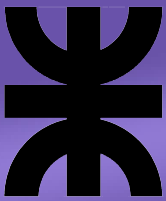


SEMÁFOROS

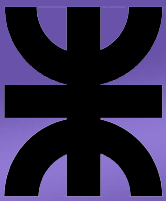


Semáforos

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



Semáforos

Semáforos

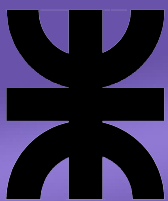
Operaciones:

- init: fijar el valor del semáforo en un número entero positivo o cero.
- wait: esperar si el valor del semáforo es igual a 0, si no restar 1 al valor actual del semáforo y continuar.
- post: sumar 1 al valor actual del semáforo.

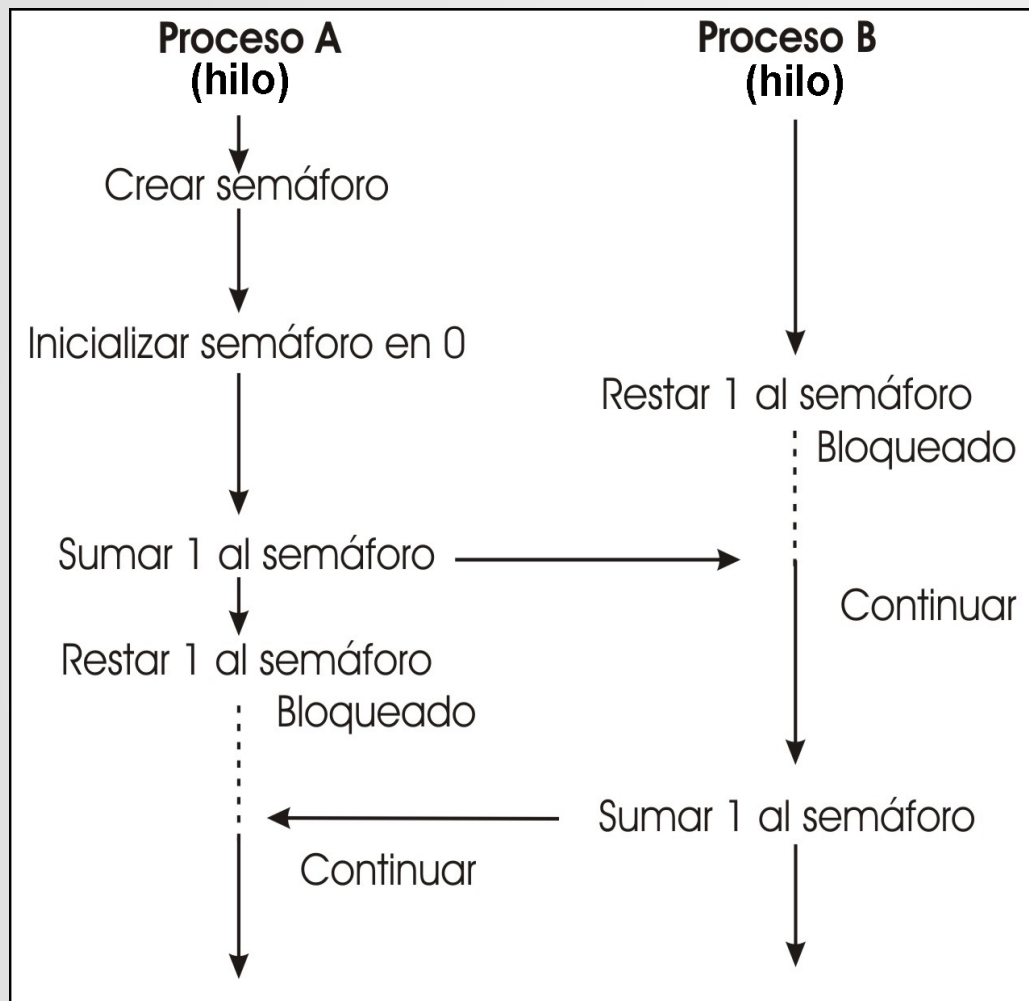
La operación wait disminuye en 1 el valor del semáforo. El kernel bloquea al proceso (hilo) que la ejecuta si el valor del semáforo es igual 0.

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

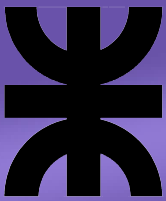
La operación post le suma 1 al semáforo. Puede producir el desbloqueo de otros procesos (hilos) a la espera de que el valor del semáforo sea mayor que 0.



Semáforos



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



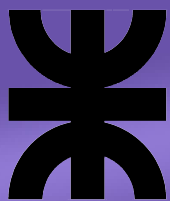
Semáforos

Semáforos

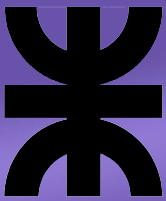
Clasificación:

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.



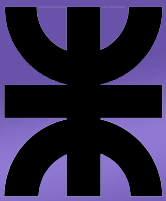
SEMÁFOROS SIN NOMBRE



Semáforos sin nombre

Semáforos sin nombre

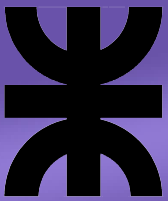
- Semáforos sin nombre se almacenan en la memoria asignada por la aplicación.
- El uso de un semáforo sin nombre nos permite evitar el trabajo de crear un nombre para un semáforo.
- El semáforo es puesto a disposición de los procesos relacionados o hilos que lo utilizan colocándolo en un área de la memoria que comparten.



Semáforos sin nombre

Semáforos sin nombre

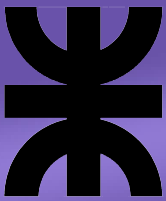
- Semáforos sin nombre se almacenan en la memoria asignada por la aplicación.
- El uso de un semáforo sin nombre nos permite evitar el trabajo de crear un nombre para un semáforo.
- El semáforo es puesto a disposición de los procesos relacionados o hilos que lo utilizan colocándolo en un área de la memoria que comparten.



Semáforos sin nombre

Uso de semáforos sin nombre:

- Un semáforo que se comparte entre los hilos no necesita un nombre.
- Un semáforo que está siendo compartido entre los procesos relacionados, no necesita un nombre. Si un proceso padre asigna un semáforo sin nombre en una **región de memoria compartida**, un hijo hereda automáticamente la asignación y, por tanto el semáforo como parte de la operación de `fork()`.



Semáforos sin nombre

Inicialización de un semáforo sin nombre

El `sem_init()` inicializa el semáforo sin nombre

```
#include <semaphore.h>
```

```
sem_t sem;
```

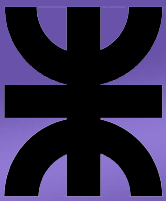
```
int sem_init(&sem, int pshared, int value);
```

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



Semáforos sin nombre

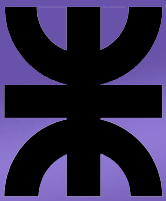
Esperando un semáforo

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 luego `sem_wait()` vuelve.



Semáforos sin nombre

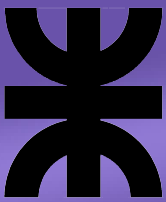
Esperando un semáforo

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`.



Semáforos sin nombre

Esperando un semáforo

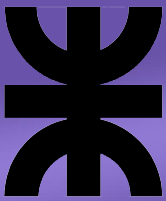
La función `sem_timedwait()` es otra variante de `sem_wait()`. Permite al que llama a la función especificar un límite de tiempo para el cual se bloqueara la llamada.

```
#include <semaphore.h>
sem_t sem;
int sem_timedwait(&sem, const struct timespec *abs_timeout);
```

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

Si la llamada a `sem_timedwait()` termina sin poder decrementar el semáforo, la llamada falla con el error de `ETIMEDOUT`.

El argumento `abs_timeout` es una estructura `timespec` que especifica el tiempo de espera como un valor absoluto en cuestión de segundos y nanosegundos



Semáforos sin nombre

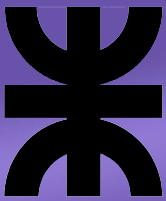
Incrementar un semáforo

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.



Semáforos sin nombre

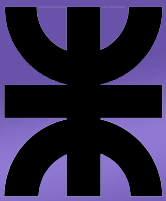
Recuperación del valor actual de un semáforo

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

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

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

Si uno o más procesos (o hilos) están bloqueados a la espera de disminuir el valor del semáforo, entonces el valor devuelto en `sval` depende de la implementación. SUSv3 permite dos posibilidades: 0 o un número negativo cuyo valor absoluto es el número de esperadores bloqueados en `sem_wait()`. Linux y otras implementaciones adoptan el comportamiento anterior.



Semáforos sin nombre

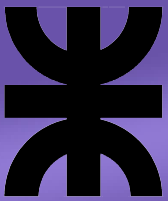
Destruir un semáforo sin nombre

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 estén 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



Semáforos sin nombre

Bibliografía

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