Programmation Système
TP n° 10 : `select()`
4-5 avril 2016

Le but de ce TP est d'écrire un petit programme de chat, sur le modèle client-serveur, *sans* création de processus ou threads auxiliaires.

Le serveur doit tout d'abord créer une *socket* destinée à recevoir les connexions des clients. Il doit ensuite, « simultanément », pouvoir accepter de nouvelles connexions, recevoir des messages à diffuser, ou constater qu'un client s'est déconnecté. Le client doit pour sa part se connecter à la soquette, puis, « simultanément », écouter les messages saisis par l'utilisateur et ceux diffusés par le serveur.

Cela sera rendu possible grâce à la fonction `select()`. On rappelle que `select()` signale parfois qu'un descripteur est prêt de manière erronée. Il est donc vivement conseillé de travailler en mode non bloquant, même si une lecture après un `select()` ne devrait pas bloquer.

Suggestion de progression :

1. Écrire un serveur rudimentaire qui crée une *socket* sur `/tmp/soquette`, se met en condition de recevoir `NB_MAX` connexions, puis entre dans une boucle (finie, disons 10 tours) durant laquelle il accepte *à la fois* les nouvelles connexions et les messages d'identification des clients déjà connectés. Attention, ne pas oublier de détruire la *socket* à la fin de la boucle !
2. Écrire un client rudimentaire qui se connecte à `/tmp/soquette`, envoie un premier message contenant son pseudo, puis dort 10 secondes avant de terminer.
3. Modifier le serveur pour qu'il repère (et gère !) les déconnexions de clients.
4. Modifier le client et le serveur pour que chaque message envoyé soit précédé d'un entier indiquant sa longueur.
5. Modifier le client pour permettre que l'utilisateur saisisse des messages, qui seront ensuite transmis au serveur.
6. Modifier le serveur pour qu'il affiche chaque message reçu, préfixé par le pseudo de l'expéditeur. Attention à bien lire le message intégralement.
7. Modifier le serveur pour qu'il diffuse les messages qu'il reçoit à tous ses clients (toujours préfixé par le pseudo de l'expéditeur). Modifier le client en conséquence.
8. Modifier le serveur pour qu'il réalise maintenant une boucle infinie, *mais* détruise tout de même la *socket* s'il est correctement interrompu (par la réception d'un signal `SIGINT`).

   Faire en sorte qu'avant de terminer, il diffuse à ses clients un message annonçant l'interruption du service. Les clients doivent alors s'interrompre également.