# Licence 3 Informatique 2009-2010
# Rapport projet Système 2
# "Mini Twitter"

Bassinot Hervé, hbassino

27 mai 2010

# Table des matières

# 1 Présentation du projet

Le but du projet “Mini Twitter”, est de permettre à un client de se connecter à un serveur afin d'interagir avec celui-ci. Le serveur doit simuler le comportement du site de micro-bloging “Twitter”. Ainsi, une fois connecté au serveur, le client doit s'authentifier à l'aide d'un simple login. Ensuite, il lui est possible par l'intermédiaire de commandes décrites ci-dessous de dialoguer avec le serveur afin de récupérer ou d'ajouter des informations. Le client peut donc, publier des Twitts, afficher les Twitts de n'importe quel utilisateur en ligne, ajouter ou supprimer des amis, connaître la liste des utilisateurs connectés au serveur et bien d'autres possibilités qui seront décrites plus en détails dans la suite du rapport. De plus, il est possible de suivre en simultané la publication des Twitts des amis que le client suit grâce au mode dynamique.

Pour ce projet nous aurons donc besoin de développer trois applications différentes : une application serveur, nécessaire pour la gestion des Twitts et des clients, une application cliente, depuis laquelle le client accède au serveur et interagit avec celui-ci et enfin une application pour arrêter le serveur. En effet, le serveur devra être exécuté en mode 'deamon' pour qu'il ne soit rattaché à aucun terminal de contrôle et qu'il puisse tourner en tache de fond. Il est également possible de préciser aux applications cliente et serveur des arguments afin qu'elles puissent modifier leurs comportements par défaut.

# 2 Compilation des executables

Pour compiler l'ensemble des programmes du projet, il suffit de se positionner dans le répertoire principal du projet et de taper dans la console la commande :

make

Cette commande permet de créer trois éxecutables différents dans le répertoire /bin qui sont :

- server : le programme faisant office de serveur.
- client : le programme faisant office de client.
- killserver : le programme permettant de mettre fin au serveur.

Les programmes sont compilés avec les options -Wall -ansi et -pedantic de GCC.

# 3 Execution des programmes

## 3.1 Lancement du programme server

Pour lancer le serveur du 'Mini Twitter' avec les paramètres par défaut, il faut se placer en mode console dans le répertoire principal du projet et taper la commande :

./bin/server

Par défaut, le serveur n'autorise pas le mode dynamique et ouvre le port 1963.

Il est possible de changer le numero de port par défaut du serveur. En effet, il suffit de lancer le programme de cette facon :

./bin/server -p numero

Où numero correspond au numero de port du serveur qu'il faut ouvrir.

Pour lancer le serveur en mode dynamique, il faut employer l'argument -d dans les paramètres du programme :

./bin/server -d

### 3.2 Lancement du programme client

Pour lancer le programme client avec les paramètres par défaut, il faut taper dans le répertoire principal en mode console la commande :

./bin/client

Par défaut, le client ouvre le port 1963 et se connect sur le serveur en local en 127.0.0.1.

Pour spécifier un autre numero de port (le même que celui du serveur), il faut lancer le programme de la manière suivante :

./bin/client -p numero

Où numero correspond au numéro de port du serveur qu'il faut ouvrir.

Pour changer l'adresse où se situe le serveur, il faut taper la commande :

./bin/client URI

Où URI est l'adresse du serveur sur lequel on souhaite se connecter.

### 3.3 Lancement du programme killserver

Pour lancer le programme killserver depuis le répertoire principal il suffit de taper dans la console, la commande suivant :

./bin/killserver

Une fois ce programme lancé, il tue le serveur et se termine immediatement après.

## 4 Présentation des executables

### 4.1 Présentation du programme server

Le programme 'server', doit gérer toutes les connections entre les différents clients du serveur ainsi que la base de données de 'Twitts' des clients et les informations de ces derniers.

Nous avons décidé de gérer la base de données des Twitts grâce à une liste chaînée d'utilisateur. Ainsi, chaque utilisateur de la liste possède une liste de Twitts qui lui est propre

ainsi qu'une liste de Following correspondant aux identifiants des clients que l'utilisateur suit. Pour simplifier la gestion des listes, nous avons developpé un module de listes chaînées génériques pouvant contenir des cellules de types Twitts ou de type Following. Le fait de ranger les utilisateurs dans une liste chaînée, nous donne une complexité linaire dans la recherche du client qui est en train de communiquer avec le serveur. Une fois l'utilisateur identifié, il reste à récupérer la commande qu'il a tapé et à l'éxecuter en parcourant soit la liste des utilisateurs, soit en mettant à jour sa propre structure d'utilisateur dans la liste. Nous conservons également dans la structure utilisateur, le descripteur correspondant au client afin de pouvoir communiquer avec lui à tout moment.

A chaque coupure du serveur, tous les twitts postés sont définitivement supprimés. Nous avons choisit de supprimer tous les twitts postés par un client lorsque celui-ci se déconnecte du serveur.

Nous gérons le multiplexage des clients du serveur grâce à la fonction 'select'. Ainsi, lorsque cette fonction detecte une communication de la part d'un client, celle-ci debloque le programme. Ensuite, il n'y a plus qu'a tester chacun des descripteurs correspondant aux clients, afin de savoir quel client a lancé une commande. A chaque tour de boucle du serveur, nous rajoutons à l'ensemble des descripteurs de la fonction 'select' tous les descripteurs correspondant aux clients. Si un nouveau client vient se connecter au serveur, la fonction 'accept' permet d'accepter la connection du client et de lui attribuer un descripteur pour permettre la communication avec ce client.

Afin d'être informé de ce qui se passe sur le serveur, nous avons créé deux fichiers logs. Un premier fichier pour l'affichage des erreurs commises lors de l'éxecution du serveur, un second pour l'affichage des differentes informations utiles telles que les connexions et déconnexions des clients. Nous avons donc modifié la sortie standard (stdout) et la sortie standard d'erreur (stderr) pour les rediriger vers nos fichiers logs, grâce à la fonction 'dup2'. A la fin de l'éxécution du serveur, nous refermons les descripteurs de fichiers correspondant aux fichiers logs.

Pour nous permettre de quitter le programme serveur de façon propre, nous avons choisi de faire en sorte de capturer le signal de fin d'éxécution de programme émis par le programme 'killserveur'. Ainsi, une fois le signal de fin capturé, nous refermons tous les descripteurs et terminons l'éxécution du programme. Le signal que nous capturons, est celui équivalent à Ctrl+C soit 'SIGINT'. En effet, le programme chargé de mettre fin à l'application enverra un signal identique.

## 4.2 Présentation du programme client

Le programme 'client', permet de communiquer avec le serveur. Celui-ci récupère les informations saisies par l'utilisateur et les transmet au serveur en attendant une réponse en retour. Ainsi, l'utilisateur peut par l'intermediaire de ce programme communiquer, acceder et mettre à jour les informations relatives à son compte Twitter du serveur.

Pour l'identification du client, nous n'acceptons comme login seulement ceux constitués de caractères alphanumériques. Ainsi dans le cas contraire, le client reçoit un message d'erreur du serveur. De plus, le client doit veiller à inscrire un login qui n'est pas déjà utilisé par un autre utilisateur du serveur. Ainsi si le client saisit un login existant, il recoit un autre message d'erreur du serveur.

Pour obtenir le mode dynamique, nous avons choisi d'utiliser la fonction 'select' du coté client. Ainsi on ajoute dans l'ensemble des descripteurs de lecture, le descripteur correspondant à la socket du serveur et le descripteur correspondant à l'entrée standard (stdin). Lorsque select détecte quelque chose sur la socket (serveur) ou bien sur l'entrée standard, il débloque le client et on agit en consequence.

### 4.3 Présentation du programme killserver

Ce programme permet de mettre fin à l'éxécution du programme server qui tourne en 'deamon' sur la machine. Ainsi, le programme est très simple, il appelle la commande système 'pkill' qui permet de tuer un processus indiqué par son nom. Il suffit alors de récupérer le nom du processus qui correspond au serveur et de le transmettre à la commande 'pkill'.

Ce programme lance un signal 'SIGINT' spécifique qui est intercepté par le programme serveur afin de quitter proprement l'application.

## 5 Organisation du projet

Pour des raisons pratiques, nous avons choisi d'organiser le projet dans des répertoires bien specifique.

- bin/ : répertoire contenant les trois executables du projet.
- obj/ : répertoire contenant les objets binaires.
- inc/ : répertoire contenant tous les headers du programme.
- src/ : répertoire contenant les sources des programmes client, server et killserver
- log/ : répertoire contenant les fichiers logs du serveur.

Le makefile se situe dans le répertoire principal du projet, il nous permet de faire de la programmation modulaire. Nous avons essayé de developper le projet le plus proprement possible en prenant soin de toujours vérifier les retours des fonctions système.

## 6 Présentation des commandes

Les commandes reconnaissables par le serveur sont sensibles à la casse. De plus toutes les commandes fond 3 caractères de long. Il y a en tout 11 commandes disponibles sur le serveur. Les commandes sont les suivantes :

- NAM : affiche les identifiants des clients connectés sur le serveur.
- ADD : ajoute un Twitt à la liste des Twitts du client.
- GET : affiche tous les Twitts d'un client.
- FOL : ajoute un client à la liste des Following.
- FQR : supprime un client de la liste des Following.
- CLE : supprime tous les clients de la listes des Following.
- ALL : affiche tous les Twitts de tous les clients suivis.
- LIS : active le mode dynamique.
- STI : désactive le mode dynamique.
- OUT : quitte l'application client.
- HLP : affiche l'aide sur les commandes du server.

Toutes les commandes doivent être écrite sur une seule ligne.

Nous avons rajouté la commande OUT pour permettre au client de se déconnecter du serveur de façon plus propre. Ainsi que la commande HLP pour l'affichage de l'aide sur les commandes pendant l'éxécution du serveur.

De plus, à chaque commande est renvoyé un code de retour par le serveur comme indiqué sur le sujet. Ensuite, la partie 'client' récupère le code de retour du serveur et affiche à l'utilisateur la signification du code en français. Afin de gérer tous les cas d'erreur possible, nous avons décidé d'ajouter des codes de retour pour la commande 'GET', pour gérer le cas par exemple où l'identifiant communiqué avec la commande 'GET' serait invalide ou autre.

# 7 Conclusion

Ce projet, nous a permis d'apprendre à créer une application client/serveur fonctionnelle. Nous avons appronfondi l'utilisation des sockets et mannipulé un bon nombre de fonctions d'appel système, telle que la fonction permettant de capturer un signal ou encore dup2 qui permet de créer une copie du descripteur de fichier, pour la création des fichiers logs. De plus nous avons vu comment détacher un programme d'un terminal afin qu'il tourne en mode 'demon'. En conclusion ce projet nous a fait pratiquer de manière concrète les differentes leçons acquises en cours et en TP.

Ce programme n'est pas adapté pour une utilisation à grande echelle. Le projet est perfectionnable, ainsi nous pouvons imaginer rajouter certaines commandes telles que afficher les noms des amis, sauvegarder les twitts dans un fichier de sauvegarde, ou bien ajouter une liste des utilisateurs qui nous suivent pour simplifier certaines commandes comme pour le mode dynamique.