SISTEMAS OPERATIVOS: COMUNICACIÓN Y SINCRONIZACIÓN ENTRE PROCESOS

Procesos concurrentes y problemas en la comunicación y la sincronización

Contenido

- Concurrencia.
- Condiciones de carrera.
- Exclusión mutua y sección crítica.
- Semáforos.
- El problema del productor consumidor.
- El problema de los lectores escritores.

Proceso concurrente : Dos procesos ejecutandose en el mismo momento

- Dos procesos son concurrentes cuando se ejecutan de manera que sus intervalos de ejecución se solapan. No que uno de paso a otro.

Si hay concurrencia

No hay concurrencia

Tipos de concurrencia

- Concurrencia aparente: Hay más procesos que procesadores.
  - Los procesos se multiplexan en el tiempo.
  - Pseudoparalelismo

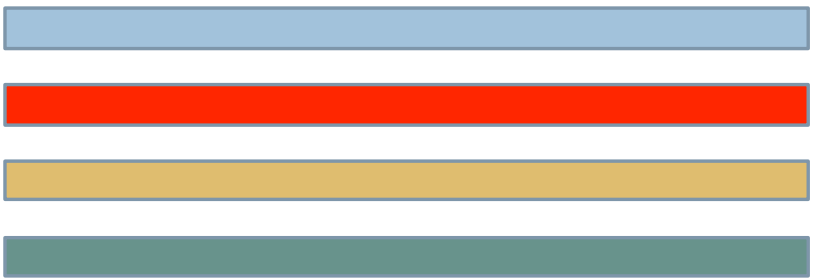
1 CPU

2 CPUs

Tipos de concurrencia

- Concurrencia real: Cada proceso se ejecuta en un procesador.
  - Se produce una ejecución en paralelo.
  - Paralelismo real.

4 CPUs



Modelos de programación concurrente

- Multiprogramación con un único procesador
  - El sistema operativo se encarga de repartir el tiempo entre los procesos (planificación expulsiva/no expulsiva). Lo puede incluso expulsando, esto es inherente.
- Multiprocesador
  - Se combinan paralelismo real y pseudoparalelismo.
    - Normalmente más procesos que CPU's.
- Sistema distribuido
  - Varios computadores conectados por red.

Ventajas de la ejecución concurrente

- Facilita la programación.
  - Diversas tareas se pueden estructurar en procesos separados.
  - Servidor Web: Un proceso encargado de atender a cada petición.
- Acelera la ejecución de cálculos.
  - División de cálculos en procesos ejecutados en paralelo.
  - Ejemplos: Simulaciones, Mercado eléctrico, Evaluación de carteras financieras.
- Mejora la interactividad de las aplicaciones.
  - Se pueden separar las tareas de procesamiento de las tareas de atención de usuarios.
  - Ejemplo: Impresión y edición.
- Mejora el aprovechamiento de la CPU.
  - Se aprovechan las fases de E/S de una aplicación para procesamiento de otras.

Tipos de procesos concurrentes

- Independientes.
  - Procesos que se ejecutan concurrentemente pero sin ninguna relación.
    - No necesitan comunicarse.
    - No necesitan sincronizarse.
    - Ejemplo: Dos intérpretes de mandatos de dos usuarios ejecutados en distintos terminales.
- Cooperantes.
  - Procesos que se ejecutan concurrentemente con alguna interacción entre ellos.
    - Pueden comunicarse entre si.
    - Pueden sincronizarse.
    - Ejemplo: Servidor de transacciones organizado en proceso receptor y procesos de tratamiento de peticiones.

Comportamiento indeterminista, no siempre hace lo mismo.

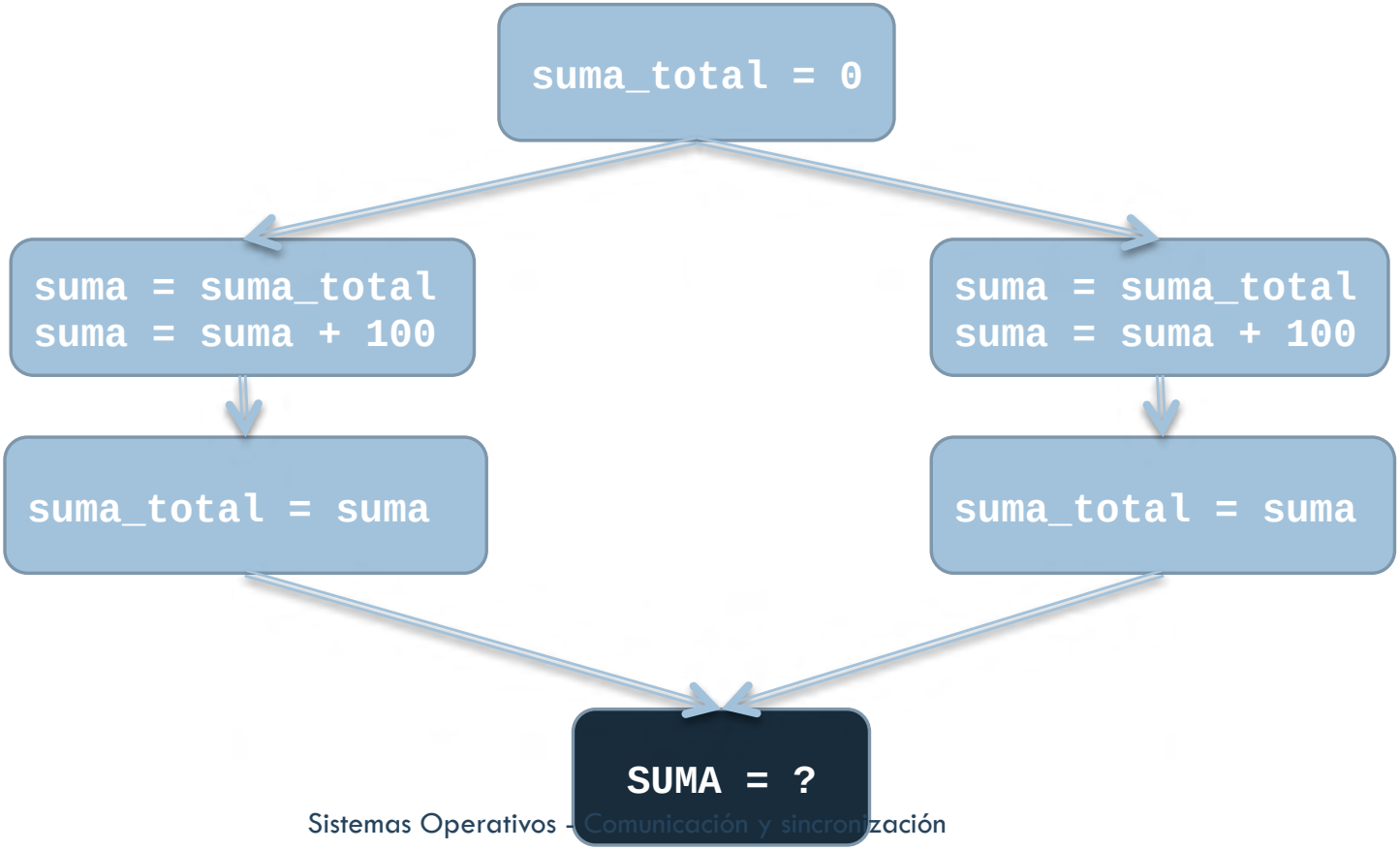
Interacciones entre procesos

- Acceso a recursos compartidos.
  - Procesos que comparten un recurso.
  - Procesos que compiten por un recurso.
  - Ejemplo: Servidor de peticiones en la que distintos procesos escriben en un registro de actividad (log).
- Comunicación.
  - Procesos que intercambian información.
  - Ejemplo: Receptor de peticiones debe pasar información a proceso de tratamiento de petición.
- Sincronización.
  - Un proceso debe esperar a un evento en otro proceso.
  - Ejemplo: Un proceso de presentación debe esperar a que todos los procesos de cálculo terminen.

Contenido

- Concurrencia.
- **Condiciones de carrera.**
- Exclusión mutua y sección crítica.
- Semáforos.
- El problema del productor consumidor.
- El problema de los lectores escritores.

Condiciones de carrera



Condiciones de carrera


```
#include <stdio.h>
#include <pthread.h>

#define NUMTH 10
int suma_total = 0;

void suma() {
  int i,n;
  int suma=suma_total;
  suma = suma + 100;
  n=rand()%5;
  for (i=0;i<n;i++)
   {printf(".");}
  suma_total=suma;
}
```

```
int main() {
  pthread_t th[NUMTH];
  int i;
  for (i=0;i<NUMTH;i++) {
    pthread_create(&th[i],
      NULL,(void*)suma, NULL);
  }

  for (i=0;i<NUMTH;i++) {
    pthread_join(th[i], NULL);
  }

  printf("Suma=%d\n",
    suma_total);
}
```

¿Resultado?

Resultado


```
[jdaniel@tucan ~]$ ./test2
................Suma=200
[jdaniel@tucan ~]$ ./test2
................Suma=600
[jdaniel@tucan ~]$ ./test2
................Suma=500
[jdaniel@tucan ~]$ ./test2
................Suma=300
[jdaniel@tucan ~]$ ./test2
................Suma=600
[jdaniel@tucan ~]$ ./test2
................Suma=600
[jdaniel@tucan ~]$ ./test2
................Suma=500
[jdaniel@tucan ~]$ ./test2
................Suma=600
[jdaniel@tucan ~]$ ./test2
................Suma=600
[jdaniel@tucan ~]$ ./test2
................Suma=600
[jdaniel@tucan ~]$ ./test2
................Suma=500
```

- Cada vez se obtiene un resultado distinto.
- Nunca se obtiene el resultado correcto.
- ¿Qué está pasando?

Secuencias posibles


```
suma_total = 0
```

```
suma1 = suma_total
suma1 = suma1 + 100
```

```
suma_total = suma1
```

```
suma2 = suma_total
suma2 = suma2 + 100
```

```
suma_total = suma
```

```
suma_total = 0
```

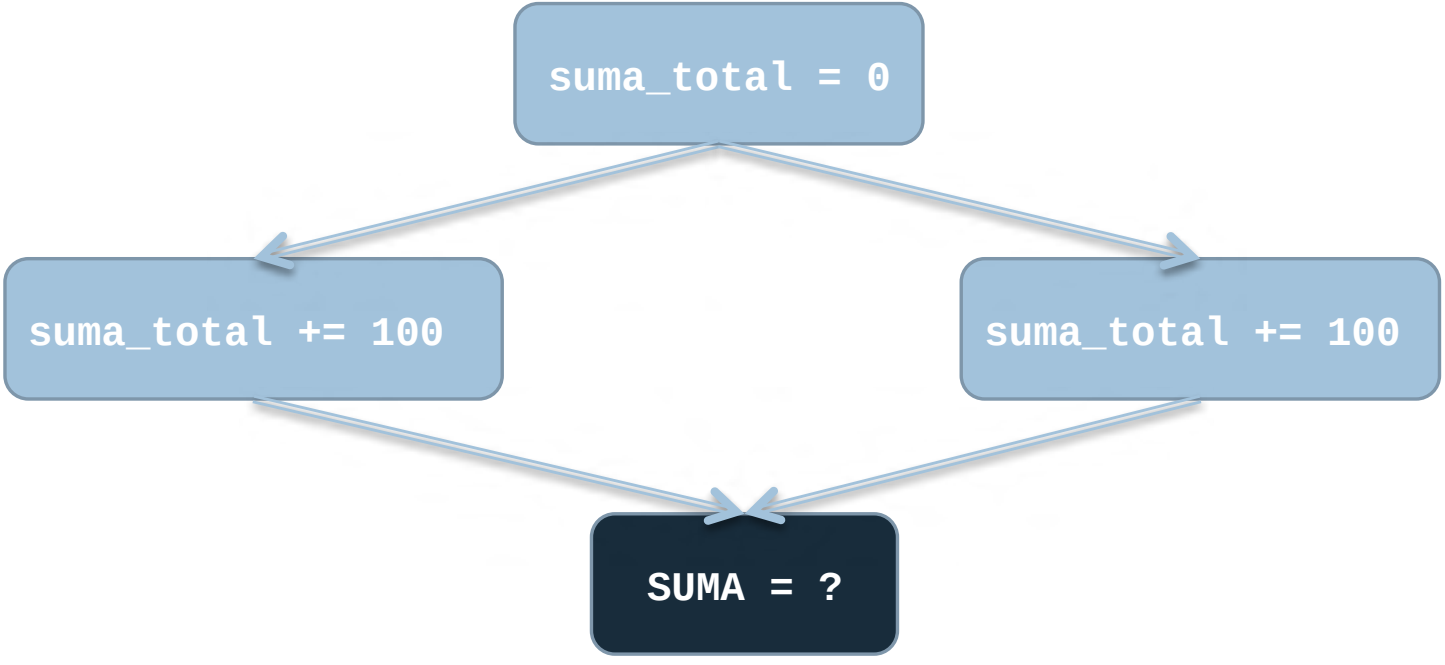
```
suma1 = suma_total
suma1 = suma1 + 100
```

```
suma2 = suma_total
suma2 = suma2 + 100
```

```
suma_total = suma
```

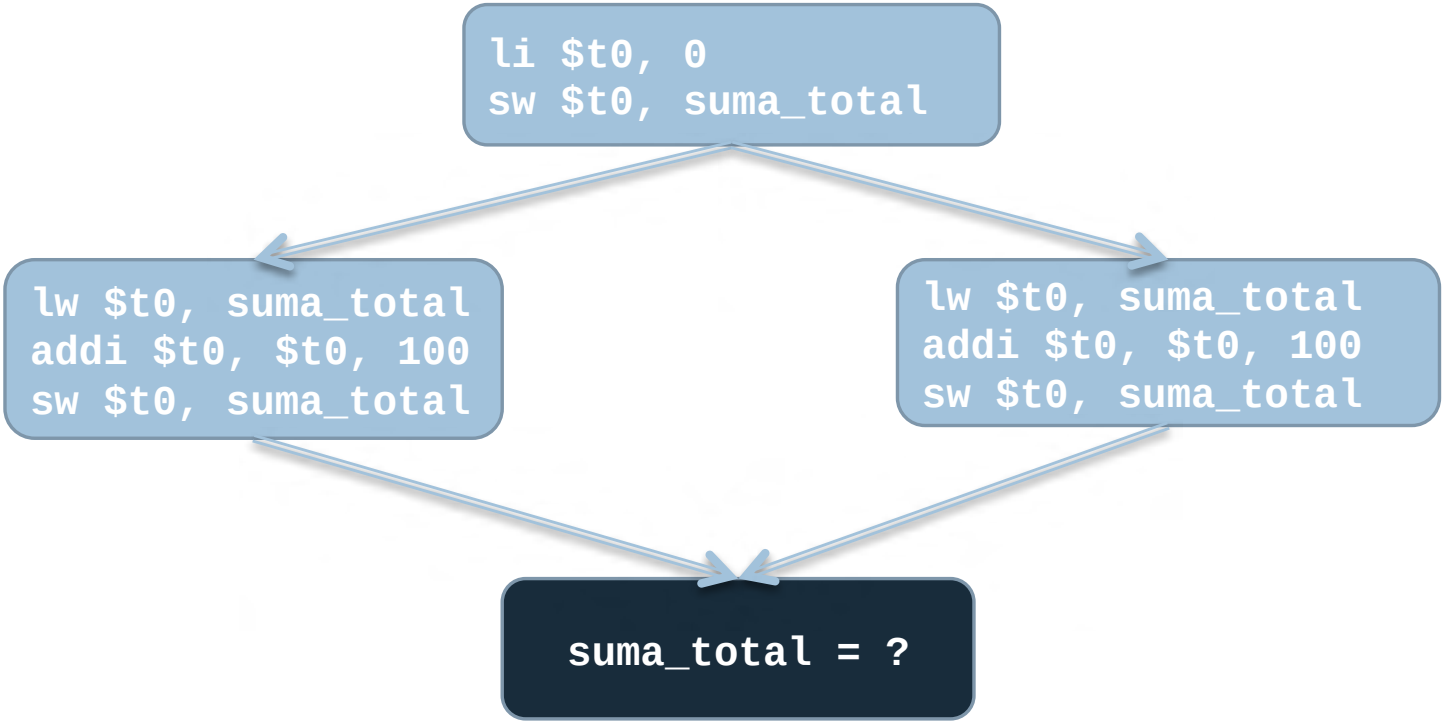
```
suma_total = suma1
```

Otra alternativa



¿Pueden darse condiciones de carrera?

Instrucciones máquina



¿Se puede dar en un multiprocesador?

Condiciones de carrera

- El funcionamiento de un proceso y su resultado debe ser independiente de su velocidad relativa de ejecución con respecto a otros procesos.
  - Es necesario garantizar que el orden de ejecución no afecte al resultado.

- Solución: Conseguir que un conjunto de instrucciones se ejecute de forma atómica.

Exclusión mutua

Cuando alguien usa un recurso compartido nadie más lo puede usar.

Contenido

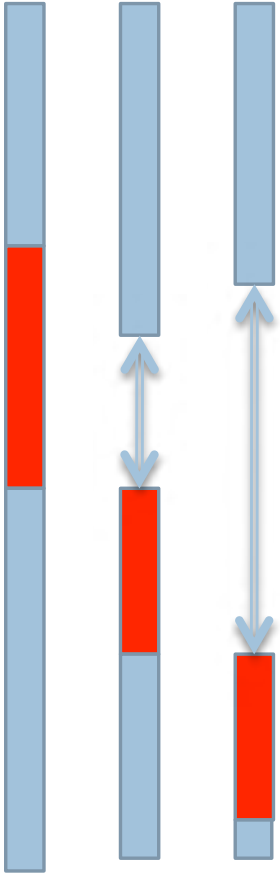
- Concurrencia.
- Condiciones de carrera.
- **Exclusión mutua y sección crítica.**
- Semáforos.
- El problema del productor consumidor.
- El problema de los lectores escritores.

Puede haber concurrencia entre hilos o entre procesos.

Exclusión mutua

- **Sección crítica**: Segmento de código que manipula un recurso y debe ser ejecutado de forma atómica.
- Se asocia a un recurso un mecanismo de gestión de exclusión mutua.
- Solamente un proceso puede estar simultáneamente en la sección crítica de un recurso.

Se organizan para acceder.



Problemas de la sección crítica

- Interbloqueos.
  - Se produce al admitirse exclusión mutua para más de un recurso.
    - El proceso P1 entra en la sección crítica para el recurso A.
    - El proceso P2 entra en la sección crítica para el recurso B.
    - El proceso P1 solicita entrar en la sección crítica para el recurso B (queda a la espera de que P2 la abandone).
    - El proceso P2 solicita entrar en la sección crítica para el recurso A (queda a la espera de que P1 la abandone).

Cuando está en una sección crítica y solicita entrar en otra, pero esa la ha pedido entrar en otra. Por lo que los dos se quedan bloqueados ya que no se liberan.

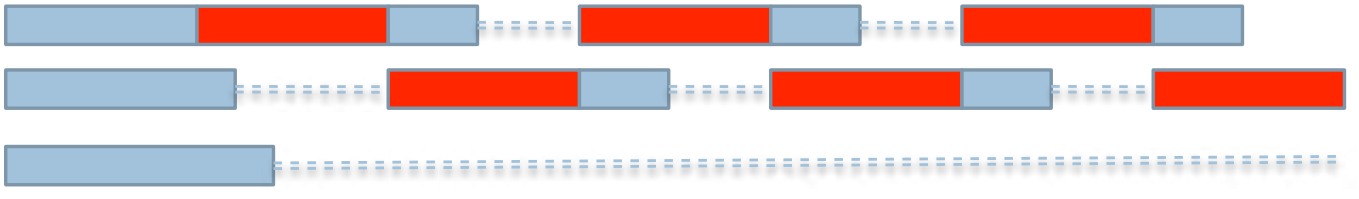
Ninguno puede avanzar

Problemas de la sección crítica

- Inanición.
  - Un proceso queda indefinidamente bloqueado en espera de entrar en una sección crítica. No dejar que entre un proceso, porque llegan otros.
    - El proceso P1 entra en la sección crítica del recurso A.
    - El proceso P2 solicita entrar en la sección crítica del recurso A.
    - El proceso P3 solicita entrar en la sección crítica del recurso A.
    - El proceso P1 abandona la sección crítica del recurso A.
    - El proceso P2 entra en la sección crítica del recurso A.
    - El proceso P1 solicita entrar en la sección crítica del recurso A.
    - El proceso P2 abandona la sección crítica del recurso A.
    - El proceso P1 entra en la sección crítica del recurso A.
    - …

El proceso P3 nunca consigue entrar en la sección crítica del recurso A

Inanición



El proceso P3 nunca llega a conseguir entrar en la sección crítica

Condiciones para la exclusión mutua

- Solamente se permite un proceso puede estar simultáneamente en la sección crítica de un recurso.
- No debe ser posible que un proceso que solicite acceso a una sección crítica sea postergado indefinidamente.
  - Ser justo para que todos puedan entrar.
- Cuando ningún proceso este en una sección crítica, cualquier proceso que solicite su entrada lo hará sin demora.
- No se puede hacer suposiciones sobre la velocidad relativa de los procesos ni el número de procesadores.
- Un proceso permanece en su sección crítica durante un tiempo finito.

Sección crítica: Mecanismo de sincronización

- Cualquier mecanismo que solucione el problema de la sección crítica debe proporcionar sincronización entre procesos.
  - Cada proceso debe solicitar permiso para entrar en la sección crítica
  - Cada proceso debe indicar cuando abandona la sección crítica.

```
Código no crítico
…
<Entrada en sección crítica>
Código de sección crítica
<Salida de sección crítica>
…
Código no crítico
```

Alternativas de implementación

- Desactivar interrupciones.
  - El proceso no sería interrumpido.
  - Solamente sería válido en sistemas monoprocesador.
- Instrucciones máquina.
  - Test and set o swap.
  - Implica espera activa.
  - Son posibles inanición e interbloqueo.
- Otra alternativa: Soporte del sistema operativo.

Solución de Peterson

SOLO para 2 procesos

- Asume que instrucciones LOAD y STORE son atómicas, no interrumpibles.
- Los 2 procesos comparten 2 variables:
  - int turno; Boolean flag[2]
- Turno: indica quien entrará en la sección crítica.
- Flag: indica si un proceso está listo para entrar en la sección crítica.
  - flag[i] = true implica que Pi está listo.

Algorithm for Process P_i

2 processes: Pi and Pj, where j=1-i

- i = 0 => j=1-i=1
- i = 1 => j=1-i=0

```
do {
        flag[i] = TRUE;
        turn = j;
        while (flag[ j ] && turn == j);
                critical section
        flag[ I ] = FALSE;
                remainder section
} while (TRUE);
```

Contenido

- Concurrencia.
- Condiciones de carrera.
- Exclusión mutua y sección crítica.
- **Semáforos.**
- El problema del productor consumidor.
- El problema de los lectores escritores.

Semáforos (Dijkstra)

- Sincronización de procesos mediante un mecanismo de señalización → semáforo.

- Se puede ver un semáforo como una variable entera con tres operaciones asociadas.
  - Iniciación a un valor no negativo.
  - **semWait:** Decrementa el contador del semáforo.
    - Si s<0 → El proceso se bloquea.
  - **semSignal:** Incrementa el valor del semáforo.
    - Si s<=0 → Desbloquea un proceso.

Los consecutivos que entran se ponen en cola.

Operaciones atómicas

Secciones críticas y semáforos

- Un semáforo asociado a la sección crítica de un recurso.
- Semáforo iniciado a 1.
- Entrada en la sección crítica: semWait.
- Salida de la sección crítica: semSignal.

```
Código no crítico
…
semWait(s);
Código de sección crítica
semSignal(s);
…
Código no crítico
```

Se declara con sem_t semaforo;

Inicio sem_init(&semaforo, 0, 1);

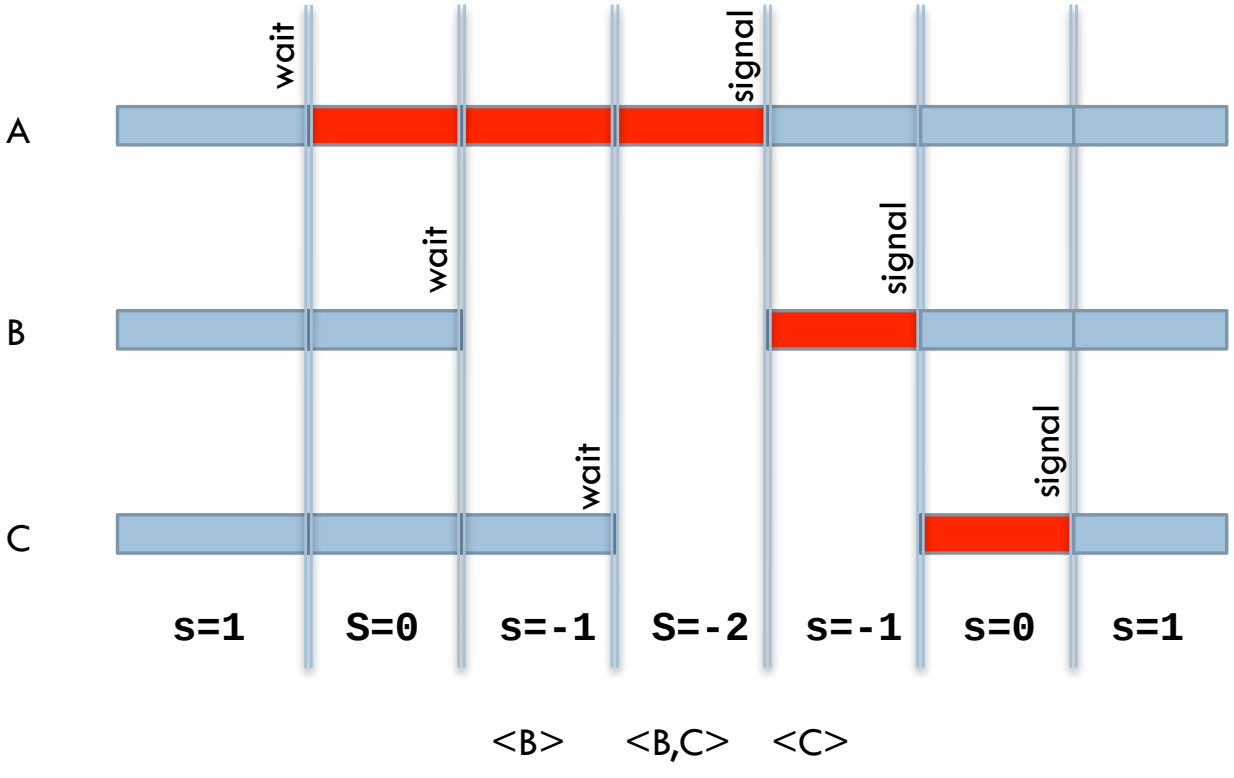
Entrada/ sem_wait(&semaforo);

Código

Salida/Libera sem_post(&semaforo);

fin sem_destroy(&semaforo);

Secciones críticas y semáforos



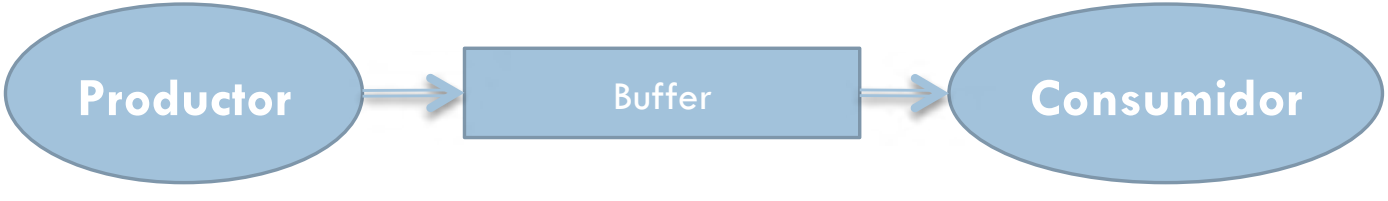
Contenido

- Concurrencia.
- Condiciones de carrera.
- Exclusión mutua y sección crítica.
- Semáforos.
- **El problema del productor consumidor.**
- El problema de los lectores escritores.

El problema del productor-consumidor

Proceso/Thread que pone algo en el buffer y otro lo está leyendo

- Un proceso produce elementos de información. (Productor)
- Un proceso consume elementos de información. (Consumidor)
- Se tiene un espacio de almacenamiento intermedio.



Búfer infinito

Productor

```
for (;;) {
   x= producir();
   v[fin] = x;
   fin++;
}
```

Hay que introducir sincronización

Consumidor

```
for (;;) {
   while (inicio==fin)
      {}
   y=v[inicio];
   inicio++;
   procesar(y);
}
```

Espera activa

Búfer infinito


```
semaforo  s=1
```

Productor

```
for (;;) {
  x= producir();
  semWait(s);
  v[fin] = x;
  fin++;
  semSignal(s);
}
```

Consumidor

```
for (;;) {
  while (inicio==fin)
    {}
  semWait(s);
  y=v[inicio];
  inicio++;
  semSignal(s);
  procesar(y);
}
```

Espera activa

Búfer infinito


```
semaforo  s=1; semaforo n=0;
```

Productor

```
for (;;) {
  x= producir();
  semWait(s);
  v[fin] = x;
  fin++;
  semSignal(s);
  semSignal(n)
}
```

Espera para escribir en la cadena

Cuando termina libera la cinta y también libera la escritura

Consumidor

```
int m;
for (;;) {
  semWait(n);
  semWait(s);
  y=v[inicio];
  inicio++;
  semSignal(s);
}
```

Espera a que acabe de escribir

Espera a que acabe de escribir en la cadena.

Libera la cadena.

Contenido

- Concurrencia.
- Condiciones de carrera.
- Exclusión mutua y sección crítica.
- Semáforos.
- El problema del productor consumidor.
- **El problema de los lectores escritores.**

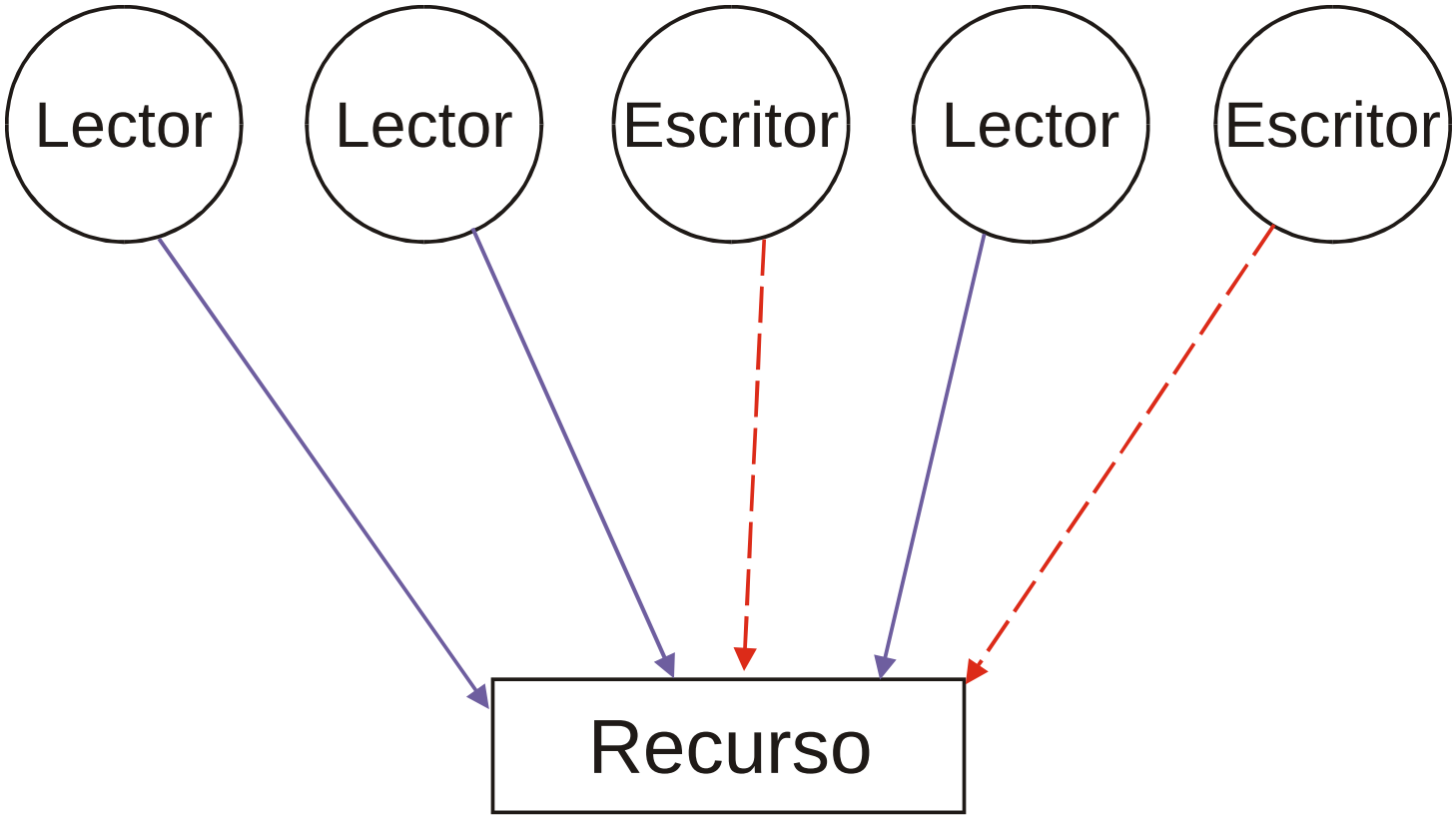
Problema de los lectores-escritores

- Problema que se plantea cuando se tiene un área de almacenamiento compartida.
  - Múltiples procesos leen información.
  - Múltiples procesos escriben información.

Le da prioridad a uno de ellos.

- Condiciones:
  - Cualquier número de lectores pueden leer de la zona de datos concurrentemente.
  - Solamente un escritor puede modificar la información a la vez.
  - Durante una escritura ningún lector puede realizar una consulta.

El problema de los lectores-escritores



Diferencias con otros problemas

- Exclusión mutua:
  - En el caso de la exclusión mutua solamente se permitiría a un proceso acceder a la información.
  - No se permitiría concurrencia entre lectores.
- Productor consumidor:
  - En el productor/consumidor los dos procesos modifican la zona de datos compartida.

- Objetivos de restricciones adicionales:
  - Proporcionar una solución más eficiente.

Alternativas de gestión

- Los lectores tienen prioridad.
  - Si hay algún lector en la sección crítica otros lectores pueden entrar.
  - Un escritor solamente puede entrar en la sección crítica si no hay ningún proceso.
  - Problema: Inanición para escritores.
- Los escritores tienen prioridad.
  - Cuando in escritor desea acceder a la sección crítica no se admite la entrada de nuevos lectores.

Los lectores tienen prioridad


```
int nlect; semaforo lec=1; semaforo = escr=1;
```

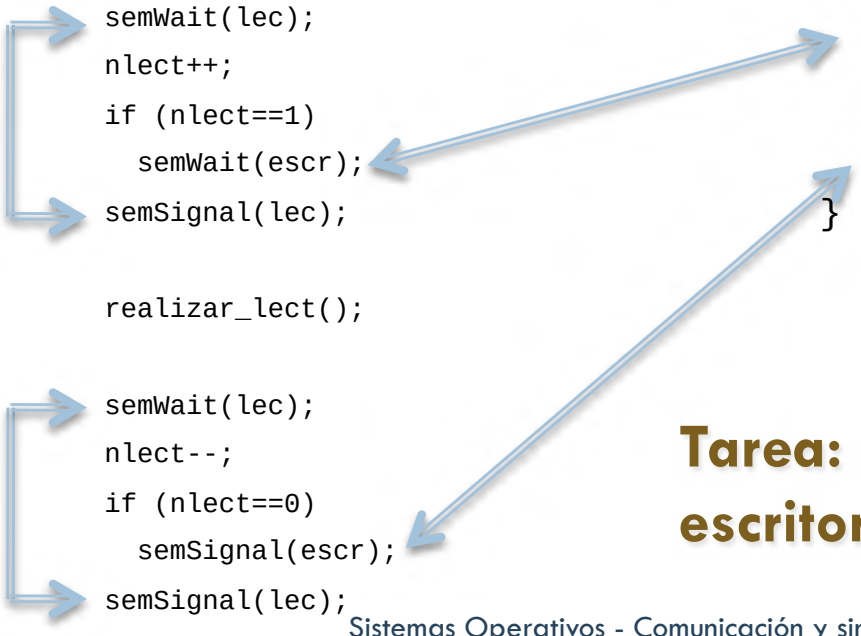
Lector

```
for(;;) {
  semWait(lec);
  nlect++;
  if (nlect==1)
    semWait(escr);
  semSignal(lec);

  realizar_lect();

  semWait(lec);
  nlect--;
  if (nlect==0)
    semSignal(escr);
  semSignal(lec);
}
```

```
for(;;) {
  semWait(escr);
  realizar_escr();
  semSignal(escr);
}
```



Tarea: Diseñar una solución para escritores con prioridad

Lecturas recomendadas

Básica

- Carretero 2007:
  - 6.1. Concurrencia.
  - 6.2. Modelos de comunicación y sincronización.

Complementaria

- Stallings 2005:
  - 5.1 Principios de la concurrencia.
  - 5.2 Exclusón mutua.
  - 5.3 Semáforos.
- Silberschatz 2006:
  - 6.1. Fundamentos.
  - 6.2. El problema de la sección crítica.
  - 6.5. Semáforos.
  - 6.6. Problemas clásicos de sincronización