Linux **Les sockets en C**

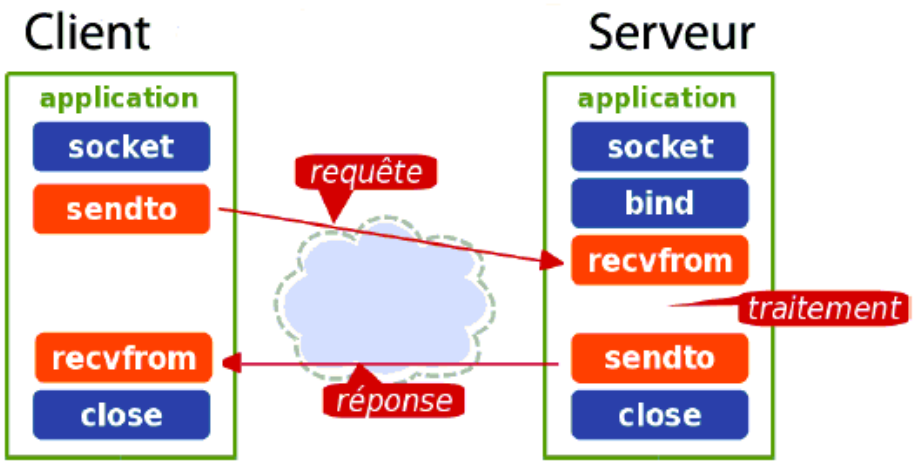
1 Introduction

Les sockets sont des flux de données, permettant à des machines distantes de communiquer entre elles sur le réseau via des protocoles.
Les différents protocoles sont **TCP** qui est un protocole dit "connecté", et **UDP** qui est un protocole dit "non connecté".

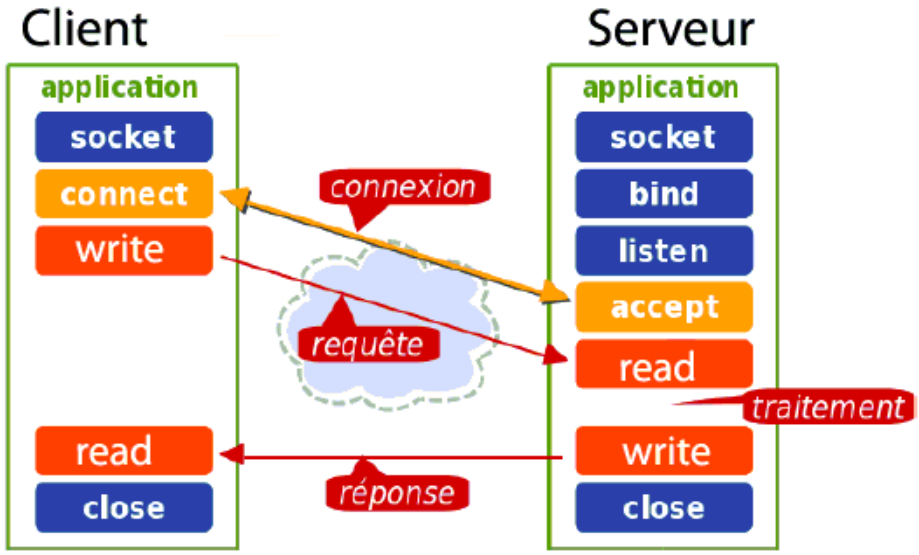
2 Les fonctions ou appels système

Sous Linux, le Langage C nous fournit un certain nombre de fonctions et structures pour manipuler les sockets.

En **UDP** (mode non connecté)



En **TCP** (mode connecté)



2.1 La fonction socket

```
int socket(int domain, int type, int protocol);
Cette fonction renvoie un entier de type int.
```

```
socketUdp = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
socketTcp = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
```

2.2 La fonction bind

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

La fonction bind lie un socket avec une structure sockaddr.

```
adresseServeur.sin_family = AF_INET;
adresseServeur.sin_port = htons(8888);
adresseServeur.sin_addr.s_addr = htonl(INADDR_ANY);

retour = bind(SocketTcp, (struct sockaddr*)
&adresseServeur, sizeof(adresseServeur));
```

2.3 La fonction listen

```
int listen(int s, int backlog);
```

Cette fonction définie la taille de la file de connexions en attente pour la socket.

```
retour = listen(SocketTcp, 10);
```

2.4 la fonction connect

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

Cette fonction connecte la socket à l'adresse spécifiée dans la structure sockaddr. Il s'agit donc d'une fonction à utiliser dans un client TCP.

2.5 la fonction accept

```
int accept(int sock, struct sockaddr *adresse, socklen_t
*longueur);
```

Cette fonction accepte la connexion d'un socket sur le socket *sock*
L'argument *adresse* sera remplie avec les informations du client qui s'est connecté. Cette fonction retourne un nouveau socket, qui devra être utilisé pour communiquer avec le client. Il s'agit d'une fonction à utiliser dans un serveur TCP.

```
fdSocketClient=accept(SocketTcp, (struct sockaddr *) &adresseClient,
&tailleClient);
```

2.6 la fonction write ou send

Ces fonctions permettent de transmettre un message à destination d'une autre socket en mode connecté (TCP). La seule différence entre **send** et **write** est la présence de *flags*.

```
retour = write(fdSocketClient, buffer, strlen(buffer));
```

2.7 la fonction read ou recv

Ces fonctions sont utilisées pour recevoir des messages depuis une socket en mode connecté (TCP). Elles renvoient la longueur du message si elles réussissent ou -1 si elles échouent.

```
retour = read(fdSocketClient, bufferReception, 512);
```

2.8 La fonction close

```
int close(int fd);
```

Cette fonction ferme le descripteur dans notre cas, elle fermera simplement la socket.

2.9 La fonction sendto

```
int sendto(int s, const void *msg, size_t len, int flags,
const struct sockaddr *to, socklen_t tolen);
```

Cette fonction permet de transmettre un message à destination d'une autre socket en mode Udp.

```
retour = sendto(fdSocket, &buf, sizeof(buf), 0, (struct
sockaddr *) &adresseClient, sizeof(adresseClient));
```

2.10 La fonction recvfrom

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

Cette fonction est utilisées pour recevoir des messages depuis une socket en mode Udp.

```
retour = recvfrom(fdSocket, &valRec, sizeof(valRec), 0, (struct sockaddr *)
&adresseClient, &tailleClient);
```

Les informations concernant celui qui a envoyé sont mises à jour via la fonction revcfrom. Attention à la fonction recvfrom qui à des passages de paramètres par adresse.