M2 FEADéP – Systèmes et réseaux
TP4 Sockets

Nicolas Chappe
École Normale Supérieure de Lyon

1 Introduction aux sockets en C

Note. Toutes ces fonctions modifient la valeur de `errno` en cas d'échec, voir leurs pages de manuel respectives pour plus de détails.

Création. `int socket(int domain, int type, int protocol)` crée un socket et renvoie un descripteur de fichier le représentant (ou -1 en cas d'erreur).

- `domain` peut valoir `AF_INET` pour la communication par IPv4, `AF_INET6` pour IPv6 ou `AF_UNIX` pour de la communication inter-processus locale.
- `type` peut valoir `SOCK_STREAM` pour une connexion TCP, ou `SOCK_DGRAM` pour communiquer par le protocole UDP. Le protocole UDP permet d'envoyer et recevoir des paquets de données, mais n'offre pas de garantie forte qu'ils seront effectivement reçus une et une seule fois dans le bon ordre. À l'inverse, le protocole TCP offre ces garanties, mais ne découpe pas les communications en paquet, les données sont considérées comme un flux continu d'informations, à la manière d'un tube.
- `protocol` peut être laissé à 0.

Les sous-sections suivantes présentent l'établissement d'une connexion client-serveur avec le protocole TCP.

1.1 Côté serveur

Attribuer une adresse au socket. `int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen)` retourne 0 en cas de succès, -1 en cas d'erreur.

- `sockfd` est le descripteur de fichier créé par `socket`.
- `addr` représente l'adresse à attribuer, de type `struct sockaddr_in`, structure documentée dans `man 7 ip` [1].

```
struct sockaddr_in {
    sa_family_t    sin_family; // famille d'adresse: AF_INET
```

1. C'est un peu différent pour l'IPv6, cf. `man ipv6`.

```
    in_port_t      sin_port;   // port avec l'endianness/boutisme du réseau (utiliser htons)
    struct in_addr sin_addr;   // l'adresse IPv4 en elle-même
};

struct in_addr {
    uint32_t       s_addr;     // adresse IPv4 avec l'endianness du réseau
};
```

Pour un serveur, l'adresse `s_addr` est l'adresse du serveur en lui-même, on peut lui donner la valeur `INADDR_ANY` pour écouter indifféremment sur toutes les interfaces réseau.

— `addrlen` est simplement la taille de l'argument précédent, c'est-à-dire `sizeof(struct sockaddr_in)`.

Attendre des connexions. `int listen(int sockfd, int backlog)` permet d'indiquer que le socket est prêt à recevoir des connexions de clients. La fonction prend en paramètre le socket et la taille maximale de la file d'attente en cas de connexions multiples de clients.

Établir des connexions. `int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen)` attend qu'un nouveau client se connecte (c'est une fonction bloquante) et renvoie un descripteur de fichiers matérialisant cette connexion. `addr` et `addrlen` permettent de récupérer l'adresse du client, nous les laisserons à `NULL`.

1.2 Côté client

Établir une connexion. `int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen)` fonctionne comme bind, à ceci près que `addr` sera l'adresse IPv4 du serveur auquel se connecter, que l'on peut obtenir en lançant la commande `ip addr` sur le serveur. Pour obtenir un `struct in_addr_t` on peut donner l'adresse du serveur à `in_addr_t inet_addr(const char *cp)` sous la forme `"a.b.c.d"`. La fonction `connect` renvoie 0 en cas de succès, -1 en cas d'erreur, notamment si le serveur refuse la connexion.

1.3 Communication

Pour envoyer et recevoir des données, utiliser les appels systèmes `read` et `write`.

Note. Le protocole UDP fonctionne sans connexion, on n'effectue donc pas de listen/accept/connect, et on remplace read/write par recvfrom/sendto.

Exercice 1

Écrire un programme qui se connecte à l'adresse IPv4 `140.77.168.21` sur le port 80, envoie la chaîne `"GET / HTTP/1.1\r\nHost: www.ens-lyon.fr\r\n\r\n"`, et affiche sur la sortie standard ce que le serveur envoie en retour. Que fait ce programme ? Quelle est sa limite ?

Exercice 2 (bonus)

Améliorer le programme précédent pour qu'il demande à l'utilisateur un nom de domaine, se connecte à son adresse IP, lui envoie la chaîne `"GET / HTTP/1.1\r\nHost: <nom de domaine>\r\n\r\n"` et affiche la réponse sur la sortie standard. Vous aurez besoin d'utiliser `getaddrinfo` pour récupérer l'adresse IP à partir du nom de domaine.

2 Messagerie instantanée en réseau

La prochaine fois !