Sockets Réseaux

Définition

- Canal d'accès aux réseaux : point de communication
- Point d'accès aux couches « protocole » du modèle OSI

Quels réseaux ?

- TCP/IP (IPv4 et IPv6)
- Autres protocoles (AX25, X25, …)
- Fichier d'échange

Sockets Réseaux

Qui les utilise ?

TCP/IP … Donc tous les acteurs des réseaux LAN/WAN

Exemples

- Protocoles HTTP, FTP, …
- Bibliothèques AJAX, WEB-Socket
- Protocoles personnels (BOT, Backdoor, ...)

Sockets (Posix -Berkeley)

Primitives du langage C

Fonction principale :

sock = socket (Univers, Protocole, Options);

Univers : Internet, Fichiers, …
Protocole : Connecté (SOCK_STREAM), Non Connecté (SOCK_DGRAM)
Options : 0

Sock : variable DESCRIPTEUR du SOCKET

Correspond à un emplacement dans une liste de connexions socket gérée par le système d'exploitation.

Sockets (Posix -Berkeley)

Primitives du langage C

Fonctions annexes :

bind (*paramètres*);

listen (*paramètres*);

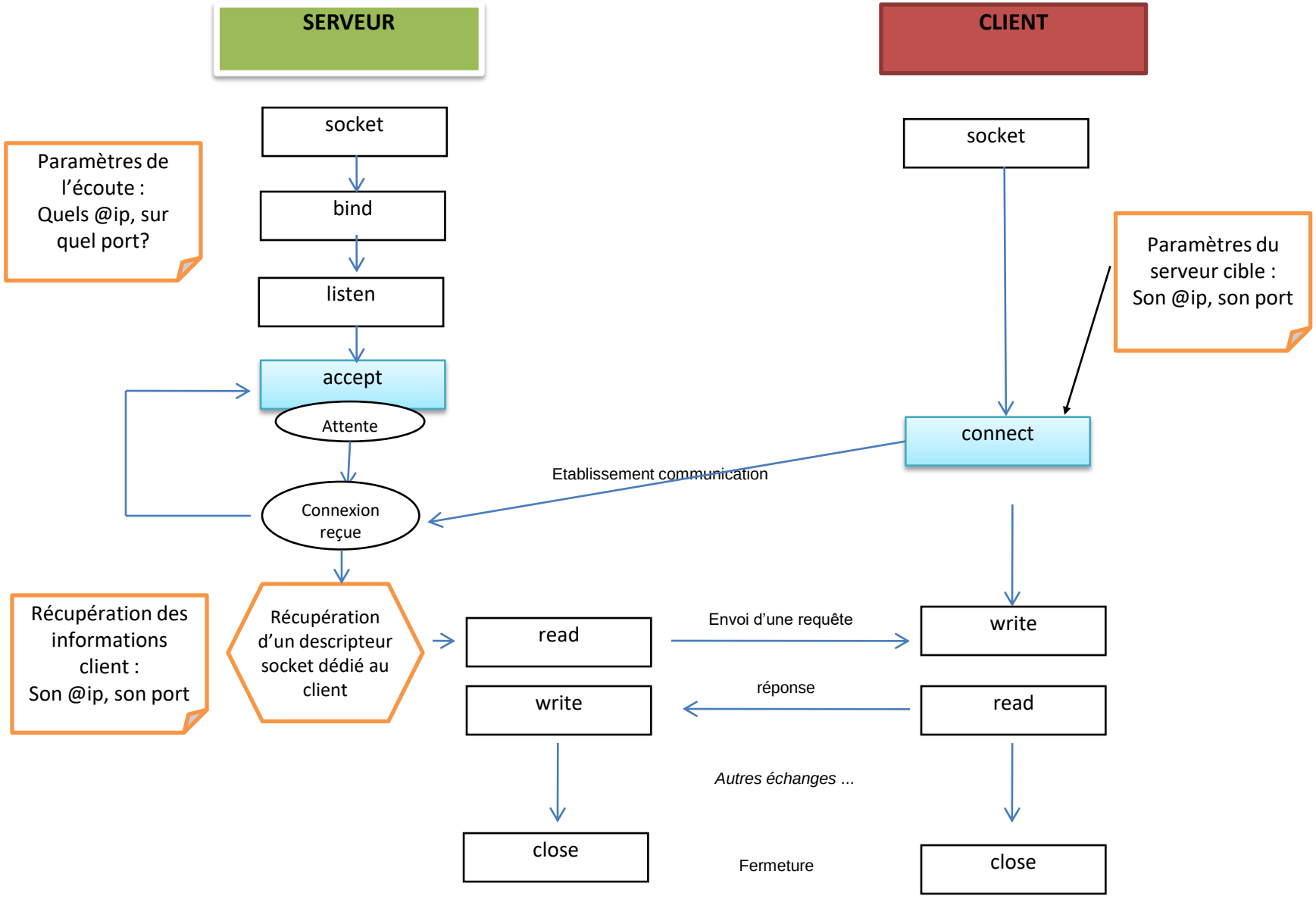
Objectif : Affiner les réglages du Socket.

Exemple :

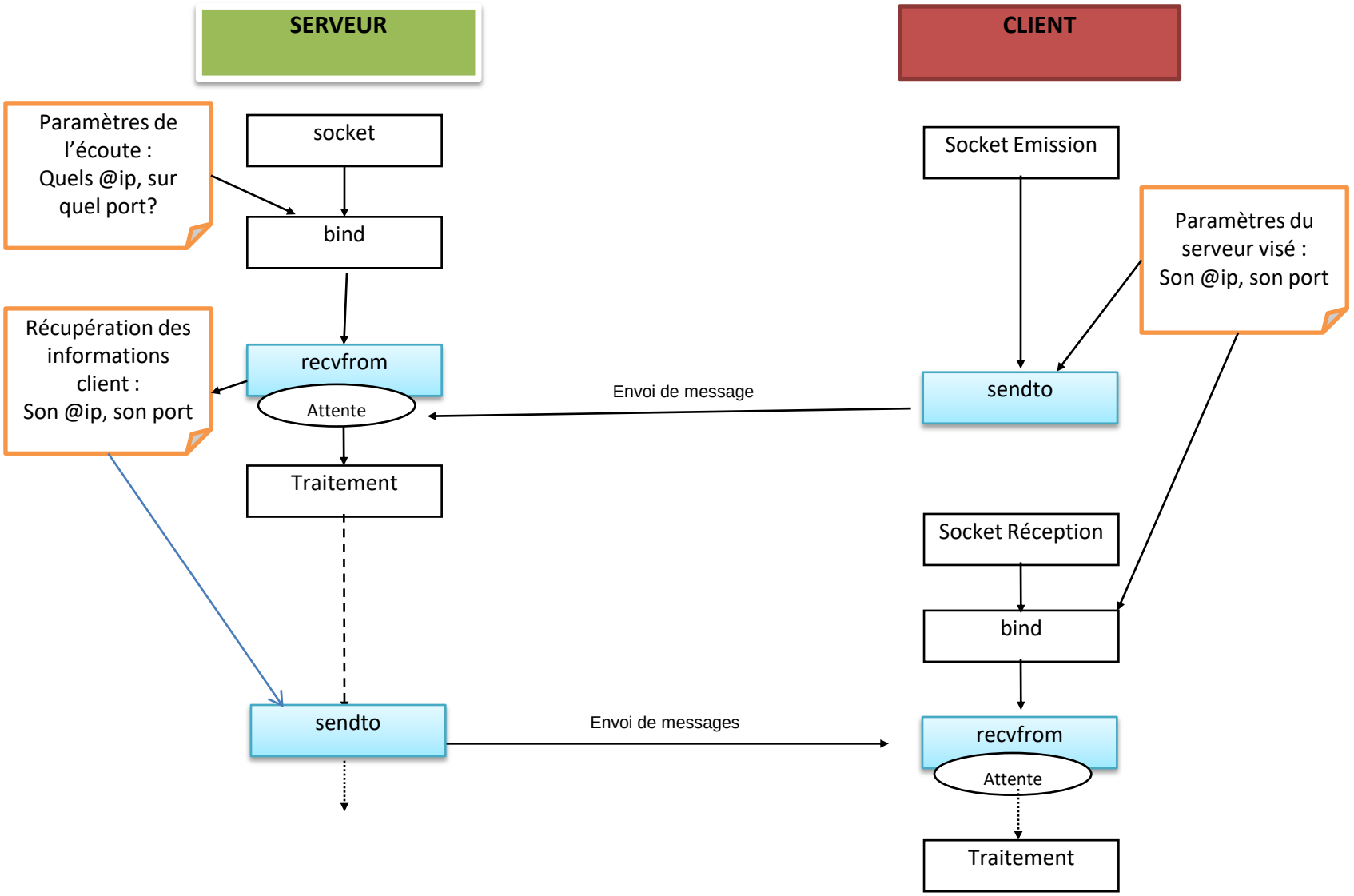
Pour un socket TCP/IP : Port de connexion

Pour un socket fichier : Nom du fichier d'échange

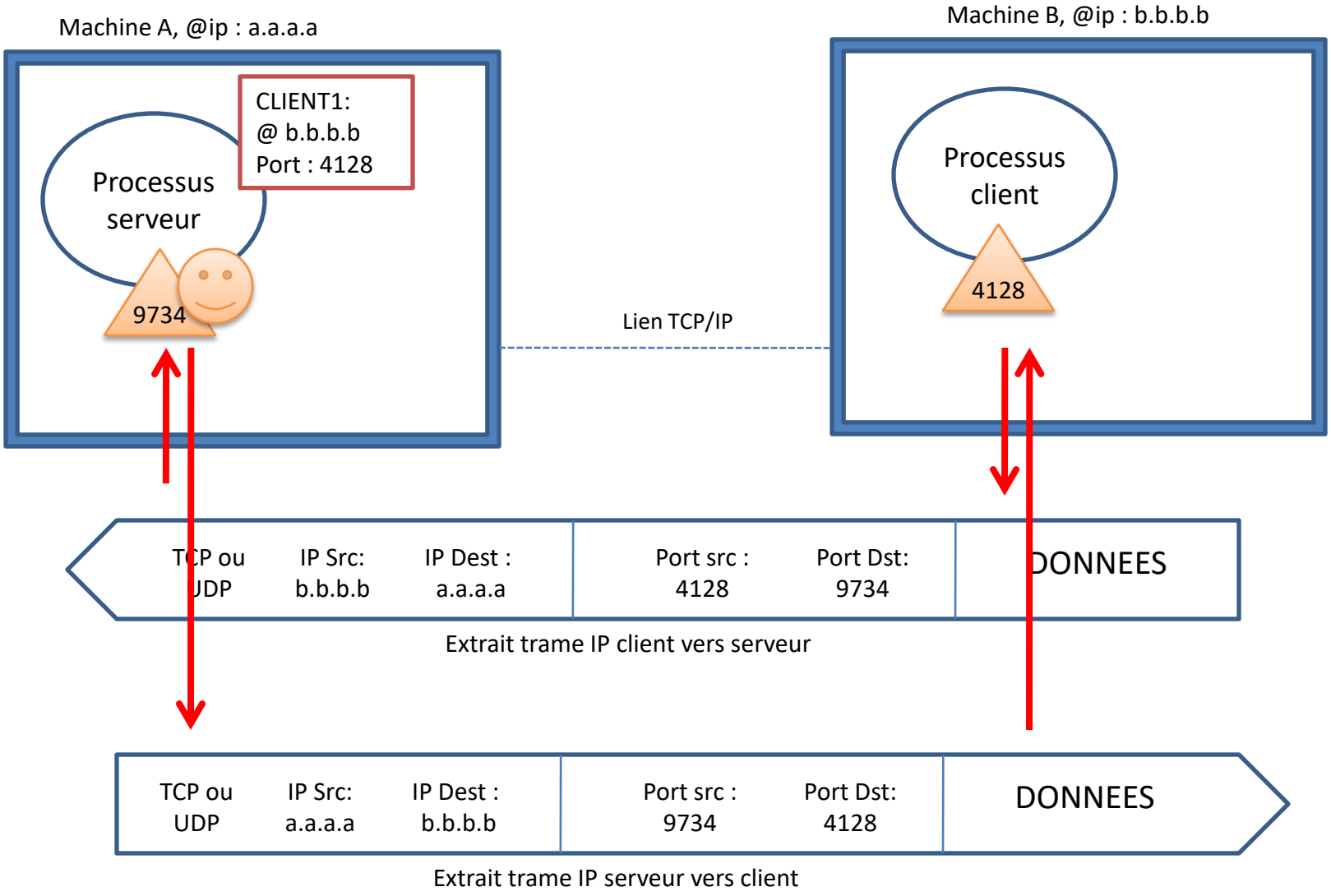
Enchainement des primitives en mode STREAM (AF_UNIX et AF_INET TCP)



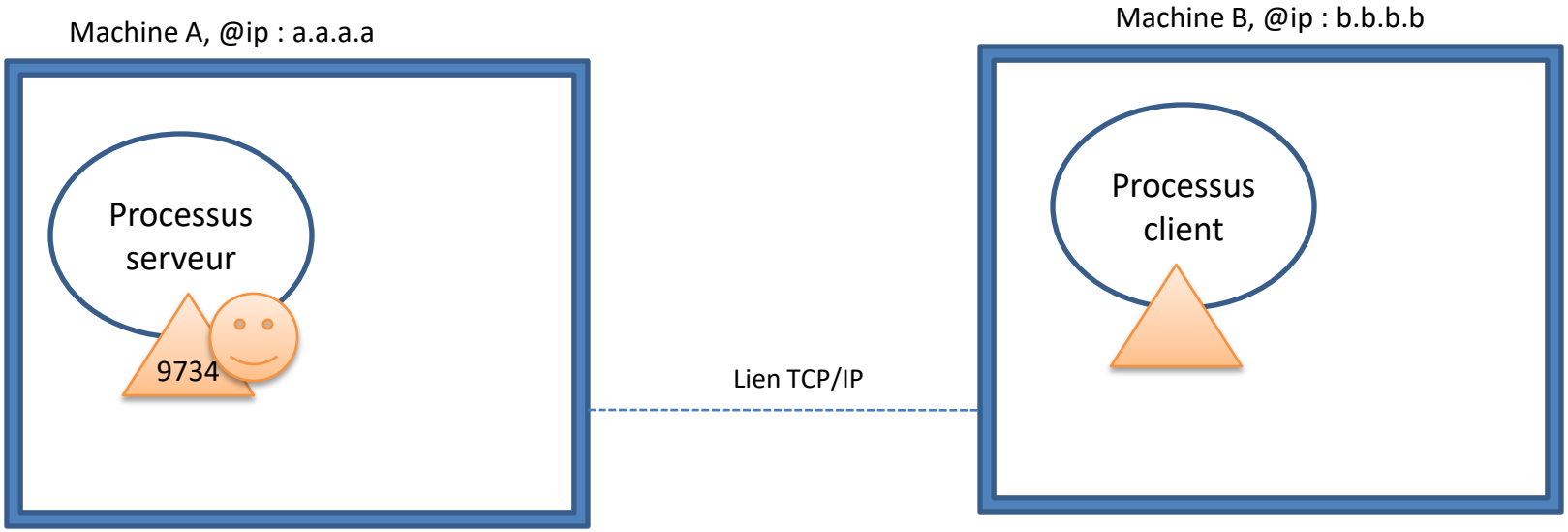
Enchainement des primitives en mode DGRAM (AF_UNIX et AF_INET UDP)



AF_INET : Rôle du port



AF_INET : Rôle du port



Extrait trame IP client vers serveur

Extrait trame IP serveur vers client

AF_INET : Serveur TCP

```
int sockfd_serveur, sockfd_client;
int long_serveur, long_client;
struct sockaddr_in adresse_serveur;
struct sockaddr_in adresse_client;

sockfd_serveur = socket(AF_INET, SOCK_STREAM, 0);

adresse_serveur.sin_family = AF_INET;
adresse_serveur.sin_addr.s_addr = inet_addr("127.0.0.1"); //CAS PARTICULIER
// adresse_serveur.sin_addr.s_addr = INADDR_ANY;
adresse_serveur.sin_port = 9734;
long_serveur = sizeof(adresse_serveur);
bind (sockfd_serveur, (struct sockaddr *)&adresse_serveur, long_serveur);

listen (sockfd_serveur, 5);

while(1)
{
  char ch;
  printf("\n Serveur en attente de requete ...");
  long_client=sizeof(adresse_client);
  sockfd_client = accept (sockfd_serveur, (struct sockaddr *)&adresse_client, &long_client);
  read(sockfd_client,&ch, 1);
  ch++;
  write(sockfd_client, &ch, 1);
  close (sockfd_client);
}
```

AF_INET : clientTCP

```
int sockfd;
 int longueur;
 struct sockaddr_in adresse;
 int resultat;
 char ch='A';

sockfd = socket(AF_INET, SOCK_STREAM, 0);

 adresse.sin_family = AF_INET;
 adresse.sin_addr.s_addr = inet_addr("127.0.0.1");
 adresse.sin_port = 9734;
 longueur = sizeof(adresse);
 resultat = connect(sockfd, (struct sockaddr *)&adresse, longueur);

 if (resultat == -1 ) {
                perror("Erreur connect");
                exit(1);
                }

 write (sockfd, &ch, 1);
 read (sockfd, &ch, 1);
 printf("\nCaractere provenant du serveur : %c", ch); fflush(stdout);
 close(sockfd);
```

Les structures d'adresse des sockets

Structure générique utilisée par les primitives : obligation du CAST (voir sources C)

```
struct sockaddr {
        u_short  sa_family ;       /* Famille d'adresse */
        char     sa_data[]         /* Données du socket*/
} ;
```

Domaine UNIX (AF_UNIX)

```
struct sockaddr_un {
        short    sun_family ;      /* pour UNIX : AF_UNIX */
        char     sun_path[]        /* Nom du fichier d échange 108c maxi*/
} ;
```

Domaine Internet (AF_INET)

```
struct sockaddr_in {
        short            sin_family ; /* pour internet : AF_INET     */
        u_short          sin_port ;  /* numero de port utilisé        */
        struct in_addr   sin_addr   /* Structure d'adresse Internet.  */
        };

struct in_addr {
        ulong s_addr ; /* INADDR_ANY pour accepter toutes adresses */
        } ;
```

Des outils pour simplifier

- Bibliothèque « maison »

  Exemple : une classe c++
- QTSocket
- Bibliothèque PHP

Exemple : Classe « maison »

Programme principal:

```
sServeurTCP  ecouter(9736);
ecouter.accepter();

sClientTCP client( ecouter.getClient() );

client.envoyer("Bonjour");
cout << client.recevoir() << endl;
```

```
class sClientTCP {

    public:
        sClientTCP(); // Client à connecter
        sClientTCP(int sd); // Client déjà connecté
        ~sClientTCP();
        bool connecter(string ip, int port); // Tenter une connexion

        bool envoyer(string);

        string recevoir();
        string recevoir(int nbmax);

        string getErreur();

    private:
        WSADATA wsaData;
        SOCKADDR_IN client;
        SOCKET sockClient;
        int port;
        string msg;

};
```

```
class sServeurTCP {

    public:
        sServeurTCP(int port);
        ~sServeurTCP();
        bool accepter(); // Attendre une connexion
        int getClient(); // Renvoit le descripteur de connexion client

        string getIPclient(); // Obtenir l'IP d'origine du dernier client
        unsigned short getPortClient();// Obtenir le port d'origine du dernier client
        string getErreur();

    private:
        WSADATA wsaData;
        SOCKADDR_IN client;
        SOCKET sockClient, sockEcoute; // Descripteurs du client et d'écoute du serveur
        int port; // Port à utiliser
        string msg;

};
```