Sujet de TP 7-8 Système et **réseau** - Sockets Licence 3

Programmation réseau avec sockets en C (Linux)

V. Felea & A. Hugeat

Les exercices sur les sockets constituent une initiation à la programmation par sockets en C, pour la communication entre processus déployés sur des machines différentes, reliées par une connexion réseau. Les tests peuvent être effectués dans un premier temps sur la même machine locale, dans de tels cas l'interface réseau locale (lo) étant utilisée (deux terminaux peuvent servir pour le lancement des deux applications communicantes). Dans un deuxième temps le test sera effectué sur deux machines physiquement distinctes, pour faire exploiter l'interface réseau Ethernet. Une des applications sera lancée dans un terminal dans lequel une connexion distante `ssh` a été établie.
Rappel. Les noms des interfaces réseaux donnés par le noyau Linux peuvent être obtenus par les commandes `ifconfig` ou `ip`.

Commande `ssh` Cette commande sous Linux correspond au nom du processus client du protocole SSH (Secure SHell). Ce dernier permet à des utilisateurs d'accéder à une machine distante (sous Linux) à travers une communication chiffrée (par un couple de clés privée/publique). Le processus serveur s'appelle `sshd`. L'administration de serveurs sous Linux se fait presque exclusivement en ligne de commande à distance, à l'aide de `SSH`.

La connexion distante sécurisée est réalisée à base d'un nom d'utilisateur et d'un nom ou adresse IP de machine distante (le nom de l'utilisateur peut être omis, auquel cas, le nom de l'utilisateur courant est pris pour la connexion) :

```
# ssh login@adresseIP_MachDist
# ssh adresseIP_MachDist
# ssh nom_MachDist
```

Le client ssh pose alors une question du genre :

```
The authenticity of host 'nom_MachDist (2001:660:7101:1::9)' can't be established.
RSA key fingerprint is 7d:8a:f3:fc:f9:2d:db:77:18:e5:2e:cf:0b:1b:7d:b3.
Are you sure you want to continue connecting (yes/no)?
```

où `nom_MachDist` est le nom de la machine dont l'adresse IP a été utilisée dans la commande ssh.

Lors de l'établissement de la connexion, des échanges de clés sont effectués pour la sécuriser. Le client ssh demande donc si l'empreinte de la clé de la machine `nom_MachDist` est bien valide. Si la réponse est `yes`, cette empreinte sera stockée dans le fichier `/.ssh/known_hosts`, sinon la connexion sera refusée.

Ce premier exercice concerne la communication entre deux processus différents en mode connecté, utilisant le protocole TCP.

Envoi d'une chaîne de taille fixe (sens unidirectionnel de communication)

Écrire une application qui envoie une chaîne de caractères, reçue par une autre application. La taille de la chaîne est fixe et connue par les deux applications (grâce à une constante). Dans cette application, c'est le serveur qui envoie le message et le client le reçoit et l'affiche. Le port du serveur est configuré par un argument en ligne de commande. Le nom de la machine serveur et le numéro de port serveur sont également arguments en ligne de commande pour le client.

Indication. Dans la communication du message (la chaîne de caractères), ne pas oublier d'envoyer le marqueur de fin de chaîne de caractères qui est le caractère '\0'.

Nous considérons dans un premier temps la famille de sockets correspondant à une communication distante, pour le protocole internet IPv4.

Q1 Développer le client, en utilisant comme programme de test de côté serveur, l'exécutable fourni (voir archive sous Moodle). Tester l'exécution de l'application (lancer le serveur en premier, suivi du client) sur la machine locale.

Q2 Essayer de lancer une première instance, suivie d'une seconde (sans arrêter la première instance), du programme serveur et commenter le résultat. Essayer de lancer le client en premier, suivi du serveur et commenter le résultat.

Q3 Développer le serveur en utilisant comme programme de test de côté client l'exécutable fourni. Tester l'exécution de l'application sur la machine locale.

Q4 Tester les deux programmes écrits par vos soins sur la machine locale. Ultérieurement, déployer (exécuter) les deux processus, client et serveur, sur deux machines physiques distinctes.

Q5 Lancer une instance de serveur sur la machine locale et une autre instance de serveur sur une machine distante. Commenter le comportement et le comparer avec la réponse à la question Q2. Quelle affirmation sur les sockets confirme-t-il ce test ?

Deux autres familles de protocoles implémentées sous Linux sont à tester (dans deux versions différentes de l'application), correspondant d'une part à la communication locale (AF_UNIX), et d'autre part à la communication distante pour le protocole internet IPv6.

Q6 Écrire et tester la version de l'application utilisant les sockets appartenant au domaine de communication locale `AF_UNIX`.

Indication. Le type d'adresse d'une socket de communication locale est défini dans la bibliothèque `sys/un.h` :

```
struct sockaddr_un {
  sa_family_t sun_family;               /* AF_UNIX */
  char        sun_path[108];            /* pathname */
};
```

où `sun_path` est un nom de fichier, terminé par un octet nul, représentant le chemin d'accès aux données de la socket (sockets nommées).

Voir aussi le manuel unix(7).

La structure du client et du serveur est identique à celle développée pour la version de communication distante.

Q7 Écrire et tester la version de l'application utilisant la communication distante basée sur le protocole IPv6.

Indication. Le type d'adresse d'une socket de communication distante en IPv6 est défini dans la bibliothèque `netinet/in.h` :

```
struct sockaddr_in6 {
  sa_family_t     sin6_family;   /* AF_INET6 */
  in_port_t       sin6_port;     /* port number */
  uint32_t        sin6_flowinfo; /* IPv6 flow information */
  struct in6_addr sin6_addr;     /* IPv6 address */
  uint32_t        sin6_scope_id; /* Scope ID (new in 2.4) */
};

struct in6_addr {
  unsigned char   s6_addr[16];   /* IPv6 address */
};
```

où

- `sin6_family` est toujours égal à `AF_INET6`,
- `sin6_port` est le numéro de port (ordre d'octets réseau),
- `sin_addr.s6_addr` est l'adresse IPv6 de la socket. Le serveur peut utiliser la variable globale `in6addr_any` pour initialiser cette adresse. Le client peut l'initialiser avec la fonction `inet_pton`.

Les autres champs peuvent être initialisés à 0.

La structure du client et du serveur est identique à celle développée pour la version de communication distante pour l'adressage IPv4.

Remarque. La fonction `inet_aton` convertit des adresses pour IPv4 (voir cours), à partir des notations diverses (décimale à point, hexadécimale, octale). La fonction `inet_pton` traite aussi des adresses IPv6, les adresses IPv4 étant uniquement en notation décimale à point. Pour rendre cette conversion plus générique, indépendamment du type d'adresse (IPv4 ou IPv6) utiliser la fonction `getaddrinfo`(3) (voir exemple dans le cours).

Statistiques réseau - netstat Les activités réseau de la machine peuvent être surveillées grâce à la commande Linux/Windows `netstat`.
Les réponses aux questions suivantes sont attendues pour le système Linux.

Q8 Quelle option permet d'afficher les processus de la machine utilisant le protocole IPv6 ? Quel est le résultat de la commande ?

Q9 Donner la commande système qui permet d'afficher la liste des adresses IP de la machine courante associées à des sockets connectées. Quel est le résultat de son exécution ? Mêmes questions pour les adresses IP des machines distantes ayant une connexion par sockets à la machine courante. Donner la commande permettant d'afficher les adresses IP de la machine courante associées à des sockets en mode non connecté.

Q10 Afficher les sockets en état LISTEN. La commande doit être fournie.

Q11 Donner la commande permettant d'afficher les ports utilisés par les sockets locales en mode connecté. Quels sont les processus qui les utilisent ?

Q12 Rajouter des temporisations dans les programmes de l'application d'envoi de message et identifier les différents états des sockets utilisées (sur une version de communication distante - IPv4 ou IPv6). Expliquer ces états.

Q13 Dans la version avec temporisations, une fois la connexion établie entre le client et le serveur, fermer l'application serveur "brutalement" (Ctrl-C) et essayer de la relancer immédiatement. Qu'obtenez-vous ? Qu'indique `netstat` ? Expliquer, en utilisant les phases de la fermeture d'une connexion TCP.

Réutilisation d'adresse Utiliser la configuration suivante sur la socket de connexion ouverte sur le serveur, après sa création.

```
int enable = 1;
if (setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR,
               &enable,sizeof(int)) < 0)
    /* error */
```

Refaire le test précédent. En conclure.