Interroger l’annuaire – getaddrinfo

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

i**nt 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

PR-APITCPC 8

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

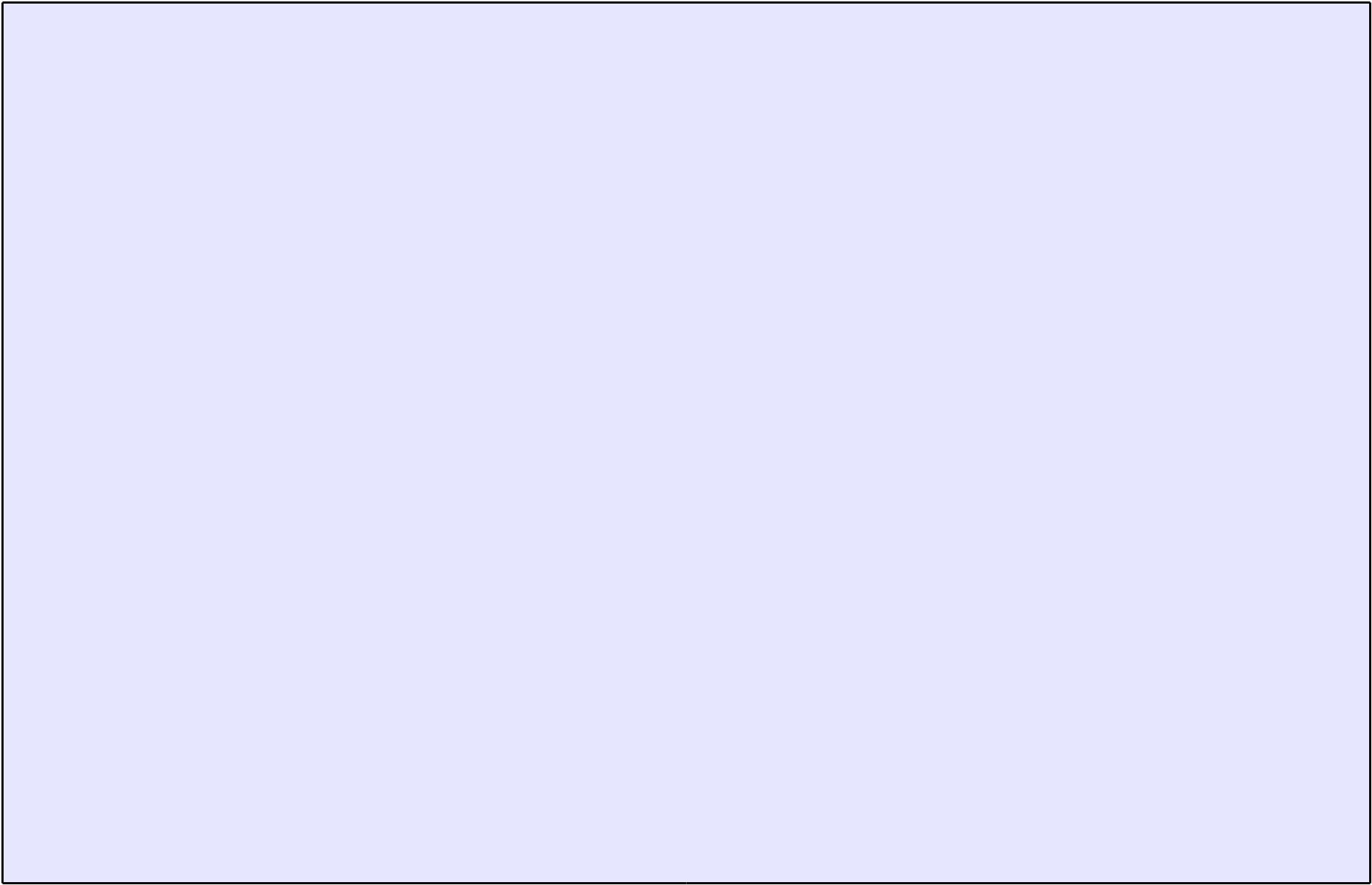
**struct addrinfo \*ai\_next;** // le pointeur vers la structure suivante

**};**

* 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 \*);**

PR-APITCPC 9

Récupération d'IP (1)

****

**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));**

**}**

**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;**

**}**

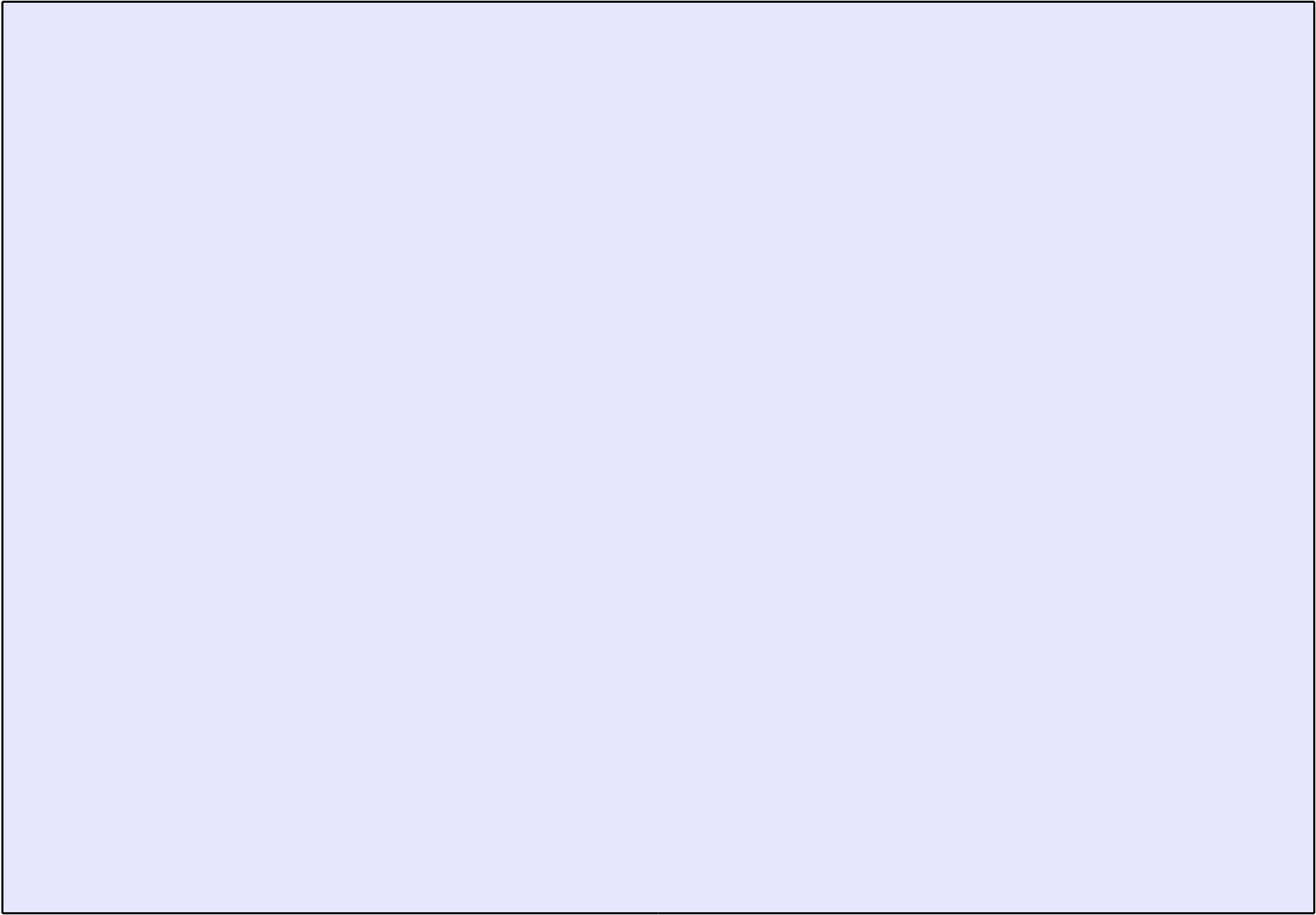
**}**

**freeaddrinfo(first\_info);**

**return 0;**

}

Récupération d'IP (2)

****

**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;**

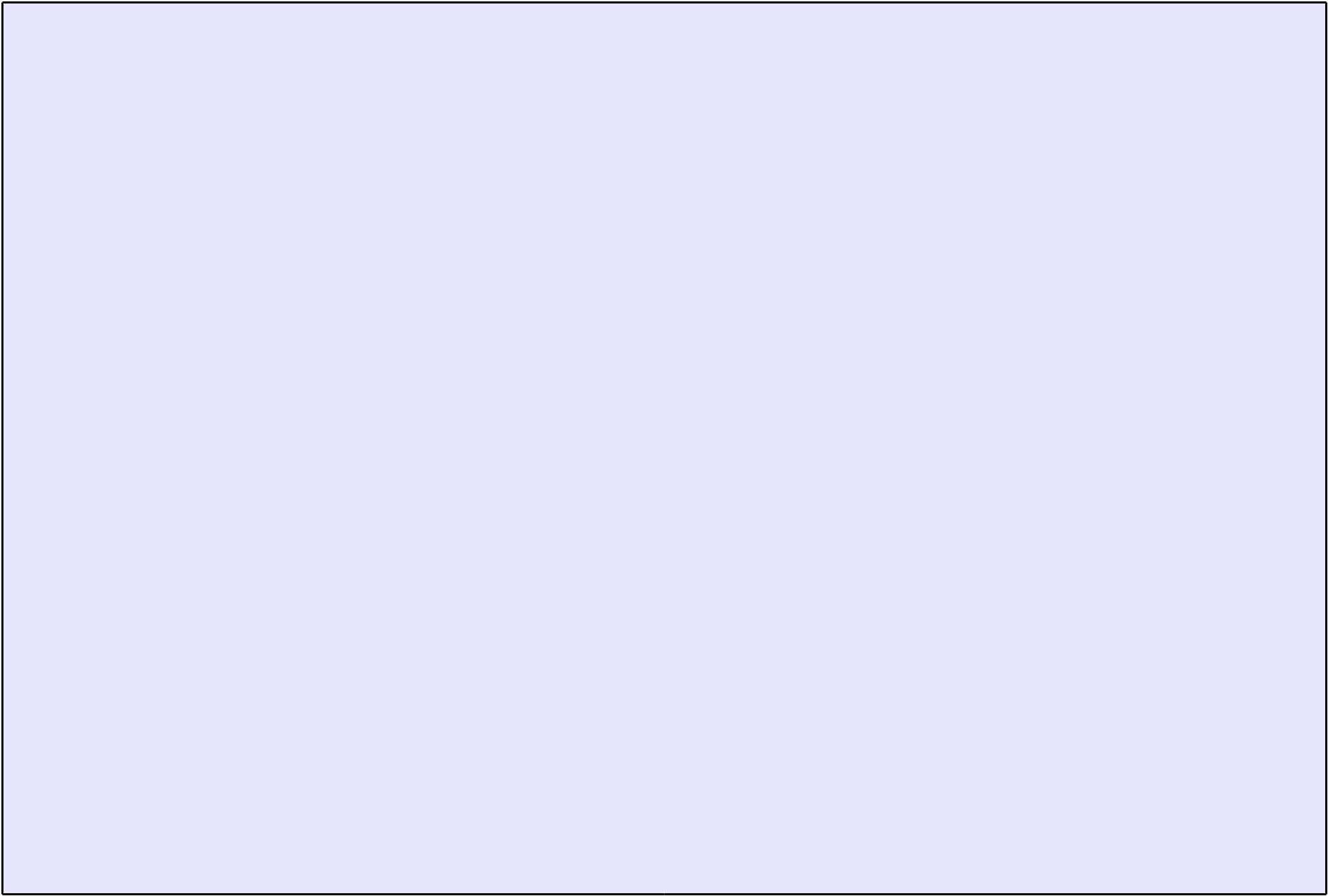
**}**

**}**

**freeaddrinfo(first\_info);**

**return 0;**

**}**

Un dernier exemple sans connaître l’Ip

**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;**

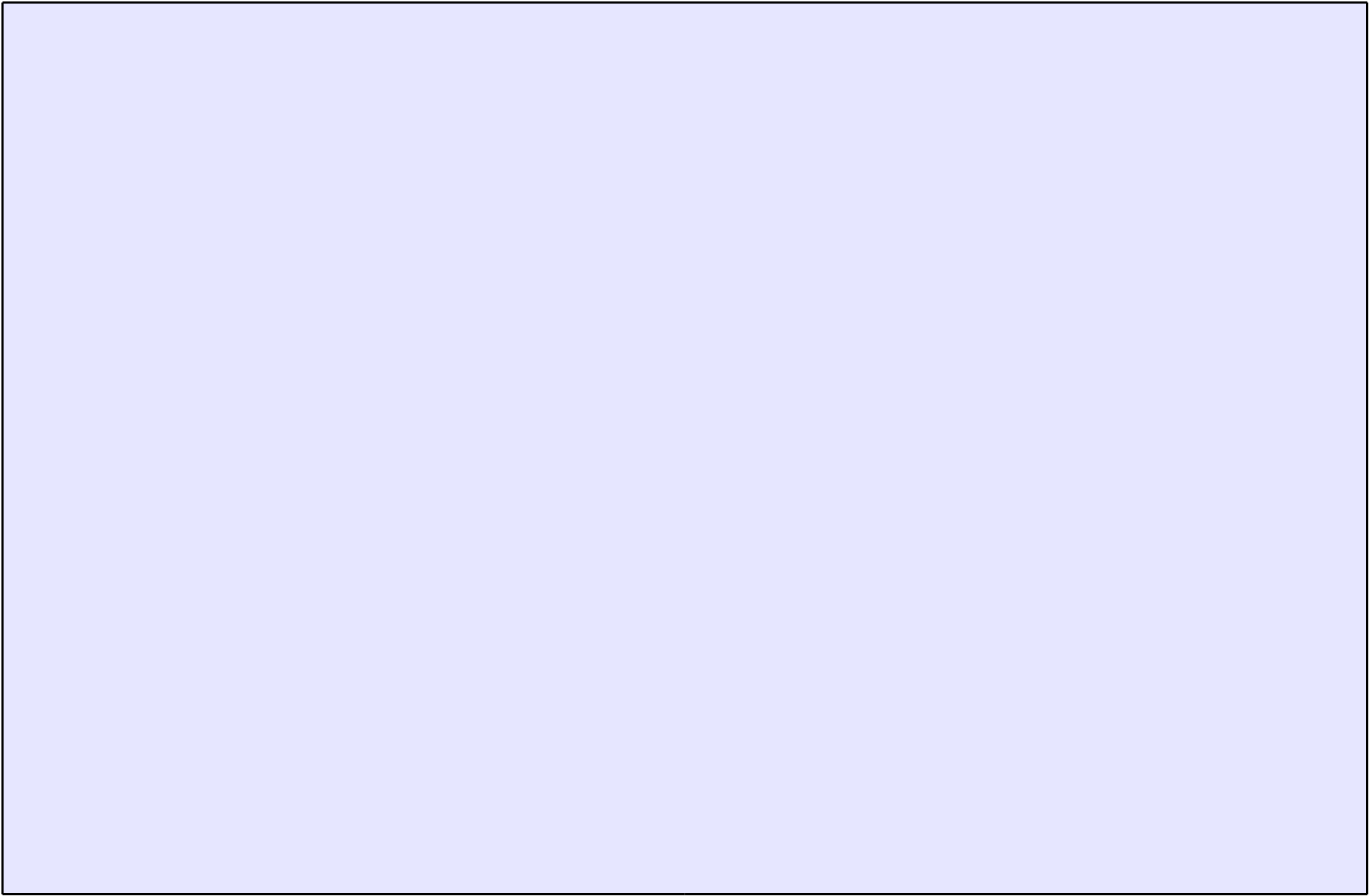
**adress\_sock.sin\_port = htons(7);**

**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));**

Côté serveur – bind, listen, accept

**int main() {**

**int sock=socket(PF\_INET,SOCK\_STREAM,0);**

**struct sockaddr\_in address\_sock;**

**address\_sock.sin\_family=AF\_INET;**

**address\_sock.sin\_port=htons(4242);**

**address\_sock.sin\_addr.s\_addr=htonl(INADDR\_ANY);**

**int r=bind(sock,(struct sockaddr \*)&address\_sock,sizeof(struct sockaddr\_in)); if(r==0){**

**r=listen(sock,0);**

**while(1){**

**struct sockaddr\_in caller;**

**socklen\_t size=sizeof(caller);**

**int sock2=accept(sock,(struct sockaddr\*)&caller,&size); if(sock2>=0){**

**char \*mess="Yeah!\n";**

**send(sock2,mess,strlen(mess)\*sizeof(char),0);**

**char buff[100];**

**int recu=recv(sock2,buff,99\*sizeof(char),0); buff[recu]='\0';**

**printf("Message recu : %s\n",buff);**

**}**

**close(sock2);**

**}**

**}**

**return 0;**

**}**

21

Récupération d'informations

**int main() {**

****

**int sock=socket(PF\_INET,SOCK\_STREAM,0);**

**struct sockaddr\_in address\_sock;**

**address\_sock.sin\_family=AF\_INET;**

**address\_sock.sin\_port=htons(4242);**

**address\_sock.sin\_addr.s\_addr=htonl(INADDR\_ANY);**

**int r=bind(sock,(struct sockaddr\*)&address\_sock,sizeof(struct sockaddr\_in));**

**if(r==0){**

**r=listen(sock,0);**

**while(1){**

**struct sockaddr\_in caller;**

**socklen\_t size=sizeof(caller);**

**int sock2=accept(sock,(struct sockaddr\*)&caller,&size); if(sock2>=0){**

**printf("Port de l'appelant: %d\n",ntohs(caller.sin\_port));**

**printf("Adresse de l'appelant: %s\n",inet\_ntoa(caller.sin\_addr));**

**}**

**close(sock2);**

**}**

**}**

**return 0;**

**}**

PR - API TCP C 22