Programmation Système
Variables de condition et sémaphores

Juliusz Chroboczek
Khouloud Zine Elabidine

17 décembre 2012

Exercice 1. Un *sémaphore comptant* est une variable entière ayant deux opérations :
- `V`, qui incrémente la valeur du sémaphore ;
- `P`, qui décrémente la valeur du sémaphore si elle est strictement positive, et bloque sinon.

1. Écrivez un programme qui créé un sémaphore (`sem_open`) de valeur 0, puis qui crée un *thread*. Le *thread* créé décrémente le sémaphore, attend un temps aléatoire entre 0 et 500 ms, affiche un message, et recommence cent fois. Le programme principal incrémente le sémaphore, attend 100 ms, et recommence cent fois. (Vous pourrez vous servir de l'appel système `nanosleep`).
2. Implémentez maintenant le sémaphore à la main, à l'aide d'une variable entière protégée par un *mutex*. Lorsqu'un processus tentera de décrémenter un sémaphore nul, il fera une attente active (une boucle avec `sched_yield`).
3. Modifiez votre programme pour qu'il évite l'attente active à l'aide d'une variable de condition.

Exercice 2. Cinq philosophes, numérotés de 0 à 4, sont dans une pièce et réfléchissent. Dans la pièce, il y a une table avec cinq assiettes (numérotées de 0 à 4) et cinq fourchettes situées entre les assiettes.

Réfléchir est un travail qui donne faim. De temps en temps (de façon imprévisible), le philosophe numéro n s'assied à sa place, prend la fourchette de gauche (numéro n), puis la fourchette de droite (numéro $n+1 \pmod 5$), mange pendant un moment, puis repose les deux fourchettes.

1. Décrivez une suite d'événements qui fait que les philosophes meurent de faim face à une table couverte de nourriture.
2. Écrivez un programme implémentant l'algorithme décrit ci-dessus à l'aide de cinq *threads* ; chaque fourchette sera représentée par un *mutex*, et le fait de manger sera représenté par un appel à `nanosleep` avec une valeur aléatoire. Pouvez-vous reproduire le problème de la mort de faim ?
3. Une solution au problème consiste à empêcher plus de quatre philosophes de s'assoir à table. Implémentez cette solution à l'aide d'un sémaphore comptant initialisé à 4.
4. Reimplémentez votre solution en implémentant un sémaphore comptant à la main à l'aide d'une variable entière, un *mutex* et une variable de condition.
5. Proposez une solution distribuée au problème.