UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence 3 Informatique
Info0601

Outils IPC

Segment de mémoire partagée

Cet article présente plusieurs programmes permettant de manipuler les segments de mémoire partagée IPC System V. Un programme (appelé serveur) crée un segment de mémoire partagée puis y place une série d'entiers. Un second programme (appelé client) récupère le segment puis lit les entiers.

1 Le programme serveur

Le serveur doit d'abord créer le segment qui contiendra les 10 entiers.

```
int shmid;
if((shmid = shmget((key_t)CLE, sizeof(int) * 10, S_IRUSR | S_IWUSR |
    IPC_CREAT | IPC_EXCL)) == -1) {
  if(errno == EEXIST)
    fprintf(stderr, "Le segment de mémoire partagée (cle=%d) existe deja
        \n", CLE);
  else
    perror("Erreur lors de la création du segment de mémoire ");
  exit(EXIT_FAILURE);
}
```

Pour utiliser le segment, il faut l'attacher :

```
int *adresse;
if((adresse = shmat(shmid, NULL, 0)) == (void*)-1) {
  perror("Erreur lors de l'attachement du segment de mémoire partagée ")
      ;
  exit(EXIT_FAILURE);
}
```

Nous pouvons ensuite l'initialiser :

```
for(i = 0; i < 10; i++)
  adresse[i] = i * 2;
```

Une fois terminé, nous pouvons le détacher :

```
if(shmdt(adresse) == -1) {
  perror("Erreur lors du détachement ");
  exit(EXIT_FAILURE);
}
```

2 Le client

Le client doit d'abord récupérer l'identifiant interne.

```
int shmid;
if((shmid = shmget((key_t)CLE, 0, 0)) == -1) {
  perror("Erreur␣lors␣de␣la␣récupération␣du␣segment␣de␣mémoire␣partagée␣
     ");
  exit(EXIT_FAILURE);
}
```

Comme le serveur, il doit ensuite l'attacher et il peut ensuite l'utiliser.

```
if((adresse = shmat(shmid, NULL, 0)) == (void*)-1) {
  perror("Erreur␣lors␣de␣l'attachement␣du␣segment␣de␣mémoire␣partagée␣")
     ;
  exit(EXIT_FAILURE);
}

printf("Client␣:␣entiers␣lus␣=␣[");
for(i = 0; i < 10; i++) {
  printf("%d", adresse[i]);
  if(i < 9) printf(",␣");
}
```

3 Suppression du segment de mémoire partagée

Pour supprimer le segment de mémoire partagée à l'aide d'un programme en C, nous devons dans un premier temps récupérer l'identifiant interne du segment de mémoire à l'aide de `shmget`. Puis nous utilisons l'appel système `shmctl` avec la commande `IPC_RMID` (le troisième paramètre est inutile).

```
if(shmctl(shmid, IPC_RMID, 0) == -1) {
  perror("Erreur␣lors␣de␣la␣suppression␣du␣segment␣de␣mémoire␣partagée␣"
     );
  exit(EXIT_FAILURE);
}
```

4 Compilation et exécution

Le `makefile` fourni permet de compiler les programmes précédents. Dans la section `ARGUMENTS ET COMPILATEUR`, la variable `CLE_SHM` correspond à la clé du segment de mémoire partagée utilisée dans les différents programmes. Elle est spécifiée par `gcc` grâce à l'option `-D`.

Pour compiler les programmes précédents, saisissez la commande suivante :

```
make
```

Pour tester les programmes, dans un premier terminal, exécutez le serveur :

```
./memoireServeur
```

Puis exécutez le client :

```
./memoireClient
```

Pour supprimer le segment de mémoire partagée, vous pouvez soit utiliser le programme `memoireSupprime`, soit la règle du `makefile` `cleanIPC`, soit utiliser la commande `ipcrm` (où X est la clé de la file de messages) :

```
./memoireSupprime
make cleanIPC
ipcrm -M X
```