Client – Serveur mode connecté

Le but de ce TDM est d'écrire un programme client (nous le nommerons strmcli) et un programme serveur (nous le nommerons strmser). La communication entre le serveur et le client se fera en mode connecté.

Description du serveur

Le serveur est de type séquentiel et effectue une boucle dans laquelle il affiche le message :

ATTENTE D UNE CONNEXION.....

Puis, il passe en attente d'une demande de connexion de la part d'un client.
Lorsqu'un client se connecte, il affiche le message :

JE TRAITE LA CONNEXION QUI VIENT D ARRIVER….

Le traitement consiste à afficher:

- Le nom et l'adresse IP de la machine du client
- Le n° de port utilisé par le client
- Le nombre de caractères envoyés par le client
- La chaîne de caractères qui lui est envoyée par le client.

Le serveur se termine lorsqu'un client lui envoie une chaîne de caractères composée du seul caractère #. Il affiche alors le message : *FIN DE SERVICE*

```
#include <stdio.h>
#include <errno.h>
#include <strings.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

#define MAX_CONNEXION 5
#define MAX_NOM 40
```



```
#define MAX_LIGNE 80

struct sockaddr_in InfoSockEcoute;
struct sockaddr_in InfoSockClient;

char NomMachineServeur[MAX_NOM],
     ligne [MAX_LIGNE];

int SockEcoute,
    SockService,
    PortServeur;
socklen_t LgInfoSockClient; /* marche aussi avec type int: socklen_t
normalise POSIX */

struct hostent *InfoMachineServeur, *infoMachineClient;

int BindOK, ListenOK, GetOK;

int nr, i;

/*=========================================================================*/
  main(int argc, char *argv[]){
/*=========================================================================*/
    if (argc <2){
        printf("\n\n  Usage %s: Numero_port\n\n", argv[0]);
        exit(1);
    }

/*=========================================================================*/
/*   RECHERCHE DE L ADRESSE DE LA MACHINE DU SERVEUR                       */
/*=========================================================================*/
    printf("\n=>DEMARRAGE DU SERVEUR....\n");
    printf("\t->RECHERCHE DE L ADRESSE DE LA MACHINE DU SERVEUR\n");

    GetOK=gethostname(NomMachineServeur, MAX_NOM);
    if (GetOK<0){
       perror("\nERREUR 1 machine serveur");
       exit(1);
    }

    InfoMachineServeur=gethostbyname (NomMachineServeur);
    if (InfoMachineServeur==NULL){
       perror("\nERREUR 2 machine serveur");
       exit(1);
    }
```

```
/*=========================================================================*/
/*  CREATION DE LA SOCKET D ECOUTE                                         */
/*=========================================================================*/

    PortServeur=atoi(argv[1]);

    printf("\t->CREATION DE LA SOCKET D'ECOUTE\n");

    SockEcoute=socket(AF_INET, SOCK_STREAM, 0);
    if (SockEcoute<0){
       perror("\nERREUR creation socket");
       exit(1);
    }

/*=========================================================================*/
/*   PREPARATION DU NOMAGE DE LA SOCKET                                    */
/*=========================================================================*/
    printf("\t->PREPARATION DU NOMMAGE DE LA SOCKET\n");

    bzero(&InfoSockEcoute, sizeof(InfoSockEcoute));

    InfoSockEcoute.sin_family=htons(AF_INET);

    InfoSockEcoute.sin_port=htons(PortServeur);

    bcopy((char *)InfoMachineServeur->h_addr,
          (char *)&InfoSockEcoute.sin_addr,
                  InfoMachineServeur->h_length);

/*=========================================================================*/
/*   NOMMAGE DE LA SOCKET D ECOUTE                                         */
/*=========================================================================*/
    printf("\t->NOMMAGE DE LA SOCKET D'ECOUTE\n");

    BindOK=bind (SockEcoute, (struct sockaddr *)&InfoSockEcoute,
sizeof(InfoSockEcoute));
    if (BindOK<0) {
       perror("\nERREUR bind serveur");
       exit(1);
    }

/*=========================================================================*/
/*   ECOUTE                                                                */
/*=========================================================================*/
    printf("\t->DIMENSIONNEMENT DE LA FILE D ATTENTE\n");

    ListenOK=listen (SockEcoute, MAX_CONNEXION);
```



```
if (ListenOK<0){
   perror("\nERREUR listen ");
   exit(1);
}

LgInfoSockClient=sizeof(InfoSockClient);

ligne[0]=' ';
while (ligne[0]!='#'){
     for (i=0;i<MAX_LIGNE;i++)ligne[i]=' '; /* Nettoyage de la ligne */

    /*=========================================================*/
    /*   PRISE EN COMPTE D'UNE DEMANDE DE CONNEXION            */
    /*=========================================================*/

     printf("\n\t=============================================\n");
     printf ("\t->ATTENTE D'UNE CONNEXION.....\n");

     SockService=accept(SockEcoute, (struct sockaddr *)&InfoSockClient,
                                     &LgInfoSockClient);
     if (SockService<0){
         perror("\nERREUR accept");
         exit(1);
     }

    /*=============================================================*/
    /*  LECTURE DES DONNEES DU CLIENT SUR LA SOCKET DE SERVICE  */
    /*  ET TRAITEMENT DE LA REQUETE                              */
    /*=============================================================*/
     printf("\n\t->UNE CONNEXION VIENT D'ARRIVER DE: %s / %d \n",
          inet_ntoa (InfoSockClient.sin_addr),
          ntohs(InfoSockClient.sin_port));

     InfoMachineServeur=gethostbyaddr (&InfoSockClient,
                                       sizeof(InfoSockClient), AF_INET);
     if (InfoMachineServeur==NULL){
         perror("\nMachine client inconnue du DNS");
         exit(1);
     }

     nr=read (SockService, ligne, MAX_LIGNE);
     if (nr<0){
         perror("\nERREUR read");
     }
     else {
           if (nr<(MAX_LIGNE-1))
              ligne[nr]='\0';
```

```
            else
               ligne[MAX_LIGNE-1]='\0';

            if (ligne[0]!='#')
                printf("\t->%d caractéres reçus:%s \n",nr, ligne);
         }/* if (nr<0) */

        /*=========================================================*/
        /*  FERMETURE DE LA SOCKET DE TRAVAIL                      */
        /*=========================================================*/
         printf ("\t->FIN DE LA CONNEXION EN COURS\n");

         printf ("\t\t.... FERMETURE DE LA SOCKET DE TRAVAIL\n");
         close (SockService);

    }/* while (ligne[0]!='#') */

/*=========================================================*/
/*  FERMETURE DE LA SOCKET D ECOUTE                        */
/*=========================================================*/
    printf ("\n\t->FERMETURE DE LA SOCKET D ECOUTE\n");

    close (SockEcoute);

    printf ("=>FIN DE SERVICE\n\n");

} /* main */
```

Description du client

Le client invite l'utilisateur à entrer une chaîne de caractères. Il se connecte ensuite au serveur puis lui envoie la chaîne de caractères entrée par l'utilisateur. Enfin, il se termine.

```
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

#define MAX_NOM 50
#define MAX_LIGNE 80
```



```
struct sockaddr_in InfoSockServeur;

char NomMachineServeur[MAX_NOM],
     ligne [MAX_LIGNE];

int SockVersServeur,
    PortServeur;

int ConnectOK;

int nw;

struct hostent *InfoMachineServeur;

main(int argc, char *argv[]){

    if (argc <3){
        printf("\n\n  Usage %s: Nom_machine_serveur Numero_port\n\n", argv[0]);
        exit(1);
    }

/*=========================================================================*/
/*   RECHERCHE DE L ADRESSE DE LA MACHINE DU SERVEUR                       */
/*=========================================================================*/
    errno=0;
    printf("\n     ->RECHERCHE DE L ADRESSE DE LA MACHINE %s\n", argv[1]);

    strcpy(NomMachineServeur, argv[1]);

    InfoMachineServeur=gethostbyname(NomMachineServeur);
    if (InfoMachineServeur==NULL){
       printf("\nERREUR machine serveur inconnue\n");
       exit(1);
    }

/*=========================================================================*/
/*  CREATION DE LA SOCKET DE DIALOGUE AVEC LE SERVEUR                      */
/*=========================================================================*/
    printf("     ->CREATION DE LA SOCKET DE DIALOGUE AVEC LE SERVEUR\n");

    SockVersServeur=socket(AF_INET, SOCK_STREAM, 0);
    if (SockVersServeur<0){
        perror("\nERREUR SOCKET ");
        exit(1);
    }

/*=========================================================================*/
```

```
/*   PREPARATION DE LA CONNEXION AVEC LE SERVEUR                          */
/*=========================================================================*/
    printf("     ->PREPARATION DE LA CONNEXION AVEC LE SERVEUR\n");

    bzero(&InfoSockServeur, sizeof(InfoSockServeur));

    InfoSockServeur.sin_family=AF_INET;
    PortServeur=atoi(argv[2]);
    InfoSockServeur.sin_port=htons(PortServeur);

    bcopy((char *)InfoMachineServeur->h_addr,
          (char *)&InfoSockServeur.sin_addr,
                  InfoMachineServeur->h_length);

/*=========================================================================*/
/*   CONNEXION AVEC LE SERVEUR                                             */
/*=========================================================================*/
    printf("     ->CONNEXION AVEC LE SERVEUR:\n");
    printf("       IP: %s / Port: %d ...\n",
                        inet_ntoa(InfoSockServeur.sin_addr), PortServeur);

    ConnectOK=connect(SockVersServeur,
                      (struct sockaddr *)&InfoSockServeur,
                      sizeof(InfoSockServeur));
    if (ConnectOK<0){
       perror("\n         ....ERREUR CONNECT");
       exit(1);
    }
    else printf("        ...CONNEXION ETABLIE\n\n");

/*=========================================================================*/
/*   ENVOI DES DONNEES AU SERVEUR                                          */
/*=========================================================================*/
    printf ("\n     ->Entrer une suite caracteres:");

    fgets(ligne, MAX_LIGNE, stdin);
    ligne[strlen(ligne)-1]='\0';// -- suppression du retour chariot

    nw=write (SockVersServeur, ligne, strlen(ligne));
    if (nw<0)
        perror("     ERREUR write ");
    if (nw==0)
         printf ("     ->Problème write\n");
    else printf ("     ->%d caractéres envoyes au serveur\n", nw);

/*=========================================================================*/
/*   FERMETURE DE LASOCKET DE DIALOGUE AVEC LE SERVEUR                     */
```



```
/*=========================================================================*/
    printf("\n     =>FERMETURE DE LA SOCKET DE DIALOGUE AVEC LE SERVEUR\n");
    close (SockVersServeur);

}// main
```

Adresse du serveur

Le serveur adoptera l'adresse de la machine sur laquelle il fonctionne. Cette adresse sera déterminée à l'aide de `gethostname` et `gethostbyname`.

N° de port du serveur

Le serveur n'étant pas répertorié dans l'annuaire, il est inutile d'utiliser `getservbyname` pour déterminer son numéro de port. Chaque étudiant fera tourner son serveur sur sa machine de travail au moment du TDM. Le numéro de port de son serveur sera déterminé de la façon suivante : 5600 + N° de son PC.
Par exemple, si le PC est pc508-12, le n° de port du serveur sera 5600 + 12 = **5612**. De cette manière, tous les serveurs développés auront un n° de port différent au moment du TDM.
Attention, si ce TDM s'étale sur plusieurs séances et que vous changez de PC, ce n° de port devra être recalculé en conséquence.

Lancement du serveur

```
strmser Numero_port_serveur
```

Localisation du client

Chaque étudiant fera tourner son client sur une machine différente de la machine du serveur.

Lancement du client

```
strmcli Nom_machine_serveur Numero_port_serveur
```

Serveur concurrent

Si vous le souhaitez, réécrire le serveur de façon concurrente.

NB :

- Les variables de type `struct sockaddr_in` seront mises à zéro avec `bzero` avant toute utilisation.
- Le champ `sin_addr` sera valué en utilisant `bcopy`.
- Le champ `sin_port` sera valué en utilisant `htons`.
- Tous les codes retour des fonctions de l'API socket seront testés et, en cas d'erreur, un message sera affiché au moyen de `perror` (cf. man).
- On se préoccupera également des valeurs NULL éventuellement retournées par `gethostbyname` qui seront signalées par un message (`printf`).

Serveur multi-threadé

En utilisant les threads, réécrire le serveur afin que :

- il réponde sur plusieurs sockets d'écoute à la fois
- il propose à l'utilisateur un menu pour:
  - afficher la liste des connexions en cours
  - arrêter le serveur