Compte rendu Semaphores Nanvix

Erwan Poncin / Maxime Bossant / Maxence Maury

Lien du git, branche semaphore : https://github.com/erwanponcin/AS/tree/semaphore2/

1. Compréhension des Sémaphores

Les sémaphores sont des mécanismes de synchronisation utilisés pour gérer l'accès concurrent aux ressources partagées dans un système d'exploitation. Dans le cadre de ce TP, nous avons implémenté les sémaphores dans le noyau du système Nanvix. Un sémaphore est constitué d'une valeur entière et d'une file d'attente de processus bloqués. Les opérations de base sur un sémaphore sont `up` (incrémenter) et `down` (décrémenter), qui permettent de contrôler l'accès aux ressources de manière ordonnée et sécurisée.

2. Fichier `sem.c`

Le fichier `sem.c` contient l'implémentation principale des sémaphores. Il définit une structure `semaphore` qui inclut la valeur du sémaphore, une unique key, une file d'attente de processus, et la longueur de cette file.

Pour chaque `semid` utilisés, on vérifie qu'il est bien entre les bornes 0 et MAX_SEM (constante taille max de sémaphores)

- **Fonction** `sem_check` **:** La fonction `sem_check` vérifie si un sémaphore correspondant à une **key** donnée existe déjà dans le tableau des sémaphores.
  - **Si le sémaphore existe**, elle retourne son `semid`.
  - **Si le sémaphore n'existe pas**, elle retourne `-1` et `sem_create` sera appelé dans `sem_get`.
- **Fonction** `sem_create` **:** Crée un nouveau sémaphore avec une valeur initiale de 1. Il alloue une file d'attente statique pour les processus bloqués.
- **Fonction** `get_sem_value` **:** Renvoie la valeur actuelle d'un sémaphore à partir de son identifiant (`semid`) et `-1` en cas d'erreur (semaphore non défini).
  - **Appelée dans :** `sys_semctl` pour la commande `GETVAL`.
- **Fonction** `set_sem_value` **:** Modifie la valeur d'un sémaphore renvoie `0` en cas de succès et `-1` en cas d'erreur (semaphore non défini).
  - **Appelée dans :** `sys_semctl` pour la commande `SETVAL`.
- **Fonction** `destroy` **:** Détruit un sémaphore en réinitialisant ses valeurs et en libérant la file d'attente. Renvoie `0` en cas de succès et `-1` en cas d'erreur (semaphore non défini).
- **Fonction** `up` **:** Incrémente la valeur du sémaphore si aucun processus n'est bloqué. Sinon, réveille le premier processus dans la file d'attente.
- **Fonction** `down` **:** Décrémente la valeur du sémaphore si elle est supérieure à zéro. Sinon, ajoute le processus courant à la file d'attente et le met en sommeil.

3. Fichier `semop.c`

Le fichier `semop.c` implémente l'appel système `semop`, qui effectue les opérations `up` et `down` sur un sémaphore en fonction de la valeur de l'argument `op`.

- **Fonction** `sys_semop` **:** Appelle `up` si `op` est positif et `down` si `op` est négatif. Cela permet d'incrémenter ou de décrémenter la valeur du sémaphore de manière atomique.

4. Fichier `semctl.c`

Le fichier `semctl.c` implémente l'appel système `semctl`, qui fournit diverses opérations de contrôle sur les sémaphores.

- **Commandes :**
  - `GETVAL` : Renvoie la valeur actuelle du sémaphore.
  - `SETVAL` : Définit la valeur du sémaphore.
  - `IPC_RMID` : Détruit le sémaphore.

5. Fichier `semget.c`

Le fichier `semget.c` implémente l'appel système `semget`, qui permet à un processus d'utiliser un sémaphore associé à une unique key.

- **Fonction `sys_semget` :** Vérifie si un sémaphore existe pour la key donnée. Si ce n'est pas le cas, il en crée un nouveau.

Compréhension et Fonctionnement

L'implémentation des sémaphores dans Nanvix repose sur la gestion atomique des opérations `up` et `down`, garantissant qu'aucune autre opération ne peut être effectuée simultanément sur le même sémaphore.

La file d'attente des processus bloqués est gérée dynamiquement, permettant aux processus de se mettre en sleep lorsqu'ils ne peuvent pas acquérir le sémaphore et d'être réveillés lorsque la ressource devient disponible. Pour améliorer notre code, nous aurions aimé sécuriser notre implémentation d'un point de vue concurrence en ajoutant un système de lock.

Malheureusement , les tests des sémaphores ne sont pas concluant :

```
Interprocess Communication Tests
general protection fault: 0
  [eax: 0xc040df60] [ebx:    0xc014bf00]
  [ecx: 0x00000004] [edx:    0x00000004]
  [esi: 0xc014bf00] [edi:    0xc040df60]
  [ebp: 0xc040df80] [esp:    0xc040df24]
  [eip: 0xc011317e] [eflags: 0x00010282]
process 3 caused and exception
PANIC: kernel running
```

L'erreur signifie qu'un processus a tenté d'accéder à une zone mémoire non autorisée, ce qui a provoqué une exception.

Cela pourrait venir de différentes erreurs :

- Accès à une adresse mémoire non allouée
- Mauvaise manipulation des sémaphores ou des queues

Nous n'avons donc malheureusement pas encore implémenté le programme utilisateur avec des processus producteurs/consommateurs.