Info2 #, Programmation système, TP4 : Communication entre processus par sockets, Enseignant : M. Nakechbandi

Exercice 1 (mode non connecté)

I. Récupérer à partir de `corton --> /tmp/info2#/tp4` le programme `creesock.c`, l'analyser, le compiler et l'exécuter. On remarque que ce programme va créer une socket, puis lui donne le nom ("l'adresse") Opera.

II. Vérifier (`ls -a`) l'existence du ficher "Opera", quel il est le type de ce fichier ?

III. Exécuter à nouveau ce programme, puis analyser le message d'erreur, proposer une solution. Indication : Utiliser la fonction `unlink`

Exercice 2 (mode non connecté)

I. On vous fournit sur `corton --> /tmp/info2#/tp4` (voir également l'annexe) deux squelettes `client_unix_dgram.c, serveur_unix_dgram.c` de code **client** et **serveur** que vous devez recopiez et compléter à partir des éléments du cours décrits plus haut.

II. Lancer le serveur en background (`serveur_unix_dgram &`) puis le client.

Remarque : Pour tuer le serveur, soit revenir en foreground (fg) et faire <Ctrl-C>, soit tuer le processus par kill après l'avoir repéré son `pid` par "`ps ax`".

III. Modifier les deux programmes précédents pour que le serveur envoie une confirmation (accusé de reception) au client.

Annexe

```
/*    programme creesock.c  */
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <errno.h>
struct sockaddr_un adresse = {AF_UNIX, "Opera"};
main ()
{int desc;               /* pour le descripteur */
 int lng;      /* longueur de l'adresse */
 if ((desc = socket(AF_UNIX, SOCK_STREAM, 0)) == -1){ perror (" socket");
exit (1); }
 adresse.sun_family = AF_UNIX;
 lng = sizeof(adresse);
 if(bind(desc, (struct sockaddr *)  &adresse, lng) == -1){
perror("bind"); exit (1); }
 close(desc);}
```

```
/*    client_unix_dgram.c : */
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
```

```
#include <errno.h>
#include <unistd.h>      /* declaration de unlink */
struct sockaddr_un agence = {AF_UNIX,"Opera"};
char msg[] = "Reservez moi main 16 places au Volcan";
int main(){
 int desc;     /* Descripteur de socket */
 int lg;  /* Longueur de l'adresse */
/* Creation d'une socket */
 if ((desc = socket(AF_UNIX, SOCK_DGRAM, 0)) == -1)
    { perror (" socket"); exit (1);}
 lg = sizeof(agence);
 /* on envoie le message a l'agence Opera */
 if (sendto(desc,msg,sizeof(msg),0, (struct sockaddr *) ..., lg) == -1)
     {perror("sendto"); exit (1);}
}
```

```
/*   serveur_unix_dgram.c  */
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <errno.h>
#include <unistd.h>
/* declaration de unlink */
struct sockaddr_un agence = {AF_UNIX,"Opera"};
#define LGMESS 80             /* longueur d'un message attendu */
char msg[LGMESS];             /* pour placer le message recu */
int main() {
 int desc;                    /* pour le descripteur */
 int lg;                 /* longueur de l'adresse */
 int lgrecu;
                         /* creation d'une socket */
 if ((desc = socket(..., 0)) == -1){
     perror (" socket"); exit (1);
     }
/* on nomme la socket Opera */
 unlink(...);
 lg = sizeof(agence);
 if (bind(desc,...    ,lg) == -1) {
     perror ("bind"); exit (1);
     }
 lgrecu = -1;                 /* on attend un message */
 while(lgrecu == -1) {             /* boucle d'attente */
     lgrecu = recvfrom(desc,msg,LGMESS,0,... , ...);
     }
 printf("Message recu du client %s \n%s\n", client.sun_path, msg);
close(desc);
 exit(0);}
```

Exercice 3 (mode connecté)

I On vous demande maintenant de modifier les programmes précédents (premières versions exercice II) du serveur et du client de telle sorte que la communication entre le nouveau serveur et le nouveau client se fasse en mode connecté, c'est-à-dire en établissant un canal de

communication préliminaire entre le client et le serveur. On nommera le serveur `serveur_unix_`stream`.c` et le client `client_unix_`stream`.c`

II Idem que l'exercice 2.III, modifier les programmes précédents pour que le serveur envie une confirmation au client.

Exercice 4 (mode connecté)
Modifier le résultat de l'exercice 3.II pour réaliser un serveur qui reçoit (en mode connecté) un message du cient1 puis l'envoie à un autre client2.

Exercice 5 (mode connecté)
Un serveur et n clients dialoguent ensemble. Un client (le numéro 1) envoie un message `m` au serveur, le serveur diffuse ce message `m` aux autres clients. Modifier les résultats de l'exercice 4 pour réaliser cette discussion à plusieurs.

(ne pas oublier d'envoyer un **CR** à nakech@free.fr, sujet tp4ex1&2)