TP Réseau - Les Sockets en C et en Java

Exercice 0 : Le fameux Hello World

Pour commencer, nous allons écrire ensemble un programme client/serveur échangeant un simple "Hello". Il vous suffit de reprendre le code qui suit, de le tester et surtout de le comprendre.

Le programme Serveur

Il faut écrire un serveur qui permet et attend la connexion de clients puis affiche pour chaque client son nom (internet) puis lui attribut un numéro.

```
/**
 *  Fichier helloServeur.c
 */
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/signal.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define TAILLE 250

int desc;

void finir(int s){
    close (desc);
    exit(1);
}

int main(int argc, char * argv[]){
  int resultat, n;
  struct sockaddr_in serveur;
  struct sockaddr_in client;
  struct hostent * host;
  int long_client, numero_client = 0;

  // creation de la socket en mode UDP
  desc = socket(AF_INET, SOCK_DGRAM, 0);
  if (desc < 0){
    perror("Erreur de création de socket\n");
    exit(-1);
  }

  serveur.sin_family = AF_INET;
  serveur.sin_addr.s_addr = htonl(INADDR_ANY); // host to network long
  serveur.sin_port = htons(5000);

  resultat = bind(desc, (struct sockaddr *)&serveur, sizeof(serveur));
  if (resultat < 0){
    perror("Erreur 2");
    exit(-2);
  }

  // mise en place d'un handler pour fermer proprement la socket
  signal(SIGINT, finir);

  long_client = sizeof(serveur);
  printf ("Lancement du serveur sur le port 5000\n");
  while (1){
    char buffer [TAILLE]="";
    n = recvfrom(desc, buffer, TAILLE, 0, (struct sockaddr *)&client, &long_client);
    printf ("%s\n",buffer);

    if (n<0){
      perror ("Erreur de reception\n");
      exit(-3);
    }

    numero_client++;
    printf("Le serveur envoi le numéro de client %d\n", numero_client);
    sendto(desc, &numero_client, sizeof(int), 0, (struct sockaddr *)&client, long_client);
  }
}
```

Le programme Client

Le client envoi un message "Hello" au serveur puis attend que le serveur lui réponde pour lui donner le numéro qu'il lui a attribué.
Chacun des deux programmes (client/serveur) devra afficher les informations envoyées et les informations reçues.

```
/**
 *  Fichier helloClient.c
 */
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/signal.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

int main(int argc, char * argv[]){
    char * nom_serveur;
    int port, desc, resultat;
    struct sockaddr_in client, serveur;
    struct hostent * hp;
    int buffer, long_serveur;

    if (argc < 3){
perror("usage commande nomServeur port\n");
exit(-1);
    }

    nom_serveur = argv[1];
    port = atoi(argv[2]);

    printf("Port : %d\n", port);
    hp = gethostbyname(argv[1]);

    serveur.sin_family = AF_INET;
    serveur.sin_addr.s_addr = ((struct in_addr *)(hp->h_addr))->s_addr;
    serveur.sin_port = htons(port);

    desc = socket(AF_INET, SOCK_DGRAM, 0);
    if (desc < 0){
perror("Erreur de creation de socket\n");
exit(-1);
    }

    client.sin_family = AF_INET;
    client.sin_addr.s_addr = htonl(INADDR_ANY); // host to network long
    client.sin_port = htons(5001);

    resultat = bind(desc, (struct sockaddr *)&client, sizeof(struct sockaddr_in));
    if (resultat < 0){
perror("Erreur 2 ");
exit(-2);
    }

    printf("Envoi au serveur Bonjour\n");
    sendto(desc, "Bonjour c est le prof", sizeof(char)*21, 0, (struct sockaddr *)&serveur, sizeof(serveur));

    recvfrom(desc, &buffer, sizeof(int), 0, (struct sockaddr *)&serveur, &long_serveur);
    printf("Le numero attribué par le serveur %d\n",buffer);

    close(desc);
}
```

Construction d'un fichier Makefile

Un fichier `Makefile` est un fichier permettant de rassembler les instructions de compilation.

```
all : hello

clean :
rm -f helloServeur
rm -f helloClient

hello : helloClient helloServeur

helloServeur : helloServeur.c
gcc -o helloServeur helloServeur.c

helloClient : helloClient.c
gcc -o helloClient helloClient.c
```

Exercice 1 : Hello en JAVA

L'objectif de cet exercice est de réécrire le programme client/serveur précédent en JAVA. Pour cela, vous devez vous aider de `java.net.DatagramSocket` et plus généralement de la Javadoc. Vous pouvez procéder en deux temps :

1. Réécrivez simplement le programme client et testez-le avec le serveur développé en C.
2. Réécrivez le serveur et croisez vos tests (Client C avec Serveur JAVA et vice-versa).

Exercice 2 : Jeu de Morpions

L'objectif de cet exercice est de développer un jeu de Morpion qui se jouera sur deux postes différents en utilisant des Sockets en JAVA. Les étapes suivantes sont à réaliser :

1. Modéliser sur papier le jeu à développer (quelles sont les informations qui seront échangées entre le client et le serveur, quelles seront les tâches du serveur).
2. Dans une première phase de développement, le serveur ne fera que jouer au hasard et seul un utilisateur utilisant un client pourra jouer (prévoir le cas où plusieurs clients joueraient en même temps). Dans une seconde phase, le serveur permettra à un autre joueur de jouer.