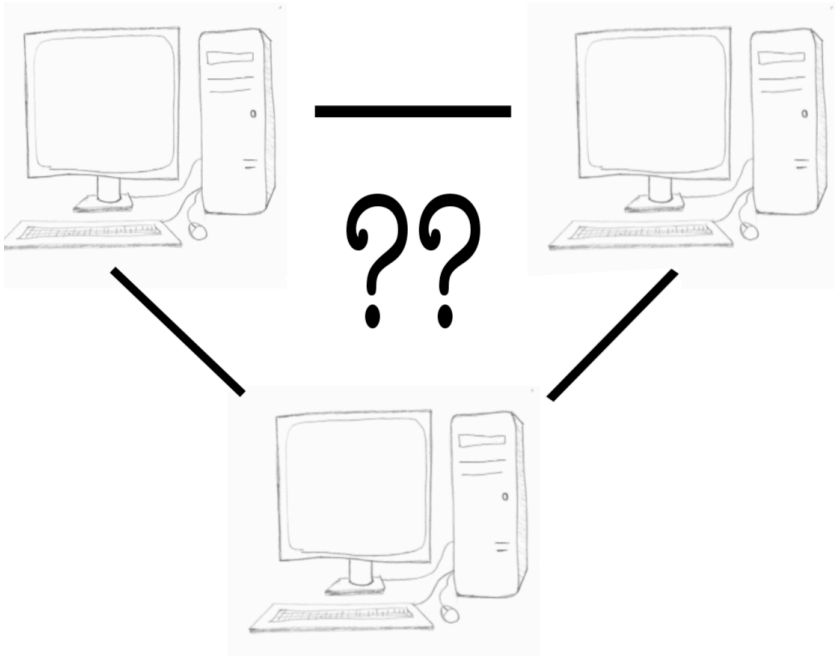
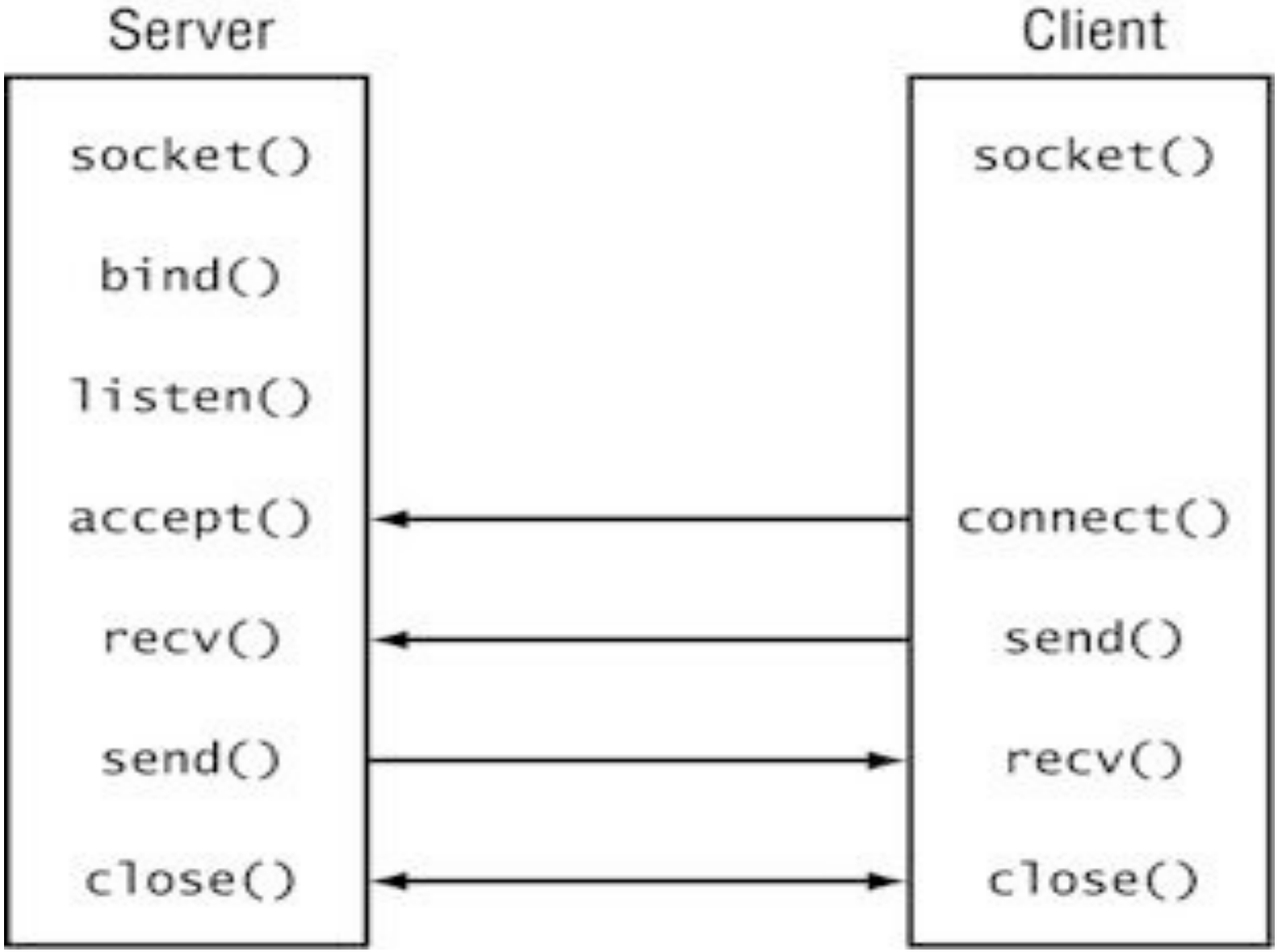
# PROGRAMMATION RÉSEAU

Arnaud Sangnier
sangnier@irif.fr

## API TCP C - II



# Schéma Client-Serveur en C



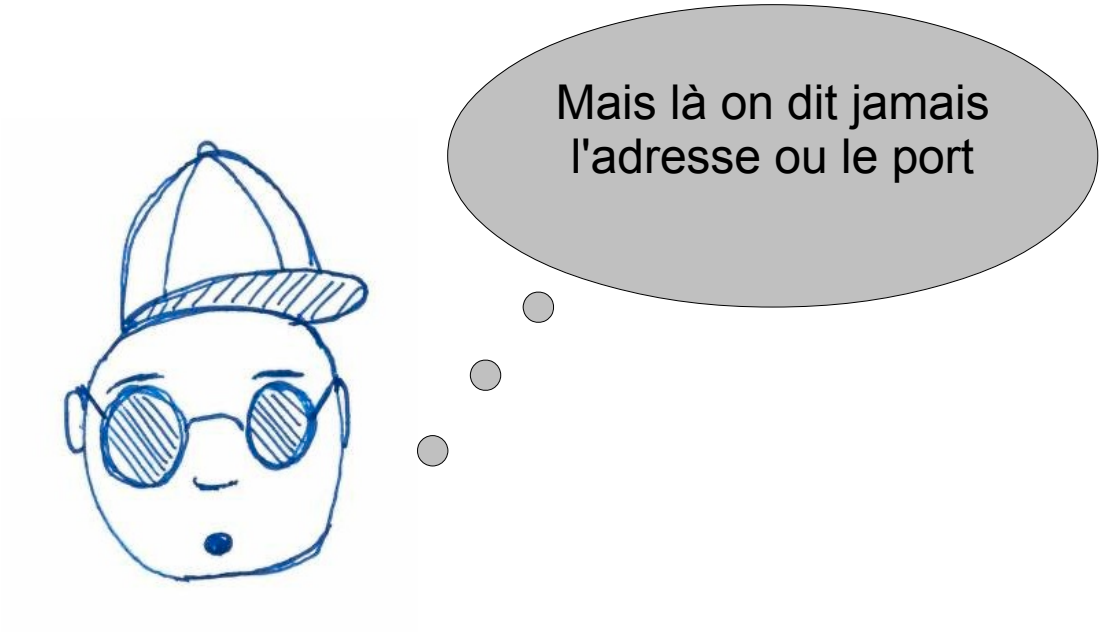
# Création d'une socket

- La création d'une socket se fait grâce à :
  - **#include <sys/socket.h >**

    **int socket(int domaine, int type, int protocol)**
- Pour nous :
  - **domaine** vaudra **PF_INET** (pour IPv4) ou **PF_INET6** (pour IPv6)
  - **type** vaudra **SOCK_STREAM** (pour les sockets TCP)
  - **protocol** spécifie le procole de communication (mais pour TCP, on peut mettre 0 et le protcole est chosi de façon automatique)
- L'entier renvoyé sera le descripteur  utilisé pour comuniquer

# Accès à une machine

**int socket(int domaine, int type, int protocol)**




- Et oui !!! On va le préciser après

# Côté client

- Il faut demander l'établissement d'une connexion à l'aide de la fonction suivante :

**int connect(int socket, const struct sockaddr *adresse, socklen_t longueur);**

- On connecte la socket correspondante
- Pour rappel dans les objets de type struct sockaddr_in, on met une adresse et un port
- Pour le dernier argument, si on est en IPv4 et que adresse est de type **struct sockaddr_in**, on pourra mettre **sizeof(struct sockaddr_in)**
- Quand on a fini la communication, on peut fermer le descripteur de socket avec la commande
  - **int close(int fildes);**

# Pour communiquer

- On va envoyer et recevoir des caractères sur le descripteur de socket
- Pour recevoir  on va utiliser
- **int recv(int sockfd, void *buf, int len, int flags);**
  - Remplit le buffer **buf**
  - **len** est la taille maximale de **buf**
  - **flags** sera la plupart du temps mis à **0**
  - renvoie le nombre de données reçu (-1 si erreur et 0 si la connexion est fermée)
- Pour envoyer on va utiliser
- **int send(int sockfd, const void *msg, int len, int flags);**
  - Même principe que recv len est la taille en octet de msg
  - flags est aussi mis à 0 ici.
- On pourrait aussi utiliser **read** et **write**

# Exemple

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

int main() {
  struct sockaddr_in adress_sock;
  adress_sock.sin_family = AF_INET;
  adress_sock.sin_port = htons(4242);
  inet_aton("127.0.0.1",&adress_sock.sin_addr);

  int descr=socket(PF_INET,SOCK_STREAM,0);
  int r=connect(descr,(struct sockaddr *)&adress_sock,
                sizeof(struct sockaddr_in));
  if(r!=-1){
    char buff[100];
    int size_rec=recv(descr,buff,99*sizeof(char),0);
    buff[size_rec]='\0';
    printf("Caracteres recus : %d\n",size_rec);
    printf("Message : %s\n",buff);
    char *mess="SALUT!\n";
    send(descr,mess,strlen(mess),0);
    close(descr);
  }
  return 0;
}
```

# Tester le client précédent

- telnet est permet de simuler un client, on peut aussi utiliser un autre outil : netcat
- **netcat lulu 7** est équivalent à **telnet lulu 7**
- Mais netcat peut aussi simuler un serveur
- Si on fait **netcat -l 4242** , on a un serveur tcp qui attend une connexion sur le port 4242 et tout ce qui est tapé ensuite et envoyé au client
- Du coup vous pouvez tester le client précédent sur votre machine en lançant d'abord un terminal avec **netcat -l 4242** et dans un autre terminal vous lancez le client

# Pour communiquer (2)

- On va envoyer et recevoir des caractères sur le descripteur de socket
- Pour recevoir  on va utiliser
- **ssize_t read(int filedes, void *buf, size_t nbyte);**
  - Remplit le buffer **buf**
  - **nbyte** est la taille maximale de **buf**
  - renvoie le nombre de données reçu (-1 si erreur et 0 si la connexion est fermée)
- Pour envoyer on va utiliser
- **ssize_t write(int filedes, void *buf, size_t nbyte);**
  - Même principe que read **nbyte** est la taille en octet de buf

# Exemple

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

int main() {
  struct sockaddr_in adress_sock;
  adress_sock.sin_family = AF_INET;
  adress_sock.sin_port = htons(4242);
  inet_aton("127.0.0.1",&adress_sock.sin_addr);

  int descr=socket(PF_INET,SOCK_STREAM,0);
  int r=connect(descr,(struct sockaddr *)&adress_sock,
                sizeof(struct sockaddr_in));
  if(r!=-1){
    char buff[100];
    int size_rec=read(descr,buff,99*sizeof(char));
    buff[size_rec]='\0';
    printf("Caracteres recus : %d\n",size_rec);
    printf("Message : %s\n",buff);
    char *mess="SALUT!\n";
    write(descr,mess,strlen(mess));
    close(descr);
  }
  return 0;
}
```