Chapitre 7 - Mémoire partagée

Table des matières

1 - Introduction

Nous avons vu dans différents exemples que les sémaphores sont des mécanismes de synchronisation puissants, qui permettent même de partager des informations dans une certaine mesure.

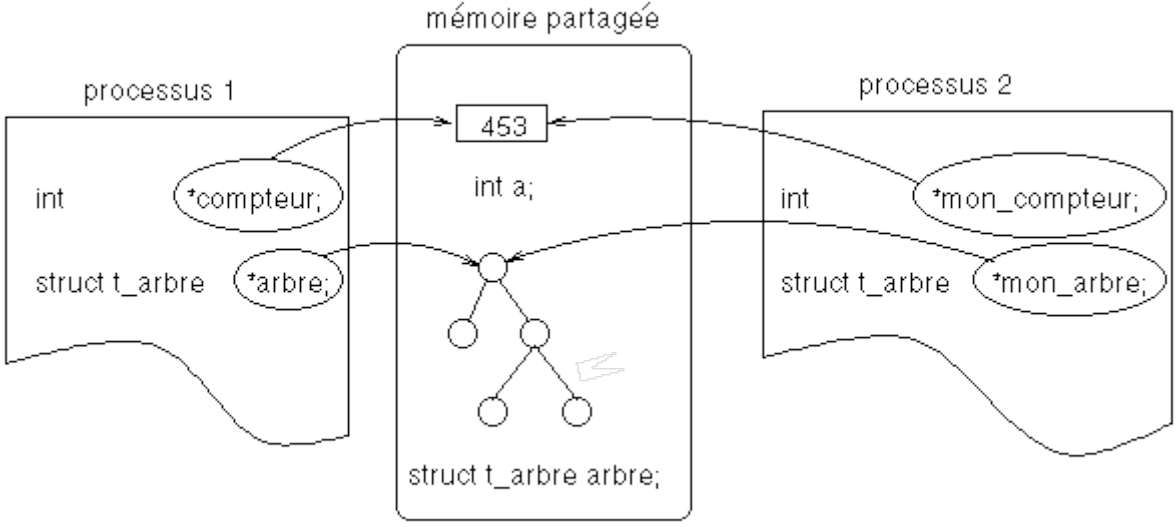
Cependant, ils se heurtent à certains problèmes et certaines limitations suivant la complexité du programme, qui demandent plus qu'un objet de type Sémaphore.

Pour résoudre ce problème, même si les processus ne partagent pas le même espace mémoire, il est possible néanmoins de déclarer un tel espace de manière explicite dans la cadre des IPC.

2 - Segments de mémoire partagée « System V »

2.1 - Le principe

Rendre commun à des processus distincts une zone mémoire définie dynamiquement par l'un d'entre eux.



2.2 - Les commandes

Informations sur les segments : `ipcs -m`.

Suppression d'un segment donné : `ipcrm –m <shmid>`

2.3 - Création d'un segment et obtention d'un shmid

```
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
int shmget(key_t key, int size, int shm_flag) ;
```

Créer ou retrouver un segment de mémoire partagée. En cas de succès, renvoie d'un *shmid* (entier > 0), sinon renvoi de –1 et *errno*…

Le paramètre *key* est une clé d'accès au segment. Pour réserver le segment au processus créateur et à sa descendance, on lui donne la valeur IPC_PRIVATE.

Le paramètre *size* est la taille du segment.

Le paramètre *shm_flag* règle les droits d'accès et certains indicateurs de création et de recherche :

- IPC_CREAT : création d'un IPC s'il n'existe pas.
- IPC_EXCL : erreur si l'IPC existe déjà.
- 0xyz : codage octal des droits d'accès.
- 0 : recherche d'un IPC supposé exister.

Ex1 : IPC_CREAT|IPC_EXCL|0666

Création d'un s.m.p. avec *rw*, avec échec si le segment existe déjà.

Ex2 : 0

Recherche de segment existant avec retour du *shmid* si existence et retour de –1 sinon.

2.4 - Opérations de contrôle d'un segment

```
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
int shmctl(int shmid, int cmd, struct shmid_ds *buf) ;
```

Consulter, modifier les caractéristiques d'un segment, ou même le supprimer. Retourne 0 si succès, -1 sinon.

Le paramètre *shmid* est un descripteur de segment.

Le paramètre *cmd* précise l'opération de contrôle à réaliser :

- IPC_STAT : Consultation.
- IPC_SET : Modification.
- IPC_RMID : Suppression.

Le paramètre *buf* est l'adresse d'un objet de structure *shmid_ds* déclaré par l'utilisateur.

Exemple : Modification des caractéristiques d'un segment

```
struct shmid_ds brou
…
shmctl(id, IPC_STAT, &brou) ;
brou.shm_perm.mode=0420 ;
shmctl(id, IPC_SET, &brou) ;
sleep(1000) ;
shmctl(id, IPC_RMID, (struct shmid_ds *)0);
...
```

2.5 - Attachement à un segment

```
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
void* shmat(int shmid, void *buf, int shmflag) ;
```

Cette fonction permet de rendre visible un segment de mémoire partagée dans l'espace d'adressage d'un processus, elle retourne l'adresse d'attachement, sinon –1.

Le paramètre *shmid* est le descripteur du segment.

Le paramètre *buf* est l'adresse d'attachement ; il est conseillé de laisser le système de choisir le point de greffe en passant zéro comme argument.

Le paramètre *shmflag* est en général égal à zéro, ce qui signifie que le segment est attaché en lecture/écriture. On peut interdire toute écriture en passant SHM_RDONLY comme argument.

2.6 - Le détachement d'un segment

```
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
int shmdt(void *buf) ;
```

Supprimer la visibilité d'un processus sur un segment. Retourne zéro ou -1. Le paramètre *buf* est l'adresse d'attachement du segment, c'est-à-dire la valeur renvoyée par la primitive *shmat*.

2.7 - Détachement et suppression

Le détachement n'affecte que la visibilité du segment depuis un processus et pas du tout son existence. Un segment pouvant être attaché à plusieurs processus, une suppression n'aura vraiment lieu que lorsque le champ *shm_nattch* de la structure *shmid_ds* décrivant le segment sera à 0. L'opération de contrôle SHM_RMID réalisé avec la primitive *shmctl* consiste donc à lever le drapeau SHM_DEST qui signifie au noyau de supprimer le segment dès que possible.

2.8 - Exemple de segment de mémoire partagé (Création)

```
/* C. Drocourt - LIS - Ch8 - ipc4.c */

int main() {
  key_t key; int flag, id; char *buf;

  flag=IPC_CREAT|0600;

  if((key = ftok("/tmp/maclef",12)) == -1) {
    perror("Probleme de cle");
    exit(2);
  }

  if ((id=shmget(key,512,flag))<0) exit(1);
  if ((buf=shmat(id,0,0))<0)       exit(2);
  strcpy(buf,"Bonjour");
  sleep(100); /* faire un ipcs -m pendant ce temps */
  shmdt(buf);
  exit(0);
}
```

2.9 - Exemple de segment de mémoire partagé (Utilisation)

```
/* C. Drocourt - LIS - Ch8 - ipc5.c */

int  main() {
  key_t key; int flag, id; char *buf;

  flag=0;
  if((key = ftok("/tmp/maclef",12)) == -1) {
     perror("Probleme de cle ");
     exit(2);
  }

  if ((id=shmget(key,512,flag))<0) exit(1);
  if ((buf=shmat(id,0,0))<0)       exit(2);
  printf("Contenu : %s \n", buf);
  shmdt(buf);
  shmctl(id,IPC_RMID,0);
  exit(0);
}
```