Sockets avec IPv4 et IPv6

M Billaud

2 septembre 2018

Table des matières

1 Introduction : pourquoi ce document ?

La notion de *socket* remonte à la RFC 1471, du temps d'ARPANET (1971). Les premières APIs proviennent d'Unix 4.2 BSD (1983). Elles permettent d'utiliser le réseau de façon uniforme, avec différentes familles de protocoles : IPv4, APPLETALK, IPX, X25, etc.

Traditionnellement, les documents pédagogiques se réfèrent à IPv4 pour la programmation réseau entre machines. Mais l'apparition d'IPv6 (RFC 2460 de 1998) change les choses, même si le déploiement en est encore limité : il convient maintenant d'écrire des applications qui fonctionnent à la fois avec IPv4 et IPv6.

Heureusement, des fonctions ont été introduites (POSIX issue 6, 2004) pour assurer cette compatibilité, rendant obsolètes les fonctions qui étaient généralement enseignées jusque-là.

Par exemple la fonction `gethostbyname`

```
struct hostent * gethostbyname(const char *name)
```

assurait la résolution des noms (traduction d'une adresse de la forme "`www.google.fr`" en adresse IP `216.58.201.227`), en retournant des adresse IPv4 contenues dans une liste chainée de structures.

Elle est évidemment inutilisable pour obtenir les adresses IPv6 qui sont pourtant fournies par les DNS.

À la place, on utilisera `getaddrinfo`

```
int getaddrinfo(const char *node, const char *service,
                const struct addrinfo *hints,
                struct addrinfo **res);
```

qui retournera une liste d'adresses internet IPv4 et/ou IPv6, selon les arguments qui lui sont données :

- `node` et `service` : nom de la machine et du port,
- une structure `hints` contenant des indications diverses.

La nouvelle API permet de traiter indifféremment des adresses et des sockets IPv6 et v4. C'est ce qu'expose ce document.

2 Conversion hôte+port <=> adresse internet

2.1 `getaddrinfo()` : Obtenir une adresse internet

La fonction `getaddrinfo` fournit des *adresses internet*, c'est-à-dire des objets qui contiennent à la fois

- une adresse IP
- un numéro de port

sous forme d'une liste chainée de structures `addrinfo`.

2.2 Nom de machine et numéro de port

Les deux premiers paramètres sont des chaines de caractères avec le nom de la machine ou son numéro ip, le nom du service (voir `/etc/services`) ou le numéro de port.

Exemples d'appels :

```
getaddrinfo("www.google.fr",     "80",    .....);
getaddrinfo("www.u-bordeaux.fr", "https", ....);
getaddrinfo("2a01:e0c:1::1",     "http",  ....);
getaddrinfo("www.facebook.com",  NULL,    ....);
```

Dans le dernier cas, on obtiendra une adresse internet avec un numéro IP renseigné, et un numéro de port non spécifié.

2.3 Liste de résultats

Les résultats sont fournis par l'intermédiaire d'une liste chainée de structures `addrinfo` :

```
struct addrinfo {
               int              ai_flags;
               int              ai_family;
               int              ai_socktype;
               int              ai_protocol;
               socklen_t        ai_addrlen;
               struct sockaddr *ai_addr;
               char            *ai_canonname;
```

```
    struct addrinfo *ai_next;
};
```

ce type de structure sert aussi pour les indications complémentaires pour la requête (voir plus loin)

Les champs qui nous intéressent, dans les résultats, sont

- `ai_family`qui indique la famille d'adresses : constante `AF_INET` pour ipv4, `AF_INET6` pour ipv6 ;
- `ai_addr` qui est un pointeur sur une adresse "générique", qui sera concrètement soit une `struct in_addr`, soit une `struct in6_addr`, selon la famille indiquée.
- `ai_addrlen`, la longueur de cette adresse ;
- `ai_next`, un pointeur sur le résultat suivant.

Attention, il faudra libérer la liste de résultats après usage, en appelant `freeaddrinfo`

```
void freeaddrinfo(struct addrinfo *res);
```

2.4 Exemple

Illustration : créer un socket TCP client connecté à un serveur WEB (protocole http), en utilisant la première adresse trouvée (et sans faire aucune vérification) :

```
char * nom_serveur = "....";

struct addrinfo * premier;                    // adresse premier résultat;
getaddrinfo(nom_serveur, "http", NULL, & premier);

int fd = socket(premier->ai_family, SOCK_STREAM, 0);
bind(fd, premier->ai_addr, premier_>ai_addrlen);

freeaddrinfo(adr_premier_resultat);
```

2.5 Les indications

Dans l'exemple ci-dessus, le troisième paramètre de `getatddrinfo()` est un pointeur nul, mais on peut s'en servir pour transmettre l'adresse d'une structure `addrinfo` qui sert à préciser ce que l'on veut.

Les champs pertinents (les plus intéressants) :

- `ai_family`qui indique la ou les familles d'adresses voulues : constante `AF_INET` pour ipv4, `AF_INET6` pour ipv6, `AF_UNSPEC` (par défaut) pour l'un ou l'autre.
- `ai_socktype` indique si on ne doit rechercher que certains types de services (`SOCK_STREAM`, `SOCK_DGRAM`) ou tous (O).
- `ai_flags` : combinaison d'indicateurs divers. Y mettre `AI_PASSIVE` pour un serveur qui doit accepter des connexions sur ses différentes adresses réseau.

Les autres doivent être initialisés, 0 est une valeur par défaut raisonnable.

L'initialisation se fait élégamment grâce aux "Designated Initializers" introduits par la norme C99. Exemple

```
struct addrinfo indication_client {
   .ai_family   = AF_UNSPEC,
   .ai_socktype = SOCK_STREAM
};

struct addrinfo indications_serveur {   // pour un serveur IPv6
   .ai_family = AF_INET6,
   .ai_flags  = AI_PASSIVE
   // les autres champs sont automatiquement initialisés à 0
};
```

2.6 `getnameinfo()` : traduction d'adresse internet en hôte+service

La fonction `getnameinfo()` réalise la conversion inverse : à partir d'une adresse internet, obtenir une description du l'adresse IP de la machine et du numéro de service/port.

```
int getnameinfo(const struct sockaddr *addr, socklen_t addrlen,
                char *host, socklen_t hostlen,
                char *serv, socklen_t servlen,
                int flags);
```

Les paramètres sont

- un pointeur sur la structure contenant l'adresse, et sa taille
- l'adresse d'un tampon pour y ranger le nom de la machine, et sa longueur
- l'adresse d'un tampon pour le port, et sa longueur.
- une combinaison d'indicateurs : `NI_NUMERICHOST` et `NI_NUMERICSERV`, pour avoir les indications sous forme numérique.

2.7 Un exemple : programme de résolution d'adresses

Le programme qui suit s'exécute en ligne de commande. Il prend en paramètre une série de noms de machines ou d'adresses

Pour chaque machine il affiche les adresses IP numériques sous forme numérique IPv4 (décimal pointé) ou IPv6 (hexadécimal).

Code Source

```
/*
 resolution.c

 Résolution d'adresses
 */

#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
```

```c
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <limits.h>

void resoudre_nom(const char nom_hote[]);

int main(int argc, char *argv[])
{
    if (argc == 1) {
        printf("usage: %s adresses...\n", argv[0]);
        return EXIT_FAILURE;
    }
    for (int i = 1; i < argc; i++) {
        resoudre_nom(argv[i]);
    }
    return EXIT_SUCCESS;
}

void resoudre_nom(const char nom_hote[])
{
    printf("* Résolution de %s\n", nom_hote);
    struct addrinfo *adr_liste_adresses;

    // le type de socket est précisé pour n'avoir qu'une réponse par numéro IP
    struct addrinfo indications = {
        .ai_family = AF_UNSPEC,
        .ai_socktype = SOCK_STREAM
    };

    int r = getaddrinfo(nom_hote, NULL,
            &indications,
            & adr_liste_adresses);
    if (r != 0) {
        printf("- échec de la résolution\n");
        return;
    };
    for (struct addrinfo *ptr = adr_liste_adresses;
            ptr != NULL;
            ptr = ptr->ai_next) {
        // conversion  adresse -> hote
        char ip_hote[HOST_NAME_MAX]; // taille max FQDN, RFC 1035
        getnameinfo(ptr->ai_addr, ptr->ai_addrlen,
                ip_hote, sizeof (ip_hote),
                NULL, 0, // port non utilisé
                NI_NUMERICHOST);
        printf("-> %s\n", ip_hote);
    }
```

```
    printf("\n");
    freeaddrinfo(adr_liste_adresses);
}
```

Exemple d'exécution

```
$ ./resolution www.google.fr  www.free.fr www.u-bordeaux.fr
* Résolution de www.google.fr
-> 2a00:1450:4007:809::2003
-> 216.58.198.195

* Résolution de www.free.fr
-> 2a01:e0c:1::1
-> 212.27.48.10

* Résolution de www.u-bordeaux.fr
-> 147.210.215.26
```