Sockets 101

Sysinfo

MARTIN Jérémie

Univesity of Geneva, Switzerland
October 30, 2018

1 Pourquoi utiliser les sockets

Communication entre deux programmes, ou plus précisément entre deux processus, via le protocol TCP/IP. Les sockets représentent le moyen de communication entre deux processus, que ce soit à travers le réseau ou en local.

2 Modèle serveur-client

On a un processus serveur, et un processus client.

En d'autres termes, de manière pratique on a deux programmes séparés, et donc deux codes C différents qui ont été compilé.

Typiquement le client – qui connait l'adresse du serveur – se connecte à ce dernier pour lui envoyer une requête.

Une fois la connection établie, le serveur et le client peuvent envoyer et recevoir des informations.

Dans un système UNIX-like comme GNU/Linux, **tout est fichier**. On peut donc voir un socket comme un fichier spécial qui représenterait une connexion entre deux processus, à travers un réseau ou non.

Une fois qu'un socket est créé, écrire des données dans ce fichier aura pour effet de transformer cette information en paquet et de les envoyer au processus cible. De la même manière, pour recevoir des informations depuis le réseau, il suffira alors de lire le fichier socket.

D'une certaine manière, les sockets sont très similaires aux pipes: ils ressemblent à des fichiers pour les programmes qui les utilisent, mais n'entraînent pas de lecture ou d'écriture sur un disque, et ils permettent de communiquer avec un autre processus (de manière local dans le cas des pipes, et potentiellement distant dans le cas des sockets). Ils offrent également une **communication bidirectionnelle**, un peu comme un pipe qui marcherait dans les deux sens.

Il existe deux grands types de sockets:

- **Stream sockets**: les communications sont un stream continue de bytes (caractères)
- **Datagram sockets**: les communications consistent en l'échange de messages complets

3 Implémentation

Les headers nécessaires pour ce TP sont

```
#include <sys/socket.h> // Structures et fonctions liées aux sockets
#include <arpa/inet.h> // Structures et fonctions liées à internet (adresse IP etc)
#include <netinet/in.h> // Structures et macros pour utiliser des protocoles internet
#include <fcntl.h> // Manipulation de files descriptors (comme des sockets)
```

Listing 1: Headers liés aux sockets

3.1 Client

Du côté client, la création d'un socket se déroule comme ceci:

```
int sock = socket(PF_INET, SOCK_STREAM, 0); // Création de socket (en mode stream)
struct sockaddr_in address; // Structure qui contient une adresse internet
address->sin_family = AF_INET;
inet_pton(AF_INET, "127.0.0.1", &(address->sin_addr)); // Converti une adresse IP texte en binaire
address->sin_port = htons(1234); // Inverse l'ordre des bits du port (big-endian)
```

Listing 2: Création d'un socket du côté client

Il suffit alors d'écrire dans le socket avec la fonction **write** pour envoyer des informations au serveur, et d'utiliser **read** pour recevoir la réponse.

3.2 Serveur

```
struct sockaddr_in address;
int sock = socket(PF_INET, SOCK_STREAM, 0);
address->sin_family = AF_INET;
address->sin_addr.s_addr = htonl(INADDR_ANY); // Adresse IP auto, lie le socket à toutes les interfaces
address->sin_port = htons(1234); // Inverse l'ordre des bits du port (big-endian)

bind(sock, (struct sockaddr *) &address, sizeof(address)) // Lie le socket à une adresse, ici celle du host+port
listen(sock, 256) // Autorise le processus à écouter sur le socket pour créer une connexion
```

Listing 3: Création d'un socket du côté serveur

On utilisera à nouveau **read** et **write** pour recevoir et envoyer des données, au client cette fois-ci.