Systèmes et C

Wieslaw Zielonka
zielonka@irif.fr
www.irif.fr/~zielonka

sémaphores POSIX

sémaphores

un sémaphore : une sorte de variables qui prend comme valeurs les entiers non-négatifs.

Deux opérations **atomiques** (non-divisibles) sur les sémaphores :

- `sem_post( &sémaphore)` qui incrémente la valeur du sémaphore
- `sem_wait(&sémaphore)` :
  - si la valeur de sémaphore est supérieure à `0` `sem_wait()` décremente cette valeur,
  - si la valeur est 0 le processus est suspendu jusqu'à ce que la valeur devienne positive.

Deux types de sémaphores : sémaphores nommés et anonymes.

sémaphores nommé

création / ouverture de sémaphore nommé

```
sem_t *sem_open(const char * name, int oflag, ...
                /* mode_t mode, unsigned int value */)
```

sem_open retourne pointeur vers le sémaphore si OK, SEM_FAILED sinon

name - nom du sémaphore, commence par '/'

oflag - 0 ouverture de sémaphore existant
O_CREAT création
O_CREAT | O_EXCL création si n'existe pas, échec si existe

mode - les permission pour le sémaphore créé, les mêmes que pour les fichiers

value - la valeur initiale du sémaphore si la création

fermeture et suppression de sémaphore nommé

```
int sem_close(sem_t sem)

int sem_unlink(const char *name)

retourne 0 si OK, -1 si erreur.

sem_close ferme le sémaphore, mais ne le
supprime pas.

C'est similaire comme pour les fichiers
ordinaire.

sem_unlink() supprime le sémaphore (quand il
n'est plus utilisé par d'autres processus)
```

fermeture et suppression de sémaphore nommé

`int sem_post(sem_t *sem)`

incrémente la valeur de sémaphore

`int sem_wait(sem_t *sem)`

`int sem_trywait(sem_t *sem)`

retournent 0 si OK, -1 si erreur.

`sem_wait` – décrémente la valeur si elle est >0, bloque le processus si la valeur est 0

`sem_trywait` - version non-bloquante de sem_wait(), décrémente la valeur si elle est > 0, retourne -1 et

`errno == EAGAIN`

`si la valeur de sémaphore est 0`

fermeture et suppression de sémaphore nommé

int sem_getvalue(sem_t *sem, int *sval)

retourne 0 si OK, -1 sinon

met la valeur de sémaphore à l'adresse val.

sem_getvalue() n'est pas implémenté dans MacOS.
Mais existe dans Linux.

Pas très utile. (Pourquoi?)

sémaphores anonymes

sémaphores anonymes

Doivent résider :

- soit dans la mémoire allouée par mmap, par exemple mmap MAP_ANONYMOUS
- soit dans un objet mémoire (shared memory object)
- soit dans les variables globales pour la synchronisation entre les threads (cela concerne uniquement les threads du même processus)

(pourquoi?)

deux nouvelles fonctions qui s'appliquent uniquement aux sémaphores anonymes :

- sem_init() pour initialiser
- sem_destroy() pour supprimer

sémaphores anonymes

```
int sem_init(sem_t *sem, int pshared,
             unsigned int value)
```

initialise le sémaphore avec la valeur **value**

pshared == 0 si le sémaphore utilisé pour la synchronisation entre les threads

pshared != 0 si le sémaphore utilisé pour la synchronisation entre les processus

```
int sem_destroy(sem_t *sem)
```

supprime le sémaphore anonyme

sémaphores anonymes

MacOS ne possède pas de sémaphores anonymes.

Sur macOS on peut utiliser les sémaphores UNIX System V (XSI extension de Single UNIX Specification).