PR6 – Programmation réseaux
TP n° 5 : Noté

Le TP doit être réalisé individuellement en `C` et est à envoyer par mail à votre enseignant de Tp au plus tard le vendredi 4 mars 2022 à 23h59.

Remarque : Dans le document le signe ␣ représentera un simple caractère d'espacement (ASCII 32). De plus les messages circulant sont indiqués entre guillemets et **les guillemets ne font pas partie du message**.

Vous devez écrire un serveur et deux clients pour le protocole de messagerie `maxint` décrit ci-dessous.
Le protocole fonctionnera de la façon suivante :

- le serveur accepte plusieurs connexions TCP de clients en parallèle ;
- lorsqu'un client se connecte au serveur, il peut envoyer au serveur un entier positif ou lui demander de lui envoyer l'entier le plus grand parmi ceux qui lui ont été envoyés.

Dans la suite `MAX_NAME` 10 et les noms d'utilisateurs sont codés en ASCII.

Le protocole `maxint` côté client

Lorsque le client se connecte, il envoie immédiatement au serveur le message « `<pseudo>` » où `<pseudo>` est le nom de l'utilisateur d'exactement `MAX_NAME` caractères. Il attend ensuite la réponse du serveur de la forme « `HELLO␣<pseudo>` ».
Le client peut alors envoyer deux types de messages au serveur :

- un message « `INT␣<val>` » où `<val>` est un entier non signé positif codé sur deux octets en big-endian,
- un message « `MAX` » pour demander le plus grand entier reçu par le serveur jusqu'à présent. À la réception de la réponse à ce message, le client devra de plus afficher le le nom de l'utilisateur, son adresse ip au format classique et la valeur contenue dans la réponse du serveur.

Ensuite le client se déconnecte.

Le protocole `maxint` côté serveur

Après avoir salué le client avec le message « `HELLO␣<pseudo>` », le serveur répond aux requêtes du client de la façon suivante :

- s'il reçoit un message de type « `INT` ». Il renvoie le message « `INTOK` » (Message OK) au client pour dire qu'il a reçu son message,
- s'il reçoit un message de type « `MAX` », il envoie au client la valeur maximale de l'entier parmi les entiers reçus. Il envoie donc « `REP<pseudo><ip><val>` » où `<pseudo>` est le nom de l'utilisateur ayant envoyé le plus grand entier, `<ip>` est son adresse ip codée sur 4 octets en big-endian, et `<val>` est la valeur du plus grand entier reçu par le serveur codé en big-endian. Il n'y a pas d'espace entre le pseudo, l'adresse ip et les données. Si le serveur n'a pas encore reçu d'entier, il répond par « `NOP` ». Si plusieurs clients ont envoyé la valeur la plus grande, le serveur renverra l'identité du dernier client ayant envoyé cette valeur.

Attention, veillez à bien gérer les accès concurrents pour les valeurs sauvegardées par le serveur.

Notes : le champ ***`sin_addr.s_addr`*** *de la structure* ***`struct sockaddr_in`*** *est un entier codé sur 4 octets en écriture big-endian.*
Pour afficher un entier sous forme hexadécimal avec la fonction ***`printf`****, utilisez la spécification de format* ***`%x`****.*

Travaux à rendre

Vous devrez rendre un fichier `serveur.c` contenant le code d'un serveur implémentant le protocole `maxint` et deux fichiers `client1.c` et `client2.c` correspondant à deux clients. Le premier client enverra en boucle 5 entiers positifs quelconques à votre serveur tournant sur la machine `lulu` de l'UFR en changeant de pseudo à chaque itération. Le deuxième client devra se connecter au serveur et demander au serveur l'entier reçu le plus grand et l'afficher avec le pseudo du client correspodant et se déconnecter. Les clients ne tourneront pas forcément sur `lulu`. Il faudra que vos programmes prennent un numéro de port en argument afin de faciliter les tests. Il faudra également rendre un `makefile` et un fichier `README` expliquant comment utiliser votre programme.
La notation prendra en compte la propreté et la lisibilité du code, ainsi que l'ergonomie de vos programmes (utilisation simple, adresse et port de connexion facilement modifiables) et la gestion des erreurs lors des appels systèmes.