TDM01 d'Informatique Répartie
Sockets

ASI4 - INSA Rouen
CORRECTION

Calculatrice distribuée sur entiers à l'aide de Sockets

Le but de ce TP est de réaliser une calculatrice distribuée à l'aide Sockets. Pour cela, vous respecterez les contraintes suivantes :

- Votre calculatrice supportera 2 types d'opération, les additions et les soustractions.
- Elle ne travaillera que sur les entiers.
- Le serveur recevra comme requête de la part du client une opération à réaliser sur 2 entiers (*i.e.* un caractère +' ou -' et un entier) et retournera la réponse (un entier).
- Le serveur est en C, le client est en Java.
- Vous travaillerez en mode non connecté.

NB : il s'agit d'Informatique répartie donc pensez à tester votre programme avec serveur et client sur des machines différentes !

Correction

serveur.c

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

static void erreur(const char *message) {
  fputs(strerror(errno), stderr);
  fputs(": ", stderr);
  fputs(message, stderr);
  fputc('\n', stderr);
  exit(1);
}

int main(int argc,char **argv) {
  char *IPServeur=NULL, calcul[512], message[512], ipClient[INET_ADDRSTRLEN], operation;
  int portServeur, socketServeur, longueur, nbOctets, e1, e2, resultat;
  struct sockaddr_in adresseServeur, adresseClient;

  // Récupération des paramètres
  if ( argc>=3 ) {
    IPServeur = argv[1];
    portServeur = atoi(argv[2]);
  }else if( argc>=2 ){
    IPServeur = "127.0.0.1";
  }else {
    IPServeur = "127.0.0.1";
    portServeur = 1024;
  }

  // Construction de l'adresse
  memset(&adresseServeur, 0, sizeof(adresseServeur));
  adresseServeur.sin_family = AF_INET;
  adresseServeur.sin_port = htons(portServeur);
  inet_pton(AF_INET, IPServeur, &(adresseServeur.sin_addr));
  if(adresseServeur.sin_addr.s_addr==INADDR_NONE)
    erreur("mauvaise addresse");

  // Création du Socket
  socketServeur = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP);
  if( socketServeur==-1 )
    erreur("socket");
```

```
  // Bind
  if( bind(socketServeur, (struct sockaddr *)&adresseServeur, sizeof(adresseServeur))==-1 )
      erreur("bind()");

  // Ecouter le Socket et traiter les requêtes
  operation = 'c';
  while( operation!='q' ) {
    longueur = sizeof(adresseClient);
    nbOctets = recvfrom(socketServeur, calcul, sizeof(calcul), 0, (struct sockaddr *)&adresseClient, &longueur);
    if(nbOctets==-1) {
      erreur("rcvfrom()");
    }else {
      calcul[nbOctets]=0;
      inet_ntop(AF_INET, &(adresseClient.sin_addr), ipClient, INET_ADDRSTRLEN);
      sscanf(calcul,"%d%c%d", &e1, &operation, &e2);
      printf("(%s:%u) : '%s'\n", ipClient, (unsigned)ntohs(adresseClient.sin_port), calcul);
      if( operation=='+' ){
        resultat = e1+e2;
      }else if( operation=='-' ){
        resultat = e1-e2;
      }else
        resultat = -1;
      sprintf(message, "%d", resultat);
      if( sendto(socketServeur, message, strlen(message), 0, (struct sockaddr *)&adresseClient, longueur)==-1 )
        erreur("sendto()");
    }
  }
  close(socketServeur);
  return 1;
}
```

Client.java

```
import java.io.*;
import java.util.*;
import java.net.*;

public class Client {

    public static void main(String[] args) throws Exception {
        InetAddress adresse = InetAddress.getByName(args[0]);
        int portUDP = Integer.parseInt(args[1]);
        byte[] requete = args[2].getBytes(), reponse = new byte[1000];
        DatagramPacket envoi = new DatagramPacket(requete, requete.length, adresse, portUDP),
            reception = new DatagramPacket(reponse, reponse.length);
        //     DatagramSocket socket = new DatagramSocket(portUDP, adresse);
        DatagramSocket socket = new DatagramSocket();

        socket.send(envoi);
        socket.receive(reception);
        String texte = new String(reponse, 0, reception.getLength());
        System.out.println("Resultat = " + texte);
    }
}
```

Travaux optionnels, à réaliser pour une bonne maîtrise des Sockets

- Réalisez un travail similaire avec un serveur Java et un client C.
- Réalisez un travail similaire en mode connecté, serveur et client dans les 2 langages.

Remarques

- Vous aurez alors accès à la correction du TDM durant une semaine à compter de la fin du TDM.
- **Déposez un compte-rendu de 2 pages `TDMSockets-NomPrenom.pdf` sur moodle chez TOUTES les personnes du binôme. Ce CR contiendra les informations que vous jugerez nécessaires.**
- Votre CR sera disponible pour vous lors de l'examen machine.