TP7 : Premiers pas en programmation réseau

1 Un serveur echo

Le but de cet exercice est de réaliser un serveur qui renvoie à chacun de ses clients ce qu'il lui avait envoyé, à l'image de la commande `echo`.

1. Créez un fichier `server.c`.
2. Créez une socket TCP/IPv4 à l'aide de l'appel système `socket(2)`.
3. Comme nous écrivons un serveur, il faut lier la socket à un numéro de port bien défini, et non pas laisser le système le faire. Choisissez un numéro de port libre (par exemple 4200), et liez la socket à ce numéro de port. On utilisera `htons(3)`.
4. Il faut maintenant écouter les demandes de connexion TCP sur le serveur, et créer une file pour stocker les demandes jusqu'à ce que le serveur puisse les accepter. Créez une file d'attente avec `listen(2)`.
5. Créez une boucle infinie (un serveur n'est généralement pas fait pour s'arrêter) dans laquelle le serveur :
   — accepte les connexions avec `accept(2)` ;
   — affiche le numéro de socket sur laquelle il reçoit la connexion ;
   — affiche l'adresse IP (voir inet_ntop(3)) et le port (`ntohs(3)`) du client ;
   — ferme gracieusement la connexion avec `shutdown(2)`, et attend une seconde ;
   — ferme la connexion avec `close(3)`.
6. Créez une fonction `void echo(int client)` qui prend en paramètre la socket d'un client, et agit comme un echo : les informations seront reçues avec `read(2)`, et envoyées avec `write(2)` [1]. On fera particulièrement attention à ce que les écritures peuvent être partielles.
7. Testez votre code avec la commande `telnet localhost <port>`.
8. Testez votre serveur avec deux instances de `telnet` en parallèle (ouvrez deux terminaux).
9. Modifiez votre serveur pour qu'il puisse gérer plusieurs connexions en parallèle. On fera bien attention à garder une mémoire propre.

2 Un mini telnet

Nous avons le serveur, programmons le client ! Je vous invite à ne pas faire de copier / coller (pour se forcer à taper tout le texte). De toutes façons, c'est assez différent.

1. Créez un fichier `client.c`.

1. Les fonctions `send(2)` et `recv(2)` peuvent aussi être utilisées. Elles ont plus d'options, mais ne s'appliquent qu'aux sockets.

2. Créez une socket TCP/IPv4, puis :
   — connectez-la au serveur avec `connect(2)`. On pourra utiliser les fonctions de la librairie : `inet_pton`, `htons`, etc. ;
   — affichez la réussite de connexion (adresse, port...) ;
   — fermez la connexion gracieusement, puis avec `close` après une seconde.
3. Écrivez une fonction `void exchange_data(int sock)` qui lit ce que tape l'utilisateur (`read`) pour l'envoyer au serveur (`write`), puis attend et lit la réponse du serveur (`read`) pour l'afficher sur la sortie standard (`write`).
4. Comment faire pour ne pas attendre la réponse du serveur ? (le code devrait être plus simple, d'aileurs).