TD Programmation Système Série 7

Correction : Communication par Socket

Exercice 1 : Test de socket

1.

```
#include<stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include<string.h>
#include<sys/un.h>
#include<stdlib.h>

int main(int argc,char* argv[])
{
        int ids;
        struct sockaddr_un adr;

  ids= socket(AF_UNIX,SOCK_DGRAM,0);
  if(ids==-1){
        perror("socket");
        exit(0);}

       printf("Identificateur de la socket : %d \n", ids);

  close(ids);
    return 0;
}
```

2.

```
int main(int argc,char* argv[])
{
        int ids,idb;
        struct sockaddr_un adresse;

  ids= socket(AF_UNIX,SOCK_DGRAM,0);

   if(ids==-1){
                perror("socket");
                exit(0);}

   /* Préparation de l'adresse de l'attachement */
   adresse.sun_family =AF_UNIX;
   strcpy(adresse.sun_path, argv[1]);

  idb=bind(ids,  (struct sockaddr *)&adresse, sizeof(adresse));
   if(idb==-1){
                perror("bind");
                exit(0);}

  close(ids);
  return 0;
}
```

3. Pour pouvoir observer par la commande netstat la socket crée en mode non connecté et son attachement au fichier passé en paramètre il faut ajouter une boucle infinie après l'attachement de la socket au fichier passé en paramètre.

a.

```
int main(int argc,char* argv[])
{
        int ids,idb;
        struct sockaddr_un adresse;

  ids= socket(AF_UNIX,SOCK_DGRAM,0);
  if(ids==-1){
                perror("socket");
                exit(0);}

   printf("Identificateur de la socket : %d \n", ids);

   /* Préparation de l'adresse de l'attachement */
   adresse.sun_family =AF_UNIX;
   strcpy(adresse.sun_path, argv[1]);

  idb=bind(ids,  (struct sockaddr *)&adresse, sizeof(adresse));
   if(idb==-1){
                perror("bind");
                exit(0);}

   while(1) ; // créer une boucle infinie

  close(ids);
  return 0;
}
```

b. Lancer l'exécution avec le fichier /tmp/test (./excutable **/tmp/test**)

Exécuté la commande **netstat :**

```
Connexions Internet actives (sans serveurs)
Proto Recv-Q Send-Q Adresse locale        Adresse distante       Etat
Sockets du domaine UNIX actives(sans serveurs)
Proto RefCnt Flags       Type       State         I-Node   Chemin
unix  7      [ ]         DGRAM                    4889     /dev/log
unix  2      [ ]         DGRAM                    2279     @/org/kernel/udev/udevd
unix  2      [ ]         DGRAM                    5458     @/org/freedesktop/hal/udev_event
unix  2      [ ]         DGRAM                    7956     /tmp/test
unix  3      [ ]         STREAM     CONNECTE      7787     /tmp/ksocket-mohammed/konqueror33Jylc.slave-
socket
unix  3      [ ]         STREAM     CONNECTE      7786
unix  3      [ ]         STREAM     CONNECTE      7770     /tmp/.ICE-unix/2324
unix  3      [ ]         STREAM     CONNECTE      7769
unix  3      [ ]         STREAM     CONNECTE      7768     /tmp/.X11-unix/X0
unix  3      [ ]         STREAM     CONNECTE      7767
unix  3      [ ]         STREAM     CONNECTE      7762     /tmp/.ICE-unix/dcop2313-1464877830
unix  3      [ ]         STREAM     CONNECTE      7761
unix  3      [ ]         STREAM     CONNECTE      7754     /tmp/ksocket-mohammed/kwritecs8Tya.slave-socket
unix  3      [ ]         STREAM     CONNECTE      7753
unix  3      [ ]         STREAM     CONNECTE      7727     /tmp/ksocket-mohammed/klauncher3Y5NSa.slave-
socket
unix  3      [ ]         STREAM     CONNECTE      7726
unix  3      [ ]         STREAM     CONNECTE      7552     /tmp/.famqeGm5n
```

Exercice 2 : communication interprocessus par socket

1. Communication en mode UDP unidirectionnelle

Processus Client	Processus Serveur
#include<stdio.h> #include <unistd.h> #include <sys/types.h> #include <sys/socket.h> #include<string.h> #include<sys/un.h> #include<stdlib.h> int main(int argc,char* argv[]) { int ids,idb; struct sockaddr_un adresseS; char c; ids= socket(AF_UNIX,SOCK_DGRAM,0); if(ids==-1){ perror("socket"); exit(0);} /* Préparation de l'adresse du Processus Serveur pour Envoi*/ adresseS.sun_family =AF_UNIX; strcpy(adresseS.sun_path, argv[1]); c=getchar(); sendto(ids, &c, sizeof(char), 0, (struct sockaddr *) &adresseS,sizeof(adresseS)); close(ids); return 0; }	#include<stdio.h> #include <unistd.h> #include <sys/types.h> #include <sys/socket.h> #include<string.h> #include<sys/un.h> #include<stdlib.h> #include <ctype.h> int main(int argc,char* argv[]) { int ids,idb; struct sockaddr_un adresseS, adresseC; char c; ids= socket(AF_UNIX,SOCK_DGRAM,0); if(ids==-1){ perror("socket"); exit(0);} /* Préparation de l'adresse du Processus Serveur pour Attachement*/ adresseS.sun_family =AF_UNIX; strcpy(adresseS.sun_path, argv[1]); idb= bind(ids, (struct sockaddr *) &adresseS, sizeof(adresseS)); if(idb==-1){ perror("bind"); exit(0);} /* adresseC adresse de l'expéditeur */ recvfrom(ids, &c, sizeof(char), 0, (struct sockaddr *) &adresseC,(socklen_t*)sizeof(adresseC)); printf("caractère reçu :%c\n", toupper(c)); close(ids); return 0; }

2. Communication en mode UDP bidirectionnelle

Processus Client	Processus Serveur
#include<stdio.h> #include <unistd.h> #include <sys/types.h> #include <sys/socket.h> #include<string.h> #include<sys/un.h> #include<stdlib.h> int main(int argc,char* argv[]) { int ids,idb; struct sockaddr_un adresseC, adresseS; char c; ids= socket(AF_UNIX,SOCK_DGRAM,0); if(ids==-1){ perror("socket"); exit(0);} /* Préparation de l'adresse du Processus Client pour Attachement*/ adresseC.sun_family =AF_UNIX; strcpy(adresseC.sun_path, argv[1]); idb=bind(ids, (struct sockaddr *) &adresseC, sizeof(adresseC)); if(idb==-1){ perror("bind"); exit(0);} /* Préparation de l'adresse du Processus Serveur pour Envoi*/ adresseS.sun_family =AF_UNIX; strcpy(adresseS.sun_path, argv[2]); /* Communication Bidirectionnelle */ c=getchar(); sendto(ids, &c, sizeof(char), 0, (struct sockaddr *) &adresseS,sizeof(adresseS)); recvfrom(ids, &c, sizeof(char), 0, (struct sockaddr *) &adresseS,(socklen_t*)sizeof(adresseS)); printf("caractère reçu :%c\n", c); close(ids); return 0; }	**#include<stdio.h> #include <unistd.h> #include <sys/types.h> #include <sys/socket.h> #include<string.h> #include<sys/un.h> #include<stdlib.h> #include <ctype.h> int main(int argc,char* argv[]) { int ids,idb; struct sockaddr_un adresseS, adresseC; char c; ids= socket(AF_UNIX,SOCK_DGRAM,0); if(ids==-1){ perror("socket"); exit(0);} /* Préparation de l'adresse du Processus Serveur pour Attachement*/ adresseS.sun_family =AF_UNIX; strcpy(adresseS.sun_path, argv[1]); idb=bind(ids, (struct sockaddr *) &adresseS, sizeof(adresseS)); if(idb==-1){ perror("bind"); exit(0);} /* adresseC adresse de l'expéditeur */ int taille=sizeof(adresseS); recvfrom(ids, &c, sizeof(char), 0, (struct sockaddr *) &adresseC,(socklen_t*) &taille); c=toupper(c); printf("caractère reçu :%c\n", c); sendto(ids, &c, sizeof(char), 0, (struct sockaddr *) &adresseC,taille); close(ids); return 0; }**