Projet PolyPeer

Et déployer vos images VirtualBox n'a jamais été aussi simple ...

Retrouvez toute l'aide pour configurer le projet dans le fichier CONFIG

L'aide à l'installation se trouve dans le fichier INSTALL

Le manuel utilisateur est disponible sur l'interface web.

Système de déploiement d'images VirtualBox en pair à pair

Le système PolyPeer permet de déployer des images VirtualBox (http://en.wikipedia.org/wiki/VirtualBox) sur un parc de machine et ce en pair à pair tout en garantissant une charge du réseau raisonnable.

Technologies utilisées

PolyPeer utilise les technologies suivantes :

- La STL de C++
- TinyXML, bibliothèque de manipulation XML
- Mongoose, embedded webserver

Cahier des charges

Le cahier des charges est disponible sur le wiki du projet : https://github.com/KenTiN/PolyPeer/wiki .

Développement de PolyPeer

La documentation générée par Doxygen est disponible dans doc/doxydoc (ouvrez alors le fichier index.html).

Arboresence du projet

Chaque sous projet doit être placé à la racine du projet principal. Le sous projet doit être un dossier sans majuscules.

```
projet1/
    include/     -- Contient les headers
    src/         -- Contient les sources
    Makefile     -- Nécessaire à la compilation de l'exemple
    .gitignore   -- Nécessaire pour éviter de push les .o et l'executable
```

Dans chaque sous projet :

- Un dossier src/ avec les sources
- Un dossier include/ avec les headers
- Un makefile permettant de compiler un exemple simple !
- **OPTIONNEL** un readme pour le sous projet pour expliquer rapidement.
- Pensez à créer un .gitignore pour ne pas pusher les .o et l'executable du sous projet.

/! Pas de fichiers .cpp en dehors de src/ /!

Arborescence actuelle des projets

Le projet est actuellement divisé en sous-projets que voici :

```
algorithm/   -- Contient les sources pour le gestionnaire de déploiement
data/        -- Contient les sources pour l'entité Data
datamanager/ -- Contient les utilitaires de gestion de fichiers XML et de peuplement des str
callback/    -- Contient les fonctions d'actions sur les paquets
connection/  -- Contient le gestionnaire de connexions
data/        -- Contient l'utilitaire de gestion des données
filemanager/ -- Contient le gestionnaire de fichier
logger/      -- Contient l'utilitaire de gestion des logs
mutex/       -- Contient la bibliothèque de gestion des mutex
packet/      -- Contient les paquets et leur gestionnaire
ppclient/    -- Contient le client polypeer
ppserver/    -- Contient le serveur polypeer
tcp/         -- Contient la bibliothèque de gestion des Sockets
webserver/   -- Contient le gestionnaire de l'interface web
```

A noter : l'executable du client est a construire dans ppclient/ et l'executable du serveur dans ppserver/ (via Makefile).

Conventions de nommages

Exemple de code standard :

```
#include <iostream>
#include <sstream>

#include <WebServer.hpp>

using namespace std;
```

```
int main(int argc, char* argv[])
{
    // Exemple de code simple
    WebServer* server = WebServer::getInstance();

    // Lancement
    server->start();

    return 0;
}
```

Dans cet exemple :

- Pensez d'abord à inclure les fichiers d'entête de la librairie standard et des bibliothèques tierces
- Puis inclure les fichiers d'entête du projet
- Déclarer quel namespace utiliser
- Déclarer vos fonctions

```
#ifndef H_MYCLASS
#define H_MYCLASS

class MyClass
{
private:
    int myAttribute;

public:
    MyClass();
};

#endif
```

Dans cet exemple :

- Pas de tabulation pour les mots clés private, public, et protected.
- Bien protéger vos headers de l'inclusion infinie.
- Respecter la casse, une classe commence par une majuscule, chaque mot clé a sa première lettre en majuscule.
- Les attributs et méthodes démarrent pas une minuscule, chaque mot clé a sa première lettre en majuscule.

Documentation

Chaque méthode, attribut de classe et classe elle même doit être documentée comme suit :

```
/**
 * Ceci est une classe d'exemple
 */
class Example
{
public:
    /**
     * Constructeur d'un exemple
     * @param int
     *  le paramètre du constructeur
     */
     Exemple(const int param);

     /**
      * Méthode permettant de retourner le paramètre
      * @return int
      *     retour du paramètre
      */
      int getMyParam();

private:
    /**
     * Attribut permettant de gérer un paramètre
     */
    int myParam;
};
```

Lors d'un passage compliqué dans un code source, n'hésitez pas à expliquer.

Besoin de plus d'infos ?

Ce projet est réalisé dans le cadre du module d'Ingénierie Logicielle (S6).

Interlocuteur projet : Lebourgeois Quentin

Pour nous contacter :

- Contact par mail : quentin.lebourgeois at etu dot univ-nantes dot fr
- Contact sur GitHub : http://github.com/KenTiN

Membres du projet :

- Blin Oliver : Algorithme de déploiement et gestion du protocole de communication
- Daumont Charles : Implémentation des structures de données du projet
- Lebourgeois Quentin : Réalisation de l'interface homme-machine et structuration du projet, fichiers d'aide, de config et d'installation
- Matrat Erwan : gestion des fichiers et wrapping des objets pour compilation cross-platform

Licence du projet

Document rédigé par Quentin Lebourgeois