Synthèse Socket en C

Client TCP

La socket

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

Taille de la structure
struct sockaddr_in

Adresse mémoire d'une variable de type **struct sockaddr_in** contenant les données correspondant au destinataire : IP, numéro de port, etc

Client TCP

```
int fdSocket;
struct sockaddr_in informationServeur;
int retour;
int tailleClient;

informationServeur.sin_family = AF_INET;
informationServeur.sin_port = htons(5555);
informationServeur.sin_addr.s_addr = inet_addr("172.17.83.110");

fdSocket = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
if (fdSocket == -1)
{
    printf("pb socket : %s\n", strerror(errno));
    exit(errno);
}
    //demande de connexion au serveur
retour = connect(fdSocket, (struct sockaddr *) &informationServeur,
                 sizeof(informationServeur));
if (retour == -1)
{
    printf("pb connect : %s\n", strerror(errno));
    exit(errno);
}
```

Client TCP

adresse en mémoire
de la donnée
à envoyer

La socket

```
ssize_t write(int fildes, const void *buf, size_t nbyte);
```

#include <unistd.h>

taille, en octets,
de la donnée
à envoyer

Client TCP

```
int fdSocket;
struct sockaddr_in informationServeur;
int retour;
int tailleClient;
int valeurEnv, valRetour ;

fdSocket = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
...
retour = connect(fdSocket, (struct sockaddr *) &informationServeur,
                    sizeof(informationServeur));
...
// envoyer donnees au serveur
valeurEnv = 32 ;
retour = write(fdSocket, &valeurEnv, sizeof(valeurEnv));
if (retour == -1)
{
    printf("pb write : %s\n", strerror(errno));
    exit(errno);
}
```

Client TC

adresse de la variable recevant la valeur envoyée

La socket

```
ssize_t read(int fildes, void *buf, size_t nbyte);
```

nombre d'octets, au maximum , réceptionnables

Client TCP

```
int fdSocket;
struct sockaddr_in informationServeur;
int retour;
int tailleClient;
int valeurEnv, valRetour ;

fdSocket = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
...
retour = connect(fdSocket, (struct sockaddr *) &informationServeur,
                    sizeof(informationServeur));
...
// envoyer donnees au serveur
valeurEnv = 32 ;
retour = write(fdSocket, &valeurEnv, sizeof(valeurEnv));
...
// reception en provenance du serveur
retour = read(fdSocket, &valRetour, sizeof(valRetour));
if (retour == -1)
{
    printf("pb read : %s\n", strerror(errno));
    exit(errno);
}
printf("valeur provenant du serveur :%d\n",valRetour) ;
close(fdSocket) ;
```

Client TCP

- A retenir
  - Une seule structure sockaddr_in
  - C'est la fonction ***connect*** qui va permettre la liaison au serveur (c'est elle qui effectue le handshake).
  - read et write, fonctionnent comme des fonctions de lecture et d'écriture dans un fichier.

Serveur TCP

```
int socketFileAttente;
int socketCommunicationClient;
struct sockaddr_in infoServeur;
struct sockaddr_in infoClient;
int retour;
int tailleClient;
int valRec;
socketFileAttente = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
if (socketFileAttente == -1)
{
     printf("pb socket : %s\n", strerror(errno));
     exit(errno);
}

infoServeur.sin_family = AF_INET;
infoServeur.sin_port = htons(2222);
infoServeur.sin_addr.s_addr = htonl(INADDR_ANY); //IP du serveur dans l'ordre des octets du reseau
//attachement ip-port
retour = bind(socketFileAttente, (struct sockaddr*) &infoServeur, sizeof(infoServeur));
if (retour == -1)
{
     printf("pb bind : %s\n", strerror(errno));
     exit(errno);
}

tailleClient = sizeof(infoClient);
```

Serveur TCP

```
int socketFileAttente;
int socketCommunicationClient;
struct sockaddr_in infoServeur;
struct sockaddr_in infoClient;
int retour;
int tailleClient;
int valRec;
fdSocket = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
...

// mise en place d'une file d'attente de 10
retour = listen (socketFileAttente,10) ;
if (retour == -1)
{
     printf("pb listen : %s\n", strerror(errno));
     exit(errno);
}
```

Client TCP

Adresse mémoire d'une variable ayant été initialisée avec la taille de la structure struct sockaddr_in

La socket du serveur

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

Le descripteur de socket permettant la communication avec le client

Adresse mémoire d'une variable de type **struct sockaddr_in** qui sera mise à jour avec les données de celui qui se connecte par la fonction **connect**

Serveur TCP

```
int socketFileAttente;
int socketCommunicationClient;
struct sockaddr_in infoServeur;
struct sockaddr_in infoClient;
int retour;
int tailleClient;
int valRec;
socketFileAttente = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
...
tailleClient = sizeof(infoClient);
while (1==1){
     // attente de connexion
     socketCommunicationClient=accept(socketFileAttente, (struct sockaddr *)&infoClient,&tailleClient) ;
     if (socketClient== -1)
     {
          printf("pb accept : %s\n", strerror(errno));
          exit(errno);
     }
     retour = read(socketCommunicationClient, &valRec, sizeof(valRec));
     if (retour == -1)
     {
          printf("pb read : %s\n", strerror(errno));
          exit(errno);
     }
     valRec=-valRec ;
     retour = write(socketCommunicationClient, &valRec, sizeof(valRec));
     if (retour == -1)
     {
          printf("pb read : %s\n", strerror(errno));
          exit(errno);
     }
}
```

Serveur TCP

- A retenir
  - La fonction **listen** permet de fixer la longueur de la file d'attente
  - La fonction **accept** retourne le descripteur de socket qui **doit** être utilisé pour communiquer avec le client