IPC - Sémaphore, Mémoire partagée et File de message

TP - Concepts des systèmes d'exploitation

Sereysethy Touch

Université de Namur



Référence

- [1] Christohoper Blaess. *Programmation système en C sous Linux*. 2003.
- [2] Jean-Noël Colin. Travaux Systèmes Exploitation. 2003.
- [3] Brian W. Kernighan. *The C Programming Language*. Ed. by Dennis M. Ritchie. 2nd. Prentice Hall Professional Technical Reference, 1988. ISBN: 0131103709.
- [4] J. F. Lalande. *Programmation C.* URL: http://www.univ-orleans.fr/lifo/Members/Jean-Francois.Lalande/teaching.html (visited on 08/01/2017).
- [5] N. Matthew and R. Stones. *Begining Linux Programmation*. 4rd Edition. Wiley Publishing, Inc., Indianapolis, Indiana, 2008.
- [6] Emmanuel Viennet. Introduction au système UNIX. 2017.

Agenda

Communication Inter Processus

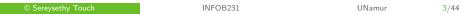
Sémaphore

Segment de mémoire partagés

File de messages

Commandes utilitaires de IPC

Exercice



Introduction

IPC - un mécanisme de communication permettant à des groupes de processus de se communiquer. Deux types de communications:

- sur la même machine (communication locale)
- sur des machines différentes (communication à distante)

Communication locale:

- signal
- tubes (pipes)
- files de messages (message queues)
- segment de mémoire partagé (shared memory)

Communication à distant - client/serveur:

Socket

Problème de synchronisation lors de l'accès à des resources partagées:

Sémaphore

Agenda

Communication Inter Processus

Sémaphore

Segment de mémoire partagés

File de messages

Commandes utilitaires de IPC

Evercice



Sémaphore

- Sémaphore est utilisé pour protéger une section ou une resource critique où à un moment donné un seul processus peut l'exécuter ou y accéder.
- Deux opérations possibles:
 - P(semaphore): décrémente la valeur de sémaphore si semaphore > 0, sinon le processus est suspendu (attend).
 - V(semaphore): libère un processus suspendu s'il existe, sinon incrémente la valeur de sémaphore.
- On s'intéresse à un sémaphore binaire qui ne peut avoir qu'une valeur 0 et 1.

Utilisation

```
semaphore sv = 1;
P(sv);
section critique
V(sv);
```

Sémaphore sur Linux

3 fonctions pour manipuler un sémaphore sur Linux:

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

int semctl(int semid, int semnum, int cmd, ...);
int semget(key_t key, int nsems, int semflg);
int semop(int semid, struct sembuf *sops, size_t nsops);
```

Ces fonctions ont été créés pour opérer sur un tableau de sémaphores, ce qui rend ces opérations plus compliquées.

Notez que **key** sert comme un nom de fichier, il représente une resource que les programmes utilisent pour collaborer, s'ils se mettent en accord pour un nom commun.

semget()

On crée un sémaphore à l'aide d'une fonction semget() int semget(key_t key, int nsems, int semflg); qui prend des arguments suivants:

- une clé key (un entier);
- un nombre de sémaphores nsems à créer;
- un flag semflg; s'il s'agit de IPC_CREAT, la fonction crée un nouvel ensemble de sémaphores associé à la clé key.
 Il faut aussi spécifier la permission comme la création d'un fichier: rwxrwxrwx pour propriétaire, groupe et le reste du monde.

© Sereysethy Touch INFOB231 UNamur 10/44

semget()

La fonction retourne

- un identifiant d'un ensemble de sémaphores en cas de succès (un entier positif);
- -1 en cas d'erreur. Par exmemple, on veut créer un sémaphore qui existe déjà avec le flag IPC_CREAT.

Un élément d'un ensemble de sémaphores commence à partir de 0.

Initialisation d'un sémaphore

Avant de pouvoir utiliser un sémaphore, il faut initialiser sa valeur initiale avec la fonction **semctl()**.

```
int semctl(int semid, int semnum, int cmd, ...);
```

Ses arguments peut être 3 ou 4 en fonction de la variable *cmd*. Quand, il a 4 arguments, le 4ième a le type union semun. Il faut définir ce type comme suivant:

Initialisation d'un sémaphore

fonction semget();

semid l'identifiant de l'ensemble de sémaphores obtenu par la

- semnu le numéro de sémaphore sur lequel on veut faire une opération;
- cmd peut avoir plusieurs valeurs en fonction de l'opération qu'on veut faire. Pour initialiser la valeur de sémaphore, utilisez une commande SETVAL.
- Pour obetnir la valeur de sémaphore, utilisez la commande GETVAL

Pour plus ample information, consultez le manuel en ligne de la fonction semctl.

© Sereysethy Touch INFOB231 UNamur 13/44

Exemple

```
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#define SEM KEY 1234
/* supposons qu'on définit l'union semun */
int main() {
  int sem id:
 union semun sem union;
 // créez un ensemble d'un sémaphore
  sem_id = semget((key_t) SEM_KEY, 1, 0666 | IPC_CREAT | IPC_EXCL);
  if (sem id == -1) {
   perror("semget()");
   exit(EXIT FAILURE);
  sem union.val = 1; // valeur initiale à 1
  if (semctl(sem id, 0, SETVAL, sem union) == -1) {
   perror("semctl()");
   exit(EXIT FAILURE);
```

Opérations P(sem) et V(sem)

```
int semop(int semid, struct sembuf *sops, size_t nsops);
```

- semid l'identifiant d'ensemble de sémaphore obtenu par semget();
- struct sembuf sops contient un tableau d'opérations;
- nsops indique le nombre d'éléments pointé par sops.

```
// p(sem)
struct sembuf lock[1];
lock[0].sem_num = 1;
lock[0].sem_op = -1;
lock[0].sem_flg = 0;
semop(semid, lock, 1)
```

```
// v(sem)
struct sembuf unlock[1];
unlock[0].sem_num = 1;
unlock[0].sem_op = 1;
unlock[0].sem_flg = 0;
semop(semid, unlock, 1)
```

Supprimer un sémaphore

- Le sémaphore doit être supprimé après son utilisation.
- Le sémaphore qui n'est pas proprement supprimé reste toujours dans le système même s'il n'est plus référencé.
- Pour le supprimer, la même fonction semctl est utilisée avec la commande IPC RMID.

```
int semctl(int semid, int semnum, int cmd);
```

- semid l'identifiant de l'ensemble de sémaphores;
- semnum est ignoré;
- cmd est IPC_RMID:

© Sereysethy Touch INFOB231 UNamur 16/44

Exercice

Considérons 3 processus suivants:

Ajouter des opérations sur des sémaphores tel que:

- Le résultat d'affichage est R I O OK OK OK
- La valeur finale de sémaphores est identique à sa valeur initiale.

source: http://people.rennes.inria.fr/Alan.Schmitt/teaching/index.html

© Sereysethy Touch INFOB231 UNamur 17/44

Agenda

Communication Inter Processus Sémaphore

Segment de mémoire partagés

File de messages

Commandes utilitaires de IPC

Evercice



Motivation

- Mémoire partagée (shared memory) est un des moyens permettant à différents processus d'accéder à un espace commun de mémoire.
- Mémoire partagée permet à des processus en cours d'exécution de partager des données.
- Accès à la mémoire partagée nécessite un contrôle, car la mémoire partagée elle-même ne dispose pas de moyen de synchronisation.

Fonctions pour la mémoire partagée

```
#include <sys/ipc.h>
#include <sys/types.h>
#include <sys/shm.h>

void *shmat(int shmid, const void *shmaddr, int shmflg);
int shmctl(int shmid, int cmd, struct shmid_ds *buf);
int shmdt(const void *shmaddr);
int shmget(key_t key, size_t size, int shmflg);
```

shmget

Création d'un segment de mémoire partagée.

```
int shmget(key_t key, size_t size, int shmflg);
```

- Comme sémaphore, le programme doit fournir une clé key pour identifier le segment de mémoire partagé.
- size spécifie le nombre d'octets nécessaires pour le segment de la mémoire partagé;
- flag spécifie le 9 mode d'accès comme dans un fichier. Un bit spécial IPC_CREAT doit être utilisé avec les modes d'accès pour créer un nouveau segment de mémoire partagée.

La fonction retourne -1 en cas d'erreur et un entier positif en cas de succès.

shmat

Le segment de mémoire partagé est créé mais il n'est pas encore accessible par auncun processus. Il faut l'attacher à un espace d'adressage du processus.

```
void *shmat(int shmid, const void *shmaddr, int shmflg);
```

- shmid est l'identifiant retourné par la fonction shmget;
- shmaddr spécifie l'adresse à la quelle le segment doit être attaché.
 Sa valeur doit être toujours un pointeur NULL; ceci permet au système de choisir un espace d'adressage convenable.
- Si la valeur de shmaddr est un pointeur NULL, shmflg doit être également égal à 0. Il est possible de spécifier que ce segment est en lecture seule (SHM_RDONLY).

La fonction retourne un pointeur d'adresse sur un segment en cas de succès et un pointeur (void *) -1 en cas d'erreur.

shmdt

Cette fonction permet de détacher un segment de mémoire partagée du processus courant.

```
int shmdt(const void *shmaddr);
```

La fonction prendre un pointeur retourné par la fonction **shmat** La fonction retourne -1 en cas d'erreur et 0 en cas de succès.

shmctl

C'est une fonction de contrôle de segment de mémoire partagée.

```
int shmctl(int shmid, int cmd, struct shmid_ds *buf);
```

- shmid est l'identifiant retourné par shmget;
- cmd indique l'opération à faire sur le segment; les valeurs possibles sont:
 - IPC_STAT enregistre les paramètres du segment dans une structure shmid_ds;
 - IPC_SET initialise les paramètres du segment avec la valeur de la structure shmid_ds;
 - IPC_RMID supprime le segment.
- buf est un pointeur sur la structure shmid_ds, pour plus d'info tapez man shmctl.

```
struct shmid ds {
  struct ipc_perm shm_perm;
                              /* Ownership and permissions */
                              /* Size of segment (bytes) */
  size_t
                 shm_segsz;
                shm atime:
                              /* Last attach time */
  time t
 time_t
                shm_dtime;
                              /* Last detach time */
 time_t
                shm_ctime;
                              /* Last change time */
 pid t
                 shm_cpid;
                              /* PID of creator */
                 shm lpid:
                              /* PID of last shmat(2)/shmdt(2) */
    © Sereysethy Touch
                                              INFOB231
```

DEMO

Un consommateur et producteur partagent un segment de mémoire partagée qui est une structure suivante:

```
struct command {
  int start;
  char value[25];
};
```

- Le consommateur attend jusqu'à start = 1 et il affiche la valeur contenue dans la variable value;
- Le producteur lit une chaîne et met start = 1, et il attend jusqu'à start = 0 pour lire une nouvelle chaîne;
- Les programmes s'arrêtent lorsque la chaîne "end" est lue.

Agenda

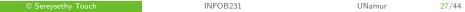
Communication Inter Processus Sémaphore

Segment de mémoire partagés

File de messages

Commandes utilitaires de IPC

Exercice



Motivation

- Une file de message (message queue) est un mécanisme qui permet d'échanger des messages par bloc entre deux processus non-reliés.
- Plus simple que le tube.
- Pas de problème de synchronisation.

#include <sys/types.h>

Fonctions

msgget

La fonction **msgget** permet de retourner un identifiant d'une file de messages.

```
int msgget(key_t key, int msgflg);
```

- Comme un sémaphore et un segment de mémoire partagée, le premier paramètre key est un identifiant;
- msgflg est le 9 permissions comme pour un fichier. Un bit spécial IPC_CREAT doit être également spécifié avec les permissions.

msgsnd

La fonction msgsnd permet d'ajouter un message à la file.

```
int msgsnd(int msqid, const void *msgp, size_t msgsz, int msg
```

- La structure de message doit respecter les contraintes suivantes:
 - sa taille doit être inférieure à la limite du système;
 - il faut commencer toujours par une variable de type **long int**.
- msgid est l'identifiant retourné par msgget;
- msgp est un pointeur sur la structure de message qui devra être envoyée;
- msgsz est la taille de message pointé par msgp, exclue la taille de long int;
- msgflg indique le comportement lorsque la file est pleine. Si IPC_NOWAIT est indiqué, la fonction retourne immédiatement, sinon le processus sera suspendu.

La valeur retourne -1 en cas d'erreur et 0 en cas de succès.

msgrcv

La fonction msgrcv permet de retirer un message à partir de la file.

- msgid est l'identifiant retourné par msgget;
- msgp est un pointeur sur le message retiré;
- msgz indique la taille de message à retirer;
- msgtyp indique l'opération à faire, si sa valeur est:
 - 0 le premier message dans la file est lu;
 - > 0 le premier message dans la file du type msgtyp est lu sauf que si le flag MSG_EXCEPT est spécifié dans msgflg, alors le premier message dont son type n'est pas égale à msgtyp est lu;
 - < 0 le premier message dont son type est <= à msgtyp est lu.</p>

La fonction retourne le nombre d'octets lu en cas de succès et -1 en cas d'erreur.

La structure générique de message

msgctl

Cette fonction est très similaire à la fonction shmctl.

int msgctl(int msqid, int cmd, struct msqid_ds *buf);

© Sereysethy Touch INFOB231 UNamur 35/44

DEMO

Un producteur envoie des messages (une chaîne) à des consommateurs à travers d'une file de message.

La structure de message est la suivante:

```
struct message {
  long int type;
  char data[MSG_SIZE];
};
```

- Un producteur génère une chaîne aléatoire 10 fois et l'ajoute à la file de message.
- Un ou plusieurs consommateurs lisent le message de la file et l'affiche sur l'écran.
- Le producteur supprime la file de message une fois elle n'est plus utilisée.

Agenda

Communication Inter Processus

Segment de mémoire partagés

File de messages

Commandes utilitaires de IPC

Evercice



ipcs

ipcs est une commande sous Linux permettant de connaître l'information de IPC sur un système.

Affichez le status de sémaphore

Affichez le status de mémoire partagée

Affichez le status de file de message

ipcrm

ipcrm permet de supprimer des resources de IPC. Syntax général

```
$ ipcrm [options]

-M shmkey, -m shmid
-Q msgkey, -q msgid
-S semkey, -s semid
-a (tout)
```

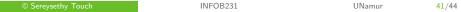
Consultez le manuel en ligne pour plus ample d'information.

Agenda

Communication Inter Processus Sémaphore Segment de mémoire partagés

Commandes utilitaires de IPC

Exercice



Exercice

- 1. Écrivez 2 programmes qui vont envoyer et recevoir des messages, et construire le dialogue suivant:
 - (Processus 1) Envoie le message "Are you hearing me?"".
 - (Processus 2) Reçoit le message et répond "Loud and Clear"".
 - (Processus 1) Reçoit la réponse et renvoie "I can hear you too".
- 2. Écrivez un programme serveur et deux programmes client tel que le serveur peut communiquer en privé avec chacun de client via une seule file de message.



UNIVERSITE

sereysethy.touch@unamur.be www.unamur.be