Conservatoire National des Arts et Métiers
FOAD Ile De France

Les Socket Unix

Exercices et Travaux Pratiques
11 novembre 2020

Première partie

Compléments de cours : Glossaire

Termes utilisés dans le contexte "Unix Sockets"

- **packet :** Objet élémentaire véhiculé par le réseau (Datagram)
- **message :** Données transportées par un "packet" (User Datagram ou Segment)
- **channel :** Canal de communication entre 2 applications (Session UDP ou TCP).
- **socket :** Point terminal de communication (spécifique Unix).
- **socket address :** Adresse IP / n° de port / protocol family
- **socket descriptor :** Identificateur ou descripteur de socket (Utilisable comme un descripteur de fichier avec read, write flush etc.).
- **stream :** mode connecté (TCP)
- **datagram :** mode non connecté (message UDP)
- **protocol family :** pile de protocole utilisée
  - TCP/IP (paramètre PF_INET)
  - SNA,
  - DECNET, X25 etc.
- **protocol :** Le protocole spécifique de la famille utilisé

Compléments de cours : Primitives

recvfrom : (spécifique datagram, sans connect virtuel)

- fonction : demande de réception en mode DATAGRAM

sendto : : (spécifique datagram, sans connect virtuel)

- fonction : demande d'émission en mode DATAGRAM

send / recv :

- variantes de **write /read** et **sendto / recvfrom** permettant de gérer les options et en particulier l'urgence et le flush.

ioctl :

- fonction : configuration télécom de la communication.

fcntl :

- Configuration temps réel de la communication (mode bloquant ou non ...).

getsockname :

- fonction : Obtenir l'adresse socket locale a partir du descripteur socket local.

getpeername :

- fonction : Obtenir l'adresse socket distant a partir du descripteur du socket distant.

getsockopt / setsockopt :

- fonction : Consulter / modifier les options d'un socket.

gethostname :

- fonction : Obtenir le nom de la machine locale.

gethostbyname :

- fonction : Obtenir l'adresse d'un hôte à partir de son nom (consultation /etc/hosts).

getservbyname :

- fonction : Obtenir le numéro d'un service à partir de son nom (consultation /etc/services).

Exercice 1 : Réception UDP

Faites fonctionner l'envoi et la réception d'un message UDP sans pseudo-connexion.

Le programme de réception possède 2 paramètres d'entrée :
- adresse IP distant
- numéro de port distant

Les 2 hôtes Les 2 hôtes d'égal à égal se trouveront sur le même ordinateur et utiliseront l'adresse IP de rebouclage 127.0.0.1

```
/*****************************************************************
   Role de ce module

   Boucle d'attendre et de reception d'un message UDP

   -> En argument d'entree :
              - numero de Port d'écoute,

   -> Sur reception de message :
              - affichage du message et de l'émetteur,
              - sortie.

      CE PROGRAMME NE SE REMET PAS EN ATTENTE DU MESSAGE SUIVANT
 *****************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <errno.h>
#define MAXMESS 100

int main(int argc, char** argv) {
  int sock;
  char message[MAXMESS+1];
  struct sockaddr_in local, distant;
  unsigned short port;
  unsigned short distant_port;
  char distant_adresse[50];
  char* ptn_adresse = distant_adresse;
  int len, n;
  errno = 0;

  /* Contrôle des arguments */
  if (argc != 2) goto usage;
  port =  (unsigned short) atol (argv[1]);

  /* Création socket */
  if ((sock = socket(AF_INET,SOCK_DGRAM, 0)) < 0) goto faute1;

  /* Bind */
  memset(&local, 0, sizeof(&local));      //precaution
  local.sin_family = AF_INET;
```

```
  local.sin_addr.s_addr = INADDR_ANY;    // une des adresses IP locales
  local.sin_port = htons(port);          // numéro de port
  if ( bind(sock, (const struct sockaddr *)&local, sizeof(local)) < 0)
                                                          goto faute2;

  /* Attente et Réception */
  printf("SERVEUR UDP A L'ECOUTE SUR PORT %d\n",port);
  n = recvfrom (sock, (char *)message, MAXMESS, 0,
                                    (struct sockaddr *)&distant, &len);
  message[n] = '\0';
  ptn_adresse = inet_ntoa(distant.sin_addr);
  distant_port = ntohs(distant.sin_port);
  printf("Reçu de %s/%d : %s\n", ptn_adresse,distant_port,message);

  /* Retour OK */
  close(sock);
  exit(0);

  /* USAGE */
  usage:
  printf ("Usage : serveur <Port>\n>");
  exit (-1);

  /* CAS DE FAUTES */
  faute1:
  perror("faute socket : ");
  exit(-2);
  faute2:
  perror("faute bind : ");
  exit(-3);
}
```

Exercice 2 : Emission UDP

Faites fonctionner l'envoi et la réception d'un message UDP sans pseudo-connexion.

Le programme d'émission possède 3 paramètres d'entrée :
- adresse IP distant
- numéro de port distant
- texte du message

Les 2 hôtes d'égal à égal se trouveront sur le même ordinateur et utiliseront l'adresse IP de rebouclage 127.0.0.1

```
/*****************************************************************
   Role de ce module

   Emettre un message UDP
   En entree :
              -> adresse IP destinataire,
              -> numero de Port destinataire,
              -> Texte du message

 *****************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <errno.h>
#define MAXMESS 100

int main(int argc, char** argv) {
  int sock;
  char message[MAXMESS];
  struct sockaddr_in local, distant;
  unsigned int adresse;
  unsigned short port;
  int len, n;
  errno = 0;

  /* Contrôle des arguments */
  if (argc != 4) goto usage;
  if (inet_aton (argv[1],(struct in_addr *)&adresse) == 0) goto usage;
  port =  (unsigned short) atol (argv[2]);
  strncpy (message, argv[3],MAXMESS+1);

  /* Création socket */
  if ((sock = socket(AF_INET,SOCK_DGRAM, 0)) < 0) goto faute1;

  /* Bind */
  memset(&local, 0, sizeof(&local));      //precaution
  local.sin_family = AF_INET;
  local.sin_addr.s_addr = INADDR_ANY;     // une des adresses IP locales
  local.sin_port = 0;                 // numero de port affecté par le système
  if ( bind(sock, (const struct sockaddr *)&local, sizeof(local)) < 0)
```

```
                                                        goto faute2;

  /* Envoi message */
  memset(&distant, 0, sizeof(&distant)); //precaution
  distant.sin_family = AF_INET;
  distant.sin_addr.s_addr = adresse;      // adresse IP serveur
  distant.sin_port = htons(port);         // numéro de port serveur
  sendto(sock, (const char *)message, strlen(message), MSG_CONFIRM,
                 (const struct sockaddr *) &distant, sizeof(distant));

  /* Retour OK */
  close(sock);
  exit(0);

  /* USAGE */
  usage:
  printf ("Usage : client <IP> <Port> <message>\n>");
  exit (-1);

  /* CAS DE FAUTES */
  faute1:
  perror("faute socket : ");
  exit(-2);
  faute2:
  perror("faute bind : ");
  exit(-3);
}
```