# TD Programmation Système Série 7

# **Correction: Communication par Socket**

### Exercice 1 : Test de socket

2.

```
#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;
}
```

```
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.

```
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
                                        @/org/kernel/udev/udevd
        []
               DGRAM
                                 2279
unix 2
               DGRAM
                                 5458
                                        @/org/freedesktop/hal/udev_event
        []
unix 2
               DGRAM
                                  7956
                                        /tmp/test
        []
               STREAM
                          CONNECTE
                                        7787
                                               /tmp/ksocket-mohammed/kongueror33Jylc.slave-
unix 3
        []
socket
                          CONNECTE
unix 3
               STREAM
                                        7786
        []
               STREAM
                          CONNECTE
                                        7770
                                              /tmp/.ICE-unix/2324
unix 3
        []
               STREAM
                          CONNECTE
                                        7769
unix 3
        []
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
               STREAM
                          CONNECTE
                                        7726
unix 3
        []
unix 3
               STREAM
                          CONNECTE
                                        7552
        []
                                               /tmp/.famgeGm5n
```

## Exercice 2: communication interprocessus par socket

## 1. Communication en mode UDP unidirectionnelle

Processus Client	Processus Serveur
#include <stdio.h></stdio.h>	#include <stdio.h></stdio.h>
#include <unistd.h></unistd.h>	#include <unistd.h></unistd.h>
#include <sys types.h=""></sys>	#include <sys types.h=""></sys>
#include <sys socket.h=""></sys>	#include <sys socket.h=""></sys>
#include <string.h></string.h>	#include <string.h></string.h>
#include <sys un.h=""></sys>	#include <sys un.h=""></sys>
#include <sys dif.ii=""> #include<stdlib.h></stdlib.h></sys>	#include <sys dif.ii=""> #include<stdlib.h></stdlib.h></sys>
#include <stalib.n></stalib.n>	
int main(int argc,char* argv[]) {	#include <ctype.h></ctype.h>
int ids,idb;	int main(int argc,char* argv[])
struct sockaddr_un adresseS;	{
char c;	int ids,idb;
onar o,	struct sockaddr_un adresseS, adresseC;
<pre>ids= socket(AF_UNIX,SOCK_DGRAM,0); if(ids==-1){</pre>	char c;
perror("socket");	ids= socket(AF_UNIX,SOCK_DGRAM,0);
	if(ids==-1){
exit(0);}	
/* Préparation de l'adresse du Processus Serveur pour Envoi*/	<pre>perror("socket"); exit(0);}</pre>
- In a confirmation of the AF HARV	(* Defended to the Heaters of the December 1
adresseS.sun_family =AF_UNIX;	/* Préparation de l'adresse du Processus
strcpy(adresseS.sun_path, argv[1]);	Serveur pour Attachement*/
	adresseS.sun_family =AF_UNIX;
c=getchar();	strcpy(adresseS.sun_path, argv[1]);
condto/ide %a cizact/abor\ 0 (ctruct cookaddr *)	idb- hind/ida (atrust cookaddr *) ? adracca?
sendto(ids, &c, sizeof(char), 0, (struct sockaddr *)	idb= bind(ids, (struct sockaddr *) &adresseS,
&adresseS,sizeof(adresseS));	sizeof(adresseS));
	if(idb==-1){
close(ids);	perror("bind");
return 0;	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;
	, i
	}
	, and the second

#### 2. Communication en mode UDP bidirectionnelle

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