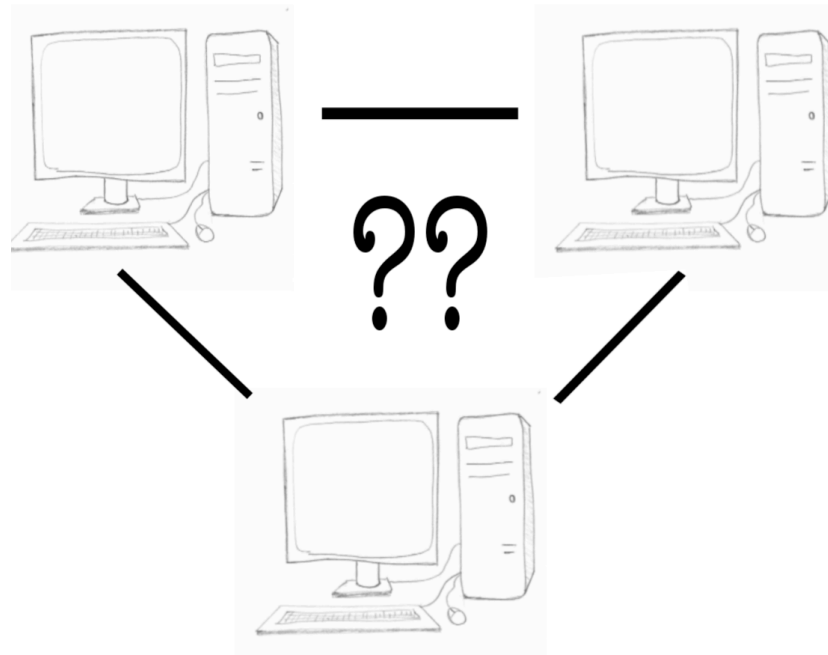


PROGRAMMATION RÉSEAU

Arnaud Sangnier
sangnier@irif.fr

API TCP C - III



Accès à une machine

```
inet_aton("10.0.0.1", &adress_sock.sin_addr);
```



Mais si on ne connaît
pas l'adresse IP ?

- Il faut interroger l'annuaire DNS

Interroger l'annuaire en C

- On peut désirer récupérer l'adresse Internet associée à un nom Internet
 - Ceci nécessite d'obtenir **la résolution de nom**
 - Il existe différentes fonction d'accès à l'annuaire DNS
 - La fonction historique
 - **struct hostent *gethostbyname(const char *name);**
 - La fonction moderne
 - int getaddrinfo(const char *node, // "www.example.com" or IP**
const char *service, // "http" or port number
const struct addrinfo *hints,
struct addrinfo **res);

La fonction gethostbyname

- **#include <netdb.h>**

struct hostent *gethostbyname(const char *name);

- L'appel à cette fonction renvoie une structure de la forme suivante :

struct hostent {

char *h_name; // Le nom canonique

char **h_aliases; // Une liste d'alias - le dernier élément est NULL

int h_addrtype; // Le type de l'adresse, qui devrait être AF_INET en général

int h_length; // La longueur des adresses en octet

char **h_addr_list; // Une liste d'adresses IP pour cet host

};

- En fait la dernière est un tableau de **struct in_addr ***, le dernier élément est NULL aussi

Exemple

- On va faire un code qui pour un nom de machine va récupérer toutes les adresses IPv4 correspondantes et les afficher. On affichera également les alias associés à un nom
- Pour cela :
 - On récupère le hostent correspondant
 - On parcourt les tableaux d'alias et d'adresses
 - Pour chaque adresse, on la traduit en chaîne de caractères grâce à la fonction :

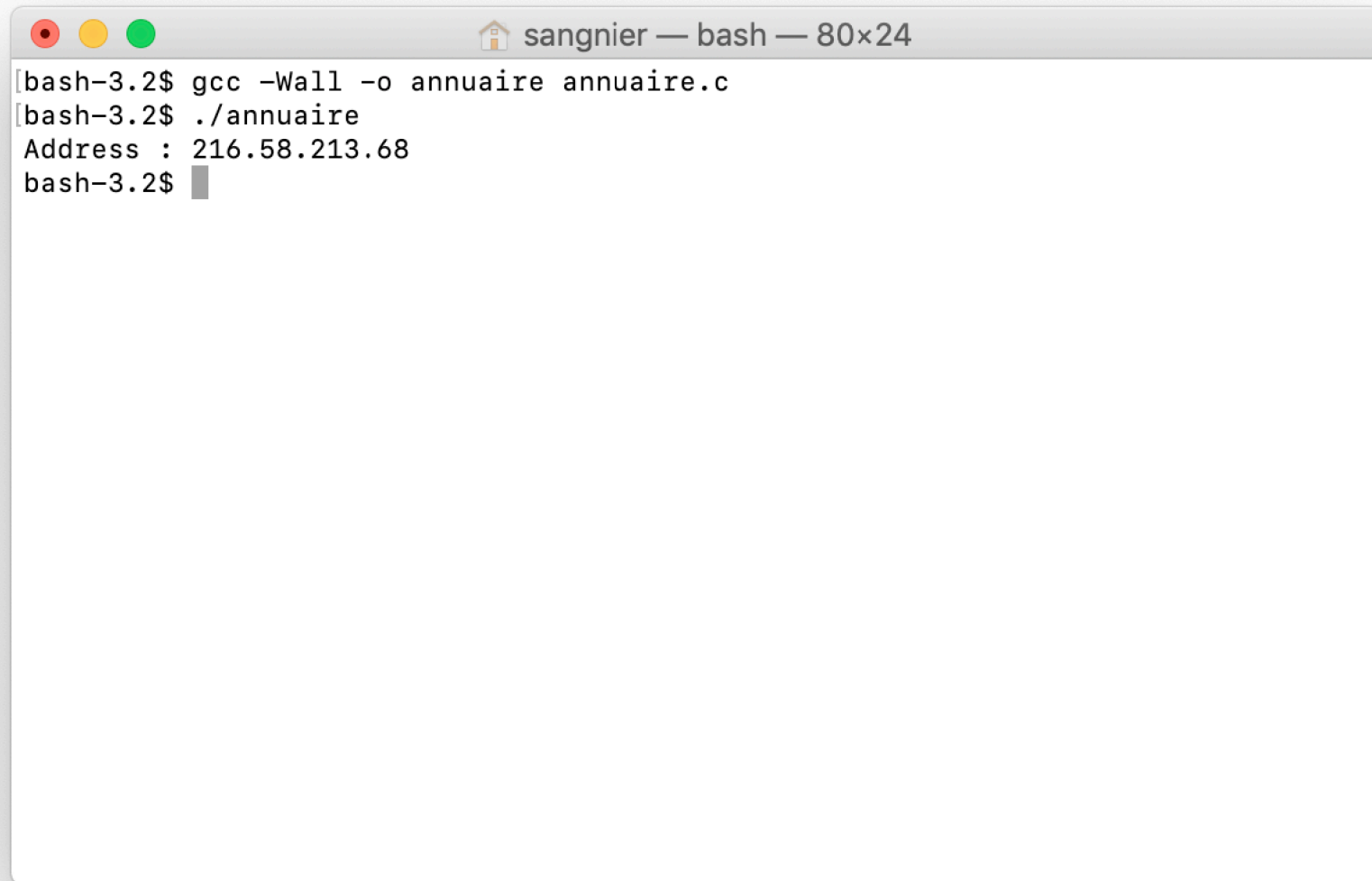
char * inet_ntoa(struct in_addr)

Récupération d'IP

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

int main() {
    struct hostent* host;
    host=gethostbyname("www.google.com");
    if(host==NULL) {
        printf("Unknown\n");
    }
    char **aliases=host->h_aliases;
    while(*aliases!=NULL) {
        printf("Alias : %s\n",*aliases);
        aliases++;
    }
    struct in_addr **addresses=(struct in_addr**)host->h_addr_list;
    while(*addresses!=NULL) {
        printf("Address : %s\n",inet_ntoa(**addresses));
        addresses++;
    }
    return 0;
}
```

Résultat



```
bash-3.2$ gcc -Wall -o annuaire annuaire.c  
bash-3.2$ ./annuaire  
Address : 216.58.213.68  
bash-3.2$
```

The image shows a terminal window titled "sangnier — bash — 80x24". The window contains the following text: "bash-3.2\$ gcc -Wall -o annuaire annuaire.c", "bash-3.2\$./annuaire", "Address : 216.58.213.68", and "bash-3.2\$". The text is displayed in a monospaced font. The terminal window has a standard macOS-style title bar with red, yellow, and green window control buttons on the left.

La fonction getaddrinfo

- Cette fonction est plus générique mais donc plus complexe à utiliser !!!
- C'est la fonction que l'on recommande d'utiliser

```
int getaddrinfo(const char *node,    // "www.example.com" or IP
               const char *service, // "http" or port number
               const struct addrinfo *hints,
               struct addrinfo **res);
```

- On ne décrira que partiellement son utilisation
- Cette fonction permet d'obtenir entre autres choses une liste d'adresses (au sens très large) associées à un nom Internet dans l'annuaire
- En pratique elle remplit une structure de type **struct addrinfo** qui est stockée dans la variable **res**
- On remarque qu'on peut donner aussi un numéro de port (mais on peut mettre NULL, si on veut juste une adresse)
- Cette fonction renvoie 0 si tout se passe bien

La structure struct addrinfo

```
struct addrinfo {  
    int  ai_flags;  
    int  ai_family; // la famille du protocole AF_xxxx  
    int  ai_socktype; // le type de la socket SOCK_xxx  
    int  ai_protocol;  
    socklen_t ai_addrlen; // la longueur de ai_addr  
    struct sockaddr *ai_addr; // l'adresse binaire  
    char*ai_canonname; // le nom canonique  
    struct addrinfo *ai_next; // le pointeur vers la structure suivante  
};
```

- Il s'agit d'une liste chaînée, **ai_next** est le successeur
- Il faut libérer la mémoire de la liste après utilisation grâce à

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

Récupération d'IP (1)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

int main() {
    struct addrinfo *first_info;
    struct addrinfo hints;
    memset(&hints, 0, sizeof hints);
    hints.ai_family = PF_UNSPEC;
    int r=getaddrinfo("www.google.com",NULL,&hints,&first_info);
    if(r==0){
        struct addrinfo *info=first_info;
        while(info!=NULL){
            struct sockaddr *saddr=info->ai_addr;
            if(saddr->sa_family==AF_INET){
                struct sockaddr_in *addressin=(struct sockaddr_in *)saddr;
                struct in_addr address=(struct in_addr) (addressin->sin_addr);
                printf("Address : %s\n",inet_ntoa(address));
            }
            info=info->ai_next;
        }
    }
}
```

Récupération d'IP (2)

```
if(saddr->sa_family==AF_INET6){
    struct sockaddr_in6 *addressin=(struct sockaddr_in6 *)saddr;
    struct in6_addr address=(struct in6_addr)
        (addressin->sin6_addr);
    char*string_address=(char*)malloc(
        sizeof(char)*INET6_ADDRSTRLEN);
    inet_ntop(AF_INET6,&address,string_address,
        INET6_ADDRSTRLEN);
    printf("Address IPv6 : %s\n",string_address);
}
info=info->ai_next;
}
}
freeaddrinfo(first_info);
return 0;
}
```

Récupération d'IP (1)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

int main() {
    struct addrinfo *first_info;
    struct addrinfo hints;
    memset(&hints, 0, sizeof hints);
    hints.ai_family = AF_INET;
    int r=getaddrinfo("www.google.com",NULL,&hints,&first_info);
    if(r==0){
        struct addrinfo *info=first_info;
        while(info!=NULL){
            struct sockaddr *saddr=info->ai_addr;
            struct sockaddr_in *addressin=(struct sockaddr_in *)saddr;
            struct in_addr address=(struct in_addr) (addressin->sin_addr);
            printf("Address : %s\n",inet_ntoa(address));
            info=info->ai_next;
        }
    }
    freeaddrinfo(first_info);
    return 0 ;
}
```

Un dernier exemple sans connaître l'Ip

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <netdb.h>
#include <unistd.h>

int main() {
    struct addrinfo *first_info;
    struct addrinfo hints;
    hints.ai_family = PF_UNSPEC;
    int r=getaddrinfo("lulu.informatique.univ-paris-diderot.fr",NULL,&hints,&first_info);
    if(r==0){
        struct addrinfo *info=first_info;
        int found=0;
        struct sockaddr *saddr;
        struct sockaddr_in *addressin;
        while(info!=NULL && found==0){
            saddr=info->ai_addr;
            if(saddr->sa_family==AF_INET){
                addressin=(struct sockaddr_in *)saddr;
                found=1;
            }
            info=info->ai_next;
        }
        if(found==1){
            struct sockaddr_in adress_sock;
            adress_sock.sin_family = AF_INET;
            adress_sock.sin_port = htons(7);
            adress_sock.sin_addr=addressin->sin_addr;
            int descr=socket(PF_INET,SOCK_STREAM,0);
            int r2=connect(descr,(struct sockaddr *)&adress_sock,
                          sizeof(struct sockaddr_in));
```