

Projet Unix (interprocessus)

1- Introduction

Les processus coopèrent souvent pour traiter un même problème. Ces processus s'exécutent en parallèle sur un même ordinateur (monoprocasseur ou multiprocesseurs) ou bien sur des ordinateurs différents. Ils doivent alors s'échanger des informations (communication interprocessus). Il existe plusieurs moyens de communication interprocessus. Nous pouvons citer, entre autres, les variables communes, les fichiers communs, les signaux, les messages et les tubes de communication.

2- Présentation du projet

Le projet consiste à mettre en place une communication (serveur-clients) via des tubes nommés. L'idée principale se manifeste dans :

- Création d'un processus démon qui est le serveur, qui écoutera les requêtes provenant des clients.
- Créer les processus clients qui ont besoin de manipuler les données fournies par les services offerts par le serveur.
- Un client a besoin d'afficher N nombres aléatoires. Ce nombre N doit être inférieur à N_{MAX} et supérieur à N_{MIN} .
- Le serveur fournit un service qui génère N nombres aléatoires entre N_{MIN} et N_{MAX} .
- Le serveur doit toujours être opérationnel, à l'écoute des demandes des clients

3- Fonctionnement de Serveur

- **Création de variables** : le but de ces variables est de gérer les demandes des clients, les réponses générées par les serveurs et autres sont des indicateurs pour vérifier si une opération échoue tout au long du processus.

- **Création des pipes nommées** : le serveur crée les FIFO nécessaires pour communiquer avec les clients. FIFO1 est la pipe que le serveur utilise pour lire les données. FIFO2 est la pipe pour écrire des données
- **Génération des nombres aléatoires** : cette tâche est essentielle pour utiliser la fonction rand().
- **Gestion des handlers** : lorsque l'utilisateur reçoit une réponse, le serveur écouterait le signal SIGUSR1, tous les autres signaux seront ignorés, le comportement par défaut du serveur est d'arrêter et de supprimer le FIFO créé.
- **Traitement du serveur** : le serveur va lire la requête entrante dans FIFO1. Le serveur générera alors une réponse qui sera envoyée en utilisant FIFO 2, et enfin le client recevra un signal SIGUSR1, il sera donc réveillé par le serveur pour continuer à faire son travail avec les données de la réponse.

4- Fonctionnement de Client

- **Création des variables** : le but de ces variables est de gérer les demandes des clients, les réponses générées par les serveurs et autres sont des indicateurs pour vérifier si une opération échoue tout au long du processus. Sauf pour les données variables qui contiennent temporairement la valeur N.
- **Ouverture des pipes** : Les canaux Fifo1 et Fifo2 sont utilisés pour écrire des requêtes et lire des réponses du serveur, respectivement.
- **Génération des nombres aléatoires** : Initialiser le générateur de nombres aléatoires : cette tâche est essentielle pour une utilisation ultérieure de la fonction rand().
- **Construction de requête** : L'objet de requête contient des données, à savoir un nombre N généré aléatoirement et l'ID de processus du client, afin que le serveur sache quel processus recevra le signal SIGUSR1.
- **Communication interprocessus** : Le client envoie l'objet requête au serveur. Il appelle ensuite pause pour dormir jusqu'à ce qu'une réponse générée par le serveur arrive. Tout d'abord, le client écrira dans fifo1, puis lira dans fifo2.
- **Traitement de la réponse** : afficher le résultat obtenu du serveur

5- Architecture de projet

La structure des dossiers peut tout dire sur l'architecture de l'application.

- ✓ Le dossier partagé contient le fichier d'en-tête "serv_cli_fifo.h" qui contient les variables, la représentation des structures de données et les constantes.
- ✓ Le fichier Serveur.c contient toute la logique et les traitements associée au serveur
- ✓ Le fichier Client.c contient toute la logique et les traitements associée au client
- ✓ Les fichiers Handler_cli.h et Handler_serv.h contiennent les handlers des signaux pour assurer le réveil de client et serveur et pour l'arrêt de serveur.

6- Les imports et les structures des données

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #define NMIN 1
4  #define NMAX 20
5  #define fifo1 "fifo1"
6  #define fifo2 "fifo2"
7
8  typedef struct question {
9      int cpid;
10     int number;
11 } question;
12
13 typedef struct reponse {
14     int spid;
15     int result[NMAX];
16 } reponse ;
17
18
19
```

```
1  #include "serv_cli_fifo.h"
2  #include "Handlers_serv.h"
3  #include <signal.h>
4  #include <sys/types.h>
5  #include <sys/stat.h>
6  #include <unistd.h>
7  #include <fcntl.h>
8
```

- **Request** : elle contient la donnée qui est le nombre N généré aléatoirement entre NMIN et NMAX. Ensuite, le champ cpid est l'ID du processus client afin que le serveur sache quel processus va être réveillé avec le signal SIGUSR1.
- **Response** : elle contient principalement le champ spid qui est l'identifiant du processus serveur afin que le client puisse notifier le serveur en cas de réception de réponse. Puis le champ size qui est le nombre de nombres générés aléatoirement. Et enfin les données de taille NMAX c'est la charge utile que le serveur va générer que l'utilisateur va afficher.

7- Processus de l'exécution

- 1- Positionnez-vous à la racine du projet.
- 2- Exécutez la commande « gcc » pour générer les exécutables serveur et client.
- 3- Ouvrir deux instances du terminal
- 4- Lancez la commande « ./server » pour **lancer le serveur**
- 5- Exécutez la commande « ./client » pour exécuter le client

8- Captures

- ✓ Génération d'un fichier Objet de Client

```
(osboxes@osboxes)-[~/Desktop/LosBlancos]  
$ gcc Serveur.c serv cli fifo.h Handlers serv.h -o server
```

- ✓ Génération d'un fichier Objet de Client

```
(osboxes@osboxes)-[~/Desktop/LosBlancos]  
$ gcc Client.c serv cli fifo.h Handlers cli.h -o client
```

- ✓ Lancement de serveur : Le serveur dans un état d'attente pour d'éventuels clients

```
(osboxes@osboxes)-[~/Desktop/LosBlancos]  
$ ./server  
Welcome to server  
Welcome to server  
PID of Client → 452032  
Choosing Number → 1  
PID of Server → 452031
```

- ✓ Lancement de Client : Le Client envoie un nombre aléatoire au serveur

```
(osboxes@osboxes)-[~/Desktop/LosBlancos]
$ ./client

_____ Client Activated _____

PID of Client -----> 452032
Choosing Number -----> 1

_____ Client Wake up _____
```

- ✓ Le serveur traite la demande du client : le serveur effectue le traitement et l'envoi pour le client

```
_____ Welcome to server _____

Handlers ch n
PID of Client -----> 452032
Choosing Number -----> 1
PID of Server -----> 452021
Random Number generated -----> 275

_____ All Sent ! ^_^ _____
```

- ✓ Le Client reçoit le traitement de serveur.

```
_____ All Received ! ^_^ _____

PID of Server -----> 452021
Random Number generated -----> 275
```

- ✓ Le serveur vérifie les données de réception et reste en attente d'un autre client

```
_____ Bingo ! All Received _____
--
_____ Another One ? _____
```

- ✓ L'enchaînement de serveur du début à la fin

```

Client :
Another One ?
PID of Client  -> 452057
Choosing Number -> 5
PID of Server  -> 452021
Random Number generated -> 684 163 489 8
112
99
All Sent ! ^_^
Bingo ! All Received
--

```

✓ L'enchaînement de client du début à la fin

```

(osboxes@osboxes)-[~/Desktop/LosBlancos]
$ ./client

Client Activated

PID of Client  -> 452057
Choosing Number -> 5

Client Wake up

All Received ! ^_^

PID of Server  -> 452021
Random Number generated -> 684 163 489 899 112

```

✓ L'enchaînement de serveur du début à la fin (pour le cas de nombre aléatoire 19)

```

_____ Bingo ! All Received _____
--
_____ Another One ? _____
_____
PID of Client -----> 452079
Choosing Number -----> 19
PID of Server -----> 452021
Random Number generated -----> 342 72 318 6
37 416 212 571 284 393 206 89 520 983 5
08 597 429 432 487 918
_____ All Sent ! ^_^ _____
_____ Bingo ! All Received _____
--
_____ Another One ? _____
_____
PID of Client -----> 452169

```

✓ L'enchaînement de client du début à la fin (pour le cas de nombre aléatoire 19)

```

(osboxes@osboxes)-[~/Desktop/LosBlancos]
$ ./client

_____ Client Activated _____

PID of Client -----> 452079
Choosing Number -----> 19

_____ Client Wake up _____

_____ All Received ! ^_^ _____

PID of Server -----> 452021
Random Number generated -----> 342 72 318 637 416 2
12 571 284 393 206 89 520 983 508 597 4
29 432 487 918

```

✓ Enfin, si le serveur reçoit un signal quelconque, il arrête le processus de programme.

```
_____ All Sent ! ^_^ _____  
_____  
-- Bingo ! All Received _____  
_____  
^C Another One ? _____  
_____  
Good Bye ! _____
```