**Exercice1**

On veut développer une application client/serveur utilisant les sockets TCP. Cette application permet de multiplier un nombre par 3. En effet, le client lit un nombre au clavier et l’envoie vers le serveur via le flux de sorite (OutputStream), le serveur reçoit cet entier via le flux d’entrée (InputStream) et calcule le résultat et l’envoie vers le client.

**Exercice 2**

On veut développer une application client/serveur utilisant les sockets TCP. Cette application permet au serveur de recevoir le nom d’une personne et de renvoyer au client une chaine de caractère de la forme « Bonjour Mr »+str.

**Exercice 3**

Refaire l’exercice 2 mais cette fois en rendant le serveur un serveur multi-clients itératif.

**Exercice 4**

Refaire l’exercice 3 mais cette fois pour un serveur multi clients concurrents.

**Exercice 5 : jeu de devinette de nombre secret en réseau en utilisant les sockets**

Dans ce premier exercice, on développera un petit jeu en réseau. Le jeu est composé d'un processus serveur qui génère et détient un nombre secret entre 0 et 100. Les processus joueurs essayent de deviner ce nombre secret. Un processus joueur envoie une tentation au processus serveur. Trois cas se présentent:

1) si c'est le bon nombre alors le serveur garde l'adresse IP du processus gagnant et termine le jeu en affichant aux autres processus joueurs un message de fin (suite à une réception d'un essaye).

2) et 3) si le nombre envoyé est plus grand (respectivement plus petit) que le nombre secret, le serveur renvoie au processus joueur un message indiquant que le nombre envoyé est plus grand (respectivement plus petit) que le nombre secret.

On procédera en trois étapes. Même si les programmes développés lors des deux premières étapes n'ont pas un sens en pratique, il nous aidera à découper le problème et de converger vers la solution finale.

Dans toutes les étapes, on ne codera pas les processus clients / joueurs. On utilisera à la place un client Telnet.

a) Dans cette première étape, on développera un programme qui représente le serveur. On se limitera pour cette première version à un seul joueur.

b) Cette deuxième version consiste à étendre la version précédente en permettant à plusieurs clients de jouer d'une façon itérative (pas en même temps mais un après l'autre). Pour que cette deuxième version aye un sens, on déclarera perdant tout joueur au bout de 5 tentatives non abouties.

c) On étendra la version précédente pour permettre à plusieurs processus de joueur d'une façon concurrente. Il faut utiliser les threads pour implémenter la concurrence. Il faut aussi utiliser des attributs de classe pour pouvoir communiquer l'état du jeu entre les différents threads.

d) On voudrait implémenter le même jeu en utilisant des sockets UDP. Quels sont les problèmes que ce mode de communication pose par rapport au socket TCP.