Projet Unix (interprocessus)

1- Introduction

Les processus coopèrent souvent pour traiter un même problème. Ces processus s'exécutent en parallèle sur un même ordinateur (monoprocesseur 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 NMAX et supérieur à NMIN.
- Le serveur fournit un service qui génère N nombres aléatoires entre NMIN et NMAX.
- 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 écoutera 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 ficher Client.c contient toute la logique et les traitements associée au client
- ✓ Les fichiers Handler_cli.h et Handler_serv.h contient 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

```
#include <stdio.h>
#include <stdlib.h>
#define NMIN 1
#define NMAX 20
#define fifol "fifol"
#define fifo2 "fifo2"
typedef struct question {
    int cpid;
    int number;
} question;
typedef struct reponse {
                                     #include "serv_cli_fifo.h"
    int spid;
    int result[NMAX];
                                     #include <signal.h>
} reponse ;
                                     #include <sys/types.h>
                                     #include <sys/stat.h>
                                     #include <unistd.h>
                                     #include <fcntl.h>
```

- ➤ 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 -0 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 → 452021
```

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

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

```
        PID of Client — → 452032

        Choosing Number — → 1

        PID of Server — → 452021

        Random Number generated — → 275

        — All Sent ! ^_^ — — —
```

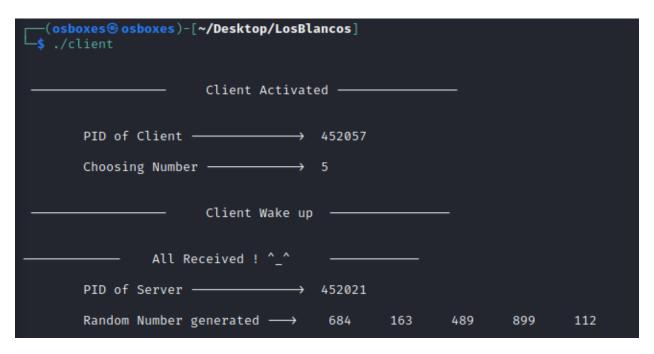
✓ Le Client reçoit le traitement de serveur.

✓ 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

✓ L'enchaînement de client du début à la fin



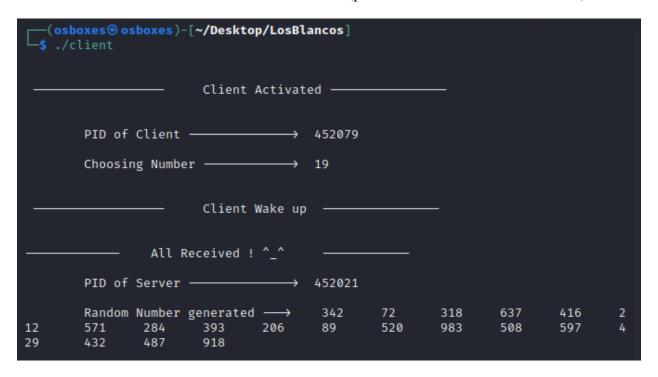
✓ L'enchaînement de serveur du début à la fin (pour le cas de nombre aléatoire 19)

```
Bingo ! All Received
                                Another One ?
        PID of Client --

→ 452079

       Choosing Number -
        PID of Server -
                                   → 452021
       Random Number generated -
                                                        342
                                                                72
                                                                        318
                                                                                6
                                                206
        416
                212
                       571
                                284
                                        393
                                                        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)



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

	All Sent ! ^_^		
	_	Bingo ! All Received	
^c	—	Another One ?	
		Good Bye !	