Introduction à la Programmation Réseaux en Langage C

11. Manipulation d'Adresses

Où chercher ?

- Fichiers d'inclusion
  - ‹ sys/types.h›
    types de données élémentaires (u_char, u_long, ...)
  - ‹ sys/socket.h›
    déclarations concernant les sockets
  - ‹ netinet/in.h›
    déclarations concernant le protocole IP
  - ‹ netdb.h› :
    déclarations concernant la résolution d'adresses

Où chercher ?

- Manuel
  - byteorder(3)

    Conversions d'entiers entre format natif et format réseau
  - inet(3)

    Conversion d'adresse entre formats texte et binaire
  - gethostbyname(3)

    Résolution d'adresses

Conversion entre Formats Natif et Réseau

- Adresse IP = entier 32 bits
  - Tous les CPU ne comprennent pas les entiers 32 bits de la même façon
  - Format MSB, LSB, mixtes …
- Une convention a été adoptée pour éviter les erreurs : Adresse IP = 32 bits MSB
  - Les machines qui n'ont pas la chance de calculer au format MSB doivent convertir
    - Adresses IP (32 bits)
    - Numéros de port (16 bits)
    - Masques de réseaux, …

Conversion entre Formats Natif et Réseau (2)

- Fonctions de conversion :
  - htons (host to network short)
  - htonl (host to network long)
  - ntohs (network to host short)
  - ntohl (network to host long)
- On ne sait pas toujours si l'ordinateur que l'on programme est MSB …
  - Conversion systématique !
  - Ces fonctions existent aussi sur les "bonnes" machines au format MSB : elle ne changent rien …

Quand utiliser les fonctions de conversion ?

- A chaque fois que :
  - L'API nous fournit une donnée de type adresse/port
    - Ex: réception d'un message venant de xxx.yyy.zzz.ttt
  - L'API demande une donnée de type adresse/port
    - Ex: envoyer datagramm vers port iiii de machine xxx.yyy.zzz.ttt ...
- Attention aux calculs arithmétiques !
  - Exemple : scan de ports
    - for (port= 1024 ; port< 2048 ; port++) ...
    - port++ est juste Ssi port au format host
  - En cas de doute : ne pas hésiter à utiliser des variables de nom explicite : hport = ntohs(nport)

Résolution d'Adresses

- Fonctions d'interface avec le DNS
  - gethostname(char * *name*, size_t *len*)
    - Retourne dans *name* le nom de la machine courante
  - struct hostent *gethostbyname(char * *name*)
    - Retourne une structure décrivant la machine de nom *name*
  - struct hostent *gethostbyaddr(char **addr*, int *len*, int *type*)
    - Retourne une structure (la même) décrivant la machine d'adresse *addr* (format réseau)
    - *len* = sizeof(addr)
    - *type* = AF_INET

La structure Hostent

```
struct hostent {
        // nom officiel
        char *h_name;
        // autres noms : vecteur de chaînes, dont le
        // dernier élément est NULL
        char **h_aliases;
        // type d'adresse (AF_INET)
        int h_addrtype;
        // longueur des adresses (32 bits = 4 si IP)
        int h_length;
        // Liste des adresses (terminée par NULL aussi)
        char **h_addr_list;
};
```

Attention : horreur !
Faux type char * (en fait void *)
Cast délicat dans le cas d'@ IP...

Structure d'adresse

```
struct sockaddr {
u_short sa_family;      /* address family: AF_xxx value */
char    sa_data [14];   /* up to 14 bytes protocol specific address */
};
```

Type générique : **jamais** instancié

```
struct in_addr {
u_long  s_addr;         /* 32 bit netid/hostid, network byte ordered */
};
```

Type instancié dans le cas d'une @IP

```
struct sockaddr_in {
short           sin_family;     /* AF_INET */
u_short         sin_port;  /* 16 bit port number, net byte ordered */
struct in_addr      sin_addr;
char            sin_zero [8];   /* unused */
};
```

Conversion d'adresses IP

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
unsigned long inet_addr (char * ptr);
char * inet_ntoa (struct in_addr inaddr);
```

- Conversion d'adresses internet entre
  - Chaîne au format "point décimal" (ex: 192.168.0.1)
  - Format binaire 4 octets (réseau)