# RFC 3493 : Basic Socket Interface Extensions for IPv6

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 19 février 2008

Date de publication du RFC : Février 2003

https://www.bortzmeyer.org/3493.html

---

A priori, l'IETF ne normalise pas d'API (et c'est pour cela que ce RFC n'a que le statut de « pour information »), se limitant aux « bits qu'on voit passer sur le câble ». Mais le monde de la normalisation n'est pas toujours si simple et, à l'époque, il était important de publier rapidement une documentation sur la nouvelle API nécessaire aux applications pour faire de l'IPv6.

Les API Unix et réseaux sont typiquement gérées par l'IEEE, via sa norme POSIX. Mais POSIX n'est pas librement téléchargeable sur le réseau, ce qui limite son utilité. Pour cette raison et pour d'autres, l'IETF a donc publié ce RFC, dont on notera qu'un des auteurs est W. Richard Stevens, auteur de "*Unix network programming*" <https://www.bortzmeyer.org/unix-network-programming.html>, **le** livre de référence sur le sujet. Il succède au RFC 2553[1] et est complété par un RFC sur l'interface « avancée », le RFC 3542.

Depuis leur apparition, les **prises** ("*socket*" dans la langue de Stevens) sont le principal mécanisme d'accès au réseau pour les programmes. Les programmes qui les utilisent sont relativement portables, bien au delà du système BSD où elles sont nées. Traditionnellement multi-protocoles, les prises ne permettaient néanmoins pas d'accéder à IPv6. C'est désormais le cas et l'interface décrite dans ce RFC est désormais largement présente (MacOS X a été le dernier système d'exploitation à la proposer).

Elle ne concerne que le langage C. Les autres langages disposent souvent de mécanismes de plus haut niveau que les prises comme les "*Streams*" <http://java.sun.com/docs/books/tutorial/networking/urls/readingURL.html> de Java.

Idéalement, la plupart des applications ne devrait pas avoir besoin de savoir si elles utilisent IPv4 ou IPv6 <https://www.bortzmeyer.org/network-high-level-programming.html>. Concentrée

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc2553.txt

dans la couche Réseau, le protocole utilisé ne devrait pas concerner l'application. Mais C est un langage de bas niveau, et expose donc le protocole de couche 3 utilisé (même si la section 2 du RFC explique que l'API cherche à être la plus générale possible).

La section 2.1 résume les principaux changements qui ont été faits pour permettre l'utilisation d'IPv6. Ils incluent une nouvelle constante (section 3.1) pour indiquer les adresses IPv6 (`AF_INET6`, puisque `AF_INET`, comme son nom ne l'indique pas clairement, étant spécifique à IPv4) et une pour le protocole IPv6 (`PF_INET6`). Notons que le RFC documente aussi `PF_UNSPEC`, qui permet à l'application de dire qu'elle se moque du protocole utilisé (ce qui devrait être le cas le plus courant). Il y a bien sûr une nouvelle structure de données (section 3.3) pour placer les adresses IPv6, quatre fois plus grandes. Les `sockaddr` sont désormais des représentations de la prise, indépendantes de la famille d'adresses, les `sockaddr_in6` étant spécifiques à IPv6. Elles contiennent :

```
struct sockaddr_in6 {
    sa_family_t     sin6_family;    /* AF_INET6 */
    in_port_t       sin6_port;      /* transport layer port # */
    uint32_t        sin6_flowinfo;  /* IPv6 traffic class & flow info */
    struct in6_addr sin6_addr;      /* IPv6 address */
    uint32_t        sin6_scope_id;  /* set of interfaces for a scope */
};
```

Les appels système sur les prises, décrits dans la section 3.5, eux, ont peu changés, `socket`, `connect`, `bind`, continuent comme avant.

Les sections 3.6. et 3.7 concernent l'interopérabilité avec IPv4. Le monde des applications a une forte inertie. Les développeurs, déjà très occupés, ne portent pas leurs applications vers la nouvelle interface instantanément. Ces deux sections expliquent donc comment assurer le recouvrement des deux protocoles. On notera qu'avec cette interface, pour faire un serveur qui écoute en IPv4 **et** IPv6, il faut créer une prise IPv6, les clients IPv4 recevront alors des adresses "*IPv4-mapped*" comme `::FFFF:192.0.2.24`. Ce n'est pas très intuitif... (Heureusement, c'est plus simple pour un client.)

La section 6 du RFC parle des fonctions de bibliothèque pour effectuer les traductions de noms en adresses et réciproquement. L'ancienne `gethostbyname`, très spécifique à IPv4 et souffrant d'autres limitations, ne devrait plus être utilisée depuis des années (le nombre de programmes qui l'utilisent encore ou de tutoriels qui l'expliquent donne une idée de l'extrême inertie d'un secteur qui se prétend novateur et réactif). Les « nouvelles » fonctions `getaddrinfo` et `getnameinfo` (section 6.4) la remplacent. Un programme typique désormais fait :

```
char *server = "www.example.org";
char *port_name = "42";

int mysocket;

/* addrinfo est défini dans netdb.h et inclus une "struct sockaddr"
(une prise, incluant l'adresse IP) dans son champ ai_addr. Sa
description figure dans la section 6.4 du RFC. */
struct addrinfo hints, *result;
hints.ai_family = PF_UNSPEC;
hints.ai_socktype = SOCK_STREAM;

error = getaddrinfo(server, port_name, &hints, &result);
if (error) {
          err_quit("getaddrinfo error for host: %s %s",
                   server, gai_strerror(error));
}
```

```
/* La première adresse IP sera dans result->ai_addr, pour les
suivantes, il faudra suivre result->ai_next */
mysocket = socket(...);
error = connect(mysocket, result->ai_addr, result->ai_addrlen);
...
```

Un des rares cas où un client réseau a besoin de manipuler des adresses IP (pas seulement de les stocker et de les passer d'une fonction à une autre) est lors de l'affichage, par exemple en mode bavard. Pour éviter que le programme ne doive connaitre les détails de syntaxe des adresses (celle d'IPv4 n'a jamais été normalisée, celle d'IPv6 est décrite dans le RFC 4291), notre RFC décrit en section 6.6 des fonctions de conversion de et vers un format texte, `inet\_ntop` et `inet\_pton`.

Enfin, la section 9 liste les différences avec son prédécesseur, le RFC 2553. Rien de très important ici.

L'inertie des programmeurs étant très forte, et celle des enseignants également, on peut parier, bien que le premier RFC sur le sujet soit vieux de onze ans, que beaucoup de cours de programmation réseaux sont encore donnés avec la vieille interface, qui ne marche qu'en IPv4... À l'appui de ce pari, notons que le service Code Search `<http://www.google.com/codesearch>` de Google trouve 255 000 occurrences de `gethostbyname` contre seulement 78 200 de `getaddrinfo`.