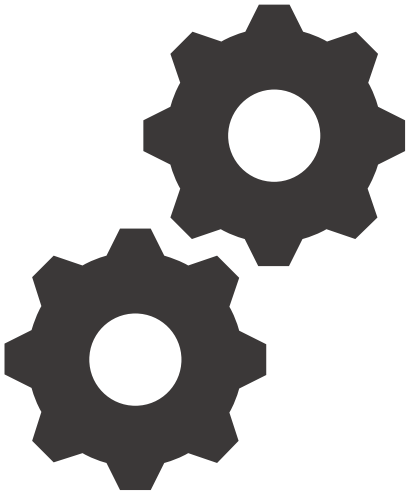
NESI TECH

MINI PROJET : Sujet 2 - Server Q&A

Dossier d'architecture technique



lien : https://alexandredemeo.github.io/ProjetOS/

ISEN P17 Marseille - HOUGUET Nathan – VALETTE Guillaume – DE MEO Alexandre – GUILLARD Dorian

I. Le Modèle client-serveur

Le client-serveur représente un dialogue entre deux processus informatiques par l'intermédiaire d'un échange de messages.

Le modèle client-serveur repose sur une communication d'égal à égal entre les applications, la communication étant réalisée par dialogue entre processus deux à deux.

- Un processus client
- Un processus serveur

Les processus ne sont pas identiques mais forment un système coopératif se traduisant par un échange de données.

Rôle du Client : Le client effectue une demande de service auprès du serveur (requête). De plus il initie le contact, ouvre la session.

Rôle du serveur : Le serveur est la partie de l'application qui offre un service. Le serveur est à l'écoute des requêtes clientes, il répond au service demandé par le client. Un programme serveur est en permanence en attente de requêtes et ne peut répondre qu'à un client en même temps.
Le Client et le serveur sont généralement localisés sur deux machines distinctes.

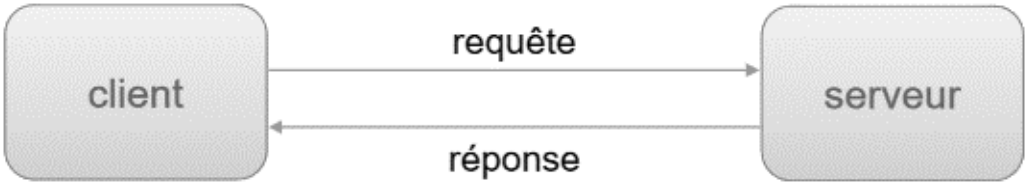


Figure 1 : Le modèle client/serveur

Comme déjà évoqué, celles-ci communiquent par messages : **les requêtes** (paramètres d'appel, spécification du service requis) et **les réponses** (résultats, indicateur éventuel d'exécution ou d'erreur).

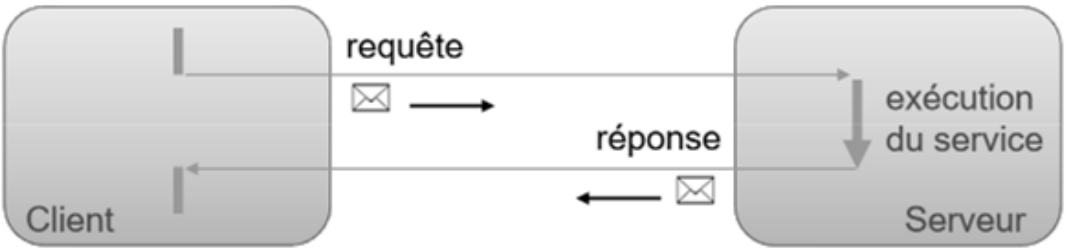


Figure 2 : Représentation de la communication client/serveur

II. Itération 1 : Fonctionnement avec un unique client.

Lien web : https://alexandredemeo.github.io/ProjetOS/41_progsys_tp1_proc/

1. Cahier des charges :

- Un processus server active un client sur une autre console
- Les 2 processus se parlent via un **pipe nommé** dont le nom est passé en paramètre au fils
- Le processus serveur est en écoute sur le pipe
- Le processus client lit des caractères tapés au clavier et les écrit dans le pipe
- Quand le combo **control + D (EOF)** est tapé au clavier, le client se termine
- Le processus serveur ne se termine pas, il est toujours en attente d'un client éventuel
- On peut relancer un client manuellement en lui donnant le nom du pipe et le logiciel fonctionne de nouveau.
- Le processus serveur affiche les données reçues précédées du PID du client.

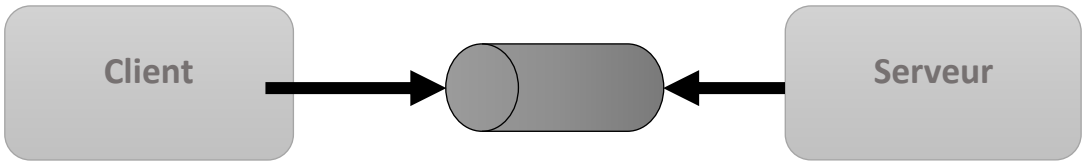
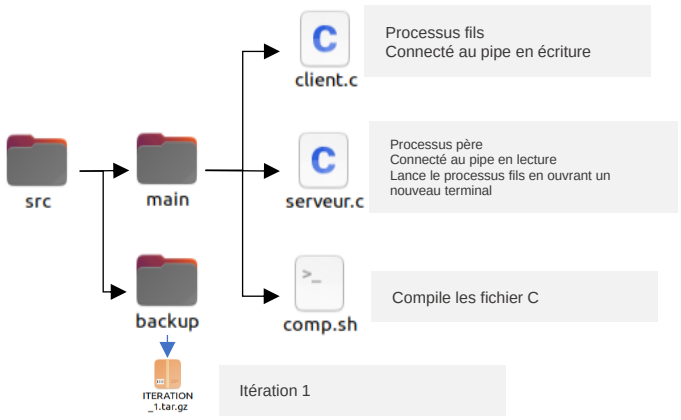


Figure 3 : Représentation de la communication client/serveur

1.2 Fonctionnement :

L'itération 1 met en place une communication entre le serveur et un unique client.
Le fichier correspondant à cette itération se trouvera dans le dossier Backup > (cf cheminement ci-dessous).



Organisation

Les processus père (serveur) et fils (client) communiquent comme souhaité via un pipe nommé dans ce projet « **pipe** ».
Après compilation et exécution du fichier **serveur.c**, la communication est fonctionnelle :

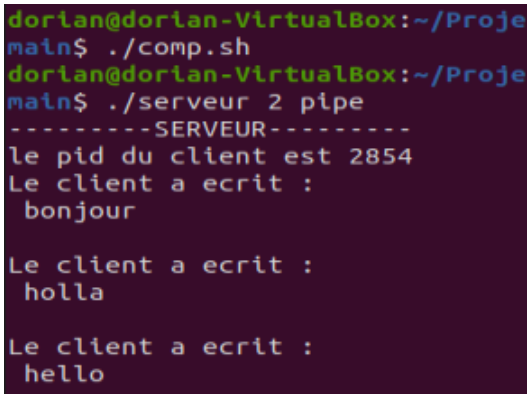


Figure 4 : Terminal serveur IT1

Figure 5 : Terminal client IT1

Un client est appelé via un nouveau terminal de commande.
La fermeture du client s'effectue grâce au combo **CTRL + D** (message ctrld tracé sur le terminal serveur).

DETAILS TECHNIQUES ??

III. Itération 2 : Fonctionnement multi-client.

Lien web : https://alexandredemeo.github.io/ProjetOS/40_progsys_signals/

2.1 Cahier des charges :

- Le programme server prends maintenant **2 paramètres**
    - Le nom du pipe
    - Le nombre N de clients
- Au démarrage il instancie N clients qui peuvent lui parler (**1 client = 1 shell**)
- Des clients peuvent être crées manuellement en plus par la suite
- On protège la mort du server par CTRL+C
- Le serveur sait répondre à des questions basiques des clients (fixées) :

```
- Q : Qui suis-je ? -> R : Ecrit le PID du demandeur
- Q : Qui es tu ? -> R : Je suis le serveur de PID 12234
- Q : Qui est la ? -> R : liste des clients
- ...
```

2.2 Fonctionnement

L'itération 2 met en place une communication entre le serveur et plusieurs clients.
Les fichier correspondant à cette itération se trouvera dans le dossier main.

Les processus père (serveur) et fils (client) communiquent comme précédemment via un pipe nommé « **pipe** ».
Après compilation et exécution du fichier **serveur.c**, la communication est fonctionnelle.
Afin d'instancier un nombre N de clients souhaité, l'exécution se fera grâce à la commande suivante : **./serveur X pipe** (la variable X représentant le nombre de client voulu).

```
dorian@dorian-VirtualBox:~/ProjetOS/ServeurQA/src/main$ ./comp.sh
dorian@dorian-VirtualBox:~/ProjetOS/ServeurQA/src/main$ ./serveur 3 pipe
---------SERVEUR---------
Le client 5817 a ecrit :
 hello

Le client 5818 a ecrit :
 bonjour

Le client 5819 a ecrit :
 salut
```

Figure 6 : Terminal serveur IT2

```
client
voici mon pid : 5817
Saisir la commande a envoyer au serveur :hello
Saisir la commande a envoyer au serveur :
```

Figure 7 : Terminal client n°1 IT2

```
client
voici mon pid : 5818
Saisir la commande a envoyer au serveur :bonjour
Saisir la commande a envoyer au serveur :
```

Figure 8 : Terminal client n°2 IT2

```
client
voici mon pid : 5819
Saisir la commande a envoyer au serveur :salut
Saisir la commande a envoyer au serveur :
```

Figure 9 : Terminal client n°3 IT2

DETAILS TECHNIQUES ??