Systèmes d'Exploitation

L3 Informatique & L3 MI-Info

TP 3 - Communication interprocessus

Léa Brunschwig <lea.brunschwig@univ-pau.fr>

Le but de ce TP est de se familiariser avec l'utilisation des appels systèmes de base pour la communication interprocessus.

Objectifs :

- Apprendre à utiliser la librairie shm.h pour les espaces de données communs
- Comprendre l'utilisation des signaux
- Comprendre les tubes de communication nommés

Espace de données communs

Nous souhaitons mettre en place une communication interprocessus en utilisant la librairie `shm.h`. Pour cela, vous allez créer deux processus, un processus parent et un processus enfant. Le processus parent va créer une zone de mémoire partagée et y écrire une chaîne de caractères. Le processus enfant va accéder à cette zone de mémoire partagée et lire la chaîne de caractères.

Pour réaliser ce scénario, vous devez mettre en place les étapes suivantes :

1. Créer un processus enfant à partir du processus parent à l'aide de la fonction `fork()`.

2. Dans le processus parent, créer une zone de mémoire partagée à l'aide de la fonction `shmget()`.
3. Attacher cette zone de mémoire partagée à l'espace d'adressage du processus parent à l'aide de la fonction `shmat()`.
4. Écrire une chaîne de caractères dans la zone de mémoire partagée.
5. Dans le processus enfant, attacher la zone de mémoire partagée à l'espace d'adressage du processus enfant à l'aide de la fonction `shmat()`.
6. Lire la chaîne de caractères de la zone de mémoire partagée dans le processus enfant.
7. Afficher la chaîne de caractères dans le processus enfant.

Les signaux

Vous allez créer deux processus qui communiquent en utilisant des signaux. Le processus parent envoie un signal au processus enfant pour l'informer de la fin de son exécution.

1. Écrire un programme en C qui crée deux processus : un processus parent et un processus enfant.
2. Dans le processus parent, créer un handler pour le signal `SIGUSR1` qui affiche "Signal reçu" sur la sortie standard.
3. Dans le processus parent, envoyer le signal `SIGUSR1` au processus enfant en utilisant la fonction `kill`.
4. Dans le processus enfant, créer un handler pour le signal `SIGUSR1` qui affiche "Signal reçu" sur la sortie standard.
5. Attendre que le processus enfant reçoive le signal `SIGUSR1` en utilisant la fonction `pause`.
6. Dans le processus parent, attendre que le processus enfant se termine en utilisant la fonction `wait`.
7. Compiler et exécuter le programme en vérifiant que le message "Signal reçu" est bien affiché.

Les tubes de communication nommés

Vous allez implémenter l'exemple du Producteur/Consommateur vu en cours en implémentant un système de communication interprocessus utilisant des tubes de communication nommé.

Programme du Producteur :

1. Créer un tube de communication nommé, en utilisant la fonction `mkfifo()`. Ce tube sera utilisé pour la communication entre le processus producteur et le processus consommateur.
2. Écrire une boucle infinie qui lit une chaîne de caractères depuis l'entrée standard, et écrit cette chaîne de caractères dans le tube de communication nommé. Pour cela, utiliser la fonction `open()` pour ouvrir le tube en écriture, puis la fonction `write()` pour écrire dans le tube.

Programme du Consommateur :

3. Écrire un deuxième programme C qui lit les chaînes de caractères écrites par le processus producteur dans le tube de communication nommé. Pour cela, utiliser la fonction `open()` pour ouvrir le tube en lecture, puis la fonction `read()` pour lire les données.
4. Afficher les chaînes de caractères lues sur la sortie standard.