for j:=6 to 150 do x:= x+1;
end;
```

Condiciones para la concurrencia

condiciones de Berstein

Condiciones suficientes (pero no necesarias)

sólo se pueden compartir variables de lectura.

Nos interesa: Buscar soluciones al indeterminismo sin que se cumplan las condiciones de Bernstein, es decir pudiendo compartir *variables de escritura*.

sincronización

Problemas en lenguajes de alto nivel

- Una instrucción de alto nivel se convierte en un conjunto de instrucciones máquina. Ejemplo ++x :
  - Obtener x
  - Tomar la constante 1
  - Agregar a x (hacer x+1)
  - Escribir en x
- Las instrucciones de máquina son las que se ejecutan **concurrentemente**
- En ejecuciones concurrentes el resultado puede ser **incorrecto**
  - Diferentes posibilidades en cuanto al orden de ejecución (ver ejemploClase3.pdf)
  - Pueden ser necesarias ciertas restricciones al orden de ejecución

Problemas en lenguajes de alto nivel

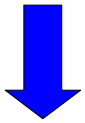
Generalmente existen partes de código con variables compartidas y que deben ejecutarse en **exclusión mutua,**

es decir tomar las operaciones que actúan sobre la variable compartida como **atómicas**.

. . .

contador.incrementar();

. . .

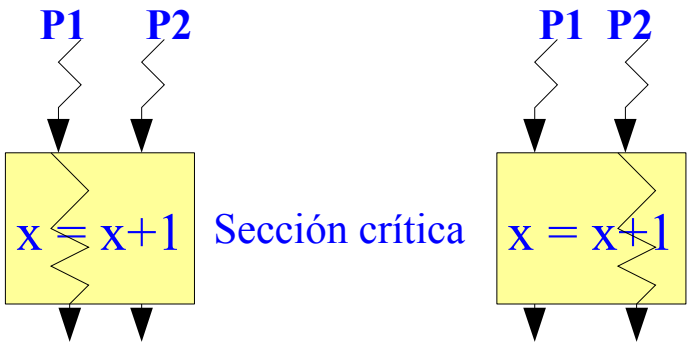


Sección crítica

Hay que sincronizar el acceso a la variable compartida

Problemas de la PC

- Exclusión mutua
  - Sección crítica: porción de código con variables compartidas y que debe ejecutarse en exclusión mutua



- Condición de sincronización
- Verificación

Programas correctos

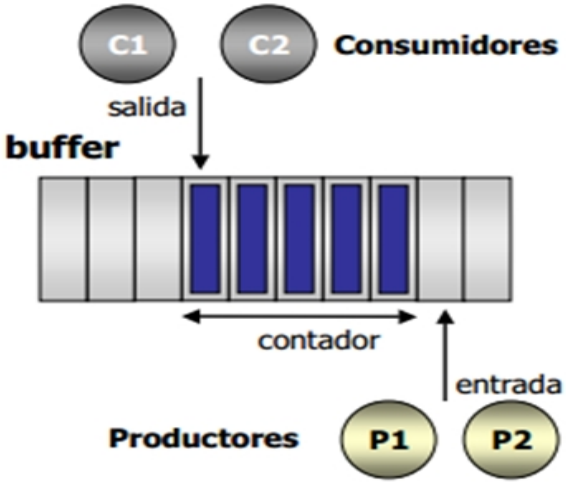
- (PS) Programa secuencial:
  - **Parcialmente correcto:** Dadas unas precondiciones correctas, **si** el programa termina **entonces** se cumplen las postcondiciones
  - **Totalmente correcto:** Dadas unas precondiciones correctas el programa termina y se cumplen las postcondiciones
- (PC)Programación concurrente:
  - Pueden no terminar nunca y ser **correctos.**
  - Puede tener múltiples secuencias de ejecución
  - Cuando es correcto es porque se refiere a **todas** sus posibles secuencias de ejecución.

PC-Propiedades

- Tipos de propiedades de la PC
  - Propiedades de seguridad **(safety)**
    - Son aquellas que aseguran que "*nada malo va a pasar durante la ejecución del programa*"
      - Exclusión mutua
      - Condición de sincronización
      - Interbloqueo (pasivo) - deadlock
  - Propiedades de viveza **(liveness)**
    - Son aquellas que aseguran que "*eventualmente algo bueno pasará durante la ejecución del programa*"
      - Interbloqueo (activo) – livelock
      - Inanición - starvation

Productores/Consumidores

Una parte produce algún producto (datos en nuestro caso) que se coloca en algún lugar (una cola en nuestro caso) para que sea consumido por otra parte.



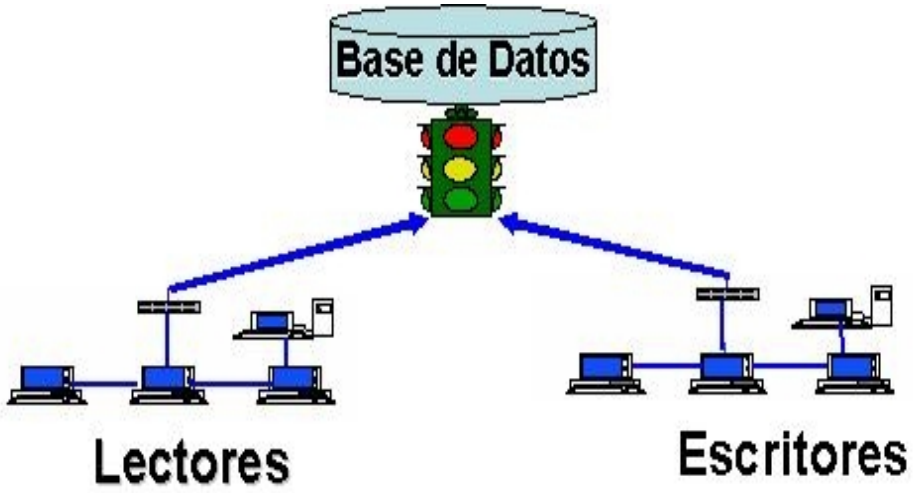
- El productor genera sus datos en cualquier momento
- El consumidor puede tomar un dato sólo cuando hay
- Para el intercambio de datos se usa una cola a la cual ambos tienen acceso, asi se garantiza el orden correcto
- Todo lo que se produce debe ser consumido

Tenemos que garantizar que el consumidor no consuma más rápido de lo que produce el productor

Acceso de los productores y consumidores de forma de asegurar consistencia de la información almacenada en la cola.
La cola tiene capacidad limitada: a) un productor no puede producir un elemento en una cola llena, b) un consumidor no puede extraer un elemento de una cola vacía

Problema de sincronización:
Lectores/escritores

Dos grupos de procesos (Lectores/escritores) que utilizan los mismos recursos (documento)



- Un grupo de lectores, sólo leen los datos.
- Los escritores, leen y escriben.
- Varios lectores pueden acceder simultáneamente a un proceso compartido
- Se debe evitar que accedan simultáneamente un proceso escritor y cualquier otro proceso.

Problema de sincronización:

Barbero dormilón

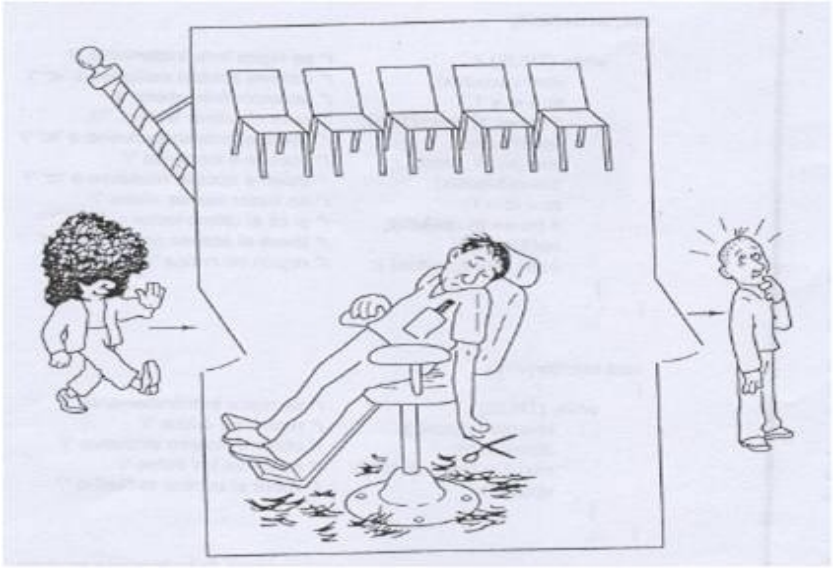
En una barbería trabaja un barbero que tiene un único sillón de barbero y varias sillas para esperar.

- Cuando no hay clientes, el barbero se sienta en una silla y se duerme.

-Cuando llega un nuevo cliente,

-si el barbero duerme: despierta al barbero o

-si el barbero está afeitando a otro cliente: se sienta en una silla



(si todas las sillas están ocupadas por clientes esperando, se va).

Problema de sincronización:

Cena de filósofos

Cinco filósofos se sientan alrededor de una mesa y pasan su vida cenando y pensando. Cada filósofo tiene un plato de fideos y un tenedor a la izquierda de su plato. Para comer los fideos son necesarios dos tenedores y cada filósofo sólo puede tomar los que están a su izquierda y derecha.

Si cualquier filósofo toma un tenedor y el otro está ocupado, se quedará esperando, con el tenedor en la mano, hasta que pueda tomar el otro tenedor, para luego empezar a comer.

Si dos filósofos adyacentes intentan tomar el mismo tenedor a una vez, ambos compiten por tomar el mismo tenedor, y uno de ellos se queda sin comer.

Si todos los filósofos toman el tenedor que está a su derecha al mismo tiempo, entonces todos se quedarán esperando eternamente. Entonces los filósofos se morirán de hambre