Nom :..... Prénom :.....

TP SIN
Rédaction d'un serveur tcp/Ip
C++ builder X10.1 (Indy 10)

Pré requis (l'élève doit savoir):

- Savoir utiliser un ordinateur
- Programme de base C++ builder

Programme

Objectif terminale :

L'élève doit être capable de réaliser un serveur tcp/IP

Matériel

- Ordinateur
- Logiciel C++ builder

1. Introduction

- L'utilité des ports

De nombreux programmes TCP/IP peuvent être exécutés simultanément sur Internet (vous pouvez par exemple ouvrir plusieurs navigateurs simultanément ou bien naviguer sur des pages HTML tout en téléchargeant un fichier par FTP). Chacun de ces programmes travaille avec un protocole, toutefois l'ordinateur doit pouvoir distinguer les différentes sources de données.

Ainsi, pour faciliter ce processus, chacune de ces applications se voit attribuer une adresse unique sur la machine, codée sur 16 bits: **un port** (la combinaison *adresse IP + port* est alors une adresse unique au monde, elle est appelée socket).

L'adresse IP sert donc à identifier de façon unique un ordinateur sur le réseau tandis que le numéro de port indique l'application à laquelle les données sont destinées. De cette manière, lorsque l'ordinateur reçoit des informations destinées à un port, les données sont envoyées vers l'application correspondante. S'il s'agit d'une requête à destination de l'application, l'application est appelée application **serveur**. S'il s'agit d'une réponse, on parle alors d'application **cliente**.

Il existe des milliers de ports (ceux-ci sont codés sur 16 bits, il y a donc 65536 possibilités), c'est pourquoi une assignation standard a été mise au point par l'**IANA** (*Internet Assigned Numbers Authority*), afin d'aider à la configuration des réseaux.

En fait, les numéros de ports sont divisés en 3 groupes (source Microsoft Q174904) :

- Les ports parfaitement définis sont dans la plage 0 à 1023.
- Les ports déposés sont ceux allant de 1024 à 49151.
- Les ports Dynamiques et/ou Privés vont de 49152 à 65535.

Les ports parfaitement définis

Ils sont assignés par l'IANA et sur beaucoup de systèmes ne peuvent être utilisés que par les process système (root) ou par des programmes exécutés par des utilisateurs privilégiés. Les ports sont utilisés par TCP [RFC793] pour nommer la terminaison d'une connexion logique véhiculant des conversations longues. Afin de donner l'accessibilité à certains services pour un utilisateur inconnu, un port de contact pour le service est défini.

Nom :..... Prénom :.....

Les ports déposés

Ils sont listés par l'IANA et sur la plupart des systèmes. Ils peuvent être utilisés par les process ordinaires d'utilisateurs ou par les programmes utilisés par les utilisateurs ordinaires. Les ports sont utilisés de la même manière que les ports parfaitement connus.

Les ports dynamiques/privés

Ils peuvent être utilisés par des applications très spécifiques.

Voici certains des ports reconnus les plus couramment utilisés:

Port	Type	Affectation
11	UDP	Systat (infos sur l'état du système)
13	TCP/UDP	DayTime
15	TCP	NetStat
20	TCP	FTP données
21	TCP	FTP
22	TCP	SSH : connexion sécurisée
23	TCP	Telnet
25	TCP	SMTP
43	TCP	WHOIS
50	TCP/IP	ESP
53	UDP	DNS
67	UDP	BootP
69	UDP	TFTP
70	TCP	GOPHER
79	TCP	Finger (infos sur les utilisateurs du serveur)
80	TCP	HTTP
88	TCP	Kerberos
109	TCP	POP / POP2
110	TCP	POP3
113	TCP	Service d'authentification
137	UDP	NetBios
139	TCP	NetBios
144	TCP	NEWS
161	UDP	SNMP
500	UDP	L2TP
546	UDP	DHCP
750	TCP/UDP	KERBEROS
1701	UDP	L2TP
NAT-T	UDP	4500

Nom :..... Prénom :.....

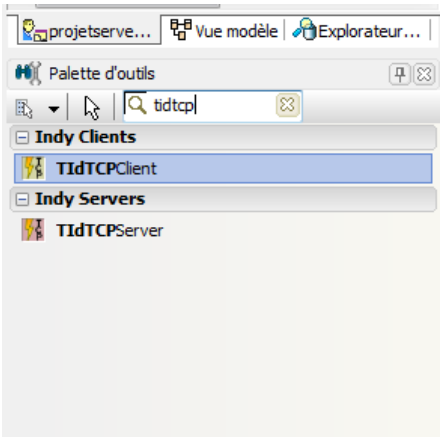
- Fonctionnement

Dans le principe, 2 ordinateurs désirant dialoguer doivent préciser chacun (avec TCP/IP), une adresse codée sur 32 bits et un numéro de port séparé par ":" (ex. 194.51.20.12:40). Cette paire d'adresses forme un canal de communication (socket).
Ainsi, un serveur (un ordinateur que l'on contacte et qui propose des services tels que FTP, Telnet, ...) possède des numéros de port fixes auxquels l'administrateur réseau a associé des services. Ainsi, les ports d'un serveur sont généralement compris entre 0 et 1023 (fourchette de valeurs associées à des services connus).
Du côté du client, le port est choisi aléatoirement parmi ceux disponibles par le système d'exploitation. Ainsi, les ports du client ne seront jamais compris entre 0 et 1023 car cet intervalle de valeurs représente les *ports connus*.

2. **Travail demandé**

Pour communiquer, on a besoin de deux sources : un programme serveur et un programme client.
Tout d'abord, le programme serveur ouvre la connexion au socket par le numéro d'identification de la connexion avec le socket serveur appelé également "Port" et se met en écoute des requêtes du programme client. Le programme client se connecte à l'ordinateur du programme serveur par l'adresse IP. Puis, ouvre la connexion au socket client par le port. Les deux sources sont maintenant relier, il ne reste plus qu'à échanger les données du programme client au programme serveur.

Pour utiliser les sockets en Delphi, vous devez utiliser deux composants qui se trouvent dans les onglets « Indy clients » et « Indy servers » dans la palette de composants :



TIdTCPServer = **class**(TIdCustomTCPServer)

Ce composant permet de spécifier le port que vous utilisez et écoute les requêtes du socket client.

TIdTCPClient = **class**(TIdTCPClientCustom)

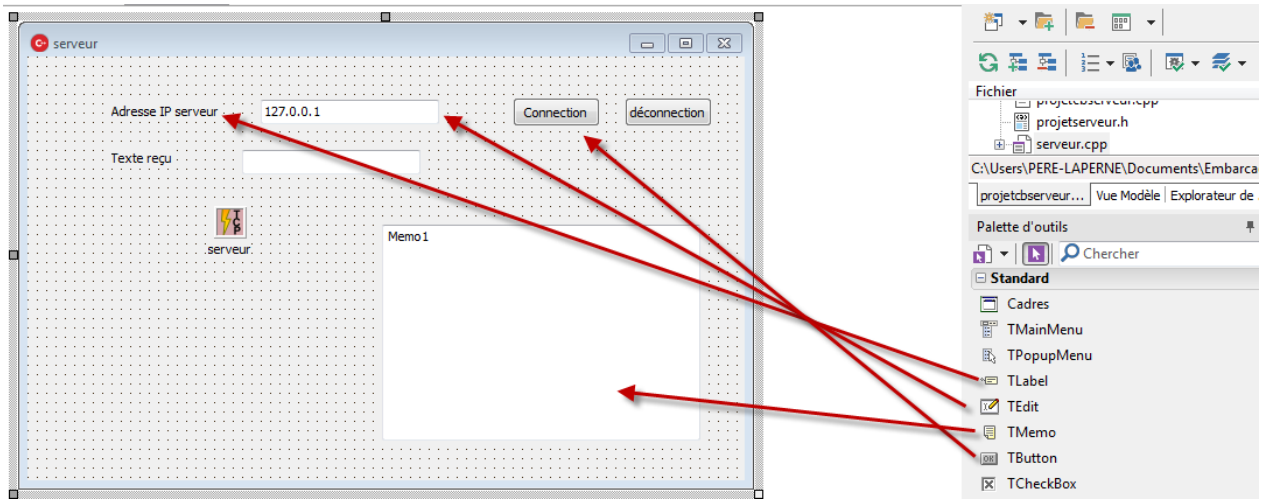
Ce composant permet de spécifier le socket serveur auquel vous voulez vous connecté, le port que vous utilisez et envoit le texte au serveur.

Nom :..... Prénom :.....

- Réaliser le programme suivant

Création du programme Serveur

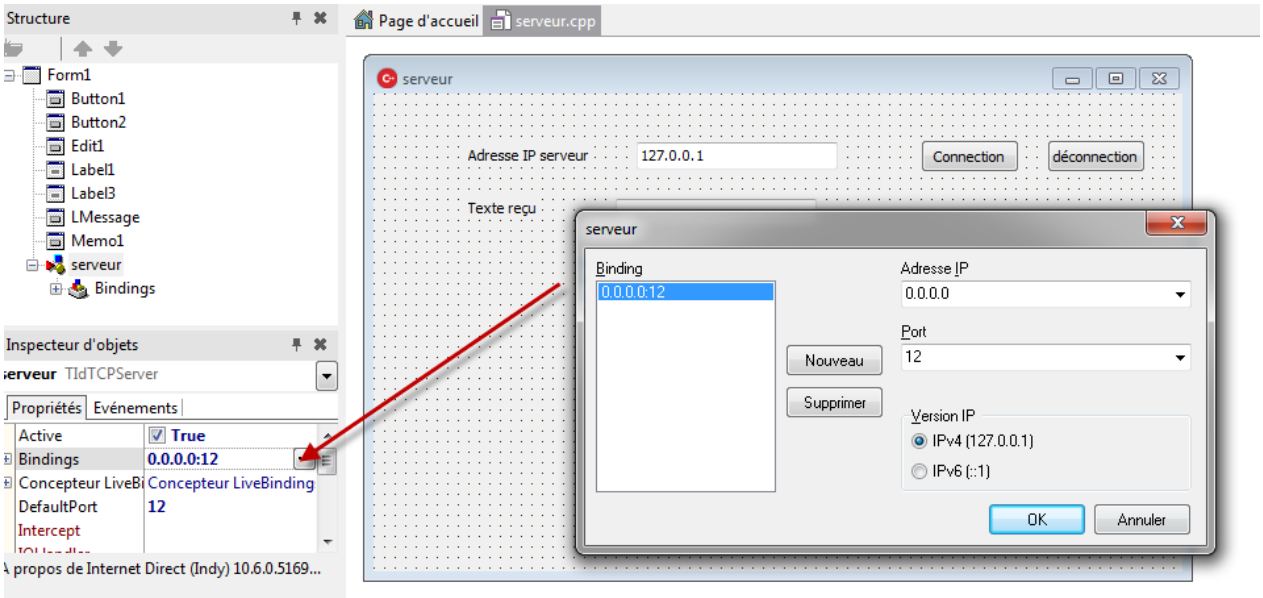
Voici à ce que ressemblera le programme Serveur.



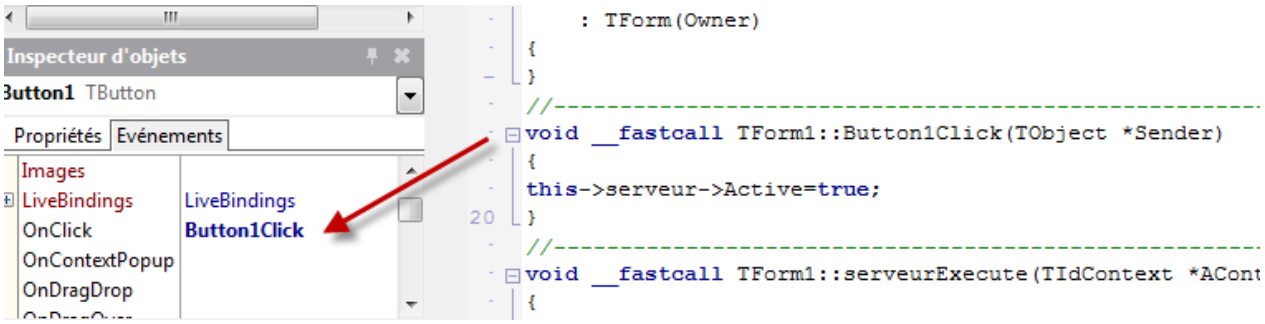
Ajouter deux TButton, deux TEdit, deux TLabel, un TMemo ,un TIdTCPServer .
"Edit1" servira à inscrire l'adresse IP
"Edit2" permet de lire le message.
"Button1" est le bouton de connexion.
"Button2" est le bouton de déconnexion.

Se connecter

Pour se connecter, le serveur doit définir l'adresse IP de l'ordinateur sur lequel est installé le serveur et le numéro du port
Le port est un numéro compris entre 1024 et 65535 inclus.
On prendra comme numéro de port part défault 12000
La méthode « active » à true active le serveur.



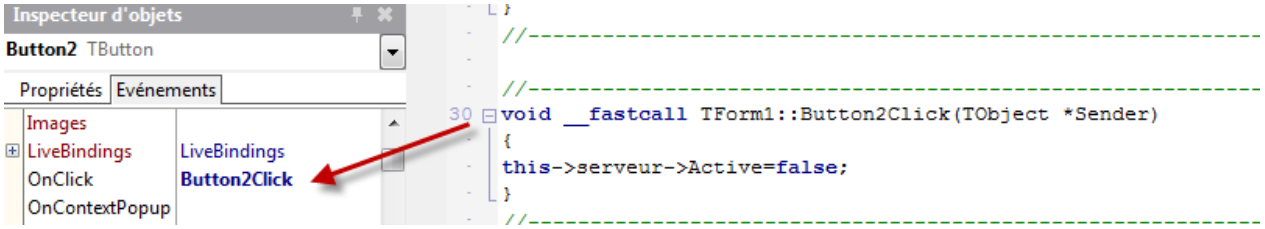
Nom :..... Prénom :.....



Remarque :
Lors du développement, il n'est pas pour autant nécessaire de disposer de plusieurs machines physiques ni même virtuelles. La même machine physique peut parfaitement héberger le serveur et un ou plusieurs clients, exactement dans les mêmes conditions: en communiquant par des ports.
Dans les domaines des réseaux informatiques, **localhost** (l'hôte local en français) est un nom habituel pour se référer à une interface logique de l'ordinateur local.
Le nom **localhost** est associé à l'adresse IPv4 **127.0.0.1** et à l'adresse IPv6 **::1**.

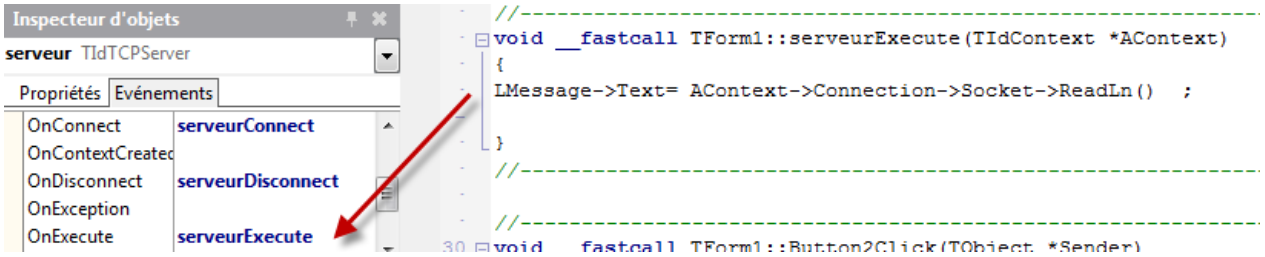
Se déconnecter

Pour se déconnecter, il suffit d'utiliser la méthode "active" à false qui arrête le serveur.



Lire les informations

L'événement onexecute permet de lire les informations reçues des sockets.
On utilisera Edit2 pour récupérer ses informations.

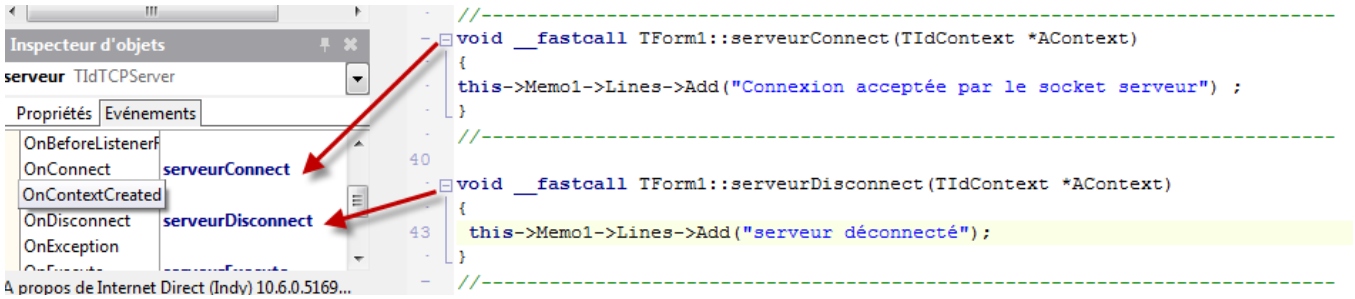


La gestion des événements

Les événement du composant TIdTCPServer permettent au programme de connaitre l'état de la connexion et gérer les erreurs si il y en a.

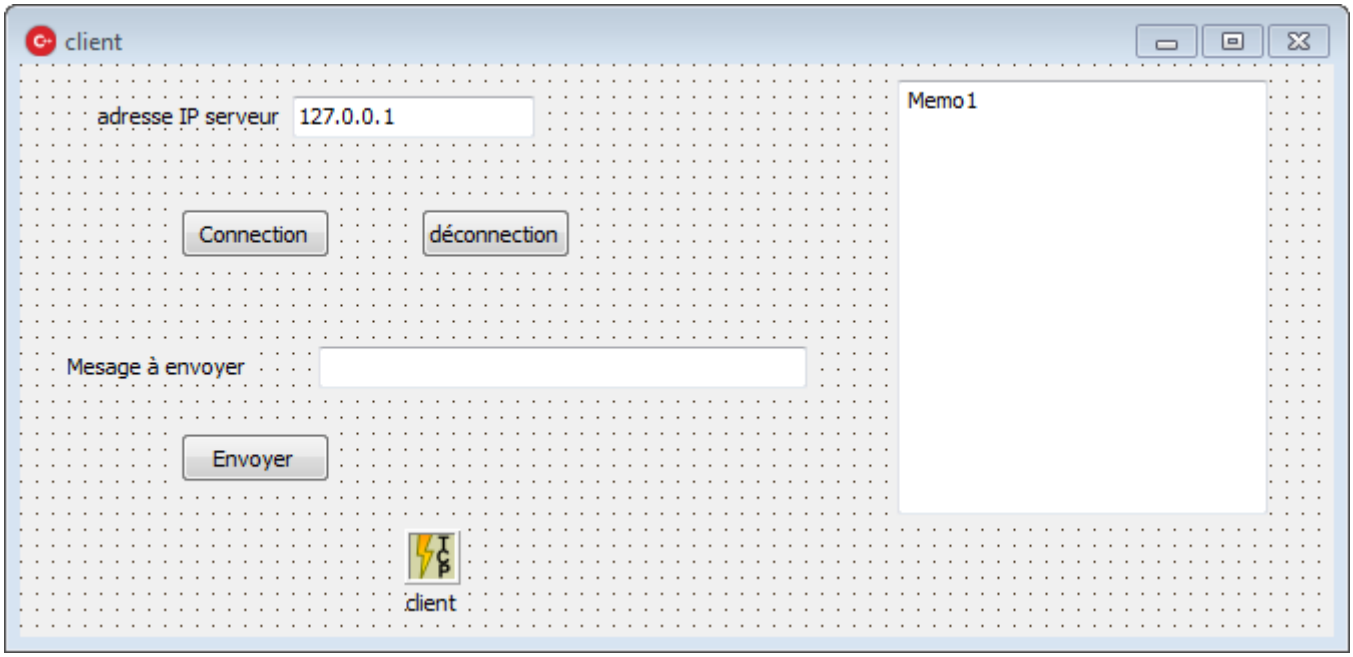
L'événement "Onconnect" se produit quand un socket client mène à bien une connexion acceptée par le socket serveur.
"OnDisconnect" se produit quand l'une des connexions avec un socket client est fermée.

Nom :..... Prénom :.....



Création du programme client

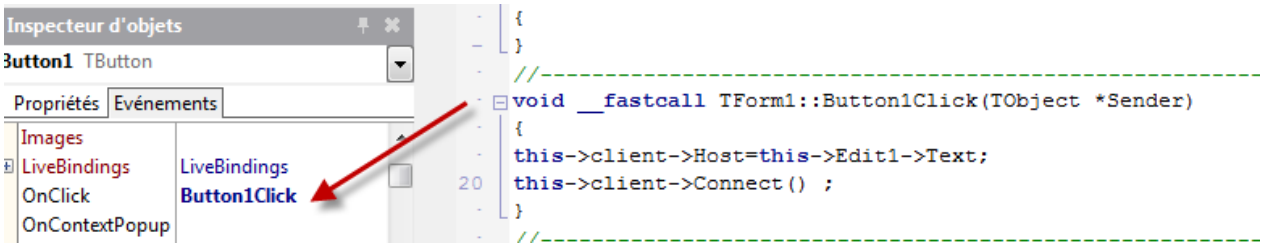
Voici à ce que ressemblera le programme client



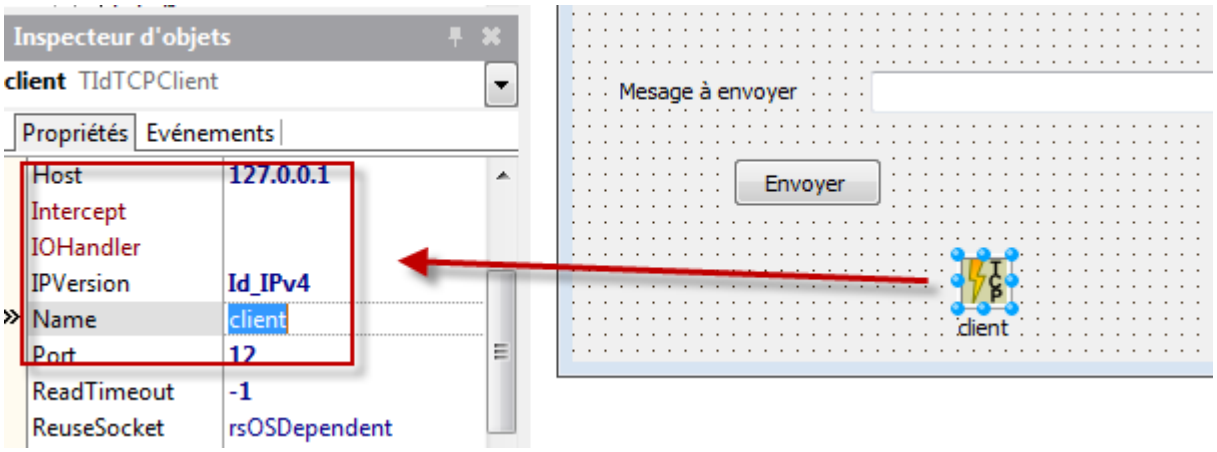
Ajouter un TMemo, trois TButton, deux TEdit, deux TLabel et un TIdTCPClient.
"Button1" est le bouton de connexion.
"Button2" est le bouton de déconnexion.
"Button3" est le bouton pour transmettre le texte au programme serveur.
"Edit2" est l'adresse IP de l'ordinateur du programme serveur.
"Edit3" est le texte que l'on veut transmettre au programme serveur.
"Memo1" donne l'état de la connexion.

Se connecter

Comme pour le composant TIdTCPserver,
"Host" définit l'adresse IP de l'ordinateur su programme serveur.
"connect" connecte au serveur.

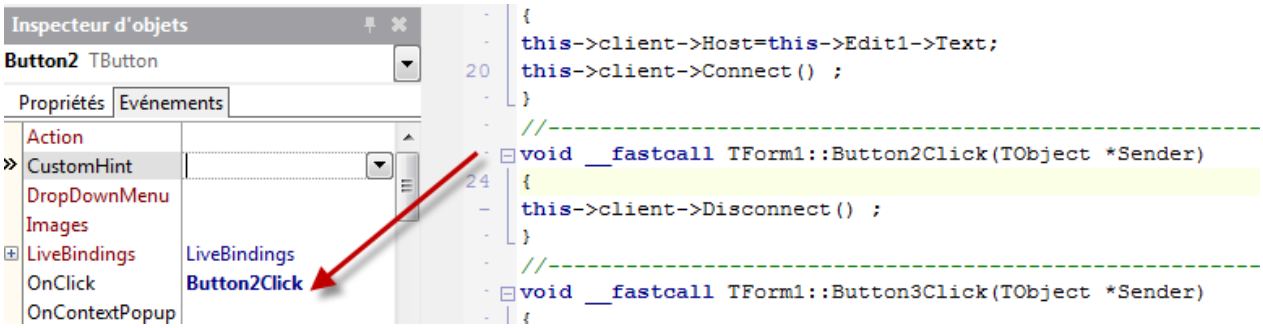


Nom :..... Prénom :.....



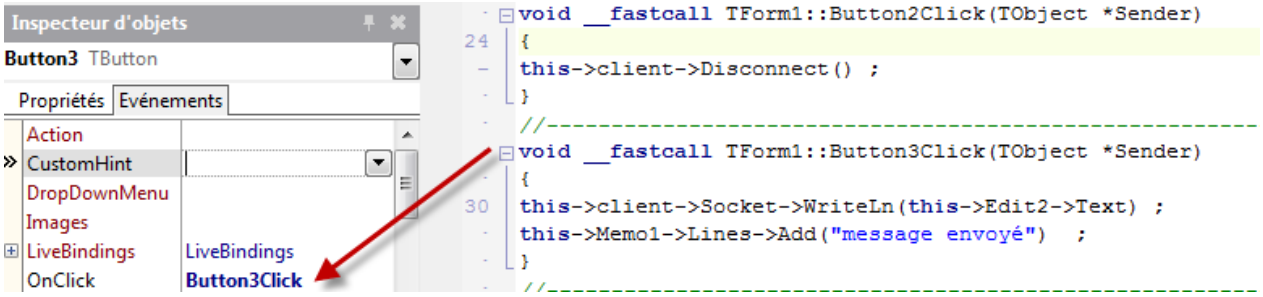
Se déconnécter

Pour se déconnecter, il suffit d'utiliser la méthode "disconnect" pour se déconnecter au serveur.



Transmettre du texte au serveur

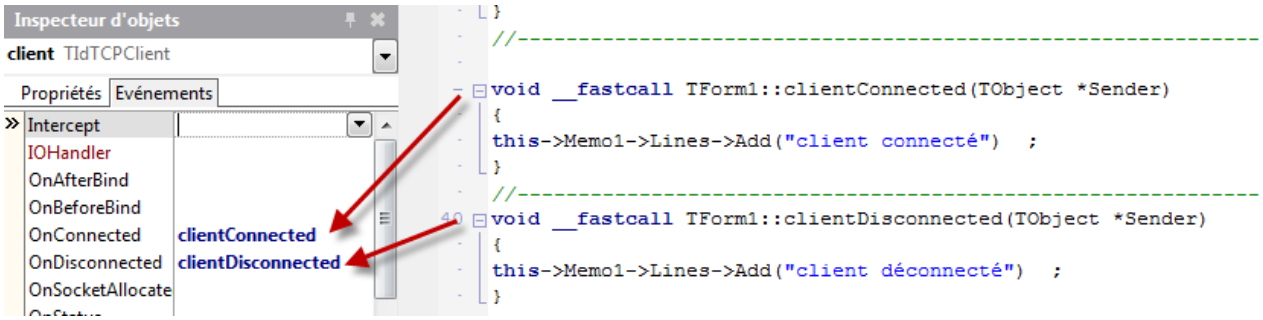
Pour envoyer du texte au serveur, on utilise "WriteLn".
Edit3 est le texte que l'on veut transmettre.



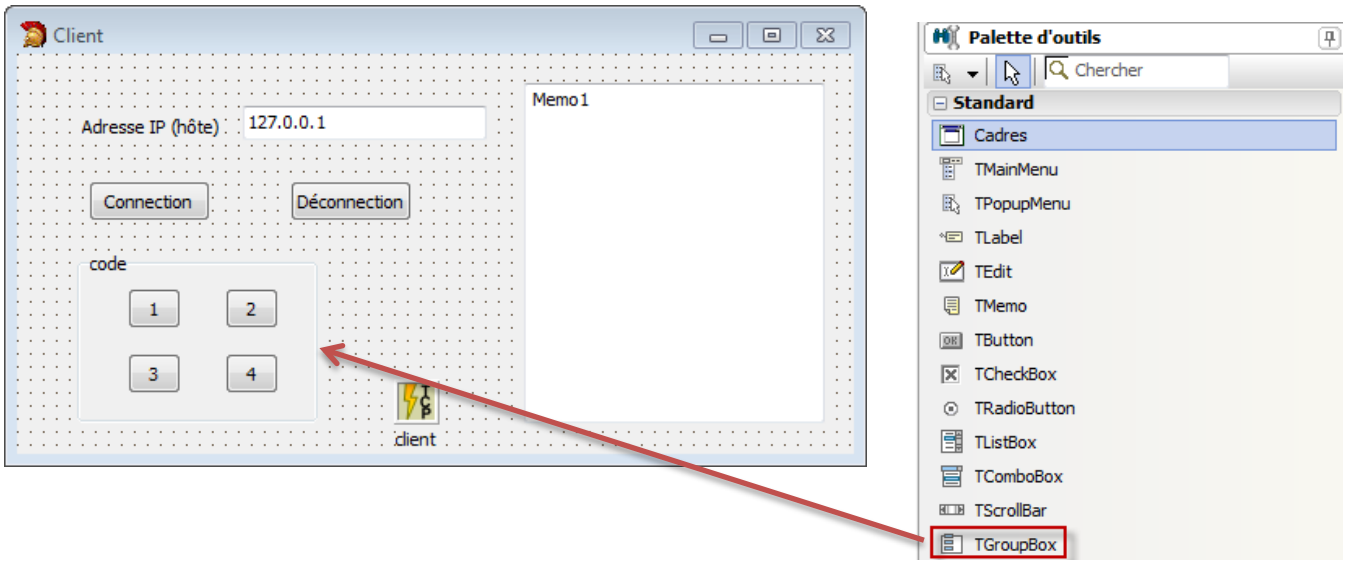
La gestion des événements

L'événement "OnConnecting" se produit une fois que le socket serveur a été localisé.
"OnDisconnect" se produit juste avant qu'un socket client ne ferme la connexion avec un socket serveur.

Nom :..... Prénom :.....



- Test de compréhension :
  - On veut réaliser un panneau de contrôle permettant de taper un code d'un chiffre



  - Modifier les programmes serveur et client, suivant le cahier des charges suivant :
    - Sur le programme client, on veut pouvoir taper un code d'un chiffre et l'envoyer au programme serveur.
    - Sur le programme serveur, on veut contrôler le code. Puis afficher le texte dans le mémo 'code correct', si le code défini est bon et code incorrect dans le cas inverse.
    - Le code est le chiffre 2