

SISTEMAS OPERATIVOS: COMUNICACIÓN Y SINCRONIZACIÓN ENTRE PROCESOS

Procesos concurrentes y problemas en la comunicación
y la sincronización

Contenido

2

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

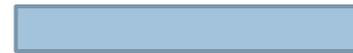
Proceso concurrente

3

- Dos procesos son concurrentes cuando se ejecutan de manera que sus intervalos de ejecución se solapan.



Si hay concurrencia



No hay concurrencia

Tipos de concurrencia

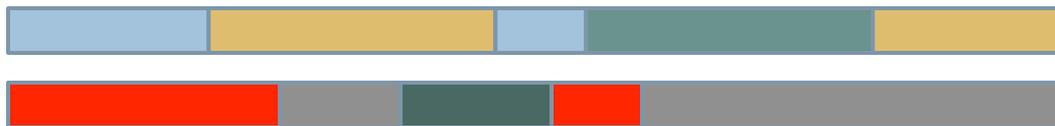
4

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

1 CPU



2 CPUs



Tipos de concurrencia

5

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

4 CPUs



Modelos de programación concurrente

6

- Multiprogramación con un único procesador
 - ▣ El sistema operativo se encarga de repartir el tiempo entre los procesos (planificación expulsiva/no expulsiva).
- 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

7

- 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.

Interacciones entre procesos

9

- 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.

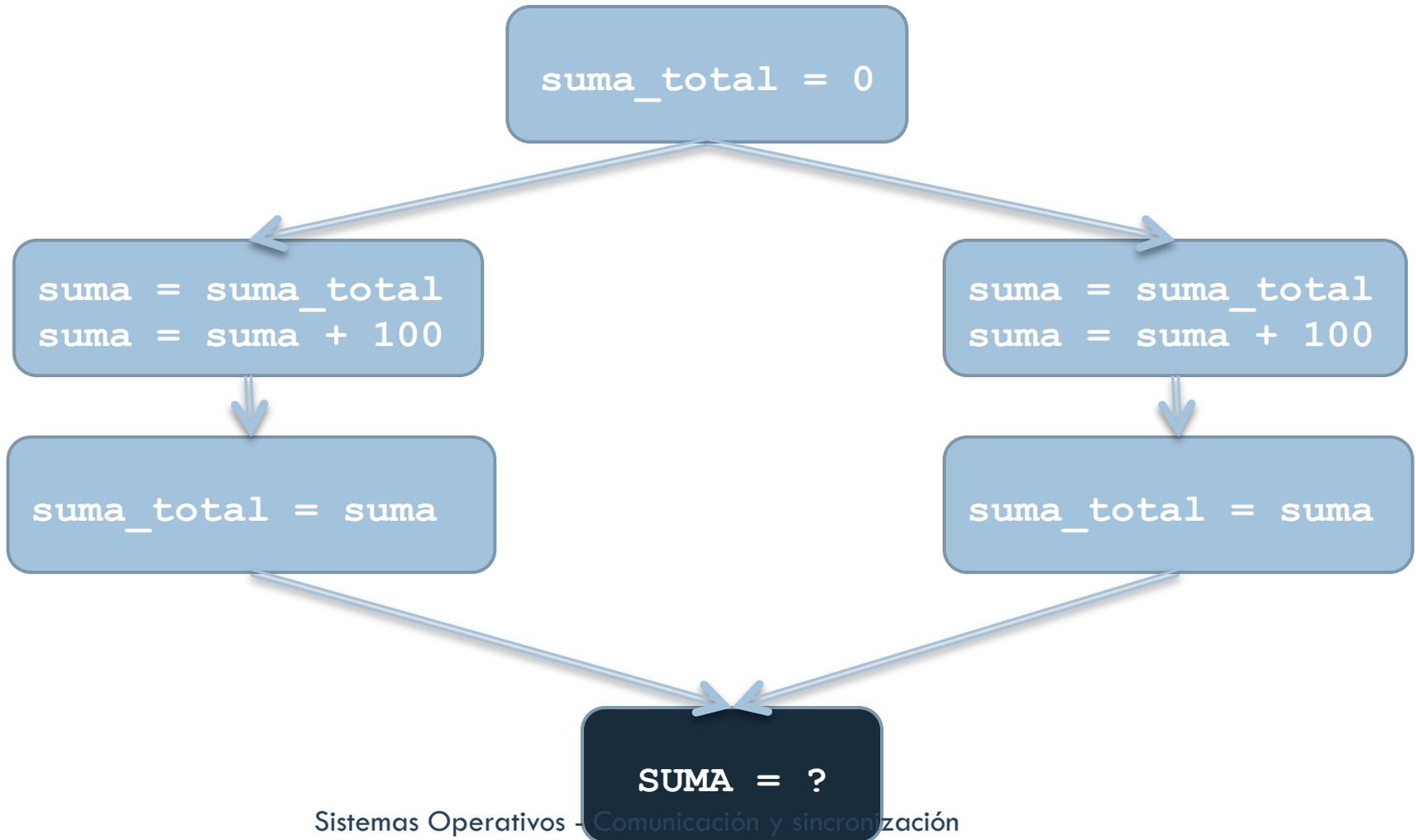
Contenido

10

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

Condiciones de carrera

11



Condiciones de carrera

12

```
#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

13

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

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

Secuencias posibles

14

```
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
```

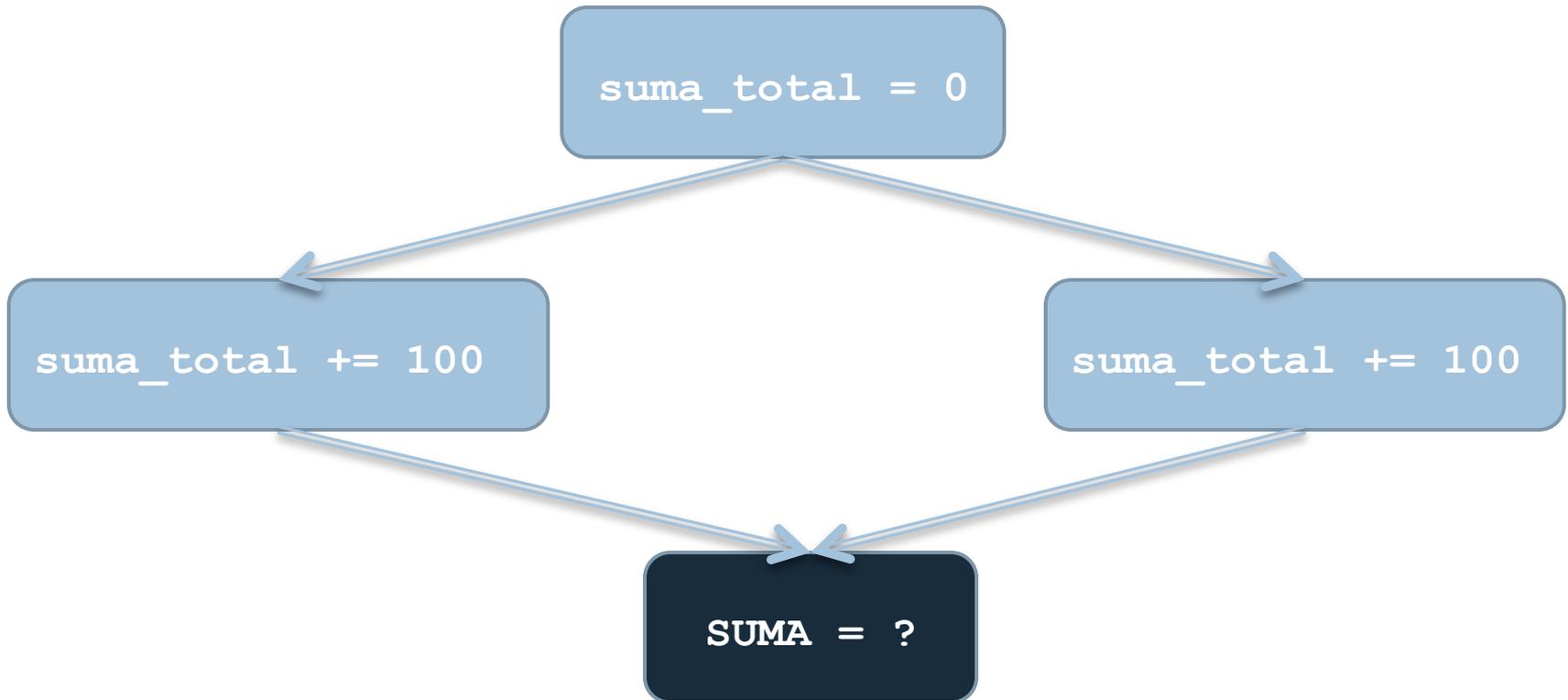
```
suma_total = suma1
```

200

100

Otra alternativa

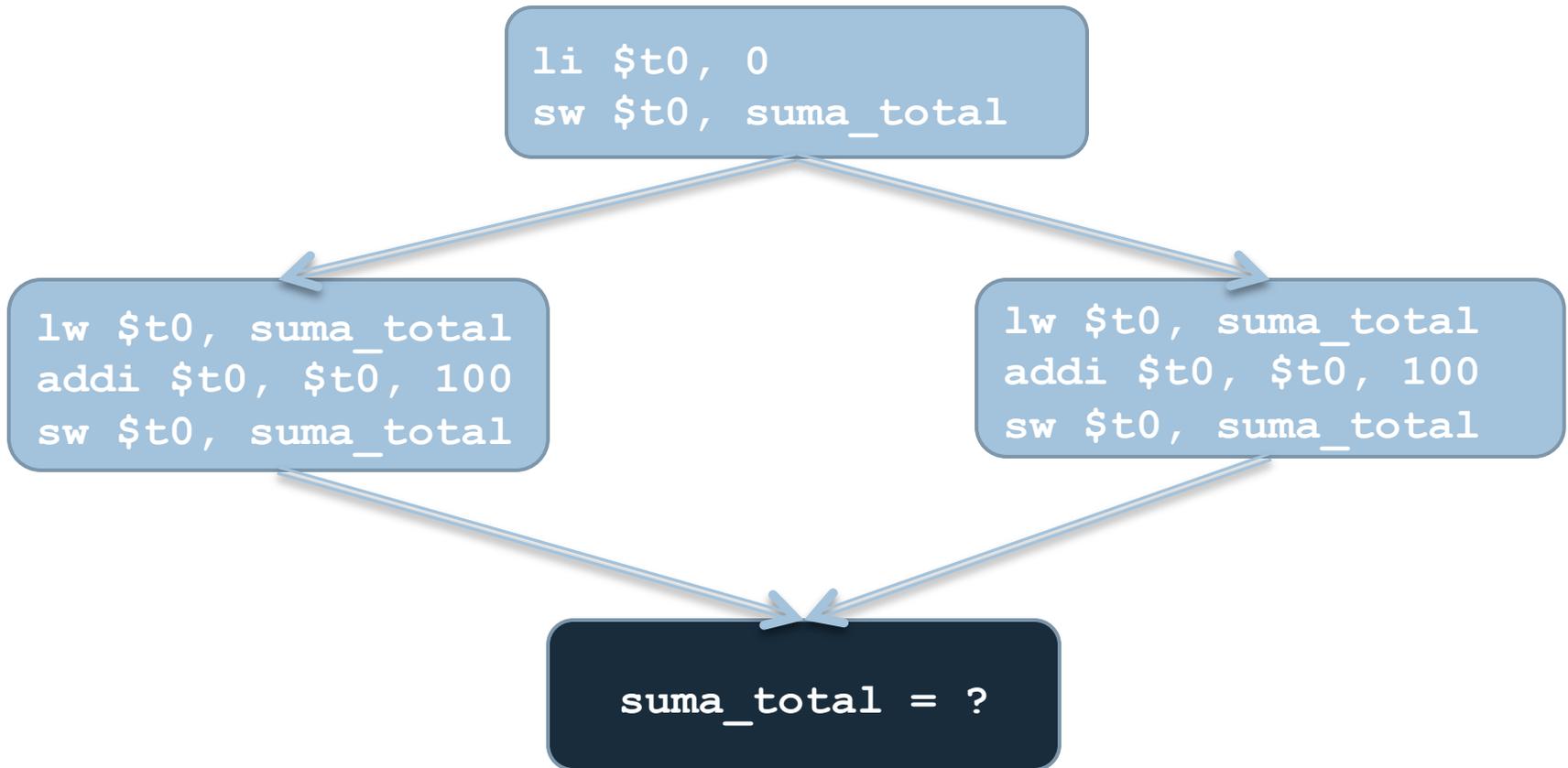
15



¿Pueden darse condiciones de carrera?

Instrucciones máquina

16



¿Se puede dar en un multiprocesador?
Sistemas Operativos - Comunicación y sincronización

Condiciones de carrera

17

- 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

Contenido

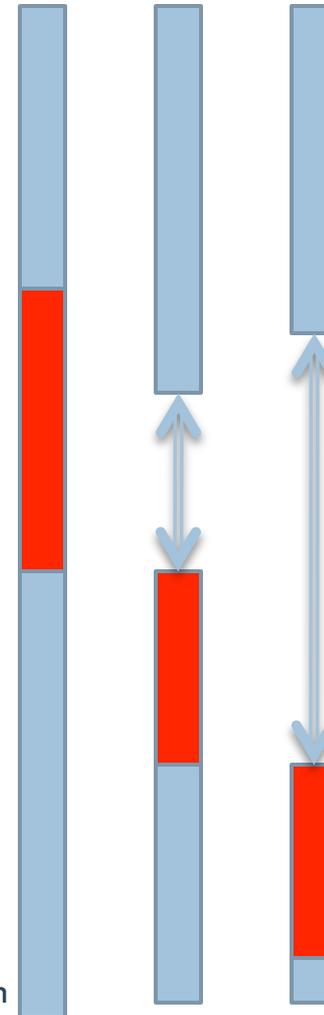
18

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

Exclusión mutua

19

- **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**.



Problemas de la sección crítica

20

□ 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).

Ninguno puede avanzar

Problemas de la sección crítica

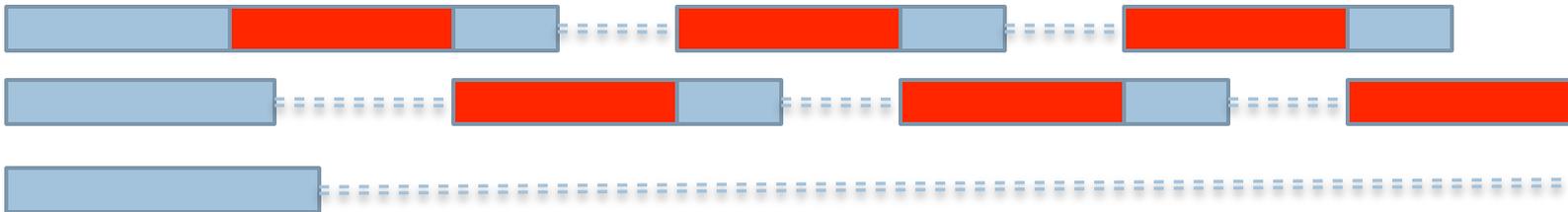
21

- Inanición.
 - ▣ Un proceso queda indefinidamente bloqueado en espera de entrar en una sección crítica.
 - 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

22



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.
- ❑ 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

24

- 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

25

- 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 P_i está listo.

Algorithm for Process P_i

2 processes: P_i and P_j , where $j=1-i$

- $i = 0 \Rightarrow j=1-i=1$
- $i = 1 \Rightarrow j=1-i=0$

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

Contenido

28

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

Semáforos (Dijkstra)

29

- 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 $s \leq 0$ → Desbloquea un proceso.

Operaciones atómicas

Sistemas Operativos - Comunicación y sincronización

Secciones críticas y semáforos

30

- Un semáforo asociado a la sección crítica de un recurso.

Código no crítico

...

```
semWait(s);
```

Código de sección crítica

```
semSignal(s);
```

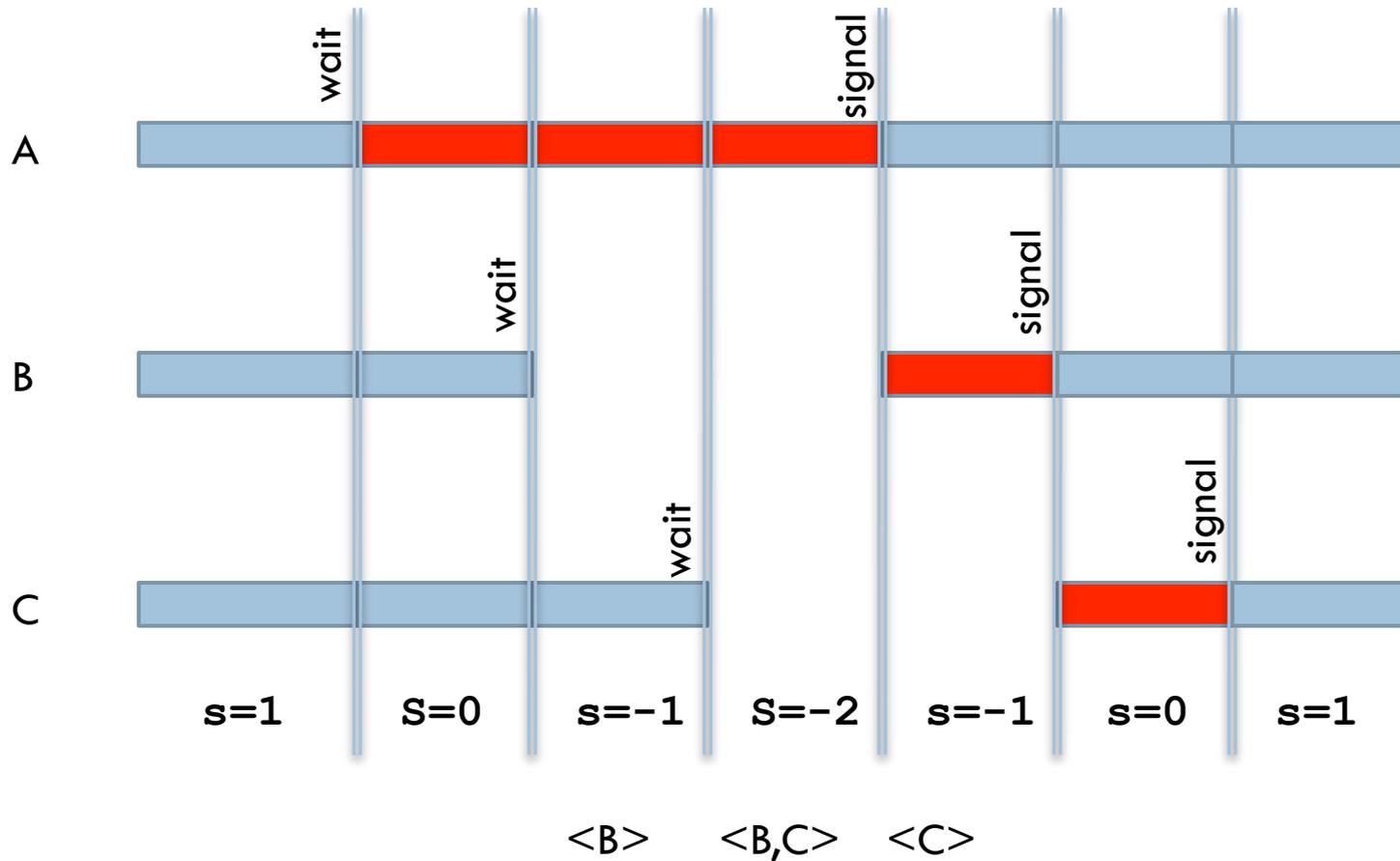
...

Código no crítico

- Semáforo iniciado a 1.
- Entrada en la sección crítica: `semWait`.
- Salida de la sección crítica: `semSignal`.

Secciones críticas y semáforos

31



Contenido

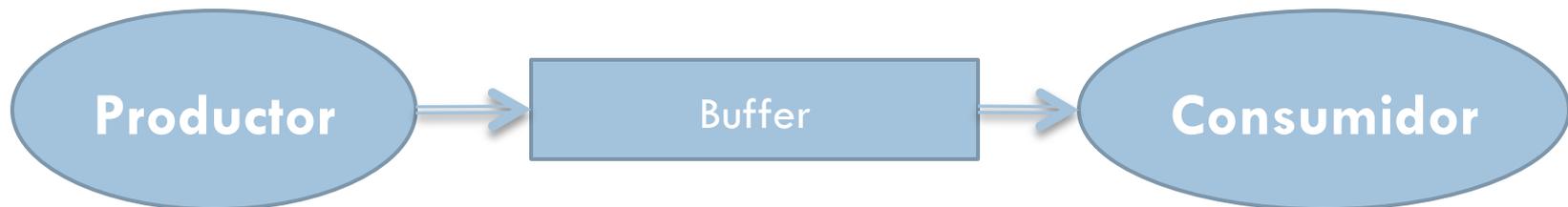
32

- 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

33

- Un proceso produce elementos de información.
- Un proceso consume elementos de información.
- Se tiene un espacio de almacenamiento intermedio.



Búfer infinito

34

Productor

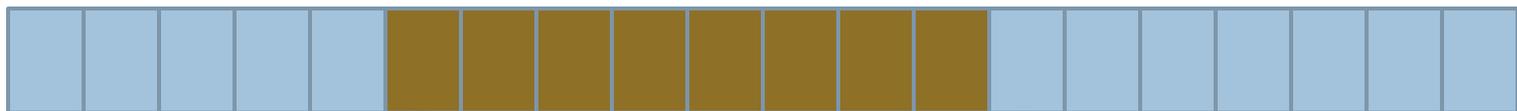
```
for (;;) {  
    x= producir();  
    v[fin] = x;  
    fin++;  
}
```

Hay que introducir sincronización

Consumidor

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

Espera activa



Búfer infinito

35

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

36

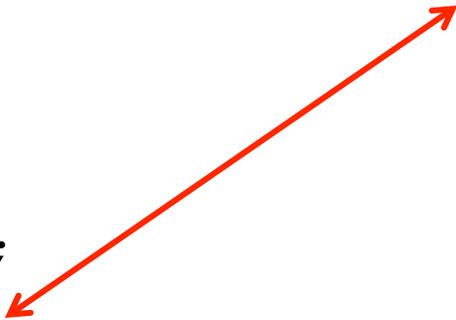
```
semaforo s=1; semaforo n=0;
```

Productor

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

Consumidor

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



Contenido

37

- 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-escriutores

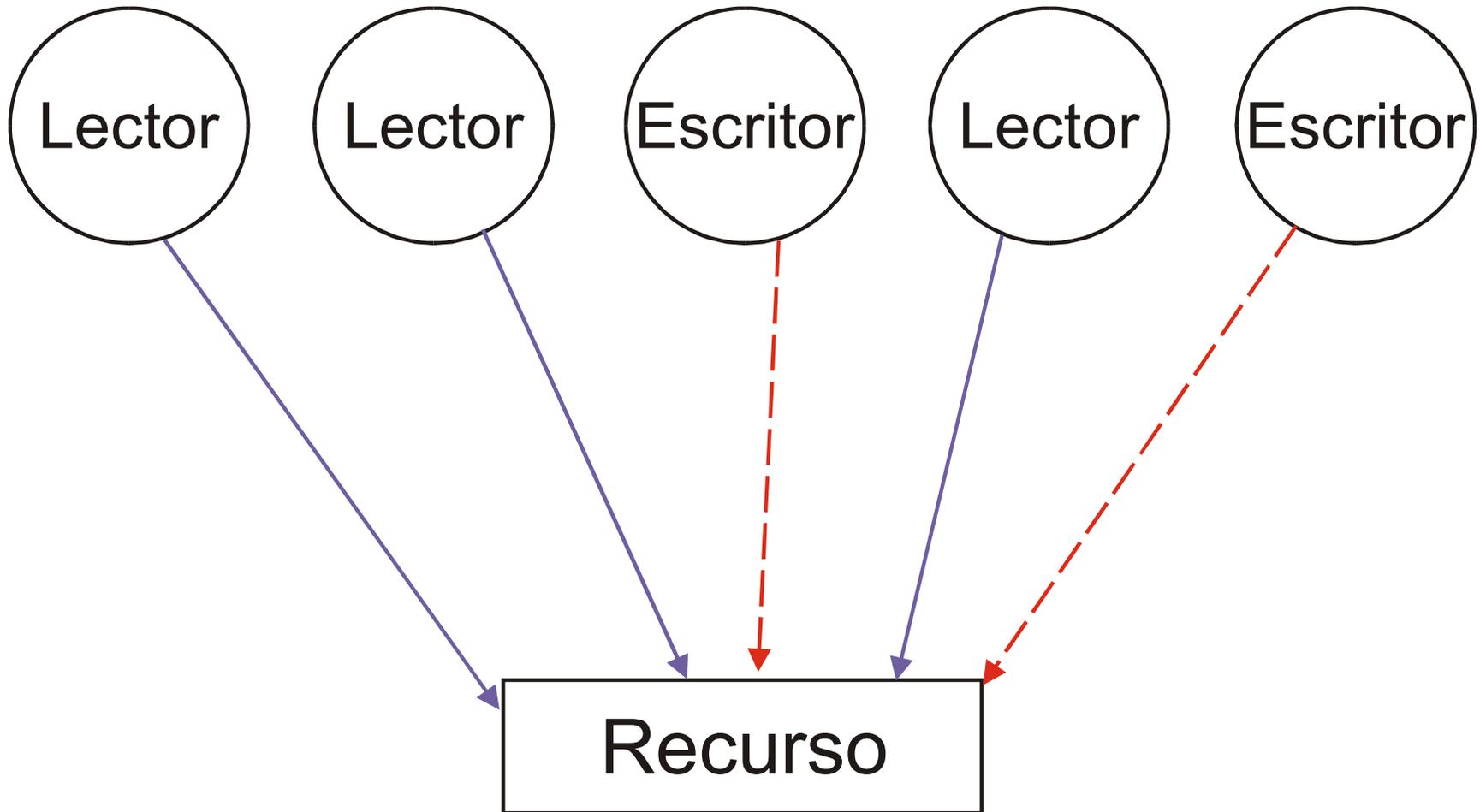
38

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

- 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

39



Diferencias con otros problemas

40

- 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

41

- 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 un escritor desea acceder a la sección crítica no se admite la entrada de nuevos lectores.

Los lectores tienen prioridad

42

```
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

43

Básica

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

Complementaria

- Stallings 2005:
 - 5.1 Principios de la concurrencia.
 - 5.2 Exclusión 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