SEMÁFOROS

Semáforos

Un semáforo es un número entero positivo que el kernel mantiene, cuyo valor se limita a ser mayor o igual a 0.

- Se utilizan cuando dos o más hilos (o procesos) quieren acceder a un mismo recurso compartido.
- Es una forma de sincronización para acceder a memoria compartida o a un recurso compartido.
- Un semáforo da acceso al recurso compartido a un solo proceso o hilo por vez.

Operaciones:

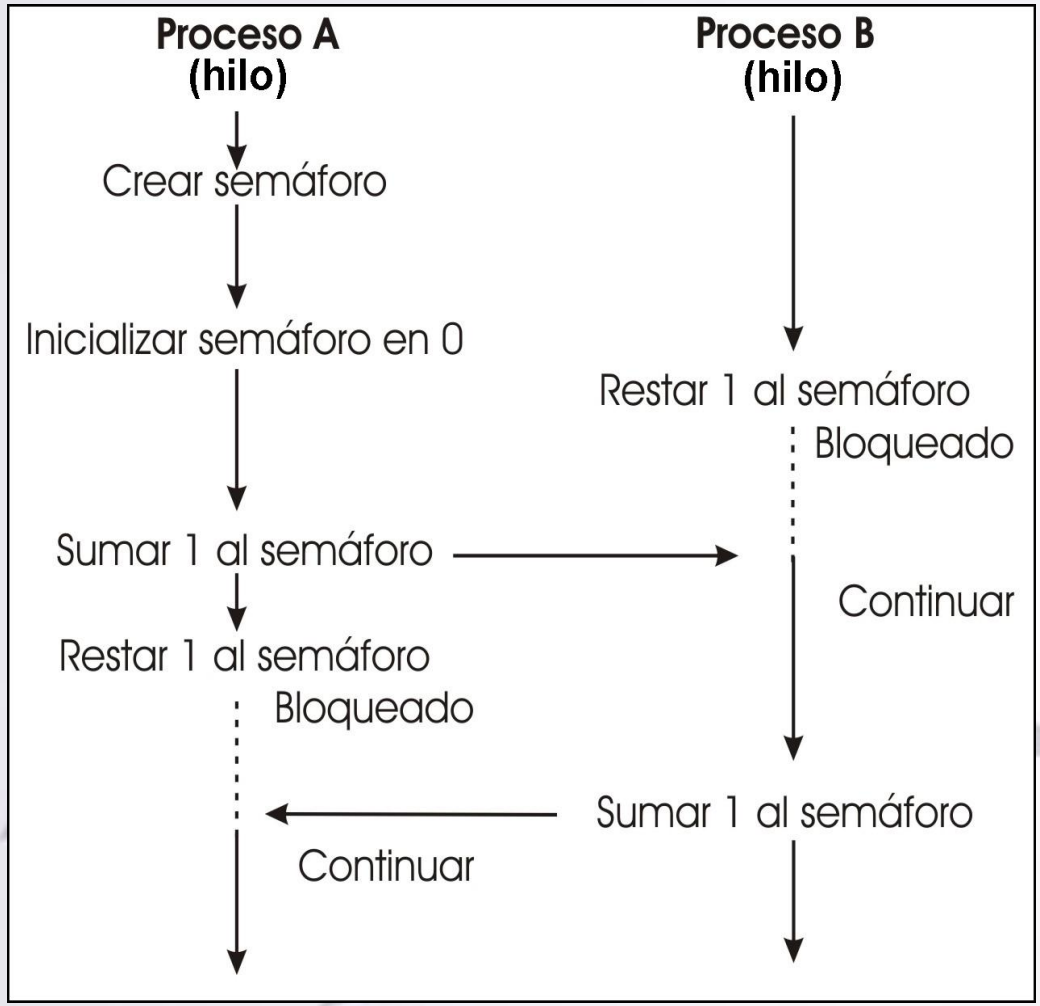
- `init`: fijar el valor del semáforo en un número entero positivo o cero.
- `wait`: Si el valor del semáforo es mayor que cero, le resta 1 al valor actual y continúa. Si el valor del semáforo es cero espera hasta que el valor del semáforo sea mayor que cero y pueda restarle 1 y continuar.
- `post`: sumar 1 al valor actual del semáforo.

La operación `wait` disminuye en 1 el valor del semáforo. El kernel bloquea el hilo (o proceso) que la ejecuta si el valor del semáforo es igual 0.

El kernel impide que el valor del semáforo sea menor a 0.

La operación `post` le suma 1 al semáforo. Puede producir el desbloqueo de otros hilos (o procesos) a la espera de que el valor del semáforo sea mayor que 0.

Semáforos



Se crea un semáforo y se inicializa en 0.
Cuando se intenta restar 1 al valor del semáforo, ese proceso (o hilo) se bloquea, hasta que otro proceso (o hilo) le sume uno y se pueda realizar la resta.

Semáforos

Clasificación:

Semáforos sin nombre: Este tipo de semáforo no tiene un nombre, sino que reside en un lugar acordado en la memoria. Semáforos sin nombre se pueden compartir entre procesos o entre un grupo de hilos. Cuando se comparten entre los procesos, el semáforo debe residir en una región de la memoria compartida. Cuando se comparten entre los hilos, el semáforo puede residir en un área de memoria compartida por los hilos.

Semáforos con nombre: Este tipo de semáforos tiene un nombre. Al llamar a sem_open() con el mismo nombre, los **procesos no relacionados** pueden acceder al mismo semáforo.

SEMÁFOROS SIN NOMBRE

Semáforos sin nombre

- Semáforos sin nombre se almacenan en la memoria asignada por la aplicación.
-
- El uso de un semáforo sin nombre nos permite evitar el trabajo de crear un nombre para un semáforo.

- Un semáforo que se comparte entre los hilos no necesita un nombre.

- El semáforo es puesto a disposición de los procesos relacionados o hilos que lo utilizan colocándolo en un área de la memoria que comparten.

- Un semáforo que está siendo compartido entre los procesos relacionados, no necesita un nombre. Si un proceso padre crea un semáforo sin nombre antes de hacer un fork(), un hijo heredará automáticamente la asignación.

Inicialización de un semáforo sin nombre

El `sem_init()` inicializa el semáforo sin nombre

```
#include <semaphore.h>
sem_t sem;
int sem_init(&sem, int pshared, int value);
```

Devuelve 0 en caso de éxito, o –1 en caso de error.

El argumento `pshared` indica si el semáforo es para ser compartido entre hilos o entre procesos. Si pshared es 0, el semáforo es para ser compartido entre los hilos de un proceso.

Si `pshared` es distinto de cero, entonces el semáforo es para ser compartido entre los procesos. En este caso, `sem` debe ser la dirección de una ubicación en una región de memoria compartida. El semáforo persiste tanto tiempo como la memoria compartida en el que reside.

El argumento `value` es el valor inicial del semáforo

Esperando un semáforo

La función `sem_wait()` decrementa en 1 el valor del semáforo referido por sem.

```
#include <semaphore.h>
sem_t sem;
int sem_wait(&sem);
```

Devuelve 0 si tiene éxito, o –1 en caso de error

Si el semáforo actualmente tiene un valor mayor que 0, `sem_wait()` vuelve inmediatamente. Si el valor actual del semáforo es 0, `sem_wait()` se bloquea hasta que el valor del semáforo se eleva por encima de 0, en ese momento, el semáforo se decrementa y `sem_wait()` se desbloquea.

Esperando un semáforo

La función `sem_trywait()` es una versión sin bloqueo de `sem_wait()`.

```
#include <semaphore.h>
sem_t sem;
int sem_trywait(&sem);
```

Devuelve 0 si tiene éxito, o –1 en caso de error

Si la operación de decremento no se puede realizar de inmediato, `sem_trywait()` falla con el error EAGAIN.

Incrementar un semáforo

La función `sem_post()` incrementa en 1 el valor del semáforo referido por sem.

```
#include <semaphore.h>
sem_t sem;
int sem_post(&sem);
```

Devuelve 0 si tiene éxito, o –1 en caso de error

Si el valor del semáforo era 0 antes de la llamada `sem_post()`, y algún otro proceso (o hilo) está bloqueado en espera para decrementar el semáforo, entonces ese proceso se despierta, y su llamada `sem_wait()` procederá a decrementar el semáforo. Si hay varios procesos (o hilos) bloqueados en `sem_wait()`, entonces, el planificador determina qué proceso despertará y permitirá incrementar el semáforo.

Recuperación del valor actual de un semáforo

La función `sem_getvalue()` devuelve el valor actual del semáforo `sem` en la variable `sval`.

```
#include <semaphore.h>
sem_t sem;
int sem_getvalue(&sem, int *sval);
```

Devuelve 0 si tiene éxito, o –1 en caso de error

Destruir un semáforo sin nombre

La función `sem_destroy()` destruye el semáforo `sem`, que debe ser un semáforo sin nombre y que se ha inicializado previamente utilizando `sem_init()`.
Es seguro destruir un semáforo sólo si no hay procesos o hilos que están esperando.

```
#include <semaphore.h>
sem_t sem;
int sem_destroy(&sem);
```

Devuelve 0 en caso de éxito, o –1 en caso de error.

Si el semáforo se encuentra en una región de memoria compartida POSIX, entonces debe ser destruido después de todos los procesos han dejado de utilizar el semáforo

Bibliografía

Kerrisk, Michael. *The linux programming Interface.* 2011. **Capítulo 53.**