Programmation système
IPC
File de message

Florent Devin

EISTI

5 novembre 2010

Plan

Introduction

- IPC : **I**nter **P**rocess **C**ommunication
- Trois outils de communication entre des processus situés sur une même machine :
  - Files de messages (`msg`) : sorte de mail interne pour la communication des processus
  - Sémaphores (`sem`) : verrou interne pour
  - Segment de Mémoires Partagées (`shm`)
- Mécanisme en dehors du système de fichier

Cadre général

- Existence d'une table système par type d'objet
- Pas de désignation par **file descriptor** $\Leftrightarrow$ Impossibilité d'utiliser les redirections standards
- Un objet : un descripteur unique (similaire au **file descriptor**)

Cadre général

- Identification par clef : nécessaire pour l'utilisation des IPCs
- Possibilité d'accéder à la liste des IPC existantes : `ipcs`
- Possibilité d'effacer des IPCs : `ipcrm`

Gestions des clefs

Fonction `ftok`

- Permet la création de clefs
- Effectue la liaison entre le système de fichier et l'espace de nommage des IPCs
- Création d'un clé unique à partir d'une référence à un fichier

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

key_t ftok(const char *pathname, int proj_id);
```

Exemple

```c
int main(int argc, char** argv)
{
  int int_retour1; // Pour la clef 1
  int int_retour2; // Pour la clef 2
  // S'il y a le bon  nombre d'argument
  if (argc == 2) {
    // Création des clefs pour les IPCS
    int_retour1 = ftok (argv[1], 0);
    int_retour2 = ftok (argv[1], 1);

    // Si les deux opérations se sont bien passé
    if ((int_retour1 != -1) && (int_retour2 != -1)) {
      //Alors on affiche les valeurs
      printf ("Clef associé au fichier avec 0 : %x\n",
              int_retour1);
      printf ("Clef associé au fichier avec 1 : %x\n",
              int_retour2);
    } else {
      // Sinon on indique qu'il y a eu un pb
      printf ("Problème dans la création de la clef\n");
    }
  }
  // Puis on quitte
  return (0);
}
```

Constantes symboliques

- Primitive d'obtention
  - `IPC_PRIVATE` : clé privée. Un nouvel objet sera crée dont l'identité ne pourra être demandée ultérieurement au système($\Rightarrow$ IPC anonyme)
  - `IPC_CREAT` : demande de création de l'IPC correspondant à la clé
  - `IPC_EXCL` : échec si la clé existe déjà
- Primitive de contrôle
- Primitive de opératoire

Constantes symboliques

- Primitive d'obtention
- Primitive de contrôle
  - `IPC_RMID` : suppression de l'IPC
  - `IPC_SET` : modification de l'IPC
  - `IPC_STAT` : Obtenir les options concernant la ressource
- Primitive de opératoire

Constantes symboliques

- Primitive d'obtention
- Primitive de contrôle
- Primitive de opératoire
  - `IPC_NOWAIT` : opération non bloquante, retour erreur si l'opération est bloquante

File de message

Introduction

- Implantation UNIX du mail
- Permet la communication indirecte (asynchrone) entre processus
- Fonctionnement de type FIFO
- Permet l'envoi/réception de messages typés
- Lecture : destructrice
- Possibilité de "**multiplexage**" via une même file de message
- Un message : informations contiguës en mémoire ($\Rightarrow$ pas d'indirection)

Structure d'une file

- Deux structures permettant la gestion :
  - `struct msqid_ds` : structure général pour la manipulation. Contient des informations sur
    - droits de la file
    - qui/quand a envoyé/reçu le dernier message
    - longueur des messages dans la file
    - nombre maximal/contenu d'octet de la file de message
  - `struct ipc_perm` : structure pour la gestion des droits sur la file de message

Messages

- Messages typés
- Possibilité de choisir le type de message
- Type du message :

```
struct msgbuf {
  long mtype; //type de message
  char mtext[...]; /* texte du message */
};
```

- Type du message : entier > 0
- Taille du message : taille de la structure - taille de l'entier

File de messages

- `MSGMNI` : nombre maximal de messages possibles sur une file
- Deux éléments importants :
  - Pointeur sur le premier élément
  - Pointeur sur le dernier élément

Utilisation

- Création de la file : `msgget`
- Récupération de l'identifiant : `msgget`
- Envoi/réception de message : `msgsnd`/`msgrcv`
- Effectuer des opérations sur la file : `msgctl`

Création

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

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

- Création/récupération de la file à partir d'une clé
- Renvoie l'identifiant de la file
- `msgflag` :
  - Droits/paramètres de la file de message
  - `IPC_CREAT` ou `IPC_EXCL`
- `key` : clef ou `IPC_PRIVATE`

Contrôle d'une file de message

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

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

- Permet d'intervenir sur la file
- `msqid` : identificateur de la file
- `cmd` : type d'opération à effectuer (`IPC_STAT`, `IPC_SET`, `IPC_RMID`, `IPC_INFO`, `MSG_INFO`, `MSG_STAT`)
- `buf` : pour récupérer la table associée

Envoi de messages

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

int msgsnd(int msqid, const void *msgp, size_t
    msgsz, int msgflg);
```

- Envoi d'un message
- `msgflag` : Option pour l'envoi
  - `IPC_NOWAIT` : ne bloque pas le processus en cas de file pleine
  - `MSG_NOERROR` : tronque le message si pas assez de place

Réception de messages

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

ssize_t msgrcv(int msqid, void *msgp, size_t msgsz,
    long msgtyp, int msgflg);
```

- Récupère un message
- `msgsz` : longueur maximale du message
- si `msgtyp` $\neq 0$: lit un message typé
- `msgflg` : même option que pour l'envoi plus `MSG_EXCEPT`

Exemple

- `cf exemple2client.c`
- `cf exemple2serveur.c`