ENSEA
Ecole Nationale Supérieure de l'Electronique et de ses Applications

2e année – Majeure Informatique

TDm 3 : Resto ?

C. BARÈS

Les manipulations proposées dans ce TDm sont à réaliser sous GNU/Linux. Référez-vous en permanence aux pages de manuel des fonctions utilisées, section 2, 3 et 7 :

2 Appels systèmes
3 Fonctions de la bibliothèque C standard
7 Panorama, conventions et divers

Remarque : Évitez les pages de manuel en français, elles ne sont pas à jour...

1 – Trouver une bonne adresse...

1.1 Recherche

Vous allez écrire un programme qui prend en argument un nom de domaine, et qui renvoie les adresses IP associées. Pour cela vous allez utiliser la fonction suivante :

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>

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

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;
};
```

Étapes :

1. Déclarez une `struct addrinfo hints`, et initialisez la avec des zéros.
2. Déclarez une `struct addrinfo *res`,
3. Appelez la fonction `getaddrinfo`, en remplaçant `service` par `NULL` et `node` par votre 1er argument.
4. Testez une éventuelle erreur de `getaddrinfo`. Un message d'erreur peut être récupéré à l'aide de la fonction `gai_strerror()`.

5. Parcourez la liste chaînée `res` et imprimez pour chaque résultat le champ `ai_family`, `ai_socktype` et `ai_protocol`.

Comment l'OS fait-il pour avoir ces résultats ? Pourquoi a-t-on autant de résultats ?

1.2 Filtrage

L'appel système `getnameinfo` permet de faire l'inverse de `getaddrinfo` : retrouver le nom d'une machine à partir d'une structure `struct sockaddr`.

```
#include <sys/socket.h>
#include <netdb.h>

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

Pendant la conversion, vous pouvez choisir de ne récupérer que des nom sous forme d'IP, il suffit d'utiliser les flags : `NI_NUMERICHOST | NI_NUMERICSERV`.

1. Paramétrez la `struct addrinfo hints` pour ne travailler qu'avec TCP.
2. Modifiez votre affichage pour imprimer les adresses IP obtenues.

---

2 – … ET COMMANDER AU SERVEUR.

Nous allons maintenant utiliser une `struct sockaddr` obtenue précédemment pour nous connecter à un serveur web.

1. Modifiez le code précédent de manière à ce que l'appel à `getaddrinfo` ne renvoie plus qu'une seule adresse IPv4 du serveur.
2. Modifiez le code précédent de manière à ce que la `struct sockaddr` contiennent aussi le numéro de port.
3. Réalisez une connexion au serveur :
   a) Créez un socket compatible avec votre `struct sockaddr` ;
   b) Ouvrez une connexion vers le serveur à l'aide de `connect` ;
   c) Envoyez une requête au serveur avec `write` ;
   d) Lisez la réponse du serveur avec `read` ;
   e) Affichez le résultat.
4. Faites le ménage :
   - coupez la connexion ;
   - fermez le socket ;
   - libérez la liste chaînée des `struct addrinfo *res`.

Qu'est-ce qu'il faudrait changer sur cette 2e partie si on voulez utiliser IPv6 ?

Rappel : Une requête valide en HTTP vers le serveur web de `perdu.com` s'écrit :

```
GET / HTTP/1.1\r\nHost: www.perdu.com\r\n\r\n
```

Annexe : Constantes utiles

Address Family (socket.h)

Nom	Valeur
AF_UNSPEC	0
AF_UNIX	1
AF_LOCAL	1
AF_INET	2
AF_INET6	10
AF_PACKET	17

Socket Type (socket.h)

Nom	Valeur
SOCK_STREAM	1
SOCK_DGRAM	2
SOCK_RAW	3
SOCK_RDM	4
SOCK_SEQPACKET	5
SOCK_DCCP	6
SOCK_PACKET	10

Protocols (netinet/in.h)

Nom	Valeur
IPPROTO_IP	0
IPPROTO_ICMP	1
IPPROTO_TCP	6
IPPROTO_UDP	17
IPPROTO_UDPLITE	136
IPPROTO_MPLS	137
IPPROTO_RAW	255