AMIENS
LA PROVIDENCE
ÉTABLISSEMENT SCOLAIRE JÉSUITE

Document TP TCP Client / Serveur

BURGUET Lucas	2SN
HAUTEMANIERE Edouard	2SN

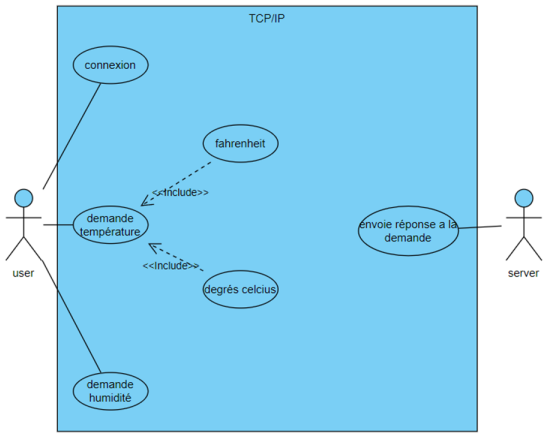
AMIENS LA PROVIDENCE
ÉTABLISSEMENT SCOLAIRE JÉSUITE

SOMMAIRE

Introduction :

La création d'une application cliente et d'une application serveur constitue une entreprise complexe mais cruciale dans le domaine de l'informatique. Ces applications reposent sur un protocole spécifique qui régit la communication entre le client et le serveur. Dans ce contexte, notre objectif est de mettre en œuvre une application respectant un protocole de communication précis, permettant à un client de récupérer des informations telles que la température en degré Celsius ou Fahrenheit ainsi que l'hygrométrie auprès d'un serveur. Notre approche s'appuiera sur l'utilisation de la bibliothèque Qt en C++, assurant une interface simple et conviviale pour les utilisateurs.

Documentation technique



Partie server :

On commence par le fichier Server.h dans lequel on instancie notre class les accès public et private ainsi que les slots permettant respectivement de :

-nouvelle connexion
- déconnexion
-écouter et répondre

```
class server : public QObject
{
    Q_OBJECT

public:
    server(QObject* parent = Q_NULLPTR);
    virtual ~server();

private :
    QTcpServer* tcpserver;

public slots:
    void onServerNewConnection();
    void onClientDisconnected();
    void onClientReadyRead();
};
```

Le main nous permet de lancer le serveur en appelant notre calss.

```
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    server srv(&a);
    return a.exec();
}
```

Server::Server(QObject* parent)// construct
Server::~Server() // destructeur

```
server::server(QObject* parent)
    : QObject(parent)
{
    tcpserver = new QTcpServer(this);
    QObject::connect(tcpserver, SIGNAL(newConnection()), this, SLOT(onServerNewConnection()));
    tcpserver->listen(QHostAddress::AnyIPv4, 4321);
}

server::~server()
{
}
```

connexion du client

```
void server::onServerNewConnection() {
    /*ui.connectionStatusLabel->setText("Un client c'est connecté");*/
    qDebug() << "client connecté";
    QTcpSocket* client = tcpserver->nextPendingConnection();
    QObject::connect(client, SIGNAL(readyRead()), this, SLOT(onClientReadyRead()));
    QObject::connect(client, SIGNAL(disconnected()), this, SLOT(onClientDisconnected()));

}
```

déconnexion du client

```
void server::onClientDisconnected()
{
    QTcpSocket* obj = qobject_cast<QTcpSocket*>(sender());
    QObject::disconnect(obj, SIGNAL(readyRead()), this, SLOT(onClientReadyRead()));
    QObject::disconnect(obj, SIGNAL(disconnected()), this, SLOT(onClientDisconnected()));
    obj->deleteLater();
}
```

gestion des envois de réponse.

```
void server::onClientReadyRead()
{
    QTcpSocket* socket = qobject_cast<QTcpSocket*>(sender());
    QByteArray obj = socket->read(socket->bytesAvailable());
    QString str(obj);
    qDebug() << str;
    QStringRef subString(&str, 2, 2);

    int val = (rand() % 5700) - 2000;
    float rval = (float)val / (float)100;
    char c[2];
    strcpy(c, "");
```

vérification si il réponds bien aux exigence pour répondre

```
if (strlen(obj) == 5) {
    if (str.endsWith("?")) {
```

qu'il termine bioen par ? et ai bien 5 caractère.

```
if (str.startsWith("Td"))
{
    qDebug() << "Temperature demandee en degres celsius";
    QString reponse = "Td";
    reponse.append(subString);
    if (rval > 0)
    {
        reponse += ",+";
        reponse += QString::number(rval);
    }
    else {
        reponse += ",";
        reponse += QString::number(rval);
    }
    socket->write(reponse.toUtf8());
}
```

pour les températures en degrés celcius fabrication de la réponse.mise en place des valeur - ou + pour l'affichage.

```
if (str.startsWith("Tf"))
{
    rval = (rval * 9.0 / 5.0) + 32;
    qDebug() << "Temperature demandee en Faraneite";
    QString reponse = "Tf";
    reponse.append(subString);
    if (rval > 0)
    {
        reponse += ",+";
        reponse += QString::number(rval);
    }
    else {
        reponse += ",-";
        rval *= -1;
        reponse += QString::number(rval);
    }

    socket->write(reponse.toUtf8());
}
```

pour les température en fahrenheit avec calcul pour passer la valeur en fahrenheit. mise en place des valeur - ou + pour l'affichage.

```
if (str.startsWith("Hr")) {
    int val = (rand() % 999);
    float rval = (float)val / (float)10;
    qDebug() << "Taux d'humidité";
    QString reponse = "Hr";
    reponse.append(subString);

    reponse += ",";
    reponse += QString::number(rval);
    reponse += "%";

    socket->write(reponse.toUtf8());
}
```

mise en place de la reponce pour l'humidité

Partie client :

On commence par le fichier InitiationQt.h dans lequel on instancie notre class les accès public et private ainsi que les slots permettant respectivement de :

- Un bouton de connection
- Une connection par socket
- Une déconnexion du socket
- Une lecture du socket
- Une déconnexion du client
- Un bouton qui sert à envoyer les info en Celsius
- Un bouton qui sert à envoyer les info en Fahrenheit
- Un bouton qui sert à envoyer les info pour l'Hygrométrie

```
#pragma once

#include <QtWidgets/QMainWindow>
#include "ui_InitiationQt.h"
#include <qtcpsocket.h>
#include <qtcpserver.h>

class InitiationQt : public QMainWindow
{
    Q_OBJECT

public:
    InitiationQt(QWidget *parent = Q_NULLPTR);
    //~InitiationQt();

private:
    Ui::InitiationQtClass ui;
    QTcpSocket * socket;
    QTcpServer * server;
public slots:
    void onConnectButtonClicked();
    void onSocketConnected();
    void onSocketDisconnected();
    void onSocketReadyRead();
    void onClientDisconnected();
    void onSendButtonCelciusClicked();
    void onSendButtonFahrenheitClicked();
    void onSendButtonHigrometrieClicked();
};
```

Le main nous permet de lancer le serveur en appelant notre calss.

```
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    InitiationQt w;
    w.show();
    return a.exec();
}
```

Questions Préliminaires

1 - Principe de la notion de client/serveur en informatique :

En informatique, le modèle client-serveur se réfère à une architecture où un programme (le client) demande des services ou des ressources à un autre programme (le serveur). Le client envoie une requête au serveur, qui répond en fournissant les informations ou les services demandés.

2 - Qu'est-ce qu'un protocole et à quoi sert-il ?

Un protocole est un ensemble de règles qui permettent à des dispositifs de communiquer entre eux de manière organisée et standardisée. Il définit le format, l'ordre et l'erreur éventuelle de transmission des messages échangés entre les dispositifs connectés.

3 - Explication de la notion de port et de socket sous TCP/IP :

Sous TCP/IP, un port est un numéro qui identifie une application ou un service spécifique sur un ordinateur. Un socket est une combinaison de l'adresse IP d'un ordinateur et du numéro de port associé, qui permet à une application d'établir une connexion réseau avec une autre application.

3 - En utilisant l'aide de Qt, quelles sont les classes de Qt permettant la création d'une application cliente et d'une application serveur ?

Pour la création d'une application cliente ou serveur avec Qt, vous pouvez utiliser les classes suivantes :

Pour une application cliente : utilisez la classe QTcpSocket.
Pour une application serveur : utilisez la classe QTcpServer.

Conclusion

La réalisation de l'application cliente et de l'application serveur a été menée avec succès. En suivant rigoureusement le protocole de communication établi, nous avons pu concevoir une application permettant la récupération et l'affichage des informations relatives à la température et à l'hygrométrie. L'utilisation des classes spécifiques de la bibliothèque Qt a permis de simplifier et de rendre plus fluide le processus de communication entre le client et le serveur. De plus, la répartition des tâches au sein de l'équipe a favorisé une meilleure gestion du projet, tout en garantissant un développement efficace des deux applications. Ce projet a été une occasion enrichissante de mettre en pratique les principes fondamentaux de la communication client-serveur et d'approfondir notre compréhension des technologies de réseau en informatique.

Lien du github : https://github.com/burguet/tcpserver