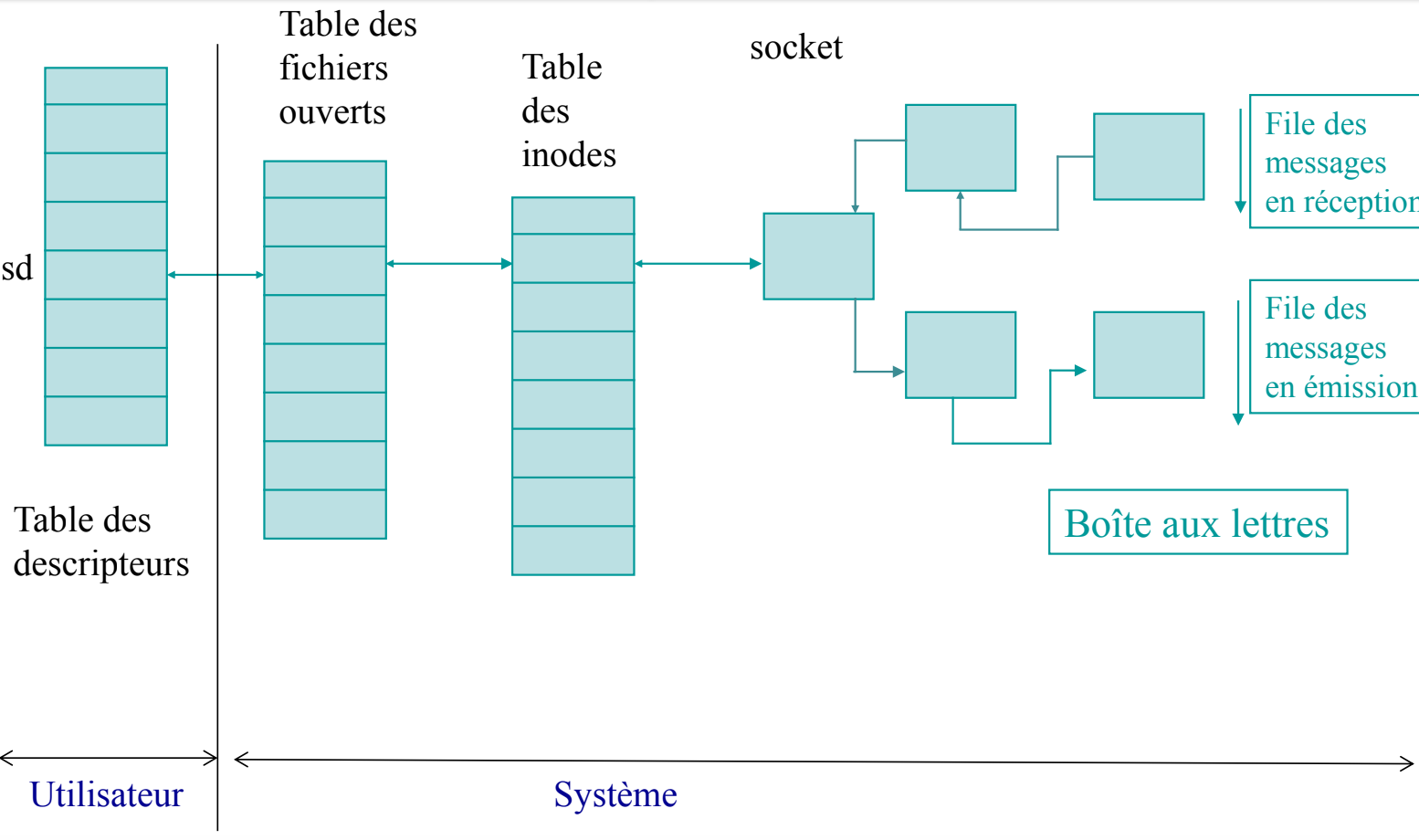
PROGRAMMATION RÉSEAU

**Socket** : interface de programmation permettant d’envoyer et recevoir des données (bidirectionnel).



* Différents types de sockets en fonction des fonctionnalités du S.E :
  + pour envoyer des msg IP : SOCK\_RAW

Plus pour utiliser des protocoles bas niveaux (type ICMP),

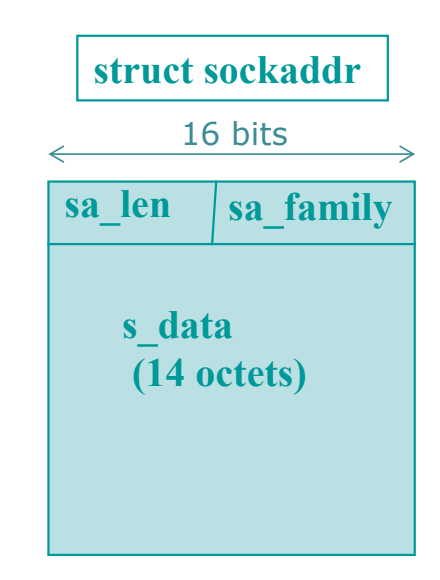
* + pour envoyer des msg TCP : SOCK\_STREAM

Comme il s’agit d’une connexion TCP, on a bien une transmission fiable, ordonnée et sans duplication. On est en mode connecté

* + pour envoyer des msg UDP : SOCK\_DGRAM

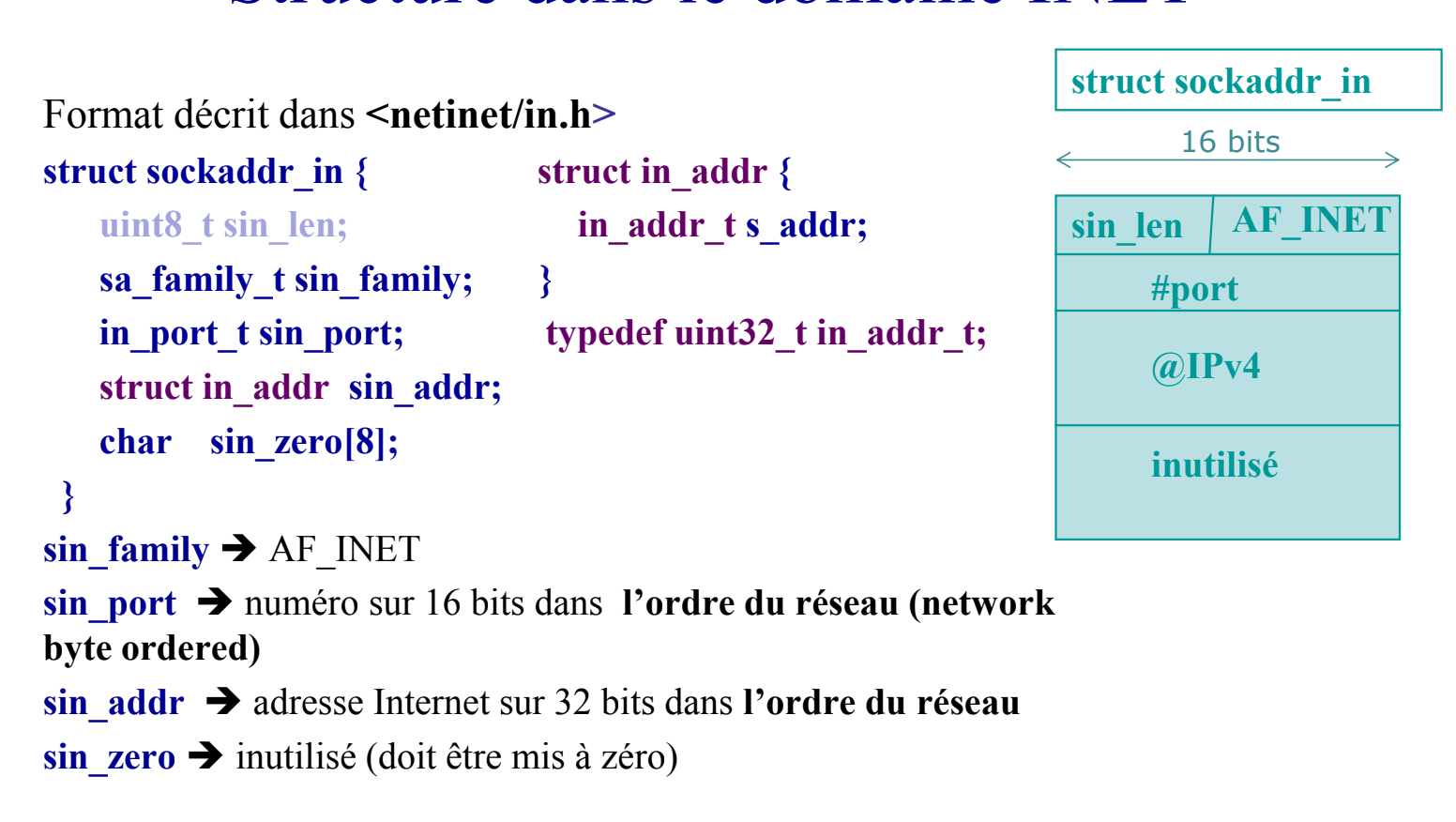
Transmission non fiable, non ordonnée, avec possibilité de duplication

* Différentes **familles (8 octets)** de sockets, en fonction des conventions des adresses permettant d’identifier les destinataires, sont définies :
  + AF-UNIX : pour des sockets locaux à la machine, pour des applications locals
  + AF\_INET : pour des sockets avec des IPv4 pour identifier le destinataire
  + AF\_INET6 : pour les sockets avec IPv6 pour identifier le destinataire.
* **Structure générique** d’une socket :



Cf diapo pour chaque structure en fonction des familles.

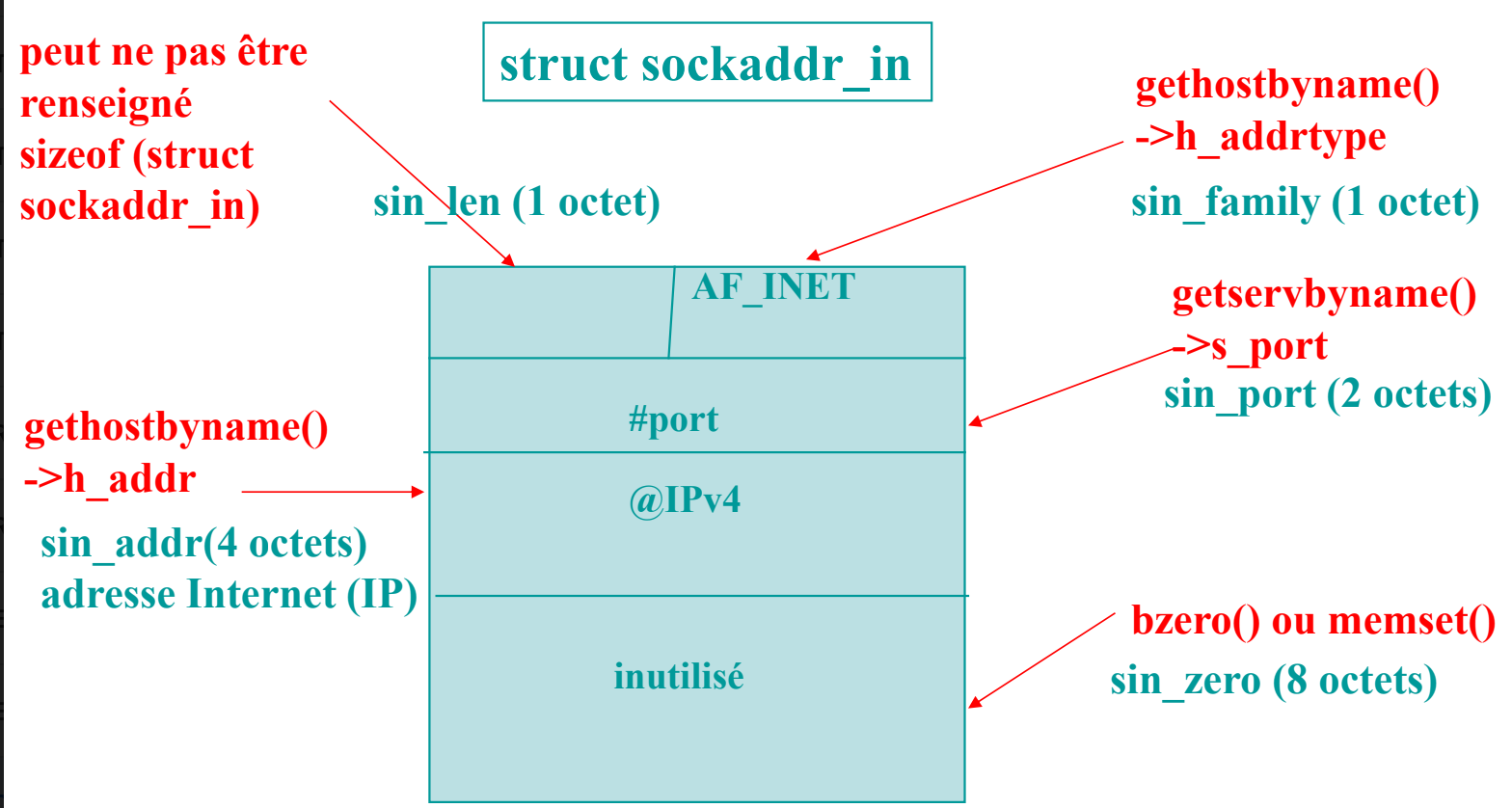
On utilisera surtout des SOCKADDR\_IN (ipv4) :



Ici, le numéro de port est retourné par getServByName() (fonction système) cf diapo pour la classe en détail. Il identifie la porte d’entrée

Ici, addr correspond à l’ipv4 qui identifie de manière unique la machine. En réponse à gethostbyname(nomMachine) par ex (cf diapo pour plus détails sur cette fonction) qui retourne une structure hostent donnant le type de l’adr, une liste d’adresses ip

VOici comment remplir la structure sockaddr\_in plus précisément :



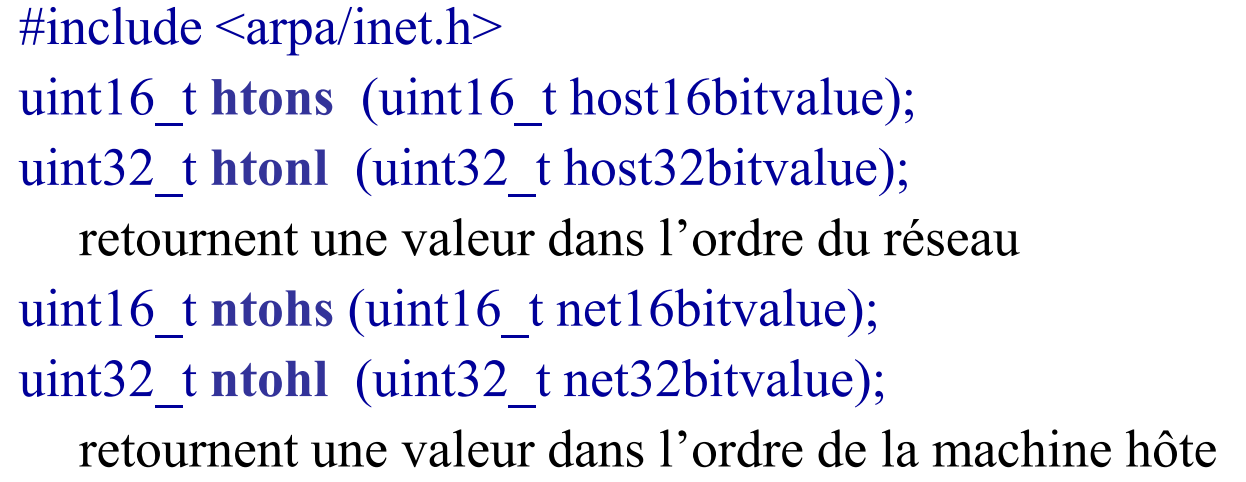
=> permet d’avoir une structure, pour créer le socket.

Format de représentation des données par les hôtes

Little Indian / Big endian

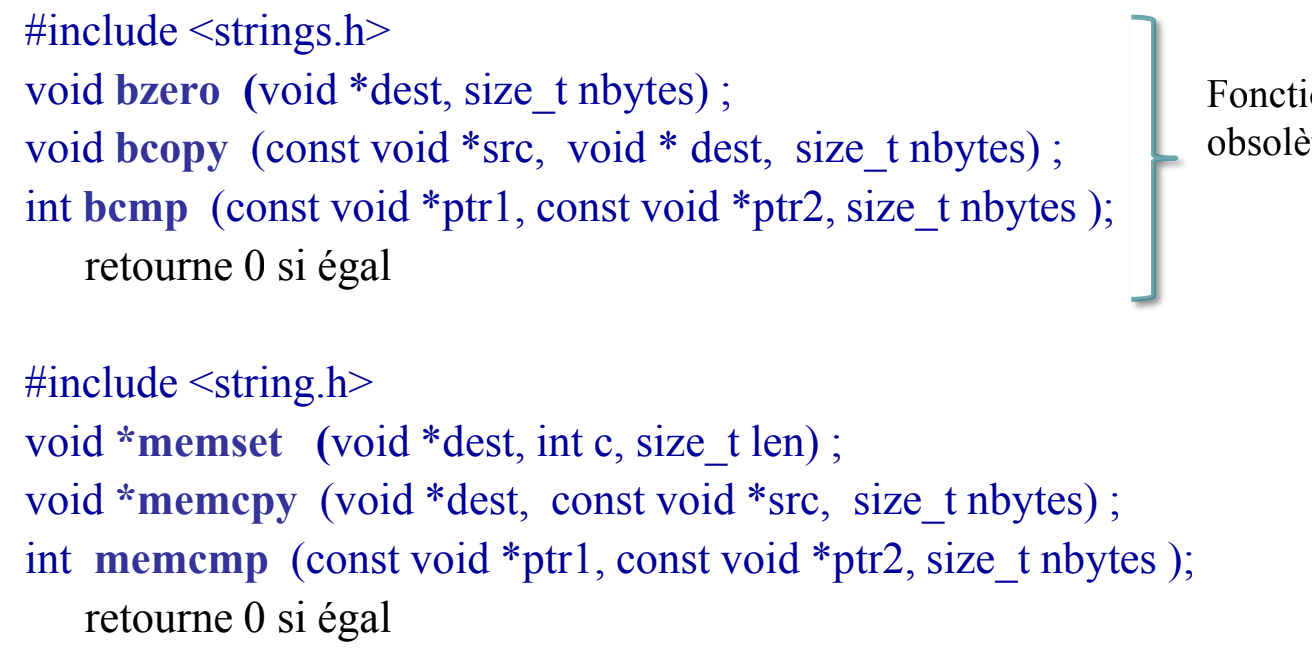
*On peut utiliser des fonctions de conversion des entiers :*

host to network to network short = htons



*On peut utiliser des fonctions de manipulation d’octets :*

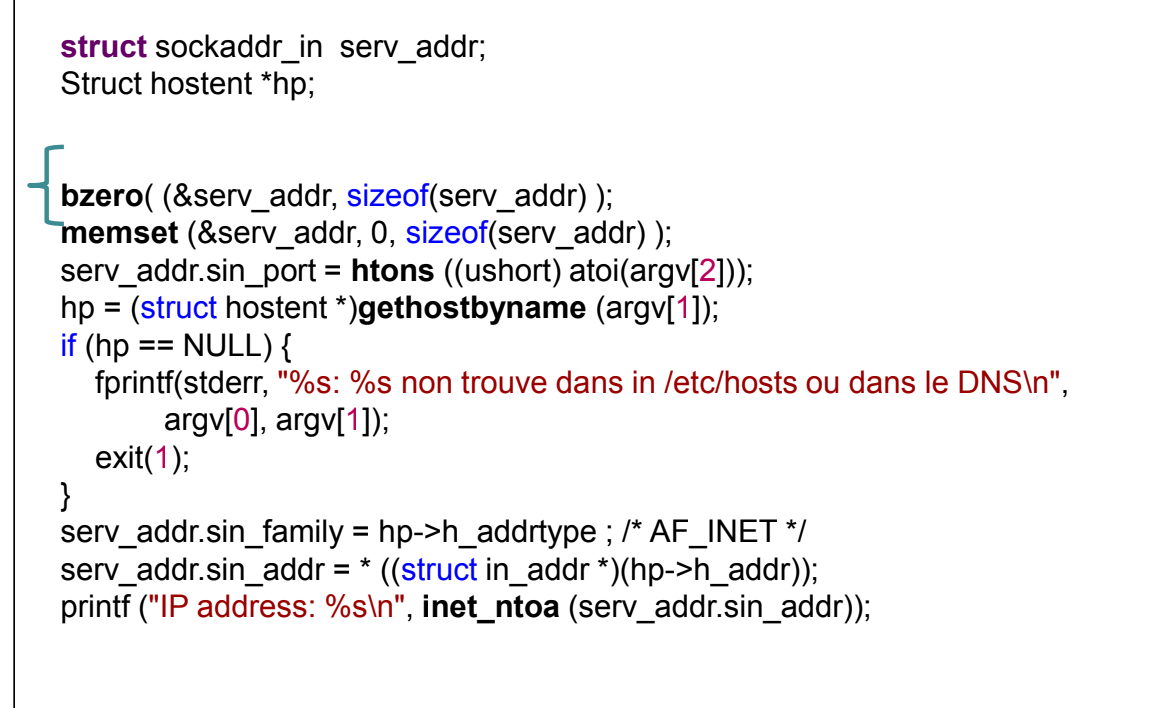
Initialiser une structure à zéro pour éviter les effets de bord :

memset

On peut utiliser des fonctions de manipulation d’adresses

Ex : convertir du format ip au format chaîne de caractères.

Exemple d’initialisation d’un socket :



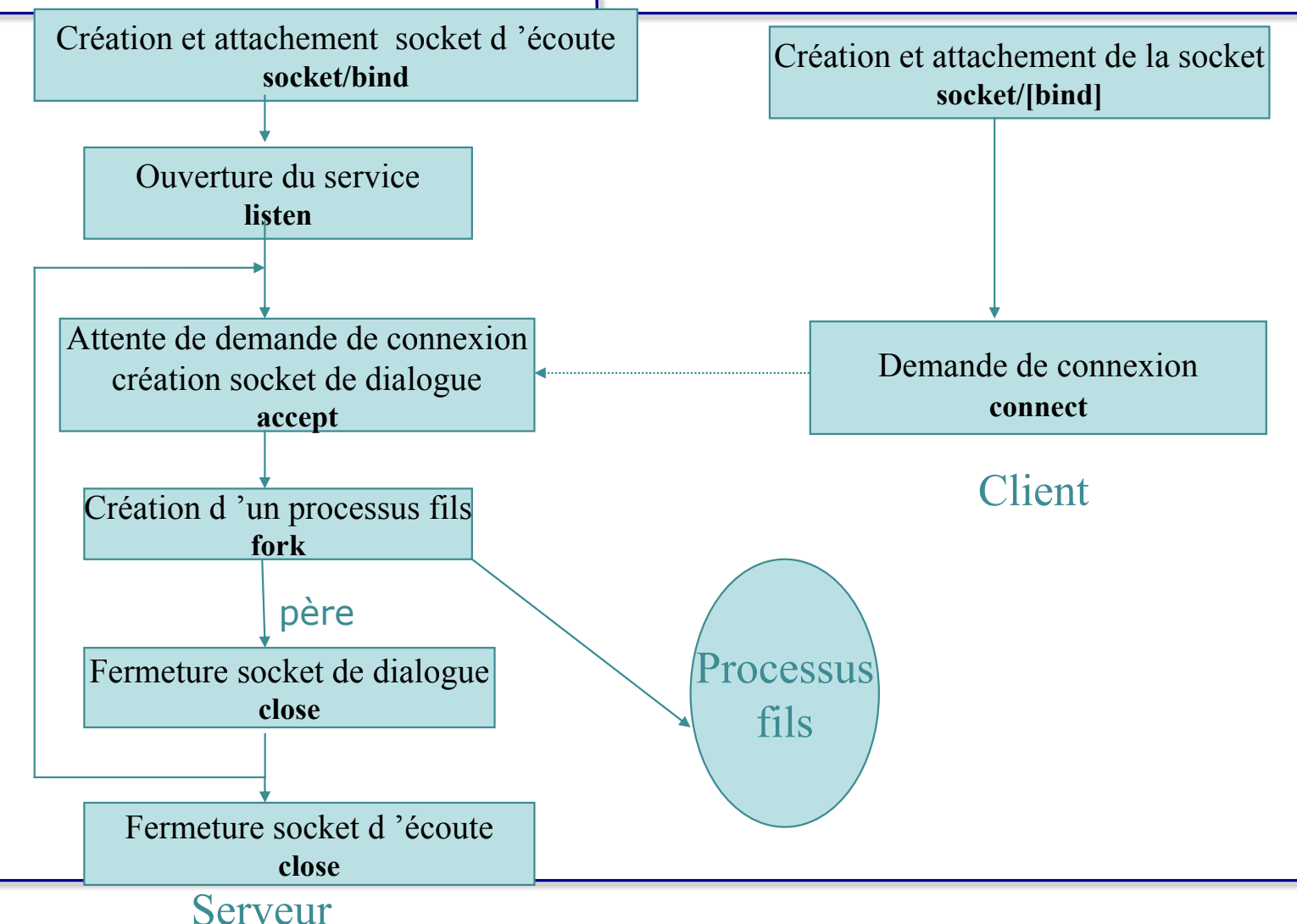
* **Comment utiliser le socket pour communiquer.**

Différents types sockets :

* **Socket TCP (PARTIE SERVEUR)**

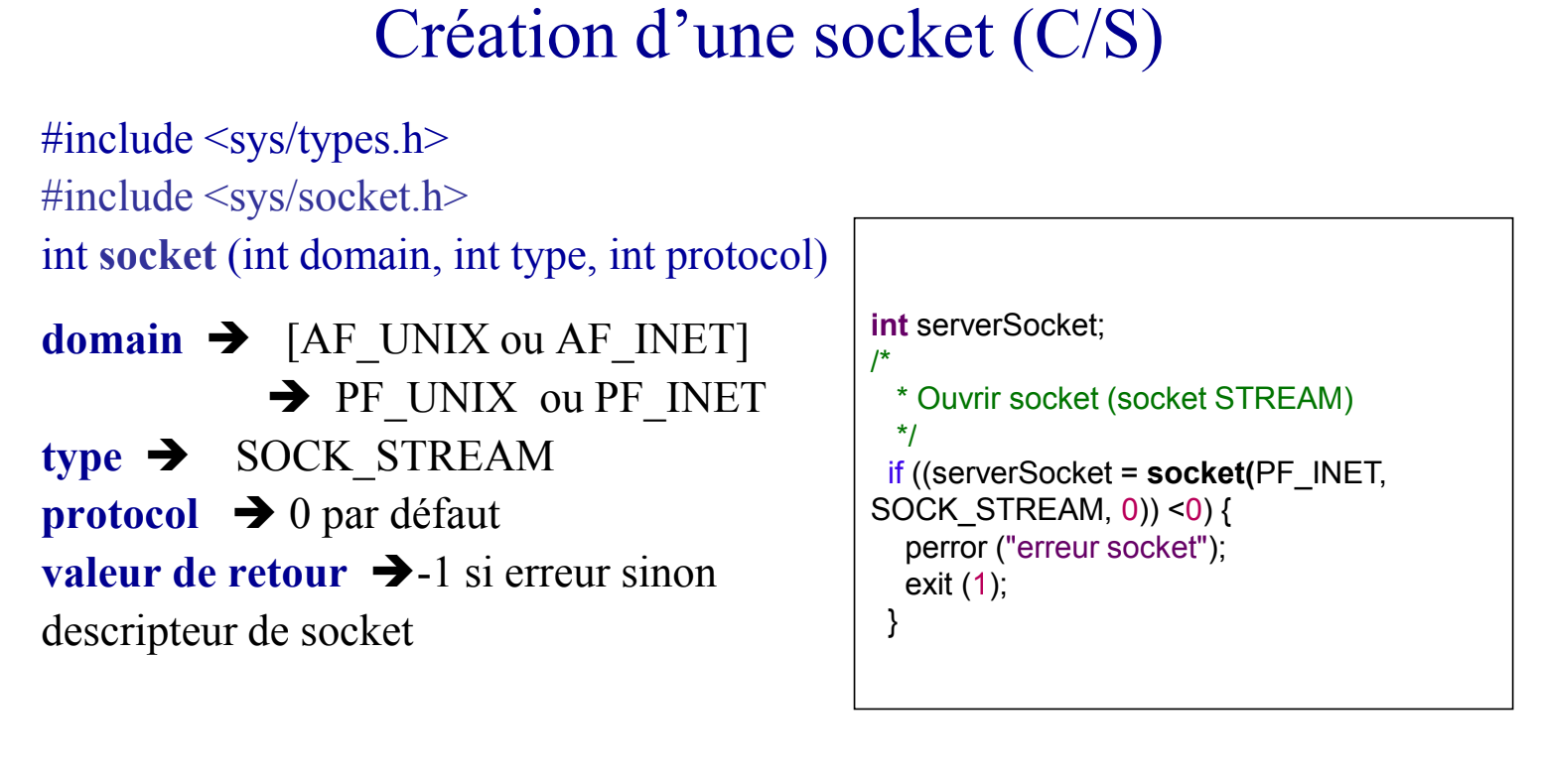
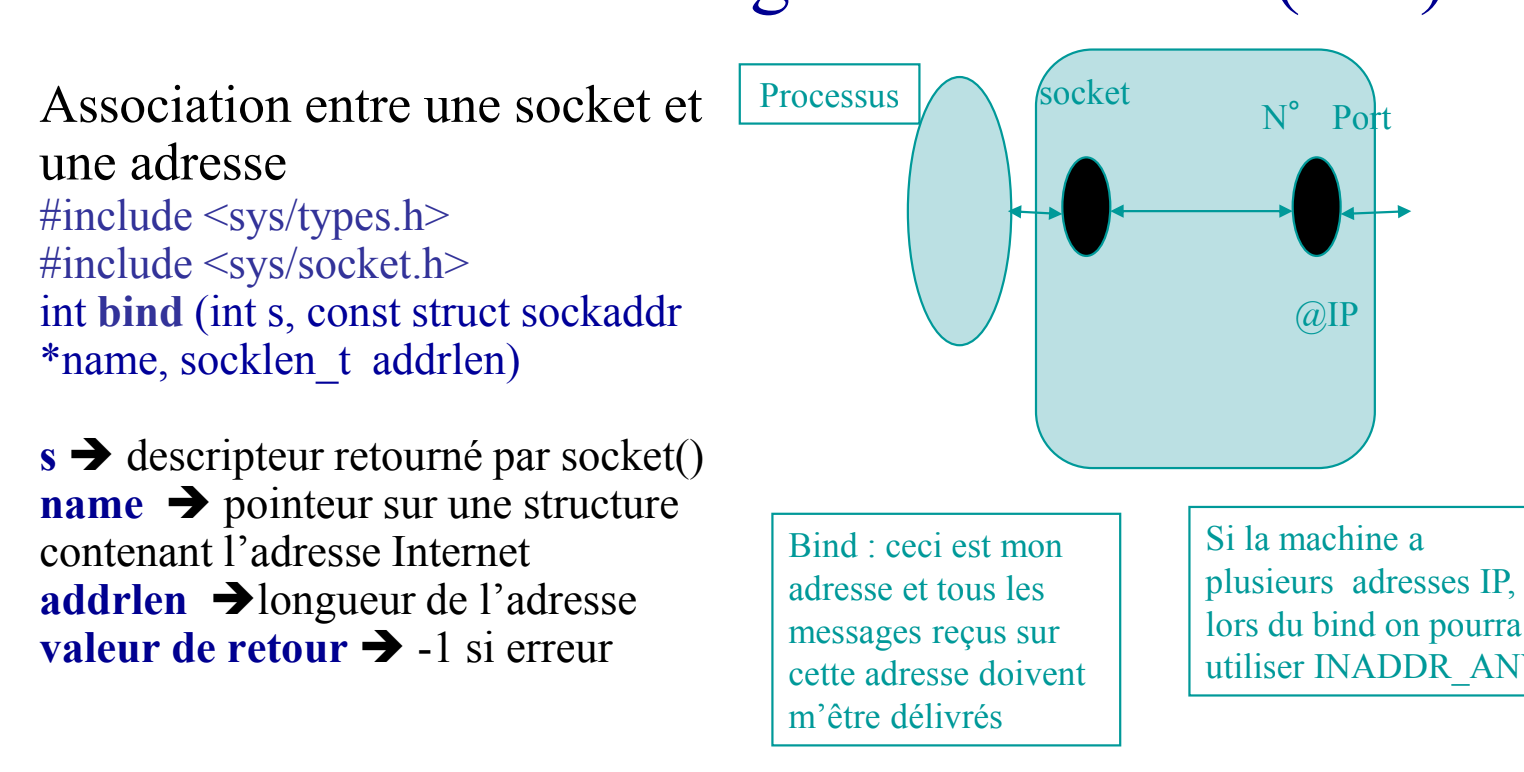
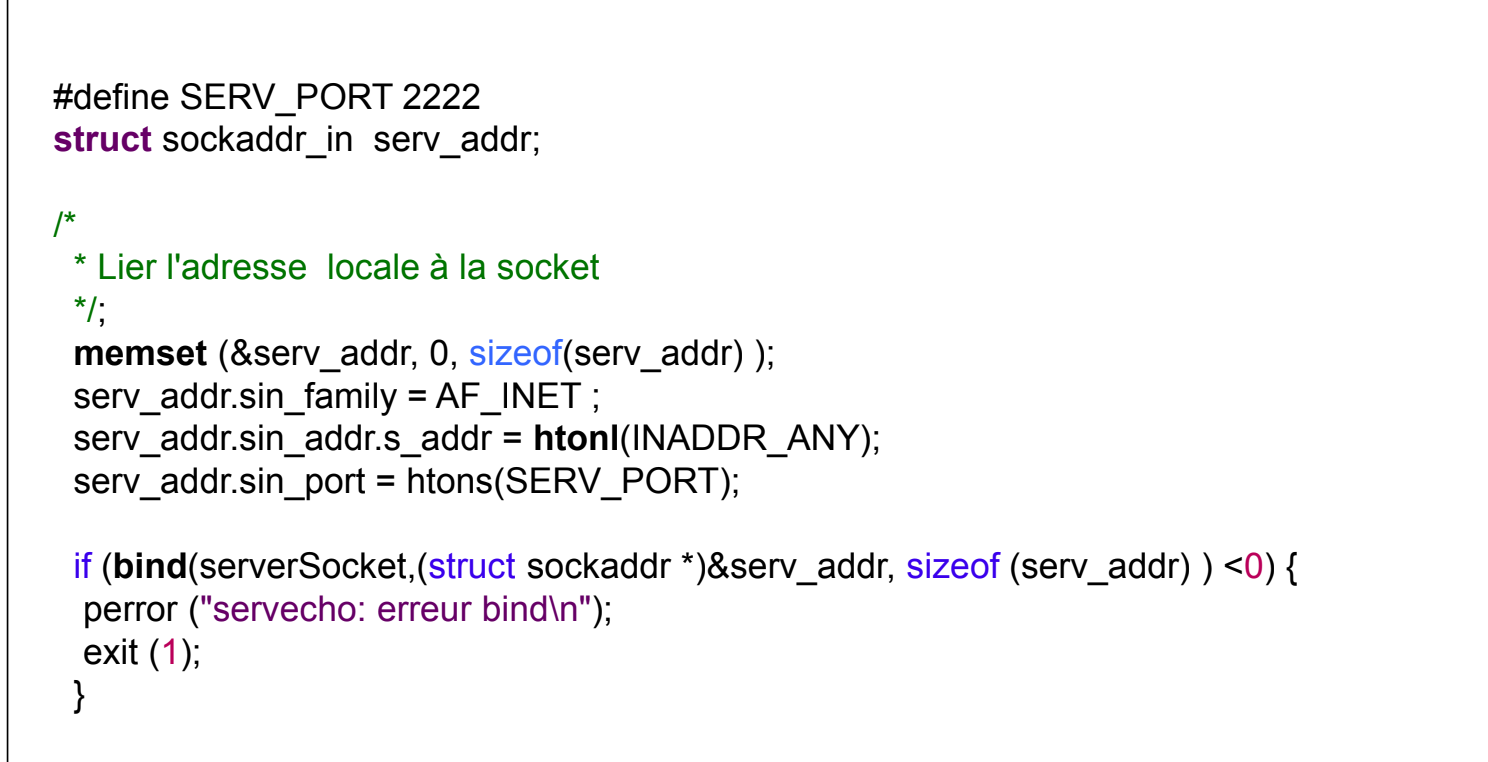
Etapes de dialogue en client-serveur TCP

1. Création socket
2. Ouverture dialogue
3. Echange données
4. Fermer socket



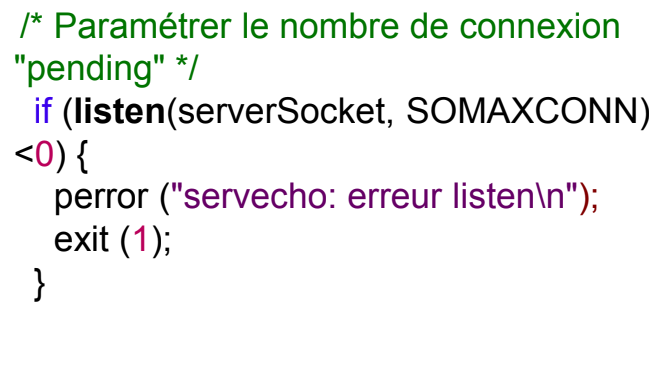
La méthode socket/bind permet le routage via le port d’entrée

HTTP écoute tt les connexions entrant sur le port 80, mais pour les pages web/image, il utilise un autre socket. On différencie le socket d’écoute et le socket de dialogue.

1. 
2. 
3. 

Validation

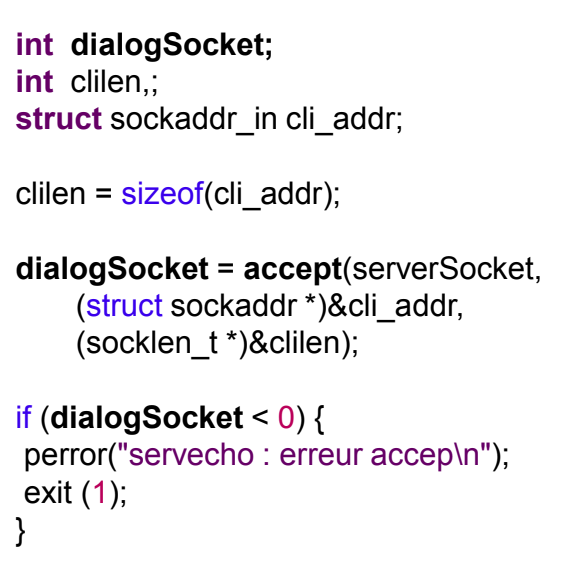
1. Ouverture du service



SOMAXCONN = variable par défaut du système

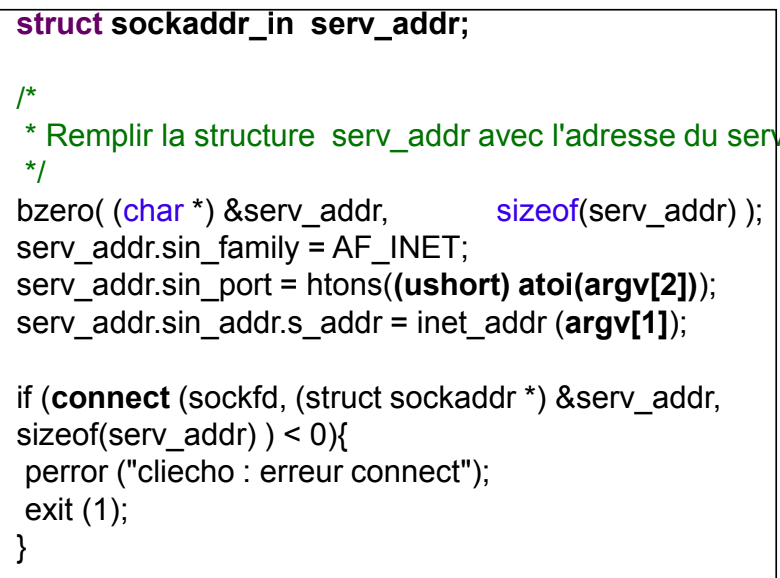
1. Méthode Accept, ouverture socket dialogue avec le client

On obtient un autre socket “dialogSocket”

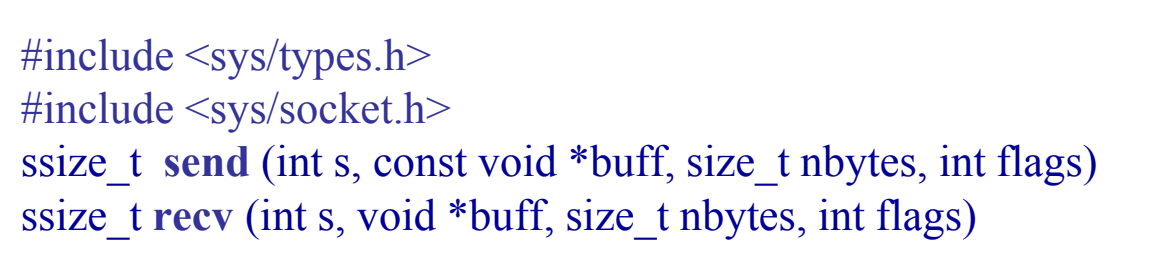


* **(PARTIE CLIENT)**

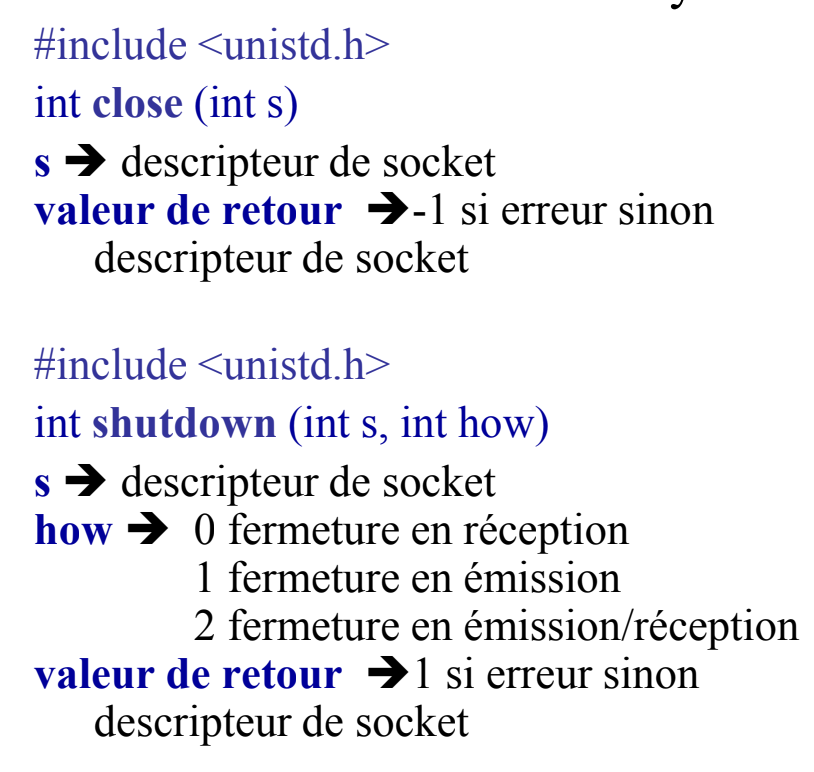
1.



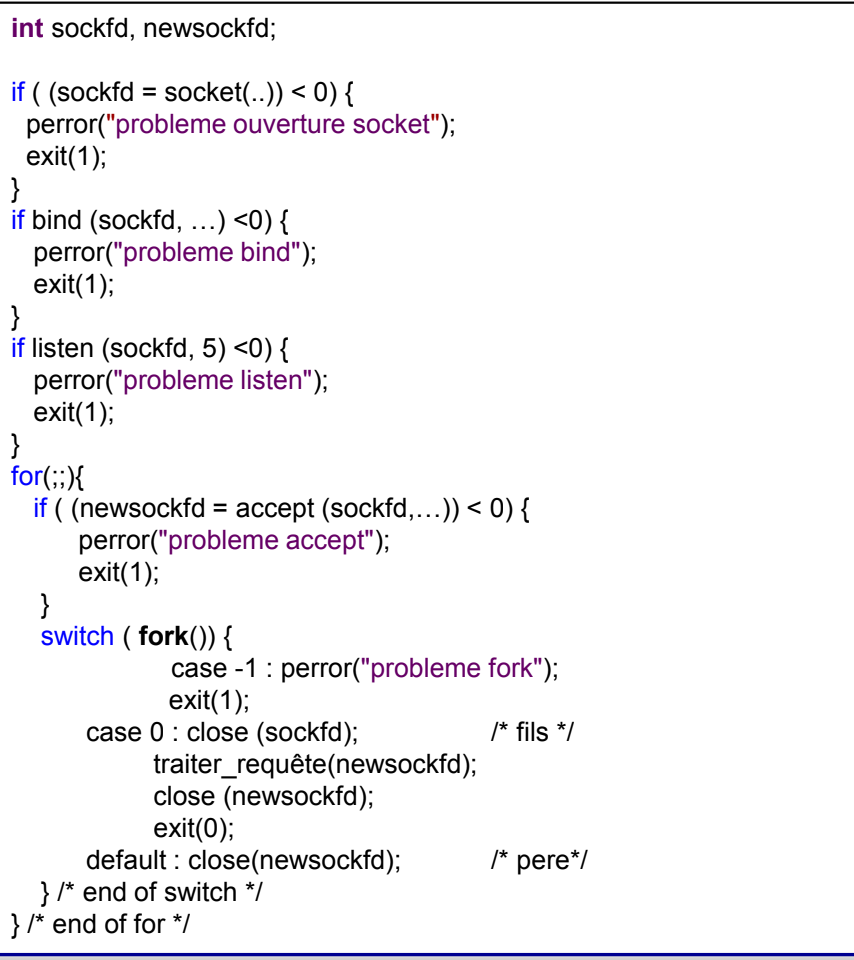
Demande de connexion

2. Dialogue

3. Fermeture



Utiliser close. Shutdown est plus spécifique (uniquement en émission ou réception).

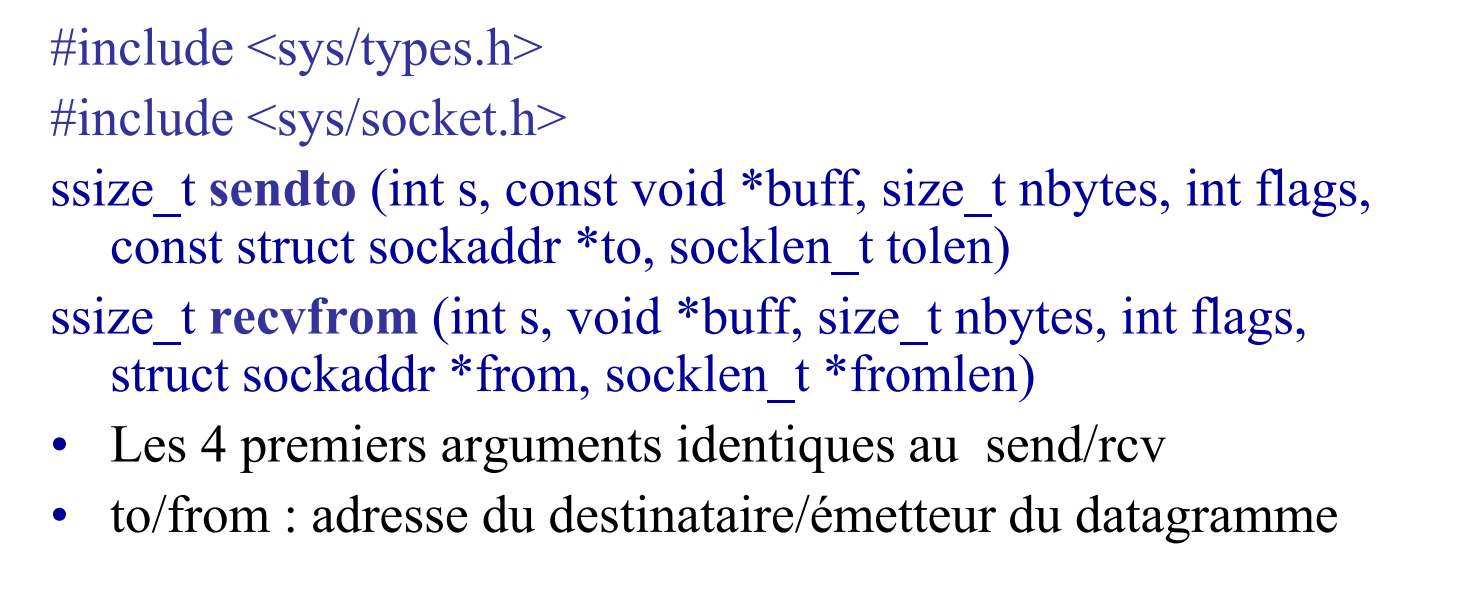
* **Techniques serveurs multi-client:**
* Bas niveau : Une fois le listen effectué (écoute des co entrantes), lors de l’acceptation du premier client, on lance “**fork**” pour qu’un processus continue d’écouter, et qu’un autre se concentre sur le socket de dialogue.

Le switch permet de check si on est dans le fils ou le père. Si fils : on ferme le socket d’écoute et on part sur le socket de dialogue. Le père va fermer par défaut le socket de dialogue.

* **Socket UDP Client-Serveur UDP**

Pas de changement sur le socket

Ce qui change ce sont les méthodes pour lire écrire dans le socket : recvfrom / sendto car la connexion est déjà établie avec l’interlocuteur en TCP. Alors qu’en UDP comme on est pas orienté connexion, on précise à chaque fois l’adr ip et le port du dest.

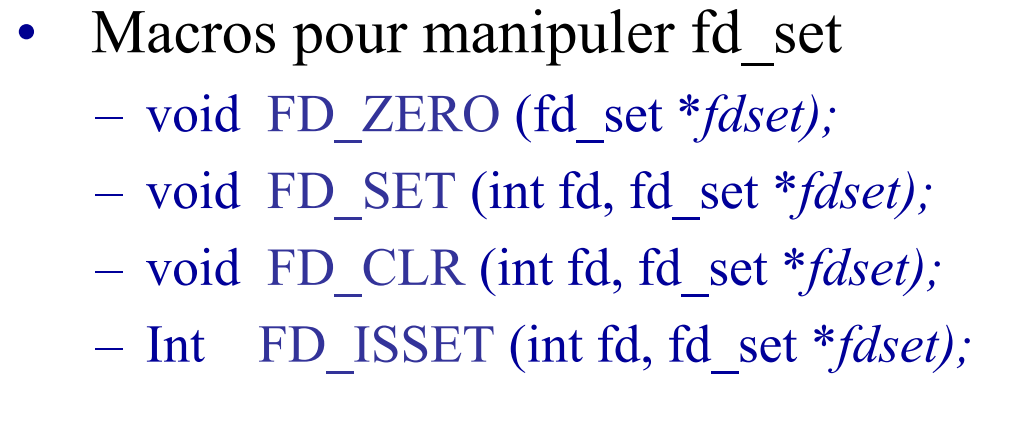


* **Entrée / sortie Multiplexée : Select**

Faire un srv multi client sans avoir à gérer le port pour que le fils gère à chaque fois. Utiliser TCP / UDP dans un même programme.

Select va prendre en charge plusieurs sockets en même temps, et mettre à jour des structures (fd\_set) pour savoir quel socket a été appelé. S’arrête dès qu’on a une activité de socket concomitante ou après le timeout précisé.

Pour appeler ces masques fd\_set :

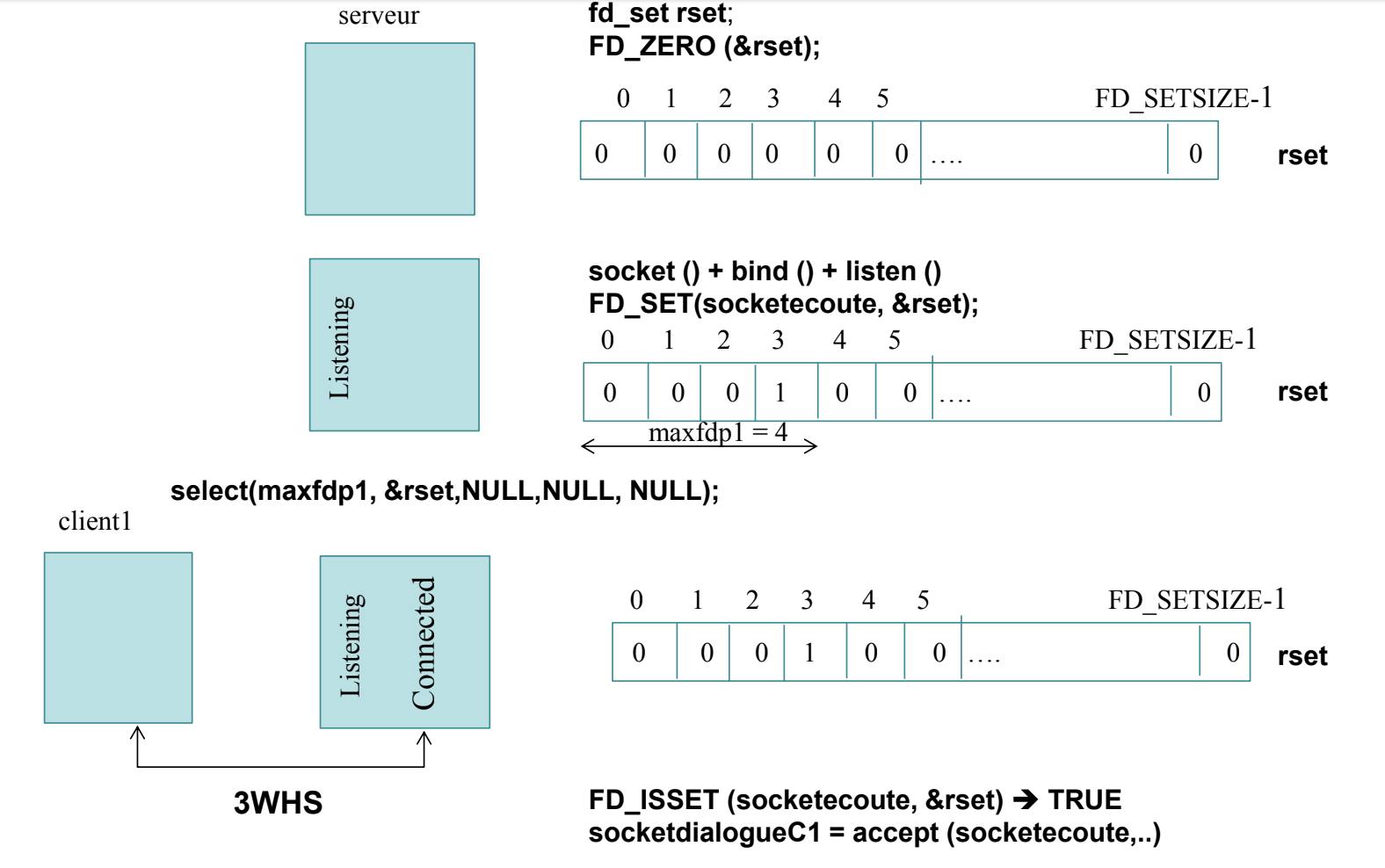


FD\_ZERO // initialiser à zéro

FD\_CLR // faire sortie le socket du select

FD\_ISSET // retrouver quels sockets ont été en activité (parcourir les sockets)

En mémoire, on a par ex :



1. FD\_ZERO
2. FD\_SET
3. Select
4. FD\_ISSET : parcourir chacun des sockets actifs, utilisés, et on regarde si select détecte une activité. Sie True = on a des données en attente de lecture sur le socket.

maxfdp : borner le parcourir mémoire (optimisation)

Timeval permet de mettre un temps limite d’activité à la fonction select().

----- Cours du 12 mai 2021 -------

* Entrées/sorties asynchrone (signal driven) : on utilise les signaux systèmes, pour savoir si le noyau a reçu les données
  + Différents types de signaux :
    - SIGIO
    - SIGURG (pas à retenir)
    - SIGPIPE (pas à retenir)

cf fonctionnement slide 122

* Options :
* Utilisation avancée des sockets. Utilisation d’options. cf 125
  + getsockopt()
  + setsockopt()
  + fcntl()
  + ioctl()
* Envoyer des datagrammes en broadcast : prendre l’option SO\_BROADCAST dans setsockopt
* Préciser la taille du buffer en réception ou émission : prendre l’option SO\_RCVBUF avec une taille (ex 4096)
* Permettre la réutilisation d’un port (ex après le crash d’un programme qu’il l’utilise) : prendre l’option SO\_REUSEADDR / SO\_REUSEPORT. Attention il enverra les paquets aux applications qui ont ce port.
* Options IP :
  + IP\_OPTIONS
  + IP\_TTL : nb de saut avant d’être jeté
  + IP\_HDRINCL : écrire soit même l’ensemble des champs de l’entête IP
* Options TCP :
  + TCP\_MAXSEG : Taille max des segments
  + TCP\_NODELAY : désactive l’algorithme de Nagle

**Socket raw (3e type de socket) :**

* Caractéristiques d’une socket raw :
  + lire et envoyer des paquets de contrôle (ICMP IGMP). IGMP pour le multi cast
  + dès que paquet arrive avec un champ “protocol” est nom traité par le noyau, il faut utiliser un socket raw
  + permet de définir sa propre en-tête
* Nécessité le mode privilégié pour être exécuté
* Création d’un socket raw. Pas d’utilisation de bind()
* Réception : utiliser recvfrom. Les paquets TCP et UDP ne sont jamais passés à un socket raw. cf slide 137
* **Le multicast** 
  + Définition : envoi d’un paquet vers les nœuds de réseaux distincts et répartis. L’information est envoyée une seule fois et dupliquée. A l’inverse, en unicast il faut envoyer autant de fois l’information que de destinataires.
  + Modèle IP multicast :
    - Implique la notion de groupe : ensemble de 0 ou n machines
      * entièrement dynamique (quitter/rejoindre à tt moment)
      * ouvert
      * ..
    - Type d’adresse ip : les adr multicast sont de classe D
    - Il existe un réseau multicast privé : de 224.0.0.0 à 224.0.0.255
      * 224.0.0.1 : tous les hôtes du réseau
      * 224.0.0.2 : contacter tous les routeurs du réseau
  + Pour qu’un noeud reçoive les données, il doit être abonné au flux multicast.
    - On utilise le protocole IGMP pour gérer cet abonnement.
    - IGMP:
      * Permet à un routeur de savoir que telle machine est intéressé à tel flux multicast : mécanisme d’abonnement / désabonnement
      * MRP permet de faire remonter l’info d’abonnement des routeurs sur internet
      * Les messages IGMP sont encapsulés dans les datagrammes IP.
      * Cf fonctionnement slide 151
      * Pour éviter la surcharge du réseau local, si le routeur détecte au moins 1 hôte abonné, il renvoie une réponse positive à la requête. Cf slide 153
      * Le routeur émet périodiquement une requête IGMP à 224.0.0.1 (tous les hôtes du réseau local).

cf de la slide 160 pour application aux sockets

* Pour rejoindre un groupe multicast il faut préciser une option : IP\_ADD\_MEMBERSHIP, avec un second paramètre précisant l’adresse imr (l’interface ou les interfaces de la machine que l’on souhaite abonner) dans la méthode setsocketoption.
* Quitter un groupe : IP\_DROP\_MEMBERSHIP dans setsocketopt avec l’interface.
* L’abonnement est un mécanisme ouvert (tout le monde peut s’abonner), on ne peut pas contrôler qui s’abonne.