PERONNET Pierre
GLEVAREC Guillaume
FEUGUEUR Yann

Projet de Système

Rapport.

Introduction

Le projet consiste à monter un mini-twitter. Pour cela on doit monter un serveur en mode démon sur un port. Une fois le serveur lancé on doit établir la communication avec un ou des clients. Chaque commande qui échoue doit comprendre un message d'erreur. Par ailleurs le serveur peut être éteint.
Les clients peuvent recevoir des messages sur la sortie standard les réponses du serveurs et donc des autres utilisateurs. Chaque utilisateur possédera un login et un identificateur le premier donné par l'utilisateur lui même et le second par le serveur. Le serveur devra répondre aux commandes du client.

Contenu

Le projet est organisé comme il était demandé.

On va maintenant les modules (.c et .h) et leurs principales fonctions.

client

Le module client est un module indépendant, des autres sources, c'est d'ailleurs à ce titre que le client est compilé et exécuté indépendamment un nombre de fois nécessaire pour chaque client qui souhaite se connecter au serveur. Grâce à ce module, nous créons une socket que nous tentons de connecter au serveur par l'intermédiaire du port passé en paramètre.

Si le port ne supporte aucun serveur ou que le serveur a atteint un nombre d'utilisateurs maximal connectés simultanément, la connexion est refusée ce qui entraîne la fermeture du client.

En revanche, si la connexion est acceptée par le serveur, nous obtenons une socket descriptor de la part du serveur, qui nous servira à communiquer avec celui-ci. Le client se place alors en lecture pour recevoir des données provenant du serveur, ou en écriture, si le client souhaite transmettre des des données au serveur.

Pour l'exécution du client, il est nécessaire d'avoir une initialisation de la socket

struct sockaddr_in init_socket(int port, char name) ;*

Puis, la connexion et l'écoute/écriture au serveur

int connect_socket(int port, char name) ;*

Pour ce module on gère la présence des followers. De leur création à leur suppression. Leur initialisation ainsi que leur affiche.

follow

Le module follow, constitue l'ensemble de liste chaînée et de son utilisation. Cette liste chaînée nous permet de représenter les « followings » c'est à dire, constituer une « mémoire » des socket descriptors (représenté par ID dans notre structure) des clients connectés au serveur dont on veut pouvoir suivre les tweets en instantané.

Structure de la liste chaînée.

```
typedef struct fol{
        unsigned short ID;
        struct fol *next;
}Follow;
```

Fonction d'allocation d'un nouveau socket descriptor (ID) dans pour une liste.
Follow allocate_follow(unsigned short ID);*

Fonction permettant de libérer une liste. C'est grâce à cette fonction qu'on peut supprimer l'intégralité des followings d'un client, lorsqu'il fait usage de la commande « CLE» ou dans le cas où l'utilisateur ferme son application.
*void free_all_follow(Follow **follow);*

Cette fonction permet de ne supprimer qu'un seul following de notre liste qui est référencé par l'ID passé en paramètre. Il s'agit simplement d'identifier la donnée visée, puis, de rechaîner la liste après la suppression.
*void free_a_follow(Follow **follow, unsigned short ID);*

La recherche d'un following se fait grâce à cette fonction, qui prend en paramètre l'ID recherché, et qui, si la recherche porte ses fruits, renvoi la macro TRUE (=1), et FALSE (=0) sinon
*int search_follow(Follow *follow, unsigned short ID);*

L'insertion d'un nouveau client dont on souhaite suivre les tweets s'effectue par une insertion en tête de liste, réduisant ainsi sa complexité à l'insertion sachant qu'on ne souhaite pas obligatoirement conserver une liste dans un ordre précis.
*void insert_follow(Follow **follow, Follow *fol);*

L'affichage de tout les followings s'effectue par cette méthode. Principalement, elle a pour but de permettre le débuggage, n'ayant que peu d'intérêt hormis si l'on souhaite d'offrir aux clients, une commande lui renvoyant la liste des gens qu'il suit.
*void print_follow(Follow *follow);*

Cette fonction d'encapsulation a pour but de faciliter l'utilisation extérieure de notre liste. En conséquence, elle effectue la vérification que l'ID n'est pas déjà présent dans notre liste de followings, puis, elle créé et insère en tête de liste, la nouvelle donnée.
*int add_follow(Follow **follow, unsigned short ID);*

killserver

A chaque démarrage de serveur, nous enregistrons dans un fichier caché .server.pid le PID du serveur venant d'être lancé.

Il ne nous reste plus qu'à tuer le processus ayant ce PID.

log

Ce module gère les logs des messages entrés par les différents clients. Ils sont classés par numéro de socket descriptor (ID). Les fichiers créés sont de la forme :

.login_ID.txt

Ces fichiers nous permettent d'établir très facilement une recherche des logs d'un ou plusieurs utilisateur(s) lorsque le client effectue une demande qui requiert d'avoir conservé un historique.

Cette fonction nous permet d'ouvrir le fichier de nom login_ID, avec les arguments nécessaire à la création, écriture ou ajout si on souhaite l'exécuter en écriture, ou nous pouvons l'exécuter en lecture simplement.

```
int open_log(unsigned short ID, int OPEN);
```

Cette fonction permet d'écrire dans un fichier les messages d'un client qui lui sont transmis.

```
void write_log(int fd, char *msg);
```

Cette fonction permet la lecture d'un fichier d'un utlisateur désignée par le fd passé en argument.

```
void read_log(int fd);
```

La fonction suivante ferme le descripteur du fichier en cours d'utilisation.

```
void close_log(int fd);
```

Les deux fonctions suivantes sont des fonctions d'encapsulation pour la fonction open_log. Cela permet un code plus propre et plus simple à utiliser à l'extérieur, car la gestion d'ouverture et de fermeture n'a plus besoin d'être manipulé par le code du serveur.

```
int writening_log(unsigned short ID, char *msg);
```

```
int reading_log(unsigned short ID);
```

server

Ce module sert à l'ouverture d'un serveur sur le terminal. Il gère aussi les messages qu'on lui transmet pour interagir avec les commandes données par l'utilisateur.

Pour se faire, nous effectuons l'initialisation d'une socket en écoute, qui écoutera alternativement les socket descriptors des clients qui lui enverront une requête. Dès qu'une requête sera soumise au serveur, il la récupérera puis la redirigera vers le traitement de commande.

Un parser nous permettra d'extraire les premiers caractères du buffer, afin d'évaluer la requête du client. De là, le serveur peut rediriger vers les sockets clientes les informations, comme l'affichage

d'un message, ou faire appel aux fonctions de logs de chargement et traitement de fichiers, ou d'ajout et suppression de followings.

User

Le module gère tout ce qui est relatif à l'utilisateur. On note que le nombre maximum d'utilisateur connectés en même temps est de 30.

L'utilisateur est défini par une structure qui donne son id et son login ainsi que la liste de ses followers.

```
typedef struct{
        unsigned short ID;
        char *login;
        Follow *following;
}Users;
```

Cette fonction nous permet tout simplement de connaître la taille précise du buffer, pour s'arrêter au caractère '\n'

```
int truncate_string(char *msg);
```

Fonction simpliste qui initialise la structure utilisateur.
void init_users();

On échange ici deux utilisateurs dans le tableau des utilisateurs.
`void swap_user(Users *first, Users *second);`

Cette fonction crée un utilisateur dans le tableau des utilisateurs connectés.
`int create_user(unsigned short ID, char *login);`

On recherche un utilisateur en fonction de son identificateur.
`int search_user(unsigned short ID);`

Permet d'afficher un utilisateur en donnant son id.
`void print_user(Users user, unsigned short ID);`

Même que la fonction précédente à la différence qu'on veut tous les utilisateurs présents.
`void print_all_user( unsigned short ID);`

La fonction supprime un utilisateur particulier.
```
void delete_user(unsigned short ID);
```

Permet la destruction de tous les utilisateurs connectés en vue de la suppression propre du programme.
```
void delete_all_users();
```