HMIN105M - TD/TP 4
Communications client-serveur et serveur multitâche

Nous avons vu en cours comment mettre en œuvre des communications en réseau entre un processus client et un processus serveur suivant différents modes et différents protocoles. L'objectif des exercices suivants est de réaliser des simples applications mettant en œuvre les protocoles UDP et TCP.

1 Ping-pong

Cet exercice est une introduction à la mise en œuvre de communications entre deux programmes/processus en utilisant le protocole UDP en mode asymétrique.

1. Ecrire un premier programme qui : 1) envoie un message sous forme de chaine de caractères saisie au clavier à un deuxième programme dont l'adresse (IP + numéro de port) sont à passer en paramètre ; et 2) reçoit et affiche une structure contenant deux entiers et une chaîne de caractères.
2. Écrire le deuxième programme. A la réception d'un message, afficher ce dernier ainsi que l'adresse IP et le numéro de port de l'expéditeur. Les données envoyées au premier processus sont à saisir au clavier.

Remarque : Lors de la réalisation des communications, veillez à ce que seules des données **utiles** soient transmises entre deux processus. Cette remarque est valable pour toute application à réaliser, peut importe le protocole utilisé.

2 Serveur multi-threads

L'objectif de cet exercice est 1) de réaliser une simple application en utilisant le mode connecté et le protocole TCP ; et 2) d'introduire la gestion simultanée de plusieurs clients en utilisant les threads.

Dans un premier temps, nous supposons un seul client. Ce client, envoie au serveur une suite de messages saisis au clavier. Ces messages sont lus et affichés par le serveur.

1. décrire le schéma algorithmique permettant l'établissement d'une connexion entre le client et le serveur ainsi que le dialogue.
2. Écrire le programme correspondant. Pour les entrées standards, utiliser la fonction suivante :

   char *fgets (char *s, int size, FILE *stream) : lit au plus size - 1 caractères depuis stream et les place dans le buffer pointé par s. La lecture s'arrête après EOF ou un retour-chariot. Si un retour-chariot (newline) est lu, il est placé dans le buffer. Un caractère nul '\0' est placé à la fin de la ligne.

Dans un deuxième temps, nous supposons deux clients qui se parlent à travers le serveur. Cela consiste à créer un serveur qui attend que deux clients se connectent puis qui transmet à chacun ce que l'autre envoie.

- Ecrire un serveur qui attend que deux clients se connectent et retransmet au client 2 ce que le client 1 envoie et vice-versa. **Note :** Il est demandé d'analyser différentes mises en œuvre du serveur.
- Ecrire le code des clients.

Enfin, nous supposons que le serveur ne transmet pas tous les messages reçus depuis un client vers le deuxième client. Sous la demande d'un client, un message est soit transmis soit simplement affiché sur le serveur.

– Modifier votre programme pour prendre en compte ce comportement.