*4IF2 : VALERO Juan, SZOSTKIEWICZ Max*

# TP Programmation Réseaux

## 1. Introduction

Ce TP concerne la construction d'une application distribuée de chat en utilisant des Sockets en C. Le but est de partir d'une version de code de base, et d'implémenter des fonctionnalités au fur et à mesure. Des pistes d'évolution du programme sont proposées par le sujet mais libre à nous d'améliorer l'application à notre guise.

Ce TP a été réalisé par Juan VALERO ainsi que Max SZOSTKIEWICZ, sur une durée d'environ une semaine. Le programme a été codé en langage C, sur Visual Studio Code.

## 2. Description des livrables

Le dossier compressé comprend ce manuel d'utilisation ainsi que le code source du programme. Le code source comprend un dossier Client et un dossier Serveur, pour bien distinguer les deux entités. Hormis les fichiers de code en c, on y trouve un fichier « client » contenant l'ensemble des clients ainsi qu'un dossier « channels » regroupant les historiques des conversations. Notons qu'un fichier Makefile a été conçu pour automatiser et assurer la compilation du programme.

## 3. Description de la compilation et de l'exécution

La compilation se réalise automatiquement grâce au fichier Makefile. Il suffit de lancer la commande « make » dans le répertoire en question pour obtenir les exécutables. Pour que le programme fonctionne, il faut exécuter à la fois le serveur, et un ou plusieurs clients. Pour exécuter le serveur, il faut se rendre dans le dossier Serveur et taper la commande « ./server2 ». Pour exécuter un client, il faut se rendre dans le dossier Client et exécuter la commande « ./client2 [adresse IP du serveur] [identifiant] [mot de passe] » (par exemple : ./client2 127.0.0.1 juan mdp). Notez que les identifiants et mots de passes des clients actuellement enregistrés sur le serveur sont :

- identifiant : juan ; mot de passe : mdp
- identifiant : max ; mot de passe : mdp
- identifiant : esteban ; mot de passe : mdp
- identifiant : kevin ; mot de passe : mdp

Une fois que les clients sont connectés, ils peuvent chatter en écrivant les messages directement dans la console. Notons qu'un message ne peut pas commencer par le caractère « / » car celui-ci est réservé aux commandes.

## 4. Fonctionnalités implantées

Nous avons implantés un grand nombre de fonctionnalités, ce qui rend le programme beaucoup plus complet. Chacune d'elle a été implémentée, testée et revisitée afin d'assurer son bon fonctionnement. Voici la liste des fonctionnalités ajoutées :

- **Channels** : Chaque client se situe à un instant t dans un channel, défini comme un groupe de discussion. Il peut créer un channel (« /c [nom du channel] ») et/ou en rejoindre un déjà existant (« /j [nom du channel] »). Quand un client se connecte au serveur, il est automatiquement placé dans le channel général nommé « global ». Quand un client écrit un message, le nom du channel apparaît entre parenthèses avant son pseudonyme.

- **Couleurs** : Une couleur est attribuée à chaque client du serveur. Ainsi, quand ce dernier écrit un message, son pseudo est affiché entre crochets avec une certaine couleur. Cela augmente grandement la lisibilité et l'ergonomie du programme.

- **Historique** : Chaque channel comporte son propre historique des messages, avec une fonctionnalité d'affichage des messages non lus en fonction de la dernière date de connexion du client. Les historiques sont stockés dans le fichiers respectifs du dossier « channels » dans Serveur.

- **Connexion** : Lorsqu'un client se connecte au serveur, il doit se connecter via un identifiant et un mot de passe. Le serveur vérifie alors les informations de connexion à l'aide du fichier « clients » dans le dossier Serveur.

- **Messages d'aide à la navigation** : Le serveur affiche dans la console la description de chaque action qui est réalisée. Cela permet une compréhension totale de chaque manipulation.