Introduction aux Sémaphores

Julien Soulé

14 octobre 2024

Objectifs

- Comprendre le concept de sémaphore dans la gestion des processus.
- Étudier les opérations de base sur les sémaphores (P et V).
- Découvrir des exemples concrets de leur utilisation.
- Expérimenter avec le problème de gestion de parking.

Qu'est-ce qu'un Sémaphore ?

Définition

Un sémaphore est une variable utilisée pour contrôler l'accès à une ressource partagée dans un environnement de traitement parallèle. Il permet de s'assurer qu'un nombre limité de processus accède à la ressource en même temps.

Qu'est-ce qu'un Sémaphore ?

Définition

Un sémaphore est une variable utilisée pour contrôler l'accès à une ressource partagée dans un environnement de traitement parallèle. Il permet de s'assurer qu'un nombre limité de processus accède à la ressource en même temps.

Types de Sémaphores

- **Compteur** : Maintient le nombre de ressources disponibles.
- **Binaire** : Agit comme un verrou, permettant un accès exclusif.

Opérations de Base

Les sémaphores sont principalement manipulés par deux opérations atomiques :

Opérations de Base

Les sémaphores sont principalement manipulés par deux opérations atomiques :

- **P (Proberen / Wait)** : Décrémente le sémaphore et bloque si la valeur est inférieure ou égale à zéro.
- **V (Verhogen / Signal)** : Incrémente le sémaphore et libère un processus en attente.

Opérations de Base

Les sémaphores sont principalement manipulés par deux opérations atomiques :

- **P (Proberen / Wait)** : Décrémente le sémaphore et bloque si la valeur est inférieure ou égale à zéro.
- **V (Verhogen / Signal)** : Incrémente le sémaphore et libère un processus en attente.

```
void P(int semid) {
    struct sembuf p_op;
    p_op.sem_num = 0;
    p_op.sem_op = -1; // P
    p_op.sem_flg = SEM_UNDO;
    semop(semid, &p_op, 1);
}
```

```
void V(int semid) {
    struct sembuf v_op;
    v_op.sem_num = 0;
    v_op.sem_op = 1; // V
    v_op.sem_flg = SEM_UNDO;
    semop(semid, &v_op, 1);
}
```

Opérations de Base

Un sémaphore permet de synchroniser des processus. Il est comparable à un ensemble de "jetons". Un processus fait un :

- P() pour "prendre un jeton" ("sonder").
  S'il n'y a plus de jetons, il attend qu'un jeton soit libéré.
- V() pour "liberer un jeton" ("incrémenter").

```
#include <stdio.h>
#include <sys/sem.h>

void P(int semid) { // (Proberen)
    struct sembuf p = {0, -1, 0}; semop(semid, &p, 1); }
void V(int semid) { // (Verhogen)
    struct sembuf v = {0, 1, 0}; semop(semid, &v, 1); }

int main() {
    key_t key = ftok(".", 65); // Clé du sémaphore (chemin + id)
    // Création (lecture/écriture + créer si existe pas)
    int semid = semget(key, 1, 0666 | IPC_CREAT);
    semctl(semid, 0, SETVAL, 1); // Initialisation à 1
    P(semid); // "Prendre un jeton"
    printf("Zone critique !\n");
    V(semid); // "Liberer le jeton"
    return 0; }
```

Exemple : Gestion de Guichet Unique

- Objectif : Limiter l'accès à un guichet unique afin qu'un seul client soit servi à la fois.
- Solution : Utilisation d'un sémaphore binaire pour restreindre l'accès.

Exemple : Gestion de Guichet Unique

- Objectif : Limiter l'accès à un guichet unique afin qu'un seul client soit servi à la fois.
- Solution : Utilisation d'un sémaphore binaire pour restreindre l'accès.

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>

int main() {
    key_t cle = ftok("path/to/file", 'G');
    int semid = semget(cle, 1, IPC_CREAT | 0666);
    semctl(semid, 0, SETVAL, 1);

    P(semid); // Accéder au guichet
    printf("Client est servi\n");
    V(semid); // Libérer le guichet

    return 0;
}
```

Conclusion

- Les sémaphores sont essentiels pour la synchronisation des processus dans des environnements concurrents.
- Ils permettent de gérer efficacement les ressources limitées.
- D'autres applications incluent la gestion des ressources mémoire, les systèmes de fichiers, et les serveurs multi-clients.