UNIVERSITÉ
DE REIMS
CHAMPAGNE-ARDENNE

Tubes nommés

Exemple d'utilisation

Cet article montre comment créer un tube nommé (appelé aussi fifo) pour échanger des données entre deux processus distincts.

1 Le serveur

Le but de ce programme est de créer un tube nommé. Il tente ensuite de l'ouvrir, ce qui le bloquera tant que le programme client ne sera pas exécuté. Il écrit ensuite 5 entiers. À la fin de son exécution, il détruit le fichier.

Dans cet exemple, le nom du tube est spécifié dans le fichier `include.h` :

```
#ifndef _INCLUDE_
#define _INCLUDE_

#define NOM_TUBE  "/tmp/montube"

#endif
```

Dans un premier temps, le père crée le tube à l'aide de `mkfifo`. Si le tube existe déjà, un simple message est affiché à l'écran :

```
if(mkfifo(NOM_TUBE, S_IRUSR | S_IWUSR) == -1) {
  if(errno != EEXIST) {
    fprintf(stderr, "Erreur␣lors␣de␣la␣création␣du␣tube␣'%s'", NOM_TUBE)
       ;
    perror("␣");
    exit(EXIT_FAILURE);
  }
  else
    fprintf(stderr, "Le␣tube␣'%s'␣existe␣déjà.\n", NOM_TUBE);
}
```

Il l'ouvre ensuite en écriture (ce qui le bloquera tant que le client ne l'ouvre pas de son côté en lecture) :

```
if((fd = open(NOM_TUBE, O_WRONLY)) == -1) {
  fprintf(stderr, "Erreur␣lors␣de␣l'ouverture␣du␣tube␣'%s'", NOM_TUBE);
  perror("␣");
  exit(EXIT_FAILURE);
}
```

Une fois le tube ouvert, le serveur envoie les cinq entiers :

```
for(i = 0; i < 5; i++) {
  if(write(fd, &i, sizeof(int)) == -1) {
    perror("Erreur lors de l'écriture d'un entier dans le tube ");
    exit(EXIT_FAILURE);
  }
}
```

Le serveur peut ensuite fermer le tube et le supprimer :

```
if(close(fd) == -1) {
  perror("Erreur lors de la fermeture du tube ");
  exit(EXIT_FAILURE);
}

if(unlink(NOM_TUBE) == -1) {
  if(errno != ENOENT) {
    fprintf(stderr, "Erreur lors de la suppression du tube '%s'",
       NOM_TUBE);
    perror(" ");
    exit(EXIT_FAILURE);
  }
}
```

2 Le client

Le client doit ouvrir le tube en lecture :

```
int fd;
if((fd = open(NOM_TUBE, O_RDONLY)) == -1) {
  perror("Erreur lors de l'ouverture du tube ");
  exit(EXIT_FAILURE);
}
```

Il lit ensuite les 5 entiers envoyés par le serveur (ici, nous utilisons un tableau pour lire les 5 entiers en un seul appel à `write`) :

```
int tab[5];
if(read(fd, tab, sizeof(int) * 5) == -1) {
  perror("Erreur lors de la lecture de 5 entiers dans le tube ");
  exit(EXIT_FAILURE);
}
```

Il peut ensuite fermer le tube avant de s'arrêter :

```
if(close(fd) == -1) {
  perror("Erreur lors de la fermeture du tube ");
  exit(EXIT_FAILURE);
}
```

3 Compilation et exécution

Le `makefile` fourni permet de compiler les programmes précédents. Tapez la commande suivante :

```
make
```

Pour exécuter les programmes, tapez l'une des commandes suivantes :

```
./serveur &
./client
```

Vous pouvez également utiliser deux terminaux différents pour éviter d'exécuter le programme serveur en tâche de fond.