Notre MSN !

Le but de ce TP est de faire un système de dialogue par le réseau. On commence petit (uniquement du texte, et chacun parle l'un après l'autre), puis on ajoutera des fonctionnalités petit à petit.

Commençons par créer un projet comme d'habitude. On créé donc un projet console vide.

Ensuite, on va créer notre classe qui permet d'envoyer et de recevoir des massages sur le réseau :

Connexion
monSocket : SOCKET
+ Connexion() + ~Connexion() + servir() + contacter(string ad) + envoyer(string msg) + recevoir() :string

Une fois le .h de créé, il va falloir faire le .cpp !

Tout d'abord, il faut rajouter une nouvelle librairie, et dire qu'on utilise une dll (à mettre dans le .h) :

```
#include <winsock2.h>
#pragma comment(lib, "ws2_32.lib")
```

Le constructeur

Le constructeur doit initialiser la classe. C'est lui qui va demander à Windows s'il peut utiliser les sockets, et ça se fait comme ça :

```
WSADATA WSAData;
int error = WSAStartup (0x0202, &WSAData);
if (error)
{
	cout<<"erreur de demarrage de WSAStartup..."<<endl ;
	exit(1);
};
if (WSAData.wVersion != 0x0202)
{
	cout<<"erreur de version de WSAStartup..."<<endl ;
	exit(1);
};
```

Maintenant que c'est fait, on peut tenter de créer un socket :

```
monSocket =socket(AF_INET,SOCK_STREAM,0);
if(monSocket ==INVALID_SOCKET)
{
	cout<<"erreur d'ouverture de socket...";
	exit(2);
};
```

On a fini le constructeur…

Le destructeur

Il faut aussi faire un destructeur pour dire à Windows qu'on a fini d'utiliser les sockets. Ca se fait comme ça :

```
	closesocket(monSocket);
	WSACleanup();
```

Servir

La fonction servir sera une fonction qui permet d'attendre que quelqu'un nous parle. Ça se fait comme ça :

```
SOCKADDR_IN soc_in;
soc_in.sin_family = AF_INET;
soc_in.sin_addr.s_addr = htonl(INADDR_ANY);
soc_in.sin_port = htons(21500);
int erreur=bind(this-> monSocket,(LPSOCKADDR)&soc_in, sizeof(soc_in));
if (erreur==INVALID_SOCKET)
{
	cout<<"bind : ne peut pas initialiser le serveur ;"<<endl;
	cout<<"erreur :"<<to_string(WSAGetLastError());
	exit(3);
};
```

Quelques explications s'imposent : `AF_INET` permet de préciser qu'on va utiliser des adresses internet (on aurait pu utiliser des adresses UNIX…). `INADDR_ANY` permet de dire qu'on veut utiliser n'importe quelle adresse (en effet, puisqu'on attend une connexion, on ne sait pas encore l'adresse du client). Enfin, on a décidé d'utiliser le port 21500, mais vous pouvez choisir celui que vous voulez…

Ensuite, il faut mettre l'ordinateur en écoute, puis en attente d'une connexion :

```
listen(monSocket,1);
int addLenght=sizeof(*soc_in);
monSocket =accept(monSocket,soc_in,&addLenght);
```

Contacter

La fonction contacter permettra de se connecter à un autre ordinateur qui est en attente. Un peut comme pour la fonction `servir`, il faudra d'abord créer une variable de type `SOCKADDR_IN` :

```
SOCKADDR_IN soc_in;
soc_in.sin_family = AF_INET;
soc_in.sin_addr.s_addr = inet_addr(ad.c_str());
soc_in.sin_port = htons(21500);
if (connect(monSocket,(LPSOCKADDR)& soc_in, sizeof(soc_in))==SOCKET_ERROR)
{
	cout<<"Ne peux pas se connecter au serveur..."<<endl;
	exit(2);
};
```

Envoyer

Pour envoyer un message, nous allons faire la fonction suivante :

```
if ((send(monSocket, msg.c_str(),msg.length(),0))<0)
{
	cout<<"erreur lors de l'envoi d'un message..."<<endl ;
	WSACleanup();
	exit(2);
}
```

Recevoir

Pour recevoir un message, nous allons utiliser la fonction suivante :

```
char *tmp=new char[512];
int nbOctet;
if ((nbOctet=recv(soc,tmp, 512,0))<0){
	MessageBox(0,
		"erreur de reception d'un message...",
		"ERREUR Client",MB_OK | MB_ICONERROR);
	WSACleanup();
	exit(2);
};
string sortie(tmp);
delete tmp;
return sortie;
```

Il est nécessaire de passer par un tableau de `char` car la fonction `recv` ne travail qu'avec des tableaux de `char`…

Exercice 1

Faites un programme pour tester ces fonctions. Mettez vous à deux : l'un fait un programme qui attend qu'on se connecte, et l'autre fait un programme qui se connecte chez lui. Ensuite, le premier envoie un message (« bonjour »), et l'autre lui répond « salut ».

Exercice 2

Améliorez la phase de connexion pour que celui qui attend une connexion demande un mot de passe, et qui envoie « bonjour » uniquement si le mot de passe est bon (envoyer « erreur » sinon).

Exercice 3

Permettez aux utilisateurs de taper un texte au clavier (à tour de rôle), et que ça s'affiche chez le voisin… Le programme se termine dès que l'un tape « bye ».