PR6 – Programmation réseaux
TP n° 5 : Noté

Le TP doit être réalisé individuellement, en `C` et est à déposer sur Moodle (en respectant votre groupe de Tp) au plus tard le dimanche 7 mars 2021 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, **les guillemets ne faisant pas partie du message**.

Vous devez écrire un serveur et deux clients pour le protocole de messagerie `mess` 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 demander au serveur de stocker le message qu'il lui envoie ou de lui envoyer le dernier message stocké.

Dans la suite `MAX_DATA` vaudra 80 et `MAX_NAME` 10. Et les noms d'utilisateurs et les messages sont codés en ASCII.

Le protocole `mess` 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 d'exactement `MAX_DATA` caractères qui doit être stocké par le serveur. Le client envoie alors le message « `PUT␣<data>` » où `<data>` est le message à stocker de `MAX_DATA` caractères,
- la demande du dernier message stocké par le serveur, en envoyant le message « `GET` ». Il affichera alors la le nom, l'adresse ip au format classique et le message contenu dans la réponse du serveur.

Le protocole `mess` 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 « `PUT` », il sauvegarde le message reçu avec le nom de l'utilisateur et son adresse ip. Il renvoit le message « `MOK` » (Message OK) au client,
- s'il reçoit un message de type « `GET` », il envoie au client le dernier message stocké. Il envoie donc « `<pseudo><ip><data>` » où `<pseudo>` est le nom de l'utilisateur ayant envoyé le dernier message stocké, `<ip>` est son adresse ip codée sur 4 octets en big-endian, et `<data>` est le dernier message stocké. Il n'y a pas d'espace entre le pseudo, l'adresse ip et les données. Si il n'y a pas de message stocké, le serveur répond par « `NOP` »

Attention, veillez à bien gérer les accès concurrents au message sauvegardé 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 `mess` et deux fichiers `client1.c` et `client2.c` correspondant à deux clients. Le premier client enverra en boucle 5 messages quelconques (de taille `MAX_DATA`) à votre serveur tournant sur la machine `lulu` de l'UFR en attendant 2 secondes entre chaque envoi (pour cela vous pourrez utilisez la fonction `sleep` de la librairie `<unistd.h>`) puis se déconnecter. Le deuxième client devra se connecter au serveur et demander au serveur le dernier message stocké et l'afficher puis se déconnecter.