M3101 – programmation réseau
TDM1 - L'API Socket

Soit les deux programmes : application cliente (page 2) et une application serveur (page 3).

1) Indiquez où se trouvent les parties correspondant à l'ouverture de la connexion, le transfert de données et la fermeture de la connexion.
2) Donnez le quintuplet <$@IP_{client}$, $Port_{client}$, $@IP_{serveur}$, $Port_{serveur}$, *Protocole Transport*> identifiant la connexion entre les deux applications.
3) A quoi sert la boucle *while* dans le code du client (lignes 34 à 48) ?
4) On parle du passage du protocole réseau IPv4 vers le protocole IPv6. Est-ce que cela pose un problème pour les codes client.c et serveur.c ? Si oui indiquez ce qu'il faut changer.
5) A quoi correspond le protocole applicatif dans cet exemple ?
6) Que faut-il rajouter pour que l'application cliente renvoie le message « OK » après avoir reçu le message de l'application serveur ?

Code Client

```
1
2   #include  <stdio.h>
3   #include  <unistd.h>
4   #include  <sys/socket.h>
5   #include  <netdb.h>
6   #include  <string.h>
7   #include  <stdlib.h>
8   #include  <stdbool.h>
9
10  #define MAXBUFFERLEN 1024
11  #define SERVERNAME = "localhost"
12  #define SERVERPORT = "12345"
13
14  int main(int argc, char* argv[]){
15        int descSock;
16        int ecode;
17        struct addrinfo *res,*resPtr;
18        struct addrinfo hints;
19        char buffer[MAXBUFFERLEN];
20        bool isConnected = false
21
22        memset(&hints, 0, sizeof(hints));
23        hints.ai_socktype = SOCK_STREAM;
24        hints.ai_family = AF_UNSPEC;
25
26        ecode = getaddrinfo(SERVERNAME,SERVERPORT,&hints,&res);
27        if (ecode){
28              fprintf(stderr,"getaddrinfo: %s\n", gai_strerror(ecode));
29              exit(1);
30        }
31
32        resPtr = res;
33
34        while(!isConnected && resPtr!=NULL){
35              descSock = socket(resPtr->ai_family, resPtr->ai_socktype,
36                                resPtr->ai_protocol);
37              if (descSock == -1) {
38                    perror("Erreur creation socket");
39                    exit(2);
40              }
41
42              ecode = connect(descSock, resPtr->ai_addr, resPtr->ai_addrlen);
43              if (ecode == -1) {
44                    resPtr = resPtr->ai_next;
45                    close(descSock);
46              }
47              else isConnected = true;
48        }
49        freeaddrinfo(res);
50        if (!isConnected){
51              perror("Connexion impossible");
52              exit(2);
53        }
54
55        ecode = read(descSock, buffer, MAXBUFFERLEN-1);
56        if (ecode == -1) {perror("Problème de lecture\n"); exit(3);}
57        buffer[ecode] = '\0';
58        printf("\"%s\".\n",buffer);
59        close(descSock);
60  }
```

Code Serveur

```
1
2   #include  <stdio.h>
3   #include  <stdlib.h>
4   #include  <sys/socket.h>
5   #include  <netdb.h>
6   #include  <string.h>
7
8   #define SERVADDR "localhost"
9   #define SERVPORT "12345"
10  #define LISTENLEN 1
11  #define MAXBUFFERLEN 1024
12
13  int main(){
14        int ecode;
15        char serverAddr[MAXHOSTLEN];
16        char serverPort[MAXPORTLEN];
17        int descSockRDV;
18        int descSockCOM;
19        struct addrinfo hints;
20        struct addrinfo *res;
21        struct sockaddr_storage myinfo;
22        struct sockaddr_storage from;
23        socklen_t len
24        char buffer[MAXBUFFERLEN];
25
26
27        memset(&hints, 0, sizeof(hints));
28        hints.ai_flags = AI_PASSIVE;
29        hints.ai_socktype = SOCK_STREAM;
30        hints.ai_family = AF_UNSPEC;
31
32        ecode = getaddrinfo(SERVADDR, SERVPORT, &hints, &res);
33        if (ecode) {
34              fprintf(stderr,"getaddrinfo: %s\n", gai_strerror(ecode));
35              exit(1);
36        }
37
38        descSockRDV = socket(res->ai_family, res->ai_socktype, res->ai_protocol);
39        if (descSockRDV == -1) {
40              perror("Erreur creation socket");
41              exit(4);
42        }
43
44        ecode = bind(descSockRDV, res->ai_addr, res->ai_addrlen);
45        if (ecode == -1) {
46              perror("Erreur liaison de la socket de RDV");
47              exit(3);
48        }
49        freeaddrinfo(res);
50
51        len=sizeof(struct sockaddr_storage);
52        ecode=getsockname(descSockRDV, (struct sockaddr *) &myinfo, &len);
53        if (ecode == -1)
54        {
55              perror("SERVEUR: getsockname");
56              exit(4);
57        }
58        ecode = getnameinfo((struct sockaddr*)&myinfo, sizeof(myinfo),
59                            serverAddr,MAXHOSTLEN,
60                            serverPort, MAXPORTLEN,
61                            NI_NUMERICHOST | NI_NUMERICSERV);
62        if (ecode != 0) {
63              fprintf(stderr, "Erreur dans getnameinfo: %s\n", gai_strerror(ecode));
64              exit(4);
65        }
66        printf("L'adresse d'ecoute est: %s\n", serverAddr);
67        printf("Le port d'ecoute est: %s\n", serverPort);
68
```

```
69          ecode = listen(descSockRDV, LISTENLEN);
70          if (ecode == -1) {
71                perror("Erreur initialisation buffer d'Ã©coute");
72                exit(5);
73          }
74
75          len = sizeof(struct sockaddr_storage);
76
77          descSockCOM = accept(descSockRDV, (struct sockaddr *) &from, &len);
78          if (descSockCOM == -1){
79                perror("Erreur accept\n");
80                exit(6);
81          }
82          strcpy(buffer, "BLABLABLA\n");
83          ecode = write(descSockCOM, buffer, strlen(buffer));
84          if (ecode == -1) {perror("Problème d'écriture\n"); exit(7);}
85          close(descSockCOM);
86          close(descSockRDV);
87    }
```