ITESO

Universidad Jesuita
de Guadalajara

Axel Roberto Orozco Hernández
NAVARRO QUINN, LUIS ROBERTO

Fundamentos de Sistemas Operativos
Profesor: Leonardo Sandoval. Gonzalez

Actividad 13

1.- Considerando las definiciones de semáforos tanto enteros como binarios, uno de los requisitos para que estos funcionen es que la ejecución de las llamadas waitsem y signalsem se haga de manera atómica. Muestra con un ejemplo por qué pueden fallar si no se asegura la atomicidad en la ejecución de ambas funciones. Por ejemplo: dos procesos/hilos ejecutan waitsem sin garantizar la atomicidad de waitsem; o un proceso ejecuta waitsem y otro signalsem sin garantizar la atomicidad de waitsem y seignalsem; o dos procesos ejecutan signalsem sin garantizar la atomicidad la función signalsem.

Los semaforos funcionan correctamente solo si las operaciones **waitsem** y **signalsem** se ejecutan de forma atomica. Sino surgen condiciones de carrera:

Dos procesos ejecutando waitsem al mismo tiempo sin atomicidad:

- Supongamos que dos procesos intentan ejecutar **waitsem** sobre el mismo semaforo. Simultaneamente. Sin atomicidad, ambos procesos prodian leer el valor de **s -> count** como mayor que 0. Ambos procederian a decrementar el contador, lo que causaria que dos procesos accedan a la seccion critica, rompiendo la conclusion mutua que los semaforos deberian proporcionar.

Dos procesos ejecutando signalsem simultaneamente:

- Si dos procesos **signalsem** sin atomicidad, ambos prodrian incrementar el contador sin detectar correctamente si hay procesos bloqueados que deberian ser desbloqueados, llevando a inconsistencias y procesos que permanecen innecesariamente.

2.- Compara las definiciones de semáforos enteros donde los semáforos enteros pueden tomar valores negativos y aquellos donde no toman valores negativos, ¿hay alguna diferencia en el efecto de las dos definiciones cuando se utilizan para sincronizar procesos? Es decir, ¿es posible sustituir una definición por la otra sin alterar el significado del programa?

- **Semaforos con valores negativos:** En esta definicion, el contador del semaforo pueden tomar valores negativos. Un valor negativo indica la cantidad de procesos que estan bloqueados esperando que el semaforo se libere. Esto permite flexibilidad por que refleja la cantidad exacta de procesos esperando.

- **Semaforos sin valores negativos:** El contador nunca toma valores negativos. Cuando el semaforo llega a 0, los procesos que intentan hacer **wait** deben bloquearse hasta que el semaforo sea liberado por otro proceso.

- **Diferencia en sincronizacion:** En general un semaforo en negativo hacen un seguimiento mas explicito de los procesos bloqueados, lo que puede ser util en situaciones donde se requiere conocer cuantos procesos estan esperando. Aun que, ambos enfoques son equivalentes en cuanto a la funcionabilidad de sincronizacionn y pueden sustituirse sin alterar el significado del programa si solo nos interesa la exclusion mutua o la sincronizacion.

3.-Considere un programa concurrente con dos procesos P y Q, definidos a continuación. la impresión en pantalla de A, B, C, D y E son sentencias arbitrarias atómicas (indivisibles). Supóngase que el programa principal ejecuta concurrentemente los procesos P y Q. Comprueba tu solución empleando semáforos en Linux.

A. Cuántas combinaciones posibles hay en la intercalación de las sentencias A, B, C, D y E.
B. Usando semáforos sincronice P y Q de manera que se asegure que las sentencias A y B se ejecutan antes que D.
C. Usando semáforos haga que el proceso P ejecute siempre las sentencia B inmediatamente después que A sin que cualquier sentencia de Q se ejecute intercalada entre ellas.

1. A, B, C, D, E
2. D, E, A, B, C
3. A, D ,B, C, E
4. A, B, D, C, E
5. D, A, E, B, C
6. D, A, B, E, C

Sincronizacion para A y B juntas, sin intercalaciones de Q:

```
Void *P(void *arg) {
        Print(“A”);
        Print(“B”);
        Sem_post(&sem); // Aquí se libera despues de A y B
        Print(“C”);
        Return NULL;
}
```

¿Qué aprendio?
Comprendimos la importancia de la atomicidad en operación sobre semaforos. Le falta de esta propiedad puede causar condiciones de carrera y comportamientos impredecibles.
Aprendimos que usar semaforos para controlar la ejecucion y sincronizacion entre procesos, asegurando que ciertas instrucciones se ejecutan en el orden deseado.