Sincronización de Hilos POSIX

Las hebras compaiten el acceso al espacio de direcciones de un proceso, por lo tanto, se necesitan primitivas de sincronización para controlar el acceso a los datos comunes.

La biblioteca de hebras de linux proporciona varias primitivas de sincronización, entre las que se encuentras los semáforos generales.

Un semáforo POSIX es un tipo de variable sem_t con operaciones atomicas tales como:

- inicalización
- wait
- signal

Existen semáforos nombrados y no nombrados. Un semaforo nombrado puede ser utilizado por cualquier proceso.

¿Qué es la sincronización y para que sirve?

La sincronización de hilos es la ejecución simultanea de 2 o más subprocesos que comparten recursos críticos.

Los hilos deben ser sincronizados para evitar conflictos de uso de recursos criticos. De lo contrario, pueden surgir conflictos cuando los hilos que se ejecutan en paralelo intentan modificar una variable común al mismo tiempo.

¿Que es la exclusión mutua?

Consiste en que un solo proceso excluye temporalmente a todos los demás para usar un recurso compartido de forma que garantice la integridad del sistema.

¿Qué son las secciones críticas?

Segmentos de código que manipulan 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.

¿Qué son los deadlocks?

Es el bloqueo permanente de un conjunto de procesos que compiten por los recursos del sistema o bien se comunican unos con otros.

A diferencia de otros problemas de la gestión concurrente de procesos, para el caso general no existe una solución eficiente.

¿Qué es la espera activa e inactiva?

Espera activa: Se denomina espera activa o espera ocupada a una técnica donde un proceso repetidamente verifica una condición, tal como esperar una entrada de teclado o si el ingreso a una sección crítica esta habilitado.

API para crear semaforos POSIX

sem_open: Devuelve un identificador que es utilizado por sem_wait, sem_trywait, sem_post y sem_get_value para referirse a dicho semaforo.

```
#include <semaphore.h>
sem_t * sem_open(const char * name, int oflag);
sem_t * sem_open(const char * name, int oflag,
                 mode_t mode, unsigned int value);
```

sem_close: Libera los recursos que el sistema asigna a un proceso cuando trabaja con un determinado semáforo.

```
#include <semaphore.h>
int sem_close (sem_t * sem);
```

No necesariamente elimina el semáforo, sólo hace inaccesible al proceso.

sem_unlink: Elimina el semaforo nombrado.

```
#include <semaphore.h>
int sem_unlink (sem_t * sem);
```

- Si otros semaforos o procesos hacen referencia al semaforo, la destruccion se pospone.

sem_wait: Decrementa el semaforo

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

sem_post: Incrementa el semaforo.

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