

Mini-projet client-serveur - Unix

Frédéric Rousseau

2024 – 2025



Mini-projet client-serveur

Sujet

- ▶ Écrire un programme client qui affiche en temps réel les données numériques générées par un programme serveur

Prérequis

- ▶ Programmation système sous Unix (CM, TD)
- ▶ Algorithmique et Programmation (S5, S6)

Les séances

- ▶ 3 x 2h les 26 nov., 3 et 10 déc. matin, salle 216

Au moins autant de travail personnel

- ▶ Projet à réaliser en individuel, ou par binôme
- ▶ Encadrement (et évaluation) : Frédéric Rousseau

Evaluation Unix IESE4

Examen écrit : 50%

- ▶ En fin de module, examen écrit 2h, 1 feuille A4 recto-verso
manuscrite

Mini-projet : 50%

- ▶ Rendu en fin de chaque séance
- ▶ Pas de rapport, pas de soutenance
- ▶ Evaluation sur les objectifs (ou le dépassement !) de programmation et d'exécution
- ▶ Evaluation individuel possible pour un binôme

Objectifs du projet

Mise en application des fonctionnalités Unix suivantes :

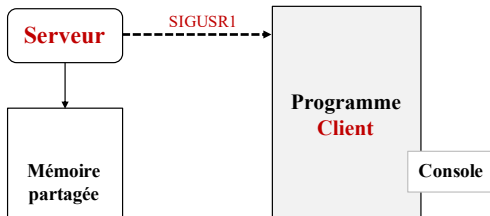
- ▶ Application multi-processus : `fork()`
- ▶ Synchronisation multiprocessus : sémaphores, signaux
- ▶ Communication multiprocessus : pipe, mémoire partagée
- ▶ Programmation multi fichiers : Makefile

Principe du projet :

- ▶ Le code source du serveur est donné
- ▶ Vous devez écrire le programme client qui respecte un cahier des charges précis
- ▶ Vous devez à *chaque séance rendre une version de votre travail*

Description générale du serveur

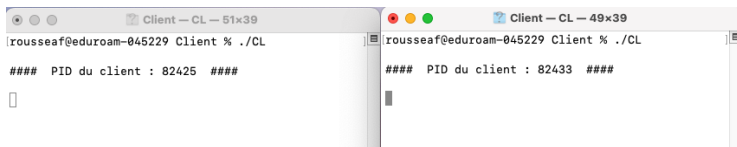
- ▶ Chaque seconde, le serveur écrit une donnée dans une mémoire partagée
- ▶ Quand la donnée est produite, il envoie un signal SIGUSR1 à tous les clients connus
- ▶ Les clients récupèrent la donnée dans la mémoire partagée et l'affiche dans leur terminal
- ▶ Après un nombre donné de cycles, le serveur envoie un signal SIGUSR2 de terminaison à tous les clients connus



Connexion client - serveur

Les étapes de démarrage :

- ▶ 1 : On lance les clients dans les terminaux. Chaque client doit afficher son PID



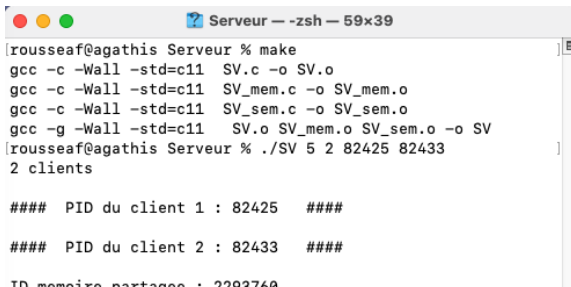
```
Client - CL - 51x39
rousseaf@eduroam-045229 Client % ./CL

#### PID du client : 82425 ####

Client - CL - 49x39
rousseaf@eduroam-045229 Client % ./CL

#### PID du client : 82433 ####
```

- ▶ 2 : On lance dans un autre terminal le serveur, en passant en paramètre dans la ligne de commande la durée de fonctionnement du serveur, le nombre de clients, le PID de tous les clients.



```
Serveur - -zsh - 59x39
rousseaf@agathis Serveur % make
gcc -c -Wall -std=c11 SV.c -o SV.o
gcc -c -Wall -std=c11 SV_mem.c -o SV_mem.o
gcc -c -Wall -std=c11 SV_sem.c -o SV_sem.o
gcc -g -Wall -std=c11 SV.o SV_mem.o SV_sem.o -o SV
rousseaf@agathis Serveur % ./SV 5 2 82425 82433
2 clients

#### PID du client 1 : 82425 ####
#### PID du client 2 : 82433 ####
```

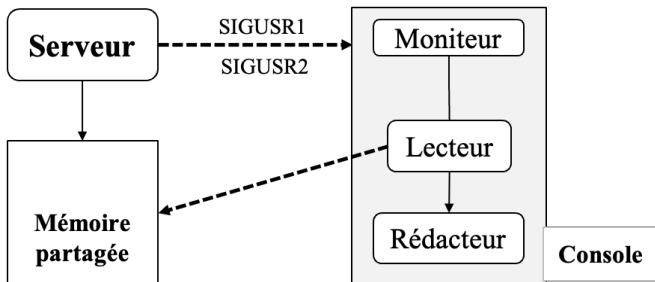
Exemple d'exécution client - serveur

- ▶ Terminaison provoquée par l'envoi de SIGUSR2 vers tous les clients

Serveur — -zsh — 53x39	Client — -zsh — 43x39	Client — -zsh — 51x39
<pre>rousseaf@agathis Serveur % make gcc -c -Wall -std=c11 SV.c -o SV.o gcc -c -Wall -std=c11 SV_mem.c -o SV_mem.o gcc -c -Wall -std=c11 SV_sem.c -o SV_sem.o gcc -g -Wall -std=c11 SV.o SV_mem.o SV_sem.o -o SV rousseaf@agathis Serveur % ./SV 5 2 81869 81873 2 clients #### PID du client 1 : 81869 #### #### PID du client 2 : 81873 #### ID memoire partagee : 2228224 Emission des données vers 2 client(s) - PID du client 1 : 81869 #### - PID du client 2 : 81873 #### Terminaison - envoi dun signal de fin aux clients rousseaf@agathis Serveur % █</pre>	<pre>rousseaf@eduroam-045229 Client % ./CL #### PID du client : 81873 #### ID memoire partagee : 2228224 Creation du fil lecteur ! Debut de reception des donnees Creation du fil redacteur ! valeur : 100 reçu a la date Thu Oct 19 19:34:59 2023 valeur : 99 reçu a la date Thu Oct 19 19:35:00 2023 valeur : 98 reçu a la date Thu Oct 19 19:35:01 2023 valeur : 97 reçu a la date Thu Oct 19 19:35:02 2023 valeur : 96 reçu a la date Thu Oct 19 19:35:03 2023 Terminaison du clients rousseaf@eduroam-045229 Client % █</pre>	<pre>rousseaf@eduroam-045229 Client % ./CL #### PID du client : 81869 #### ID memoire partagee : 2228224 Debut de reception des donnees Creation du fil lecteur ! Creation du fil redacteur ! valeur : 100 reçu a la date Thu Oct 19 19:34:59 2023 valeur : 99 reçu a la date Thu Oct 19 19:35:00 2023 valeur : 98 reçu a la date Thu Oct 19 19:35:01 2023 valeur : 97 reçu a la date Thu Oct 19 19:35:02 2023 valeur : 96 reçu a la date Thu Oct 19 19:35:03 2023 Terminaison du clients rousseaf@eduroam-045229 Client % █</pre>

Détail de l'architecture du client

- Le client est composé de 3 processus : moniteur, lecteur et rédacteur



Décomposition du développement

- ▶ Pour préparer
 - ▶ 0 : Comprendre le code du serveur (précisément ...)
- ▶ Attendus fin de séance 1
 - ▶ 1 : 1 processus *moniteur*, à la réception de SIGUSR1, affiche un message, et à la réception de SIGUSR2 se termine
 - ▶ 1bis : Le processus *moniteur* affiche la valeur lue dans la mémoire partagée
 - ▶ Mise en oeuvre : gestion des signaux systemV, accès à la mémoire partagée
- ▶ Attendus fin de séance 2
 - ▶ 2 : Le processus *moniteur* crée un processus *lecteur*. A la réception de SIGUSR1, le *moniteur* indique au *lecteur* qu'il peut aller lire la valeur produite dans la mémoire partagée et l'afficher
 - ▶ Mise en oeuvre : programmation multiprocessus, synchronisation entre processus
- ▶ Attendus fin de séance 3
 - ▶ 3 : Le processus *moniteur* crée un processus *rédacteur*. Le processus *lecteur* envoie au *rédacteur* la valeur produite par le serveur. Le *rédacteur* l'affiche en indiquant la date de réception.
 - ▶ 3bis : On protège l'accès à la mémoire partagée par un mutex
 - ▶ Mise en oeuvre : programmation multiprocessus, communication entre processus

Les questions à se poser

- ▶ Comment le serveur crée la mémoire partagée ?
- ▶ Comment le serveur crée le mutex de protection de la mémoire partagée ?
- ▶ Que doit faire le client à la réception de SIGUSR1 ?
- ▶ Que doit faire le client à la réception de SIGUSR2 ?
- ▶ Quels mécanismes de communications et de synchronisation utiliser dans le client ? Entre quels processus ?
- ▶ Dans le programme client, à quels moments dois-je créer ces IPC ?
- ▶ Comment terminer proprement le client à la réception de SIGUSR2 ?

Décomposition des étapes du projet

- ▶ Présentation du projet (1h)
- ▶ Travail personnel : étude du code du serveur - réflexion sur l'architecture du projet
- ▶ Réponse aux questions (1h) sur l'architecture client ou serveur
- ▶ 3 séances de 2h
 - ▶ Séance 1 : Création du *moniteur*. Test avec le serveur. Le client se termine à la réception du signal SIGUSR2.
 - ▶ Séance 2 : Le *moniteur* crée un processus *lecteur*. C'est le lecteur qui récupère la donnée et l'affiche.
 - ▶ Séance 3 : Le *moniteur* crée un processus *redacteur*. Le *lecteur* récupère la donnée et l'envoie au *redacteur*. C'est le *redacteur* qui affiche le résultat.
 - ▶ Autres fonctionnalités possibles :
 - ▶ En cas de CTRL-C sur le client, on arrête proprement le client
 - ▶ Le rédacteur affiche les données par groupe de 3, 4 ou 5 données
 - ▶ La date de réception de la donnée par le textitredacteur est ajouté au texte affiché.
 - ▶ ...