UE NSY103

Les Socket Unix

Sommaire

I- Introduction

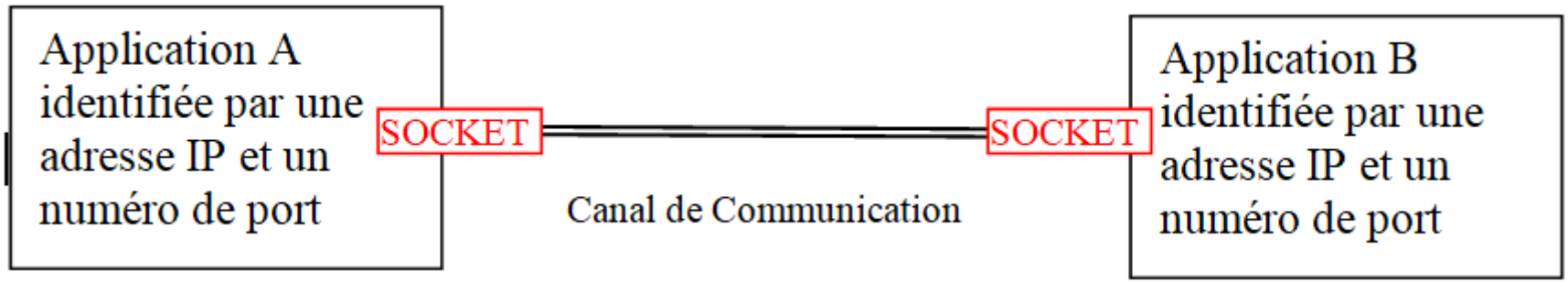
Les primitives Sockets Unix sont considérées comme la référence en matière de primitives réseau dans le monde Internet.

- Socket voulant dire : Point d'accès aux communications.

- Le modèle "Socket Unix" est aujourd'hui repris par le modèle WinSock Windows.

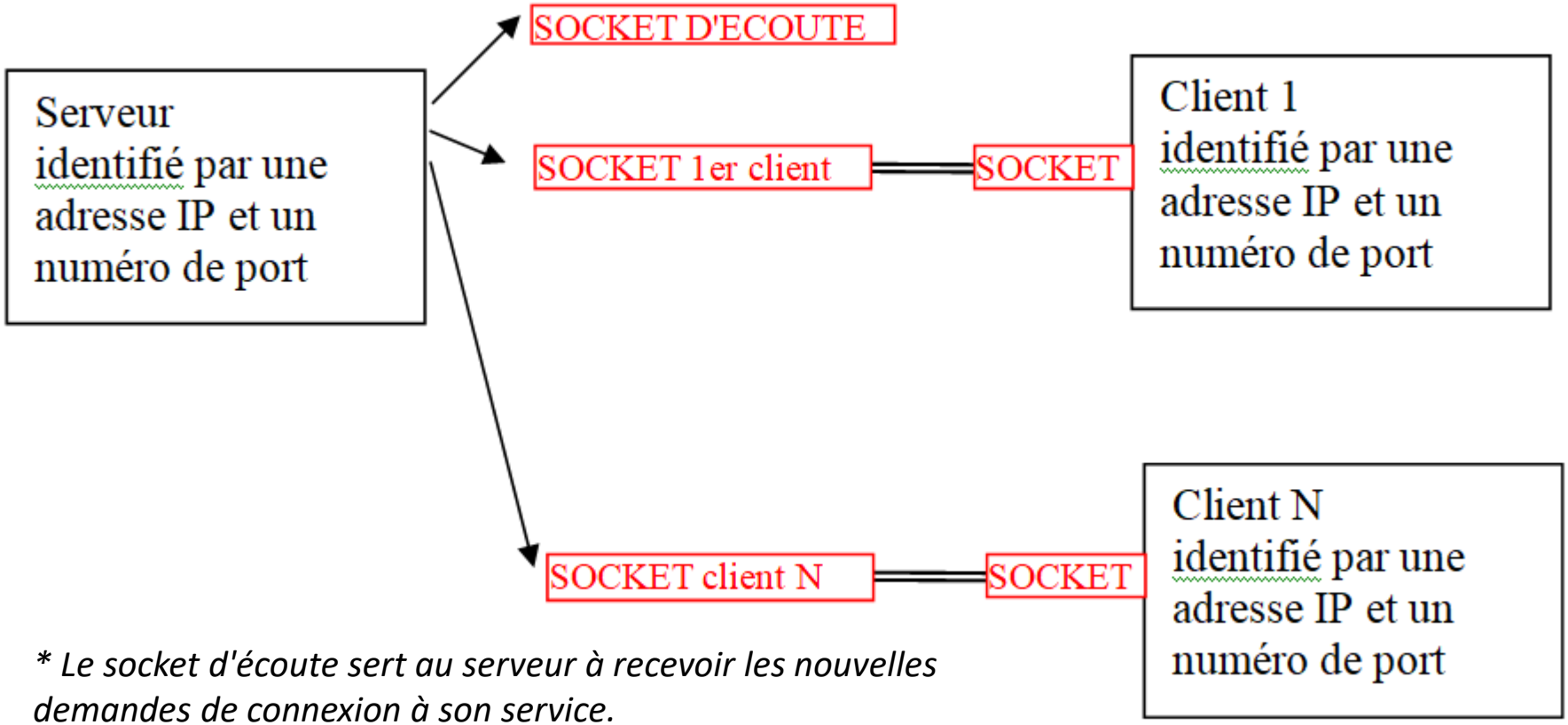
- Un Socket représente un point d'accès aux communications : Un canal de communication entre 2 applications est matérialisé par un socket à chaque extrémité.
- Pour créer un service au dessus d'un canal de communication les applications en réseau vont piloter les couches réseau à l'aide des primitives (appels système) Socket.

Modèle d'égal à égal (peer to peer)



I- Introduction

Modèle Client / Serveur



** Le socket d'écoute sert au serveur à recevoir les nouvelles demandes de connexion à son service.*

Dans ce modèle c'est toujours le client qui est demandeur de l'ouverture du canal de communication.

I- Introduction

Adresse Serveur

Pour accéder à un service réseau le client (ou le demandeur dans le cas d'un mode d'égal à égal) doit utiliser les éléments d'adressage suivants :

- Adresse IP : adresse du serveur physique.
- N° de port : adresse du service sur le serveur physique
- Protocole : type de service, donc type de protocole de communication applicatif à utiliser (HTTP, FTP etc.)
- Eventuelles informations d'adressage complémentaires spécifiques du protocole considéré.

Ce qui est précisément défini par une adresse URL, par exemple l'adresse du serveur Web grand public de la Fnac http://www.fnac.com:80

** si le n° port n'est pas précisé dans l'URL, le navigateur insèrera automatiquement le n° de port standard HTTP : 80*

I- Introduction

Adresse Client

Le client s'identifie auprès du serveur au travers d'un protocole d'échange en communiquant son adresse IP et son numéro de port.

-Protocole HTTP le plus souvent

- Le serveur devrait posséder un n° port fixe bien identifié permettant d'accéder au service qu'il propose ; le client en revanche peut utiliser un numéro de port quelconque (délivré par le système d'exploitation à chaque nouvelle ouverture de session).

I- Introduction

Rappels TCP/IP

Il y a 2 protocoles de transport spécifiés dans l'architecture TCP/IP

1- Le protocole sans connexion UDP (User Datagram Protocol)

- Protocole sans connexion
- Souple pour la diffusion de messages.
- Peu de contrôles (lettre ordinaire à la poste).
- Faible consommation de ressources système.

2- Le protocole avec connexion TCP (Transmission Control Protocol)

- Protocole connecté : établissement d'un canal de communication
- Orienté communication d'application à application (flot de données).
- Nombreux contrôles, forte fiabilité (lettre recommandée).
- Forte consommation de ressources système.

II- Les Primitives

socket

Prise d'un socket local

Paramètres: - famille de protocoles
- mode stream ou datagram
- protocole de la famille à utiliser (si 0 le système choisi pour vous)

Retour : - un descripteur (identifiant) de socket

bind

Association d'un socket local à une adresse IP et un n° de Port

paramètres : - descripteur de socket,
- famille de protocoles locale
- adresse IP locale,
- n° de port local, si 0 le système affecte automatiquement un port

II- Les Primitives

listen : (mode STREAM seulement)

Utilisation d'un socket en "écoute de demandes de connexion"

paramètres : - descripteur de socket a l'écoute,
- nombre max de connexions autorisées
- retour: 0 si OK, -1 sinon

remarque : - pas plus d'un socket à l'écoute sur un port donné

accept : (mode STREAM seulement)

Définition de l'ensemble des connexions entrantes acceptables

paramètres : - descripteur de socket,
- famille de protocoles distante
- adresse IP distante (si 0 accepte toutes adresses)
- n° de port distant (si 0 accepte tous ports)

retour : - descripteur de socket d'échange différent du socket d'écoute.

II- Les Primitives

connect :

Demande de connexion à un socket distant à l'écoute

paramètres : - descripteur de socket,
- famille de protocoles distants
- adresse IP distante,
- n° de port distant

II- Les Primitives

read :

Lecture sur le canal de communication

paramètres : - descripteur de socket de connexion
- buffer de lecture,
- nombre d'octets à lire

remarque : La lecture peut être configurée en mode bloquant ou non bloquant.
En mode bloquant read retourne le nombre d'octets lus si OK,
0 si connexion est fermée et -1 en cas de faute de lecture.

write :

Ecriture sur le canal de communication

paramètres : - descripteur de socket de connexion
- buffer d'écriture
- nombre d'octets à écrire

remarque : L'écriture peut être configurée en mode bloquant ou non bloquant.
En mode bloquant write retourne le nombre d'octets écrits si OK,
0 si la connexion est fermée et -1 en cas de faute de lecture.

II- Les Primitives

close :

Demande de fermeture "douce" de connexion, après la fin des échanges en cours.

paramètres : - descripteur de socket

remarque : - close libère le socket associé.

shutdown :

Fermeture "dure" de connexion, sans attendre la fin des échanges en cours.

paramètres : - descripteur de socket

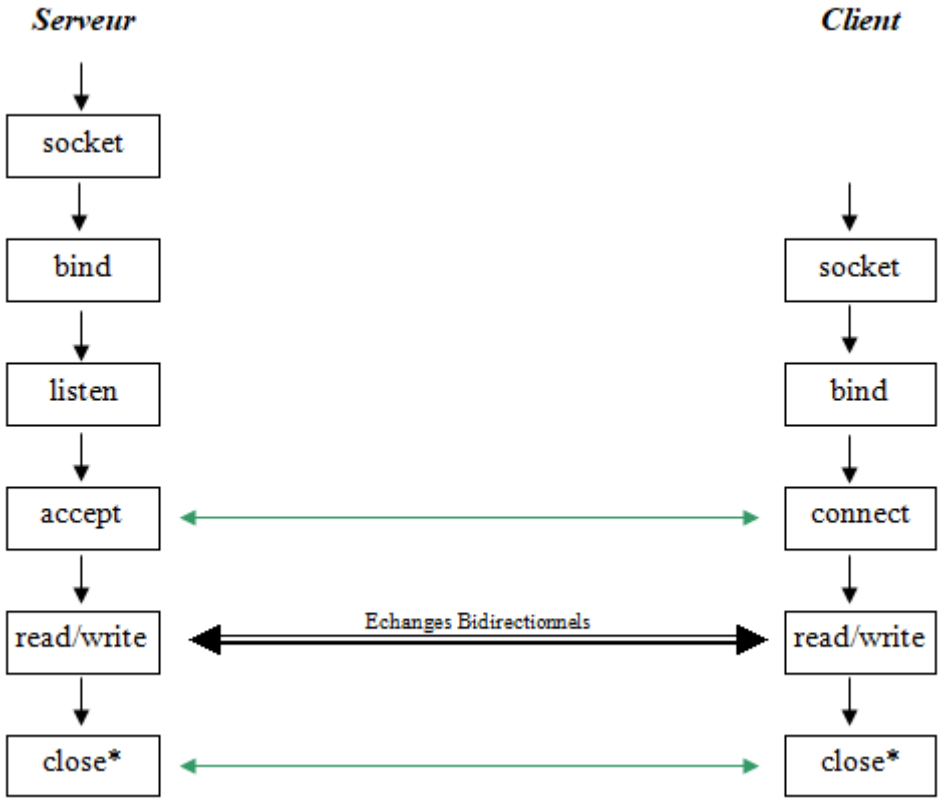
remarque : - shutdown libère le socket associé.

- Cf. le support d'exercices pour plus d'information sur l'utilisation des primitives

III- Les Scénarios d'Echange

Mode Connecté (Stream) ou Client / Serveur

Un socket "A" est à l'écoute du coté serveur, à chaque connexion "i" un nouveau socket "A_i" est affecté au nouveau canal de communication (entre le serveur et le nouveau client).



* Chaque Close ne ferme qu'un sens de la transmission

III- Les Scénarios d'Echange

Mode Client / Serveur (Suite)

La primitive "accept" retourne un descripteur de socket d'échange pour le Client considéré et le socket d'écoute reste à l'écoute.

- en cas d'une nouvelle demande de connexion la primitive "accept" retournera un deuxième socket d'échange pour le deuxième Client etc.

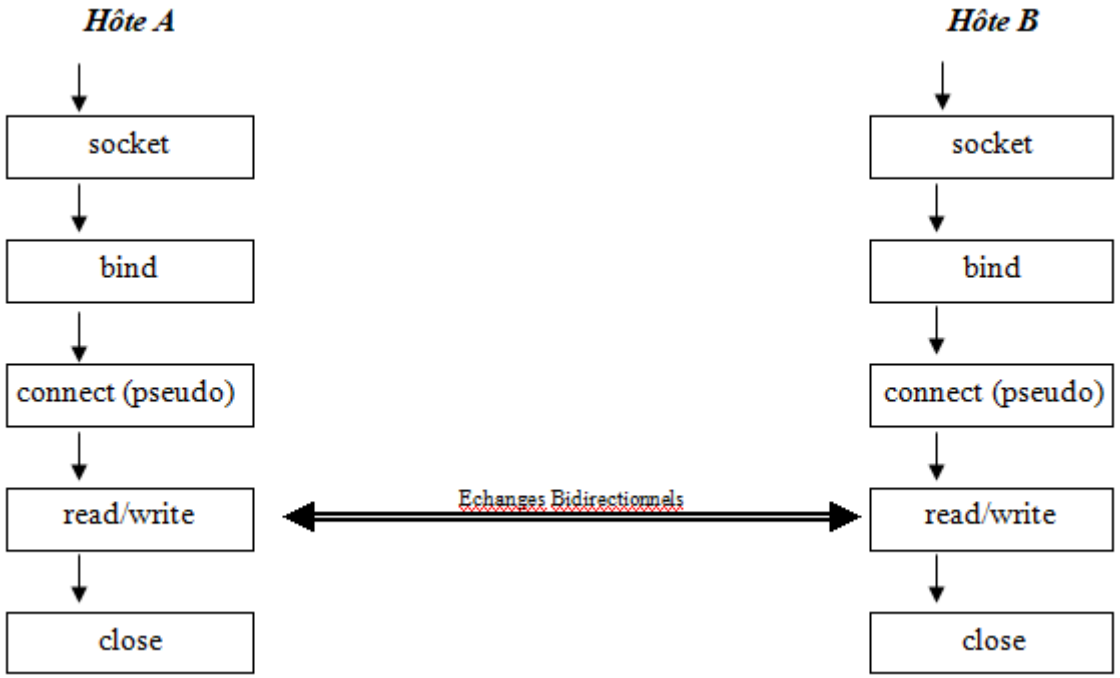
➢C'est au programmeur s'il souhaite réaliser une application Serveur de gérer la création des processus correspondant aux différents Clients afin que les échanges se déroulent en parallèle et de façon structurée (primitive "fork" pour créer un processus fils).

* Chaque Close ne ferme qu'un sens de la transmission

III- Les Scénarios d'Echange

Mode d'égal à égal (peer to peer) avec pseudo-connexion

La pseudo-connexion permet d'émettre ou de recevoir plusieurs messages avec le même socket, mais en fait chacun des messages est bien envoyé en mode non conneté (UDP).

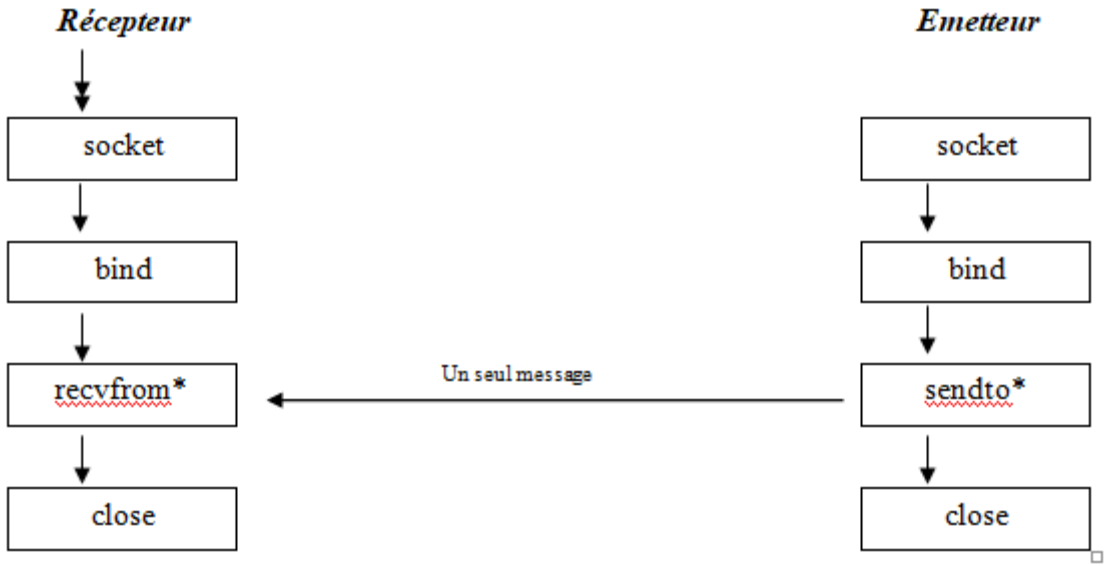


III- Les Scénarios d'Echange

Mode d'égal à égal (peer to peer) sans pseudo-connexion

Un seul message de transmis, il faudra reprendre des sockets pour le message suivant.

➢Dans ce cas on ne peut pas utiliser les primitives read / write



EXERCICES

Vous pensez avoir bien assimilé les concepts présentés dans ce cours.
Vous devez alors passer aux exercices, ce sont eux qui
vous permettront de valider vos connaissances .