TD3 – Sockets

1 ère Partie – Introduction à l'API socket

1 – Application serveur et programmation socket

Le code en ANNEXE 1 est celui d'une application de type serveur accessible via des sockets sous TCP/IP.

Sachant que les principales primitives de l'API Socket sont :

```
int socket(int domain, int type, int protocol);
int close(int fd);
int bind(int sockfd, struct sockaddr *my_addr, socklen_t addrlen)
int listen(int s, int backlog);
int accept(int sock, struct sockaddr *adresse, socklen_t *longueur);
int connect(int sock, struct sockaddr *serv_addr, socklen_t addrlen);
int send(int s, const void *msg, int len, int flags);
int write(int s, const void *msg, int  len);
int recv(int s, void *buf, int len, unsigned int flags);
int read(int s, void *buf, int len);

struct sockaddr_in {
   uint8_t         sin_len;       /* longueur totale      */
   sa_family_t     sin_family;    /* famille : AF_INET    */
   in_port_t       sin_port;      /* le numéro de port    */
   struct in_addr  sin_addr;      /* l'adresse internet   */
   unsigned char   sin_zero[8];   /* un champ de 8 zéros  */
};

struct sockaddr
Sélectionnez
struct sockaddr {
   unsigned char   sa_len;         /* longueur totale        */
   sa_family_t     sa_family;      /* famille d'adresse      */
   char            sa_data[14];    /* valeur de l'adresse    */
};

struct hostent {
   char    *h_name;       /* Nom officiel de l'hôte.   */
   char   **h_aliases;    /* Liste d'alias.            */
   int      h_addrtype;   /* Type d'adresse de l'hôte. */
   int      h_length;     /* Longueur de l'adresse.    */
   char   **h_addr_list;  /* Liste d'adresses.         */
}
```

Remarque : dans les fonctions on demande une structure de type **sockaddr**, il faudra caster **sockaddr_in** en **sockaddr**.

1) Remplacez les parties marquées de ?? par les instructions nécessaires à la mise en œuvre de cette application.
2) Testez le fonctionnement de cette application à l'aide de la commande telnet :
   telnet 127.0.0.1 12345

2 – Le client

1) Utiliser les fonctions définies précédemment pour écrire une application client qui envoie au serveur précédent une suite de messages saisis au clavier et affiche les réponses du serveur.

   Remarque : pour le client la fonction gethostbyname, permet de convertir un nom de serveur symbolique « chaine de caractères » en adresse IP « suite de 32 bits »
   Exemple :

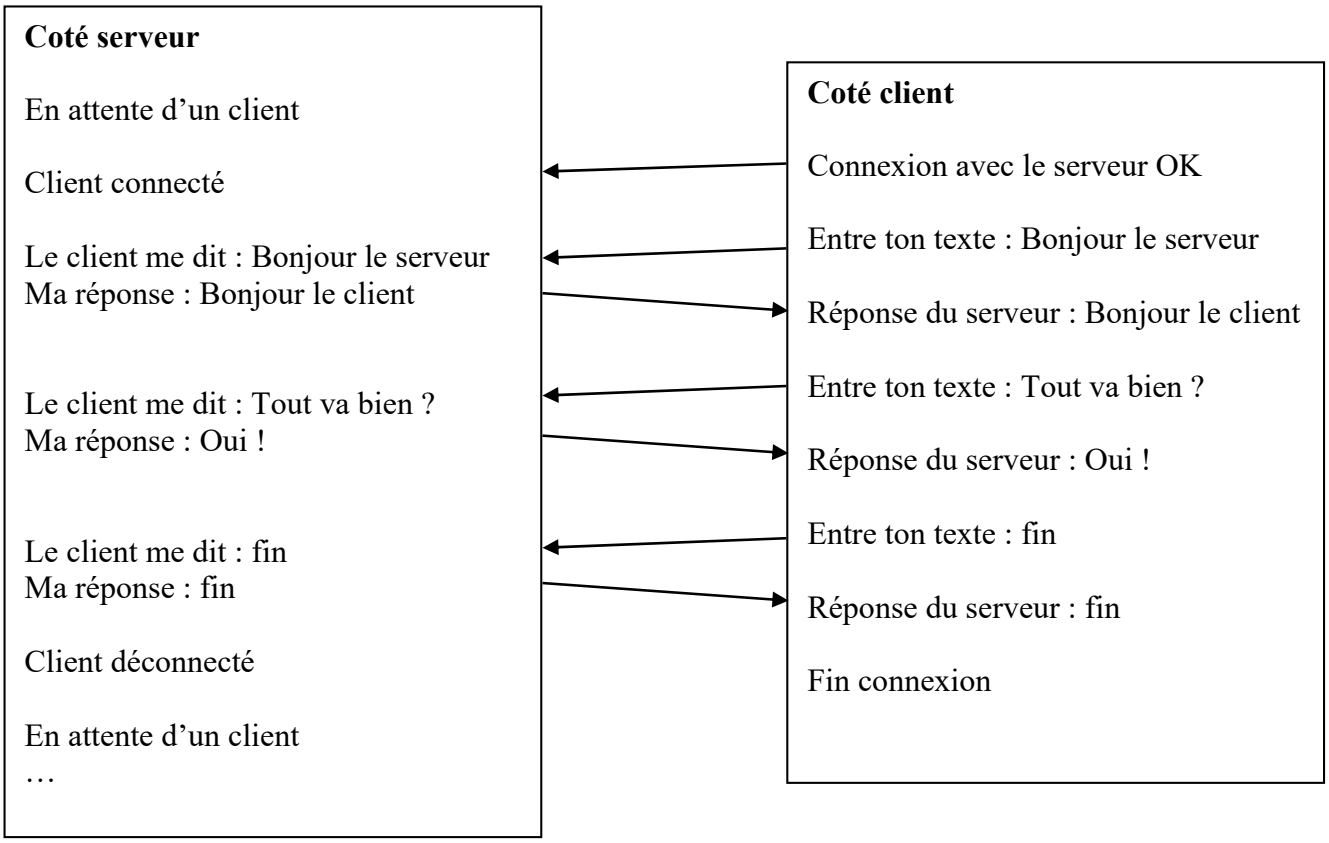
```
#define SERV "127.0.0.1"     // adresse du serveur
struct  hostent    *server;
server = gethostbyname(SERV);
```

2 ème Partie – Ecriture de « chats » et serveur multi-clients

1 – Ecriture d'un « chat » simple

1) Modifier les applications client et serveur précédentes pour qu'elles se comportent comme un véritable « chat ».

   Exemple d'exécution :



2 – Une vrai application client/serveur

Lancez l'application serveur de la question 1 puis lancez deux applications clientes en même temps.

1) Quel est le comportement du serveur ?
2) Comment faire pour que l'application serveur dialogue avec plusieurs clients en même temps ?
3) Faites les modifications nécessaires pour créer un serveur qui dialogue avec plusieurs clients en même temps.
4) Tester l'application serveur avec des applications clientes écrites par d'autres étudiants du groupe.

   Conseil : Modifiez l'application « client » pour que l'adresse IP du serveur soit saisie au clavier en début de programme ou transmise sous forme de paramètre au lancement du programme.

   Rappel : pour obtenir l'adresse IP de votre serveur vous pouvez utiliser la commande : ifconfig sous linux

3 ème Partie – Dialogue avec une service HTTP

1 - Dialogue direct avec un serveur HTTP

Le protocole de transfert hypertexte (HTTP, Hypertext Transfer Protocol) est un protocole de niveau application, de type client-serveur, pour les systèmes d'information hypermédia distribués. Les spécifications complètes sont décrites dans la RFC 2616.

Le dialogue avec un serveur web peut se faire via la commande telnet.
Le dialogue se fait en 3 phases :

- La connexion au site
- L'envoi d'une requête au serveur
- La réponse du serveur

1 La connexion au site

La connexion a un site se fait sur la base d'une adresse de site (nom ou adresse IP) et un numéro de port.
Cette connexion se fait via la commande : telnet nomsite port
Exemple : telnet ffctlr.free.fr 80 ou telnet www.google.fr 80
Remarque : le nom du site peut être remplacé par son adresse IP

2 Envoi de la requête au serveur

Après la connexion, si tout c'est bien passé, vous êtes en relation avec une application particulière.
Vous pouvez maintenant utiliser les requêtes spécifiques au protocole
Pour le cas du protocole HTTP les requêtes sont : GET , HEAD ou POST.

Exemple : GET / HTTP/1.1

3 La réponse du serveur

La réponse du serveur est généralement sous cette forme :
Un code retour pour indiquer quelle suite il donne la requête
Un entête, avec différentes informations sur le service qui répond
Les données correspondant à la réponse

Travail à faire :

Dans cette première partie vous allez vous connecter directement à un (ou plusieurs sites), en composant vous-même les requêtes du protocole HTTP

1- Connectez-vous au site ffctlr.free.fr avec la commande telnet pour exécuter différentes requêtes.

N° requête	Ligne de commande	Requête HTTP
1	telnet ffctlr.free.fr	
2	telnet ffctlr.free.fr 80	GET /

Commentez ce qui se passe dans chaque cas.

3- Recommencez ces opérations avec le site www.google.fr ou une autre site de votre choix.

Ca y est, vous commencez à parler le même langage que les services !

2 - Création d'une application C pour dialoguer avec un service HTTP

On souhaite refaire la même chose que pour la question précédente, mais cette fois à partir d'un programme C utilisant l'API sockets.

Pour cela vous reprendrez le programme client.c de la question 2 du TD3. Vous y apporterez les modifications nécessaires au programme pour :

- Saisir l'adresse IP et le numéro de port du serveur au clavier,
- Saisir au clavier la requête que vous enverrez après la connexion,
- Afficher la réponse du serveur. (Attention la réponse peut être longue, il faudra répéter plusieurs fois l'opération de lecture pour récupérer la totalité de la réponse.)

Question 1

Tester votre application en vous connectant aux sites de la question1, de l'exercice précédent, et comparez les réponses obtenues dans chaque cas.

Remarque : C'est l'adresse IP de google qui doit être saisie et non ffctlr.free.fr. Pour récupérer l'adresse vous pouvez faire un : ping ffctlr.free.fr

Question 2

Recopier les réponses obtenues dans un fichier .html (vous pouvez pour cela utiliser les redirections >) et ouvrez ensuite ce fichier directement en cliquant dessus.
Que se passe-t-il ?

Questions 3

Vous pouvez maintenant utiliser votre programme, pour parcourir les liens d'un document reçu.

ANNEXE 1

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <strings.h>
#include <unistd.h>

#define PORT 12345
int sock, socket2, lg;
char mess[80];
struct sockaddr_in local;   // champs d entete local
struct sockaddr_in distant; // champs d entete distant

void creer_socket()
{
 // preparation des champs d entete
   bzero(&local, sizeof(local));                // mise a zero de la zone adresse
   local.sin_family = AF_INET;                  // famille d adresse internet
   local.sin_port = htons(PORT);                // numero de port
   local.sin_addr.s_addr = INADDR_ANY;  // types d adresses prises en charge
   bzero(&(local.sin_zero),8);                  // fin de remplissage

   lg = sizeof(struct sockaddr_in);
   ??     // creation socket du serveur mode TCP/IP
   ??     // nommage de la socket
}

main()
{
   // creation socket
   creer_socket();

   ??     // mise a l ecoute
          ??

   // boucle sans fin pour la gestion des connexions
   while(1)
     {  // attente connexion client
          printf ("En  attente d un client\n");
          ??
          printf ("client connecte \n");
          strcpy(mess,"");
          while (strncmp(mess,"fin",3)!=0)
          { read(socket2,mess,80);
            printf ("le client me dit %s \n",mess);
           write(socket2, "message recu !",80);
          }
        close(socket2);       // on lui ferme la socket
     }
}
```