# Fichier serveur\_tcp.c pour ipv4

Liste include  
 <stdio.h><netinet/in.h><netdb.h> <sys/socket.h>,<unistd.h><string.h><stdlib.h>

int main(int argc, char \*argv[]) {

int s\_ecoute,scom, lg\_app,i,j;

struct sockaddr\_in adr;

struct sockaddr\_storage recep;

char buf[1500], renvoi[1500], host[1024],service[20];

s\_ecoute=socket(AF\_INET,SOCK\_STREAM,0);

printf("creation socket\n");

adr.sin\_family=AF\_INET;

adr.sin\_port=htons(atoi(argv[1]));

adr.sin\_addr.s\_addr=INADDR\_ANY;

if (bind(s\_ecoute,(struct sockaddr \*)&adr,sizeof(struct sockaddr\_in)) !=0) {

printf("probleme de bind sur v4\n");

exit(1); }

if (listen(s\_ecoute,5) != 0) {

printf("pb ecoute\n"); exit(1);}

printf("en attente de connexion\n");

while (1) {

scom=accept(s\_ecoute,(struct sockaddr \*)&recep, (socklen\_t \*)&lg\_app);

getnameinfo((struct sockaddr \*)&recep,sizeof (recep), host, sizeof(host),service, sizeof(service),0); //optionnelle pour info

printf("recu de %s venant du port %s\n",host, service); //optionnnelle

while (1) {

recv(scom,buf,sizeof(buf),0);

printf("buf recu %s\n",buf);

bzero(renvoi,sizeof(renvoi));

for(i=strlen(buf)-1,j=0;i>=0;i--,j++) renvoi[j]=buf[i];

renvoi[j+1]='\0';

send(scom,renvoi,strlen(renvoi),0);

bzero(buf,sizeof(buf));

if (strcmp(renvoi,"NIF") == 0) break; }

close(scom); }

close(s\_ecoute); }

**Fichier client\_tcp.c pour ipv4**

Liste include   
 <stdio.h><stdlib.h> <netinet/in.h> <netdb.h> <sys/socket.h> <string.h><unistd.h>

void main(int argc, char \*argv[])

{

char buffer[200],texte[200];

int port, rc, sock,i,c;

struct sockaddr\_in addr;

struct hostent \*entree;

if (argc !=3) {

printf("usage : client\_tcp nom\_machine numero\_port\n");

exit(1); }

addr.sin\_port=htons(atoi(argv[2]));

addr.sin\_family=AF\_INET;

entree=(struct hostent \*)gethostbyname(argv[1]);

bcopy((char \*)entree->h\_addr,(char \*)&addr.sin\_addr,entree->h\_length);

sock= socket(AF\_INET,SOCK\_STREAM,0);

if (connect(sock, (struct sockaddr \*)&addr,sizeof(struct sockaddr\_in)) < 0) {

printf("probleme connexion\n");

exit(1); }

printf("connexion passe\n");

while (1) {

bzero(texte,sizeof(texte));

bzero(buffer,sizeof(buffer));

i = 0;

printf("Entrez une ligne de texte : \n");

while((c=getchar()) != '\n')

texte[i++]=c;

send(sock,texte,strlen(texte)+1,0);

recv(sock,buffer,sizeof(buffer),0);

printf("recu %s\n",buffer);

if (strcmp("FIN",texte) == 0) break;

}

close(sock); }

# Fichier serveur\_tcp.c pour ipv6 (automatique)

#include <stdio.h>, <stdlib.h>, <netinet/in.h>,<netdb.h>,<sys/socket.h>,<strings.h>, <string.h>, <ifaddrs.h>

int main(int argc, char \*argv[])

{

int sockx,scom, lg\_app, ecode,i,j;

struct sockaddr\_in6 adr6, appelant, \*so6;

struct sockaddr\_storage recep;

char buf[1500], renvoi[1500], host[1024], service[20], serv[10];

char adresseipv6[INET6\_ADDRSTRLEN];

char adresseipv4[INET\_ADDRSTRLEN];

struct addrinfo \*res1,\*rres1, hints;

struct in6\_addr ip;

struct ifaddrs \*res = NULL;

printf("Une facon de faire automatique\n");

memset(&hints, 0, sizeof(hints));

hints.ai\_flags = AI\_PASSIVE;

hints.ai\_socktype= SOCK\_STREAM;

// hints.ai\_family = AF\_INET6; pour avoir que ipV6

hints.ai\_family = PF\_UNSPEC; // pour avoir tout ip

sprintf(serv,"%d",atoi(argv[1]));

ecode = getaddrinfo(NULL, serv, &hints, &rres1);

if(ecode) {

printf("problem %s\n",gai\_strerror(ecode));

exit(1); }

for(res1 = rres1; res1; res1 = res1->ai\_next) {

// en linux, un seul bind possible

if (res1->ai\_family == AF\_INET6) {

sockx = socket(res1->ai\_family,res1->ai\_socktype,res1->ai\_protocol);

if (bind(sockx, res1->ai\_addr, res1->ai\_addrlen) < 0) {

printf("Problerme de bind\n");

exit(1);

}

listen(sockx,5);

}

}

printf("fin de l'initialisation automatique\n");

while (1)

{

scom=accept(sockx,(struct sockaddr \*)&recep, (unsigned long \*)&lg\_app);

getnameinfo((struct sockaddr \*)&recep,sizeof (recep), host, sizeof(host),service, sizeof(service),0);

printf("recu de %s\n",host);

while (1) {

recv(scom,buf,sizeof(buf),0);

printf("buf recu %s %d\n",buf,strlen(buf));

bzero(renvoi,sizeof(renvoi));

for(i=strlen(buf)-1,j=0;i>=0;i--,j++) renvoi[j]=buf[i];

renvoi[j+1]='\0';

send(scom,renvoi,strlen(renvoi),0);

bzero(buf,sizeof(buf));

if (strcmp(renvoi,"NIF") == 0) break; }

close(scom); }

close(sockx);

}

**Fichier serveur\_tcp.c pour ipv6 (manuel)**

int main(int argc, char \*argv[])

{

int s\_ecoute, scom, lg\_app,i,j;

// meme variable…

/\* recuperation et affichage des adresses IP \*/

if (getifaddrs(&res))

printf("pb sur getifaddrs\n");

printf("Recuperation des adresse IP\n");

for(; res !=NULL;res = res ->ifa\_next)

{

if ( (res->ifa\_addr)->sa\_family ==AF\_INET6) {

so6 = (struct sockaddr\_in6 \*)res->ifa\_addr;

bcopy(so6->sin6\_addr.s6\_addr, &ip,16);

inet\_ntop(AF\_INET6,&ip,adresseipv6, sizeof(adresseipv6));

printf("adresse ipv6 %s\n",adresseipv6); }

if ( (res->ifa\_addr)->sa\_family ==AF\_INET) {

so4 = (struct sockaddr\_in \*)res->ifa\_addr;

inet\_ntop(AF\_INET,&(so4->sin\_addr),adresseipv4,sizeof(adresseipv4));

printf("adresse ipv4 %s\n",adresseipv4); }

}

printf("debut initialisation manuelle\n");

s\_ecoute=socket(AF\_INET6,SOCK\_STREAM,0);

adr.sin6\_family=AF\_INET6;

adr.sin6\_port=htons(atoi(argv[1]));

memcpy((void \*)&adr.sin6\_addr, (void \*)&in6addr\_any, sizeof in6addr\_any);

if (bind(s\_ecoute,(struct sockaddr \*)&adr,sizeof(struct sockaddr\_in6)) !=0){

printf("probleme de bind sur v6\n");

exit(1); }

if (listen(s\_ecoute,5) != 0) {

printf("pb ecoute\n"); exit(1);}

printf("en attente de connexion\n");

…..

**Fichier client\_tcp.c pour ipv6**

#include <stdio.h>,<stdlib.h>, <netinet/in.h>,<netdb.h>,<sys/socket.h>,<strings.h>, <string.h>, <ifaddrs.h>

void main(int argc, char \*argv[])

{

char buffer[200], texte[200];

int port, rc, sock,c,i;

struct in6\_addr serveraddr;

struct addrinfo hints, \*res=NULL;

if (argc !=3) {

printf("usage : clientv6 nom\_serveur port\n");

exit(1); }

memset(&hints, 0x00,sizeof(hints));

hints.ai\_flags = AI\_NUMERICSERV;

hints.ai\_family = AF\_INET6;

hints.ai\_socktype = SOCK\_STREAM;

port=atoi(argv[2]);

//on teste sil'adresse IP est numeric ou textuel

// si rc = 1 , c'est numeric donc resolution numeric

rc = inet\_pton(AF\_INET6, argv[1], &serveraddr);

if (rc ==1)

hints.ai\_flags |= AI\_NUMERICHOST;

// recuperation des infos sur le serveur distant

rc = getaddrinfo(argv[1], argv[2], &hints, &res);

if (rc != 0)

{ printf("pb %s\n",gai\_strerror(rc));exit(1);}

sock= socket(res->ai\_family, res->ai\_socktype, res->ai\_protocol);

if (connect(sock, res->ai\_addr, res->ai\_addrlen) <0) {

printf("probleme connexion\n");

exit(1); }

while (1)

{

bzero(texte,sizeof(texte));

bzero(buffer,sizeof(buffer));

i = 0;

printf("Entrez une ligne de texte : \n");

while((c=getchar()) != '\n')

texte[i++]=c;

printf("texte %s\n",texte);

send(sock,texte,strlen(texte)+1,0);

recv(sock,buffer,sizeof(buffer),0);

printf("recu %s\n",buffer);

if (strcmp("FIN",texte) == 0) break;

}

close(sock); }

**Fichier serveur\_udp.c pour ipv4**

#include <stdio.h>

#include <netinet/in.h>

#include <netdb.h>

#include <sys/socket.h>

int main(int argc, char \*argv[]) {

int s\_ecoute, recus;

char mes[20], host[200],service[10];

struct sockaddr\_in adr;

struct sockaddr\_storage recep;

int lg\_app;

s\_ecoute=socket(AF\_INET, SOCK\_DGRAM,0);

printf("la socket est cree\n");

adr.sin\_family=AF\_INET;

adr.sin\_port=htons(atoi(argv[1]));

adr.sin\_addr.s\_addr=INADDR\_ANY;

if (bind(s\_ecoute,(struct sockaddr \*) &adr, sizeof (struct sockaddr\_in)) !=0)

{

printf("Pb de connexion\n");

exit(1); }

printf("je suis en ecoute\n");

lg\_app=sizeof(struct sockaddr\_storage);

recus=recvfrom(s\_ecoute,mes,sizeof(mes),0,(struct sockaddr \*) &recep,&lg\_app);

if (recus <=0)

printf("bug\n");

else {

getnameinfo((struct sockaddr \*)&recep,sizeof (recep), host, sizeof(host),service, sizeof(service),0);

printf("message : %s venant de %s et du port %s\n",mes,host,service);

sendto(s\_ecoute,"OK",3,0,(struct sockaddr \*)&recep, lg\_app); }

close(s\_ecoute);

}

**Fichier client\_udp.c pour ipv4**

#include <stdio.h>

#include <netinet/in.h>

#include <netdb.h>

#include <sys/socket.h>

#include <string.h>

int main(int argc, char \*argv[])

{

int s\_com,emis;

char mes[20];

struct sockaddr\_in adr;

struct sockaddr\_storage recep;

struct hostent \*entree;

int lg\_app;

if (argc !=3)

{printf("usage : prog nom\_serveur port\n"); exit(1);}

s\_com=socket(AF\_INET, SOCK\_DGRAM,0);

printf("la socket est cree\n");

adr.sin\_family=AF\_INET;

adr.sin\_port=htons(atoi(argv[2]));

entree= (struct hostent \*)gethostbyname(argv[1]);

bcopy((char \*) entree->h\_addr, (char \*)&adr.sin\_addr, entree->h\_length);

printf("entrez un mot\n");

scanf("%s",mes);

emis=sendto(s\_com,mes,sizeof(mes),0,(struct sockaddr \*)&adr,sizeof(adr));

if (emis <=0)

printf("gros probleme\n");

recvfrom(s\_com,mes,sizeof(mes),0,(struct sockaddr \*)&recep,&lg\_app);

printf("message recu : %s\n",mes);

close(s\_com);

}

**Fichier serveur\_udp.c pour ipv6**

#include <stdio.h>

#include <netinet/in.h>

#include <netdb.h>

#include <sys/socket.h>

#include <string.h>

#include <stdlib.h>

int main(int argc, char \*argv[]) {

int s\_ecoute, recus;

char mes[20], host[200],service[10];

struct sockaddr\_in6 adr;

struct sockaddr\_storage recep;

int lg\_app;

s\_ecoute=socket(AF\_INET6, SOCK\_DGRAM,0);

printf("la socket est cree\n");

adr.sin6\_family=AF\_INET6;

adr.sin6\_port=htons(atoi(argv[1]));

memcpy((void \*)&adr.sin6\_addr, (void \*)&in6addr\_any, sizeof in6addr\_any);

if (bind(s\_ecoute,(struct sockaddr \*) &adr, sizeof (struct sockaddr\_in6)) !=0)

{

printf("Pb de connexion\n");

exit(1); }

printf("je suis en ecoute\n");

lg\_app=sizeof(struct sockaddr\_storage);

recus=recvfrom(s\_ecoute,mes,sizeof(mes),0,(struct sockaddr \*) &recep,&lg\_app);

if (recus <=0)

printf("bug\n");

else {

getnameinfo((struct sockaddr \*)&recep,sizeof (recep), host, sizeof(host),service, sizeof(service),0);

printf("message : %s venant de %s et du port %s\n",mes,host,service);

sendto(s\_ecoute,"OK",3,0,(struct sockaddr \*)&recep, lg\_app); }

close(s\_ecoute);

}

**Fichier client\_udp.c pour ipv6**

#include <stdio.h>, <netinet/in.h>,<netdb.h>, <sys/socket.h>, <string.h>, <stdlib.h>

int main(int argc, char \*argv[])

{

int s\_com,emis, port, rc, lg\_app;

char mes[20];

struct sockaddr\_in6 adr;

struct sockaddr\_storage recep;

struct in6\_addr serveraddr;

struct addrinfo hints, \*res=NULL;

if (argc !=3)

{printf("usage : udpcliv4 nom\_serveur port\n"); exit(1);}

memset(&hints, 0x00,sizeof(hints));

hints.ai\_flags = AI\_NUMERICSERV;

hints.ai\_family = AF\_INET6;

hints.ai\_socktype = SOCK\_DGRAM;

port=atoi(argv[2]);

//on teste sil'adresse IP est numeric ou textuel

rc = inet\_pton(AF\_INET6, argv[1], &serveraddr);

if (rc ==1) hints.ai\_flags |= AI\_NUMERICHOST;

rc = getaddrinfo(argv[1], argv[2], &hints, &res);

if (rc != 0)

{ printf("pb %s\n",gai\_strerror(rc));exit(1);}

s\_com= socket(res->ai\_family, res->ai\_socktype, res->ai\_protocol);

printf("la socket est cree\n");

printf("entrez un mot\n");

scanf("%s",mes);

emis=sendto(s\_com,mes,sizeof(mes),0,(struct sockaddr \*)res->ai\_addr,res->ai\_addrlen);

if (emis <=0) printf("gros probleme\n");

recvfrom(s\_com,mes,sizeof(mes),0,(struct sockaddr \*)&recep,&lg\_app);

printf("message recu : %s\n",mes);

close(s\_com);}

**Fichier client\_tcp.c pour ipv4 avec csocket (windows)**

#include "stdafx.h", <string.h>, <iostream>, <afxsock.h>

using namespace std;

/\* Attention : Change project settings:

- Open the Project Settings window ("Alt+F7")

- Select the General tab

- Choose use MFC in a shared DDL in the Microsoft Foundation Classes box \*/

void DieError(char\* errorMessage) {

cout << errorMessage << endl; exit(0);}

int \_tmain(int argc, \_TCHAR\* argv[]) {

if (!AfxWinInit(::GetModuleHandle(NULL), NULL, ::GetCommandLine(), 0)) {

DieError("Fatal Error: MFC initialization failed");}

cout << "debut du programmme\n";

int port = 16515;

CSocket servSocket;

AfxSocketInit(NULL);

if (!servSocket.Create(port)) { DieError("servSock.Create() failed");}

if (!servSocket.Listen(5)) {DieError("servSock.Listen() failed");}

for(;;) {

sockaddr\_in Addr; // Client address

int clntLen = sizeof(Addr); //taille du client

CSocket sock\_com;

if (!servSocket.Accept(sock\_com)) {DieError("servSock.Accept() failed");}

if (!sock\_com.GetPeerName((sockaddr \*)&Addr, &clntLen)) {

DieError("clntSock.GetPeerName() failed");}

cout << "Handling client " << inet\_ntoa(Addr.sin\_addr) << endl;

while (1)

{ int recvMsgSize, i, j;

char Buffer[100], renvoi[100];

recvMsgSize = sock\_com.Receive(Buffer, 100, 0);

if (recvMsgSize < 0) { DieError("clntSock.Receive() failed");}

cout << "message recu : " << Buffer << "\n";

for(i=strlen(Buffer)-1,j=0;i>=0;i--,j++) renvoi[j]=Buffer[i];

renvoi[j+1]='\0';

sock\_com.Send(renvoi, strlen(renvoi), 0);

if (strcmp(renvoi,"NIF")==0) break; }

sock\_com.Close();

}

servSocket.Close();

getchar();

return 0;

}

**Fichier client\_tcp.c pour ipv4 en socket (windows)**

#include "pch.h",<stdio.h>,<iostream>,<ws2tcpip.h>,"winsock2.h"

#pragma comment(lib, "ws2\_32.lib")

using namespace std;

int main(int argc, CHAR\* argv[])

{

int s\_com, i, c;

char phrase[50], buf[50];

WSADATA info;

struct addrinfo \*result = NULL,\*ptr = NULL, hints;

DWORD dwRetval;

if (WSAStartup(MAKEWORD(2, 0), &info) == SOCKET\_ERROR)

{ cout << "erreur dans l'initialisation " << endl;

WSACleanup();exit(1);}

//--------------------------------

// Setup the hints address info structure which is passed to the getaddrinfo() function

ZeroMemory(&hints, sizeof(hints));

hints.ai\_family = AF\_INET;

hints.ai\_socktype = SOCK\_STREAM;

hints.ai\_protocol = 0;

dwRetval = getaddrinfo("nom\_machine", "numero\_port", &hints, &result);

if (dwRetval != 0) {

printf("getaddrinfo failed with error: %d\n", dwRetval);

WSACleanup();

return 1; }

printf("getaddrinfo returned success\n");

s\_com = socket(AF\_INET, SOCK\_STREAM, 0);

cout << " la socket est cree\n";

if (connect(s\_com, result->ai\_addr, result->ai\_addrlen) < 0)

{

cout << "probleme de connexion";

getchar();exit(1);}

freeaddrinfo(result);

while (true)

{

printf("Entrez une ligne de texte\n");

i = 0;

while ((c = getchar()) != '\n')

phrase[i++] = c;

phrase[i] = '\0';

printf("ligne trouve %s\n", phrase);

send(s\_com, phrase, i + 1, 0);

for (i = 0; i < sizeof(buf);i++)

buf[i] = '\0';

recv(s\_com, buf, sizeof(buf), 0);

printf("message recu %s\n", buf);

if (strncmp(phrase, "FIN", 3) == 0)

break;

}

WSACleanup();

return 0;

}