

Concurrencia

Libro Silberchatz: Cap 5, 6 y 7

Vocabulario

Mecanismo: dado un problema, un mecanismo es cualquier “elemento” externo, artificio, que ayuda a resolver el problema.

Multiplexar comunicaciones: Intermitencia rotativa rápida de **comunicaciones** que da la impresión de que las comunicaciones se desarrollan en paralelo: <https://es.wikipedia.org/wiki/Multiplexaci%C3%B3n>

Multiplexación en informática: hay varios tipos

Comunicaciones: Varios dispositivos (discos, memoria, etc.) multiplexación el bus.

Al usar el cpu: varios procesos multiplexan el cpu

<https://es.wikipedia.org/wiki/Multiplexaci%C3%B3n#:~:text=Tambi%C3%A9n%20en%20inform%C3%A1tica%2C%20se%20denomina,este%20archivo%2C%20esto%20para%20respaldar>

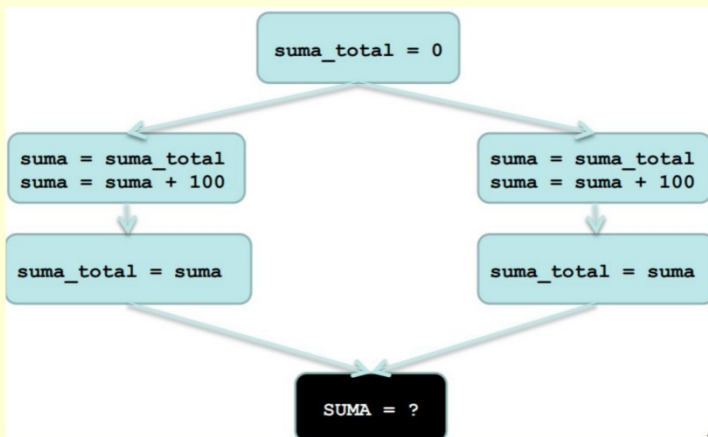
Aplicaciones

- 1) Acceso de procesos **listos** (en la RAM) al CPU, que rotan en quantum de tiempo, y son controlados por el Planificador de procesos.
- 2) Acceso de varios procesos, por turnos, a una sola unidad de almacenamiento.
- 3) Un archivo contenedor de dos o más archivos: video: sonido + imágenes
- 4) Escribir y respaldar, simultáneamente, un archivo en dos dispositivos.

Resumen de implantación de semáforos en el archivo 01Concurrencia.pdf

1) Pag 11: Planteamiento de un problema

Condiciones de carrera

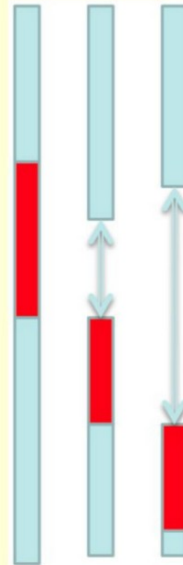


Condición de carrera: Es un posible error de cálculos que se puede producir en una concurrencia de procesos a causa de fallas en la sincronización del uso de memoria.

2) Pag. 19:

Solución con 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 ejecutar la sección crítica.

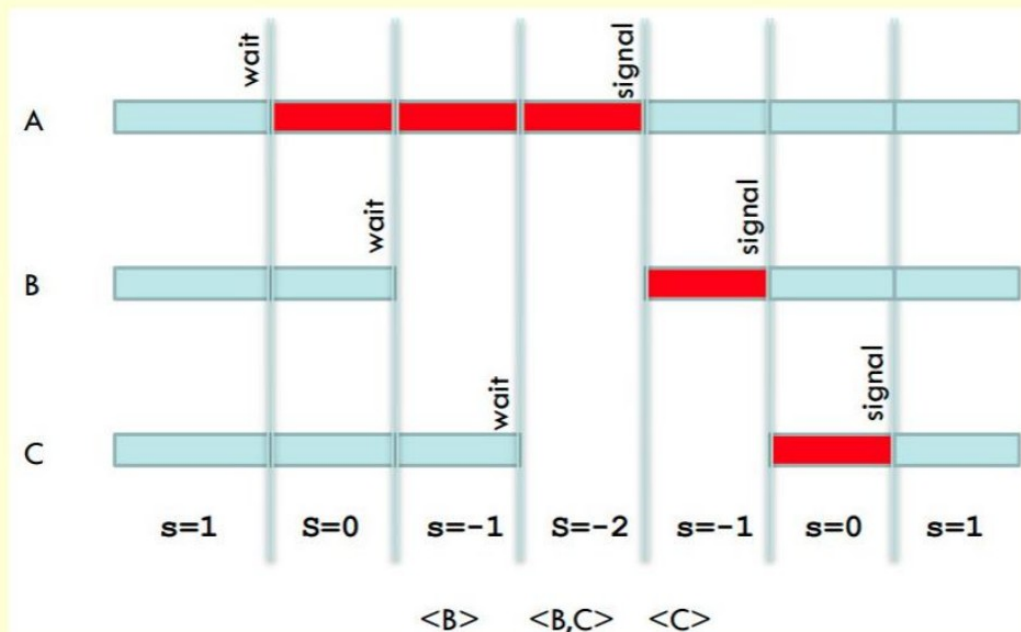


3) Pag. 30:

Código de proceso que implementa el **Mecanismo**

- | | |
|---|---|
| • Semáforo asociado a la sección crítica de un recurso. | <code>sem_t s;</code> |
| • (1) Semáforo iniciado a 1. | Código no crítico
<code>s_init(&s, 1);</code>
Código no crítico |
| • (2) Entrada en la sección crítica: | <code>s_wait(&s);</code>
Código atómico de sección crítica |
| • (3) Salida de la sección crítica: | <code>s_signal(&s);</code>
Código no crítico |

Sección crítica y aplicación de semáforo: procesos A, B y C



Significado del valor **s** del semáforos

Si $s > 0$: La sección crítica puede alojar a s procesos

Si $s = 0$: La sección crítica está llena y no hay procesos en cola de espera

Si $s < 0$: La sección crítica está llena y hay $|s|$ procesos en cola de espera

Ejemplo: Programa

```
#include <semaphore.h>
#include <thread>
#include <iostream>
#define NUMTH 10
using namespace std;
sem_t sem; // 0) Declaración de semáforo
int suma_total = 0; // 1) memoria compartida (variable global)
void suma() {
    int i;
    sem_wait(&sem); // 2) Inicio de sección crítica
    int suma=suma_total;
    suma = suma + 100;
    for (i=0;i<10000;i++); // consumidor de tiempo
    suma_total=suma;
    sem_post(&sem); // 2) Fin de sección crítica
}
```

```
}
int main() {
    thread th[NUMTH];
    int i;
    sem_init(&sem, 0, 1); // inicia el semáforo (&s,
compartidoEntreProcesos=no, inicio=1)
    for (i=0;i<NUMTH;i++) th[i] = thread(suma); // crea 10 threads
    for (i=0;i<NUMTH;i++) th[i].join(); // espera a los 10 threads
    printf("Suma Total=%d\n", suma_total);
}
```