C++

Ch 8 – Programmation réseau et Sockets

I. Sockets

A. Qu'est-ce qu'un « socket »

Un **socket** est un descripteur d'entrées/sorties permettant la communication entre programmes à travers les couches réseau. (On parle de communication inter-processus, ou IPC, InterProcess Communication).

On trouve 2 formes principales de sockets :

- En mode connecté (TCP) : garantie de la bonne réception du message envoyé
- En mode non connecté (datagrammes UDP) : envoi d'informations sans garantie de bonne réception par le destinataire

L'échange des données doit parfois tenir compte des particularités en terme d'endianisme (ordre des octets pour les nombres dont la taille excède 1 octet).

B. Mode de fonctionnement

Le mode de fonctionnement des sockets est similaire à celui du téléphone.

1. Installation du téléphone et attribution du numéro

- Créer d'un socket : **int** *socket*(**famille**,**type**,**protocole**):
  - Type d'adresse : AF_INET (utilisation des adresses Internet sur 4 octets)
  - Port
  - Type de socket : paquet (connecté) ou datagrammes (non connecté)
- Attribuer au socket une adresse à écouter : **bind**(int **descripteur**,**sockaddr localaddr**,int **addrlen**)
- Dire si on accepte une liste d'appels entrants (cas d'un serveur) : **int** *listen*(int **socket**,int **backlog**)

2. Décrocher (suite à un appel)

- accepter un appel entrant : int **accept**(int **socket**,struct sockaddr *** addr**,int *** addrlen**)

3. Composer un numéro de téléphone

- Demander la connexion : int **connect**(int **socket**,struct sockaddr *** addr**,int *** addrlen**)

4. Dialoguer

- écouter : int **recv**(int socket,char *** buffer**,int **len**,int **flags**)
- parler : int **send**(int **socket**,char *** buffer**,int **len**,int **flags**)

5. Raccrocher

- Terminer la communication : int **close**(int **socket**)

II. Programmation des sockets

A. Inclusion des prototypes

1. #include

Le fichier de prototypes pour l'utilisation des fonctions liées aux sockets :

```
#include <winsock2.h>
```

B. Codage des échanges

1. Initialisation de l'utilisation des sockets

```
WSADATA WSAData;
WSAStartup(MAKEWORD(2,0), &WSAData);
```

2. Déclaration des variables de connexion

```
// déclarations
SOCKET sock;
SOCKADDR_IN sin;
```

```
// initialisation des paramètres d'adresse/port
sin.sin_addr.s_addr = inet_addr("127.0.0.1");
sin.sin_family      = AF_INET;
sin.sin_port        = htons(4148);

// initialisation du socket
sock = socket(AF_INET,SOCK_STREAM,0);

// liaison du socket avec les paramètres d'adresses
bind(sock, (SOCKADDR *)&sin, sizeof(sin));
```

3. Coder l'écoute/réponse d'un serveur

Etablir la boucle d'écoute :

```
listen(sock, 4);
int val = 0;
while(1) // boucle infinie...
{
    val = accept(sock, (SOCKADDR *)&csin, sizeof(csin))
    if(val != INVALID_SOCKET)
    {
      // Fonctions à éxécuter sur le socket.
    }
}
```

Envoi d'information en reponse :

```
send(csock, "Hello world!\r\n", 14, 0);
```

4. Coder un client

Etablir la connexion :

```
connect(sock, (SOCKADDR *)&sin, sizeof(sin))
```

Recevoir :

```
char *buffer = new char[255];
recv(sock, buffer, sizeof(buffer), 0);
```

Fermer la connexion :

```
closesocket(sock);
```

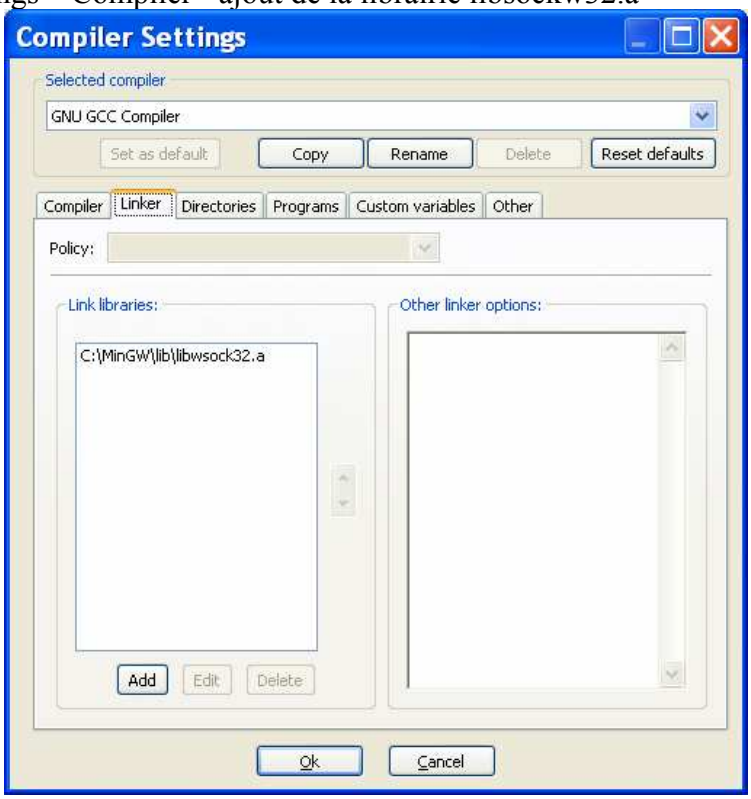
Nettoyer la connexion

```
WSACleanup();
```

III.Configurer le compilateur

La compilation (l'édition de lien) de programmes utilisant les sockets nécessitent la configuration des compilateurs et l'ajout de librairies. Par exemple pour **code::blocks** :

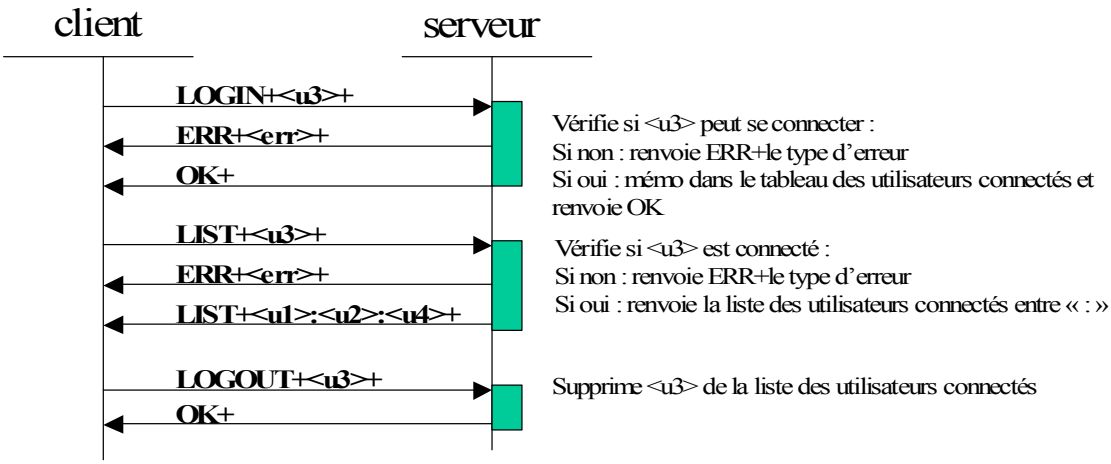
- Menu – Settings – Compiler - ajout de la librairie libsockw32.a



IV. Définir un protocole d'échange

Une fois ces aspects techniques assimilés, l'essentiel reste à faire : on va faire communiquer 2 programmes, mais quelle langue va-t-on parler ? quels mots va-t-on utiliser ? quelles questions pourra t –on se poser et quelles types de réponses allons nous autoriser ? sous quelle forme ?

Un protocole définit les différents échanges (succession et syntaxe) qui auront lieu entre un demandeur (client) et un fournisseur (serveur).
Par exemple :



Les valeurs entre chevrons (< et >) représentent des données variables : codes utilisateurs à définir, codes d'erreurs à utiliser, etc.