Bruno Cornil, Matricule : 000392857

Rapport INFO-F-201

Introduction

Le projet contient deux fichiers contenant respectivement le code du client (client.c) et du serveur (server.c). Le fichier server.h contient la définition du port utilisé par le serveur et le client et le fichier Makefile les règles pour construire les deux applications.

Description des programmes

L'implémentation du client est minimaliste; il ouvre une connexion vers le serveur, affiche tout ce qu'il reçoit du *socket* sur la console et renvoie sur le *socket* les chaines de caractères que l'utilisateur introduit par la console.

L'implémentation du serveur est basée sur l'utilisation de la commande '*select*'. Une fois un *socket* ouvert sur le port 5555, '*select*' est utilisé pour signaler de l'activité sur les ports d'entrées du serveur et des clients déjà connectés.

Une activité sur le *socket* du serveur (demande de connexion) initialise un nouveau client (les clients sont maintenus dans une liste chainée '*clientNode*' comme illustré à la figure 1 ci-dessous) tandis que de l'activité sur un *socket* client (données en provenance du client disponibles) appelle la fonction 'clientProcess' avec comme paramètre un pointeur vers le 'clientNode' correspondant.

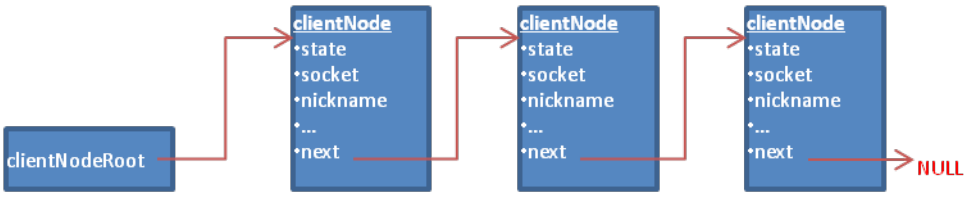

Figure 1 : Liste chainée de client

'clientProcess' implémente un machine d'état qui réagit aux données communiquées par le client (lecture garantie non bloquante grâce au '*select*') et renvoie au client ce qui doit s'afficher sur son écran en fonction des données fournies et de l'état du client

Sans activité en provenance d'un des *sockets* (timeout du select), on exécute une procédure de nettoyage de la liste (fermeture du socket et libération de la mémoire allouée pour la structure '*clientNode*') pour tous les clients qui sont dans l'état 'STATE_END' (conséquence d'un q(uit) de l'utilisateur ou, par exemple, d'une erreur de transmission/réception sur un socket).

Instructions

Construire les deux applications 'client' et 'server'

```
> make
```

Exécution du serveur de message (pas de paramètres)

```
> ./server
```

Exécution du client

```
> ./client <nom ou adresse IP du serveur>
```

Exemple de session (3 utilisateurs connectés)

Serveur

```
student@padi-vm ~/Documents/messageServer $ ./server
Socket created
Socket bound to port 5555
Listening for clients
New connection with 127.0.0.1 (port 33294)
New connection with 127.0.0.1 (port 33295)
New connection with 127.0.0.1 (port 33296)
Free resources for socket  6
Free resources for socket  5
Free resources for socket  4
```

Client 1

```
student@padi-vm ~/Documents/messageServer $ ./client localhost
*** INFO-F201 Message server ***
Enter your nickname (max. 10 char.): bruno

Welcome bruno !

Available commands
h- This help
l- List connected users
m <user> - Send a message to <user>
q- Quit

>> l
List of users: tester2, tester1
>> m tester2
Message to: tester2
Hello tester 2
>>
Message from:tester1
Good to see you Bruno
>>
Message from:tester2
Hello Bruno !
>> q
Bye !
```

Client 2

```
student@padi-vm ~/Documents/messageServer $ ./client localhost

*** INFO-F201 Message server ***
Enter your nickname (max. 10 char.): tester1

Welcome tester1 !

Available commands
h- This help
l- List connected users
m <user> - Send a message to <user>
q- Quit

>> l
List of users: tester2, bruno
>> m bruno
Message to: bruno
Good to see you Bruno
>> q
Bye !
```

Client 3

```
student@padi-vm ~/Documents/messageServer $ ./client localhost
*** INFO-F201 Message server ***
Enter your nickname (max. 10 char.): tester2

Welcome tester2 !

Available commands
h- This help
l- List connected users
m <user> - Send a message to <user>
q- Quit

>>
Message from:Bruno
Hello tester 2
>> l
List of users: tester1, Bruno
>> m Bruno
Message to: Bruno

Hello Bruno !

>> q
Bye !
```