



Communications via sockets

Les Sockets en C

- Les principes de la communication
- En pratique

Principes

Reseau ⇔ ensemble d'immeubles

adresse ⇔ adresse IP

Numéro d'appartement ⇔ numéro de port



Principes

2 Façons de communiquer



**Mode non connecté
(datagram)**





Mode connecté (STREAM)



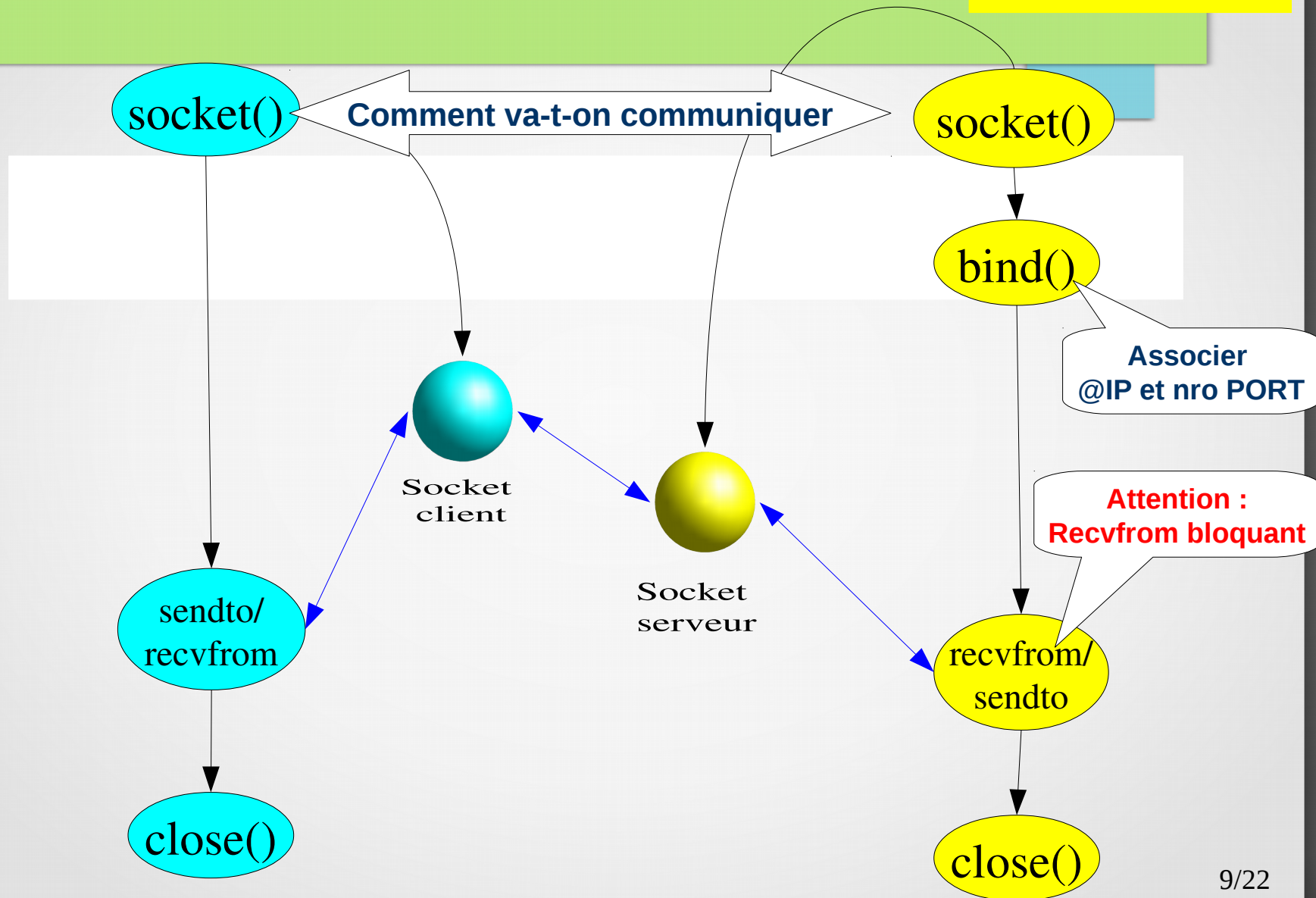
En pratique



Socket en mode non-connecté

Client UDP

Serveur UDP



Fonctions et Paramètres

- L'ensemble des ports réservés se trouve (sous linux)dans le fichier : **/etc/services**

```
chargen 19/udp
ftp-data 20/tcp
ftp 21/tcp
fsp 21/udp fspd
ssh 22/tcp # SSH Remote Login Protocol
ssh 22/udp
telnet 23/tcp
smtp 25/tcp mail
time 37/tcp timserver
time 37/udp timserver
rpl 39/udp resource # resource location
nameserver 42/tcp name # IEN 116
whois 43/tcp nickname
tacacs 49/tcp # Login Host Protocol (TACACS)
tacacs 49/udp
re-mail-ck 50/tcp # Remote Mail Checking Protocol
re-mail-ck 50/udp
domain 53/tcp # Domain Name Server
domain 53/udp
```

La commande socket()

```
#include <sys/types.h>
#include <sys/socket.h>
int socket(int domain, int type, int protocol);
```

PF_INET
(Protocol Family)



A line connects the text 'PF_INET (Protocol Family)' to the 'domain' parameter of the 'socket()' function in the code above.

SOCK_DGRAM
SOCK_STREAM



A line connects the text 'SOCK_DGRAM' and 'SOCK_STREAM' to the 'type' parameter of the 'socket()' function in the code above.

IPPROTO_TCP
IPPROTO_UDP
0



A line connects the text 'IPPROTO_TCP', 'IPPROTO_UDP', and '0' to the 'protocol' parameter of the 'socket()' function in the code above.

La commande bind()

```
#include <sys/types.h>
#include <sys/socket.h>
int bind(int sockfd, struct sockaddr *my_addr, socklen_t addrlen);
```

Le descripteur retourné
par la commande socket

La structure contenant les
informations relatives à l'adresse et
au numéro de port du serveur

Taille de la
structure
d'adresse

La structure sockaddr

```
struct sockaddr{  
    u_char sa_len;    // longueur de l'adresse  
    u_char sa_family; // famille de protocole  
    char sa_data[14]; // adresse  
};
```


La structure sockaddr_in

```
#include <netinet.h>
#include <arpa/inet.h>
struct sockaddr_in {
    sa_family_t    sin_family;    /* famille d'adresses : AF_INET
                                   sur 16 bits */
    u_int16_t      sin_port;      /* port dans l'ordre d'octets
                                   réseau */
    struct in_addr sin_addr;      /* adresse Internet
    */
    char sin_zero[8];            /* initialise à zéro */
};

struct in_addr {
    u_int32_t      s_addr; /* Adresse dans l'ordre d'octets réseau */
                                   /* INADDR_ANY afin d'ecouter sur toutes */
                                   /* les interface presentes */
};
```

Se mettre d'accord entre lilliputiens

htonl : host to network long

-> de l'hôte vers le réseau sur 4 octets

htons : host to network short

-> de l'hôte vers le réseau sur 2 octets

ntohl : network to host long

-> du réseau vers l'hôte sur 4 octets

ntohs : network to host short

-> du réseau vers l'hôte sur 2 octets

Commande sendto()

```
#include <sys/types.h>
#include <sys/socket.h>
int  sendto( int s, const void *msg, size_t len,
              int flags, const struct sockaddr *to,
              socklen_t tolen);
```

s : la socket

msg : l'adresse de ce qui doit être envoyé

len : le nombre d'octets à envoyer

flags : options diverses (vaut généralement 0)

to : adresse d'une structure **sockaddr_in** initialisée avec les informations relatives au destinataire. Cette structure est ensuite castée → (struct sockaddr *) devant.

tolen : Taille de la structure d'adressage du destinataire.

Commande recvfrom()

```
#include <sys/types.h>
#include <sys/socket.h>
int  recvfrom(int s, void *msg, int len,
               int flags, const struct sockaddr *from,
               socklen_t *fromlen);
```

s : la socket

msg : l'adresse où sera stocké ce qui doit être reçu

len : le nombre d'octets maximum que peut supporter **msg**

flags : options diverses (vaut généralement 0)

from : adresse d'une structure **sockaddr_in**. Elle sera initialisée avec les informations de celui qui envoie les données. Cela permettra de lui répondre par la suite.

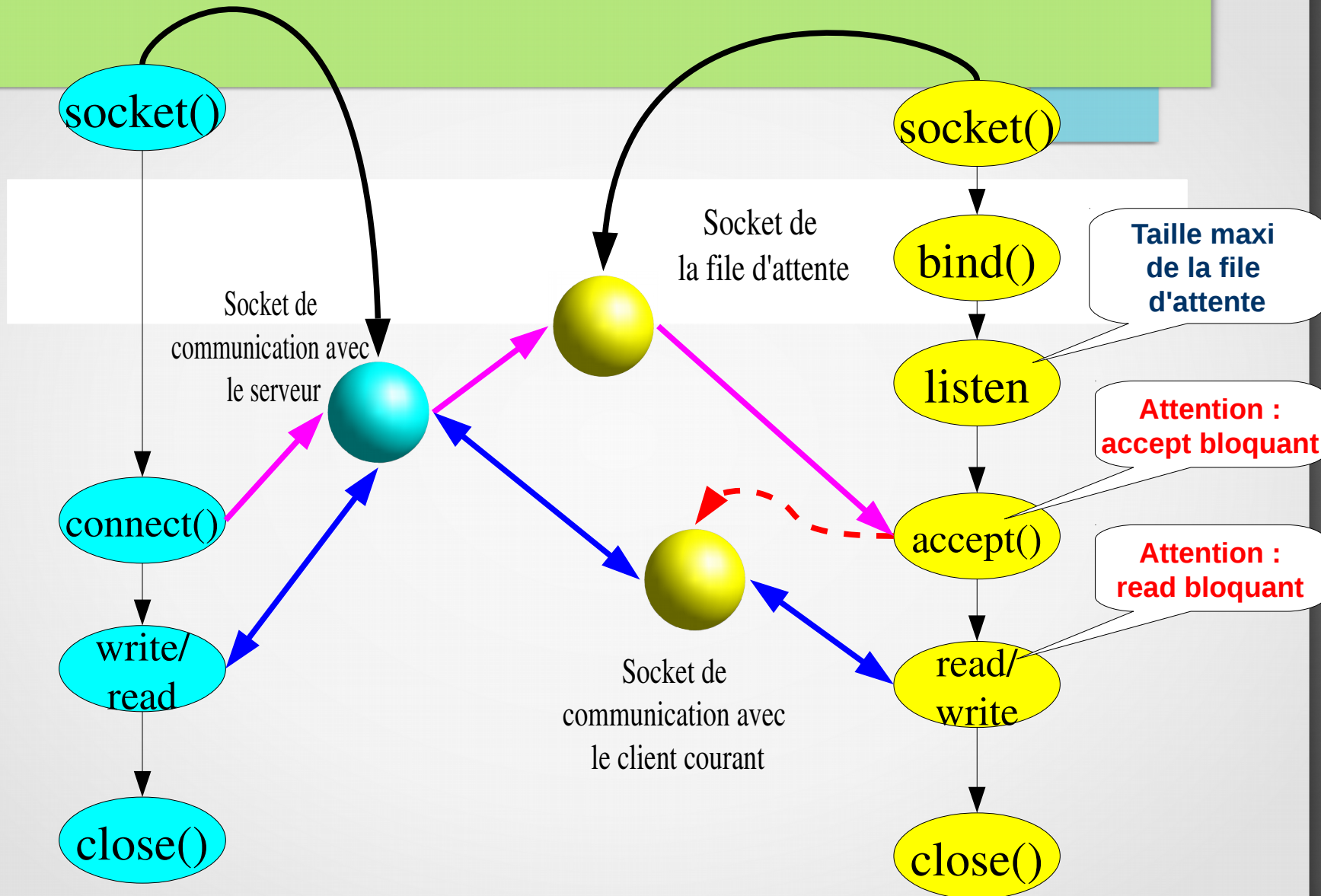
tolen : adresse d'une variable qui contiendra la taille de la structure d'adressage de celui qui envoie un message.



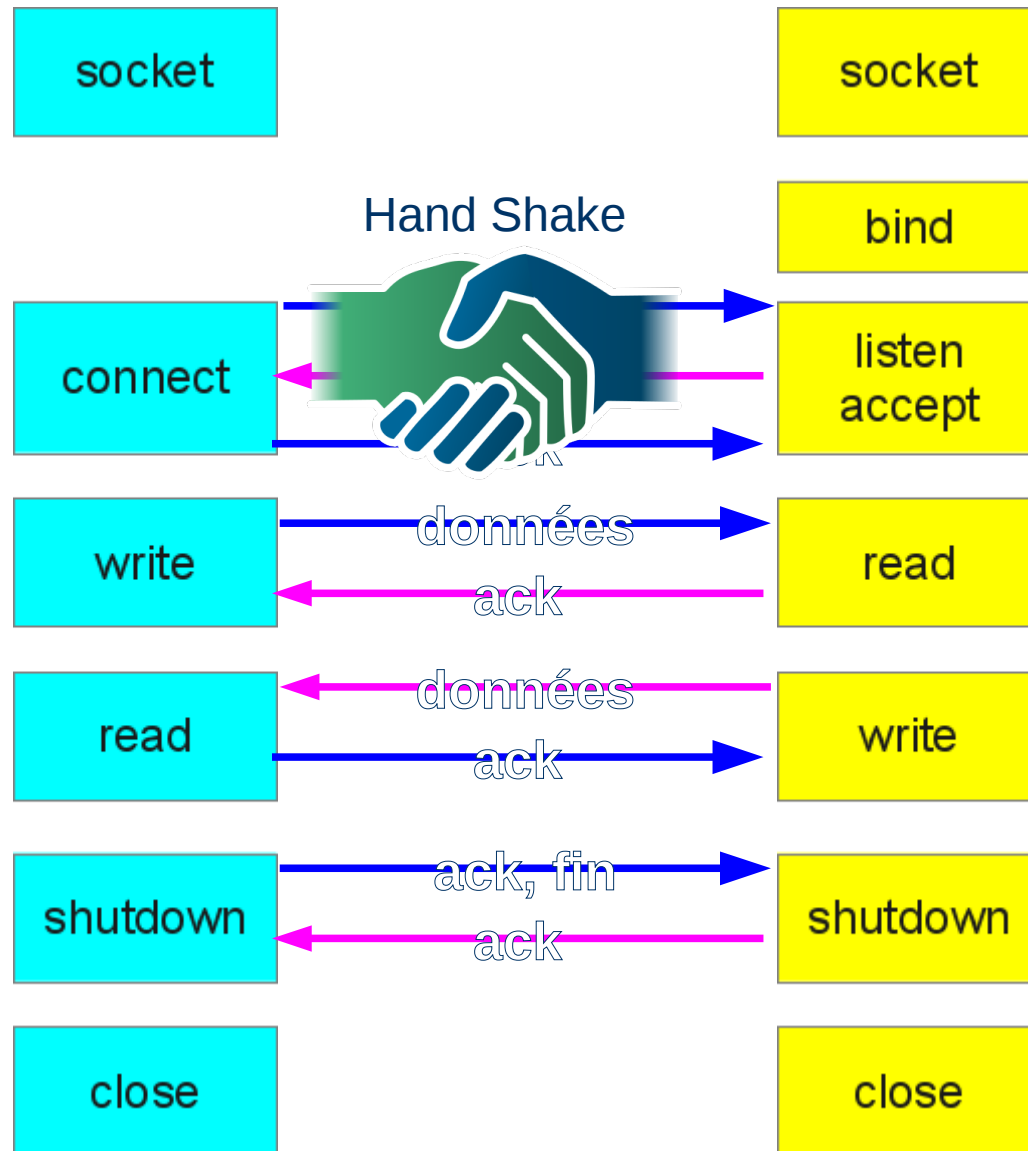
Socket en mode connecté

Client TCP

Serveur TCP



Vue avec les échanges réseau



Commande connect()

```
#include <sys/types.h>
#include <sys/socket.h>
int connect(int s, struct sockaddr *serv_addr, socklen_t len);
```

s : la socket

serv_addr : adresse d'une structure **sockaddr_in** initialisée avec les données du serveur (IP, port, etc).

len : Taille de la structure d'adressage du serveur.

Commande accept()

```
#include <sys/types.h>
#include <sys/socket.h>
int accept(int s, struct sockaddr *adresse, socklen_t *len);
```

s : la socket

adresse : adresse d'une structure **sockaddr_in**. Elle sera initialisée avec les informations de celui qui demande une connexion. Cela permettra de créer la socket de communication.

len : adresse d'une variable qui contiendra la taille de la structure d'adressage de celui qui envoie un message.

La valeur retournée par **accept** est la socket à utiliser pour communiquer avec le client.