# Système UNIX et programmation

Dr. Davy MOUSSAVOU- Consultant Linux, Certifié Linux International

# Chapitre 7: Initiation à la programmation réseau

# Contenu du  cours

Séquence 1: Introduction aux sockets

Séquence 2: Socket TCP

Séquence 3: Socket UDP

# Séquence 1 : Introduction aux sockets

## Concept de socket

- C'est approximativement un prolongement du concept des tubés nommés pour faire dialoguer des processus s'exécutant sur différentes machines.

- C'est point de communication avec l'extérieur

- Une socket est représentée sous forme d'un descripteur de fichier;

## Principe

- On peut donc écrire des données dans une socket après l'avoir associée à un protocole de communication, et les couches réseau des deux stations s'arrangeront pour que les données ressortent à l'autre extrémité.
- La seule complication introduite par rapport aux tubes classiques est la phase d'initialisation, car il faut indiquer **l'adresse** et le numéro de port du correspondant.
- Une fois que la liaison est établie, le comportement ne sera pas très différent de ce qu'on a étudié dans les tubes de communication

## Format générique de l'adresse d'une socket

```
struct sockaddr {
        uint8_t sin_len; //taille de la structure
        sa_family_t sin_family; //famille de la socket
        char sa_data[8]; //inutilisé
};
```

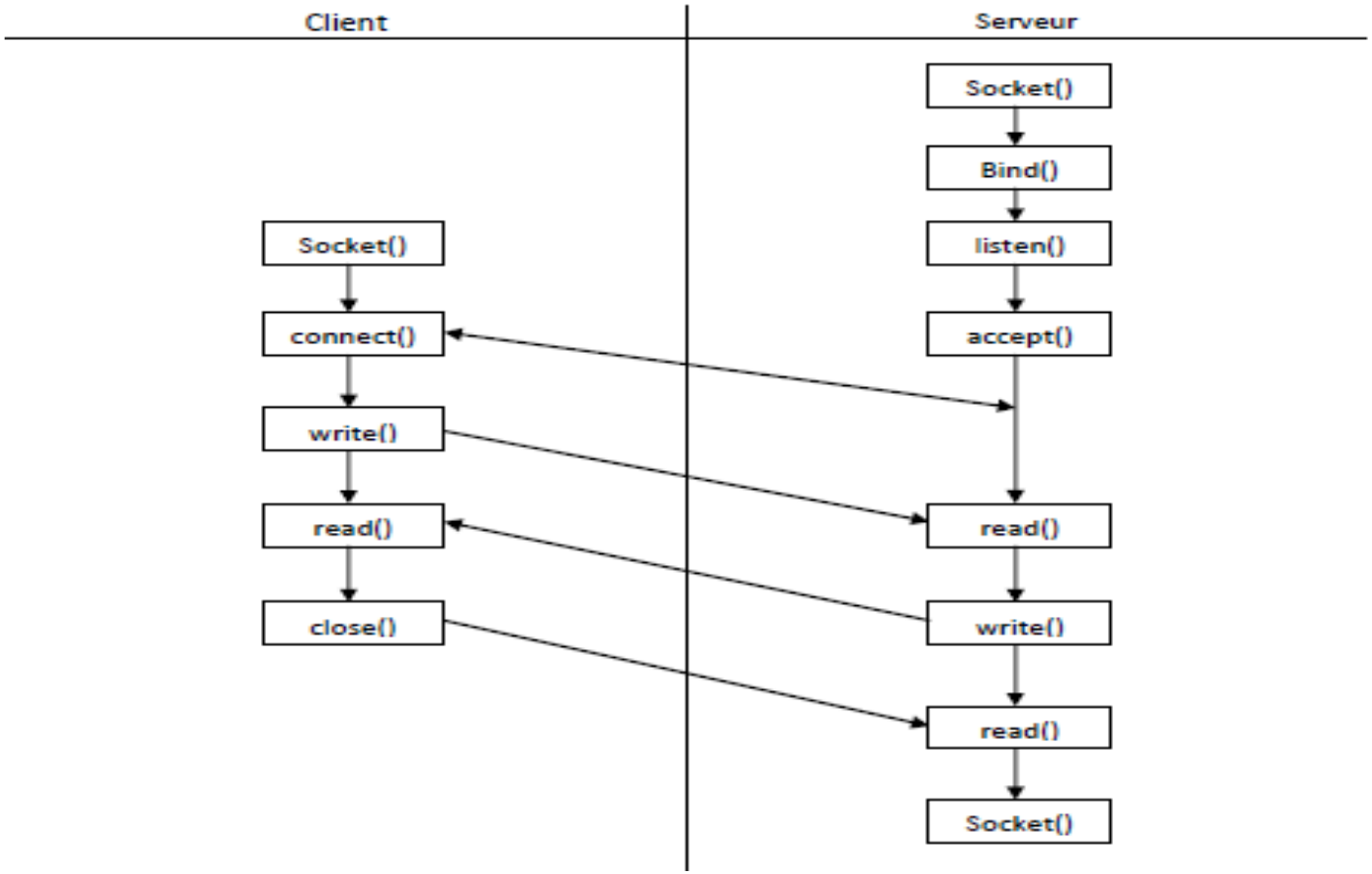
## Format de l'adresse d'un socket IPV4

```
struct sockaddr_in {
        uint8_t sin_len; //taille de la structure
        sa_family_t sin_family; //famille de la socket
        in_port_t sin_port; //numéro de port TCP ou UDP
        struct in_addr sin_addr; //adresse IPv4
        char sin_zero[8]; //inutilisé
};
```

```
 struct in_addr {
        in_addr_t s_addr; //adresse IPv4 sur 32 bits
};
```

# Séquence 2 : Socket TCP

## Algorithme


Client
Serveur
Socket()
Bind()
listen()
accept()
Socket()
connect()
write()
read()
close()
read()
write()
read()
Socket()

## **Fonction côté serveur:** bind()

### Bind associe la socket à une son adresse

```
NOM
       bind - Fournir un nom à une socket

SYNOPSIS
       #include <sys/types.h>          /* Voir NOTES */
       #include <sys/socket.h>

       int bind(int sockfd, const struct sockaddr *addr,
                socklen_t addrlen);
```

## **Fonction côté serveur:** listen()

### Se met en attente de connexion sur une socket

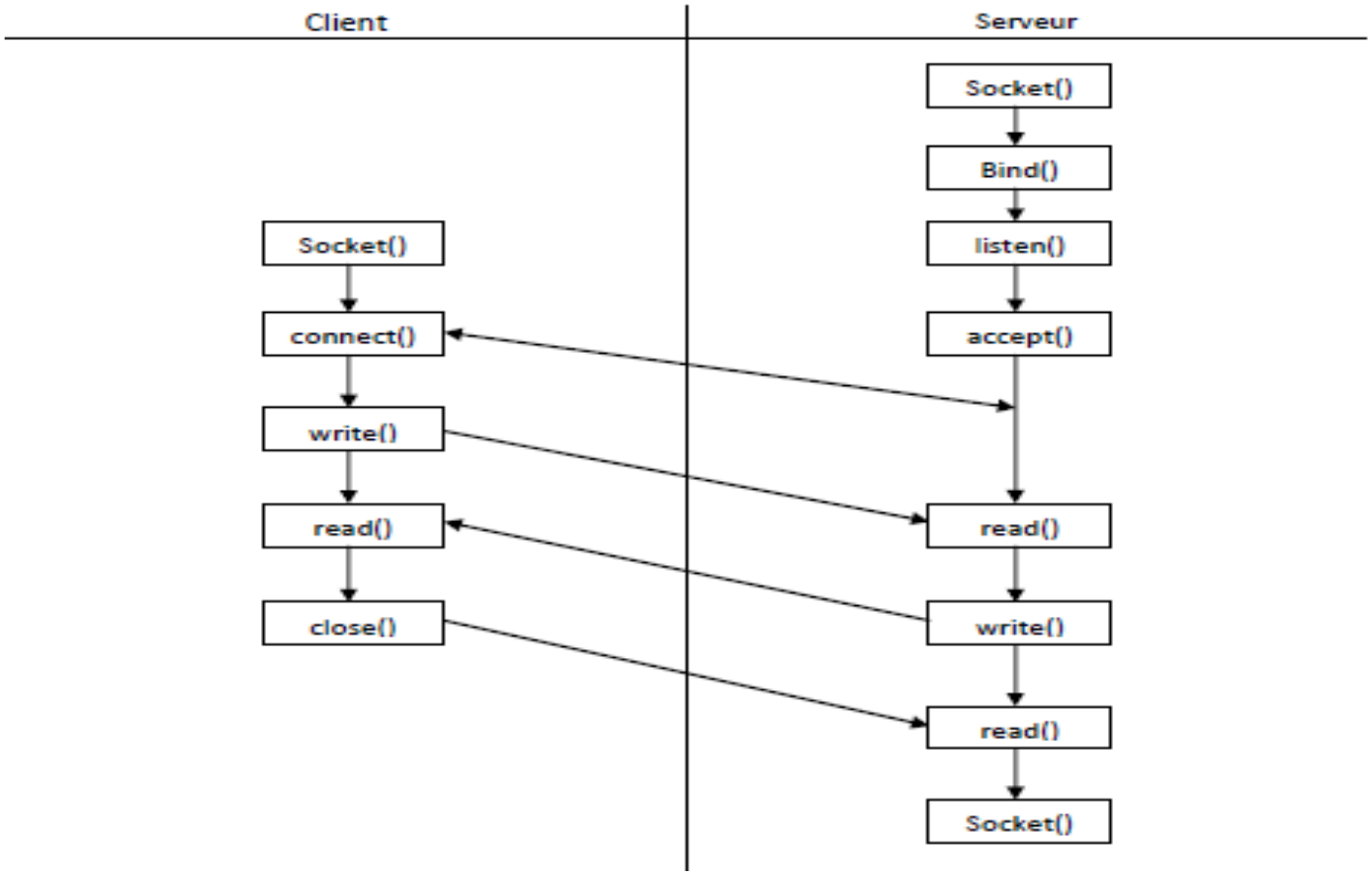
```
NOM
       listen - Attendre des connexions sur une socket

SYNOPSIS
       #include <sys/types.h>          /* Voir NOTES */
       #include <sys/socket.h>

       int listen(int sockfd, int backlog);

DESCRIPTION
       listen()  marque  la  socket référencée par sockfd comme une socket
       passive, c'est-à-dire comme  une  socket  qui  sera  utilisée  pour
       accepter   les   demandes  de  connexions  entrantes  en  utilisant
       accept(2).
```

## **Fonction côté serveur:** accept()

**Accepte une connexion sur une socket TCP et génère une nouvelle socket pour l'échange des données avec le client connecté**

```
NOM
       accept - Accepter une connexion sur une socket

SYNOPSIS
       #include <sys/types.h>          /* Voir NOTES */
       #include <sys/socket.h>

       int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
```

## **Fonction côté client:** accept()



## Fonction côté client: connect()

**NOM**

connect - Débuter une connexion sur une socket

**SYNOPSIS**

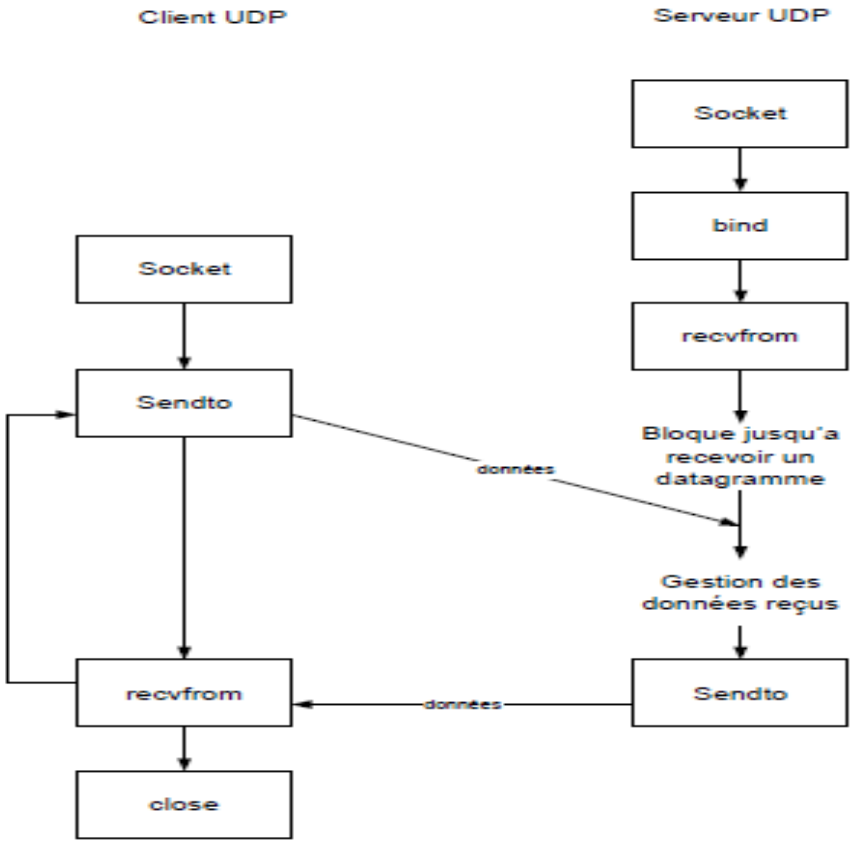
```
#include <sys/types.h>          /* Voir NOTES */
#include <sys/socket.h>

int connect(int sockfd, const struct sockaddr *addr,
            socklen_t addrlen);
```

**DESCRIPTION**

L'appel système **connect**() connecte la socket associée au descripteur de fichier sockfd à l'adresse indiquée par addr. L'argument addrlen indique la taille de addr. Le format de l'adresse addr est déterminé par la famille de la socket sockfd ; voir **socket**(2) pour plus de détails.

# Séquence 3 : Socket UDP

## Algorithme


Client UDP
Serveur UDP
Socket
bind
recvfrom
Bloque jusqu'a recevoir un datagramme
Socket
Sendto
données
Gestion des données reçus
recvfrom
données
Sendto
close

## Fonction recvfrom()

```
NAME
       recv, recvfrom, recvmsg - receive a message from a socket

SYNOPSIS
       #include <sys/types.h>
       #include <sys/socket.h>

       ssize_t recv(int sockfd, void *buf, size_t len, int flags);

       ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags,
                        struct sockaddr *src_addr, socklen_t *addrlen);

       ssize_t recvmsg(int sockfd, struct msghdr *msg, int flags);

DESCRIPTION
       The  recvfrom()  and  recvmsg()  calls are used to receive messages
       from a socket, and may be used to receive data on a socket  whether
       or not it is connection-oriented.
```

## Fonction Sento()

```
NAME
       send, sendto, sendmsg - send a message on a socket

SYNOPSIS
       #include <sys/types.h>
       #include <sys/socket.h>

       ssize_t send(int sockfd, const void *buf, size_t len, int flags);

       ssize_t sendto(int sockfd, const void *buf, size_t len, int flags,
                      const struct sockaddr *dest_addr, socklen_t addrlen);

       ssize_t sendmsg(int sockfd, const struct msghdr *msg, int flags);

DESCRIPTION
       The system calls send(), sendto(), and sendmsg() are used to trans-
       mit a message to another socket.
```

Séquence 1: Concepts théoriques
Séquence 2: Envoie d'un signal
Séquence 3: Réception d'un signal
TP