TP_LO43

ROMET Pierre

Automne 2019

1 Présentation du projet

1.1 Introduction

Le TP de LO43 va prendre la forme d'un mini-projet, que vous allez devoir réaliser en solitaire, qui va vous permettre de mettre en pratique les connaissances acquises lors du cours; il sera également un aperçu du projet Java, ainsi que ces attentes, que vous devrez rendre en fin de semestre.

Vous allez disposer de trois séances de trois heures chacune pour réaliser votre projet et étant donné la nature de l'U.V, la difficulté de ce dernier sera croissante.

1.2 Mise en contexte

Votre projet a pour but la réalisation d'un simulateur, permettant de modéliser un écosystème IOT spécialisé dans le monitoring de qualité de l'air d'espace de travail.

Cet écosystème sera composé d'un serveur communiquant avec divers types de capteurs répartis au sein de l'espace de travail, tels qu'un capteur de température, un capteur d'humidité, un capteur de lumière, ainsi qu'un capteur sonore. De plus, les informations provenant des capteurs pourront être stockées sous forme de fichier de logs ou encore partagées dans une console afin de les visualiser.

2 Le simulateur

Lors de cette première séance, vous allez être amenés à mettre en pratique les fondamentaux de la programmation de par son architecture, son écriture ainsi que sa compilation. Vous mettrez également en place des outils de suivi de code, de versioning & de travail collaboratif.

2.1 Votre premier programme

Pour votre premier programme, vous allez rédiger un grand classique de la programmation, le bien connu "Hello World !". Une fois ce dernier écrit, vous devrez le compiler, puis l'exécuter.

Concernant la compilation, vous utiliserez le compilateur GNU pour le langage C++, "g++". Vous devrez dans un premier temps renommé votre fichier compilé "HELLOWORLD" (avec l'option de compilation qui convient), puis vous devrez afficher les Warnings.

Pour savoir comment rédiger la commande g++, vous utiliserez la documentation Linux présente dans la console, le "man".

Une fois le travail effectué, vous le ferez constater par un professeur.

2.2 Première séance

Nous allons maintenant nous pencher sur le développement de notre simulateur. Comme expliqué précédemment, notre simulateur est composé d'un serveur ainsi que d'un ensemble de capteurs.

Cependant, avant de commencer à développer votre première classe, vous allez mettre en place un dépôt GitHub. GitHub est une plate-forme web permettant d'héberger votre travail, de mettre en place du versioning et de pouvoir travailler à plusieurs sur un projet de développement. C'est notamment grâce à ce dépôt, que votre travail sera rendu en milieu de semestre, afin d'être évalué. Vous devrez donc vous créer un compte, puis cloner un dépôt distant sur lequel vous allez travailler. Une fois votre compte créé, vous irez voir le prof qui vous ajoutera en tant que contributeur sur son dépôt.

Vous trouverez à cette adresse, le tuto officiel en ligne de Git

https://git-scm.com/docs

Vous allez commencer par vérifier que Git est bien installé sur votre ordinateur. Pour cela vous exécuterez la commande suivante:

git –version

qui devra vous retourner les lignes suivantes:

git version 2.20.1

Vous allez donc commencer par cloner le repo fourni par votre prof, **à l'emplacement que vous souhaitez**

git clone "URL du dépôt fourni par le prof"

Ensuite, vous devrez créer une nouvelle branche

git branch MABRANCH

Puis, vous vous déplacerez sur votre nouvelle Branch, sur laquelle vous allez pouvoir héberger votre code

git checkout MABRANCH

Enfin, vous pousserez votre nouvelle branch sur le serveur distant :

git push -–set-upstream origin MABRANCH

Cela fait, vous disposer maintenant d'un espace de travail, sur un repo distant, qui vous est dédié. Vous allez maintenant copier votre code dans le dossier du repo que vous venez de cloner.

Nous allons maintenant placer notre fichier sous "suivis de version", ce qui va permettre de traquer les changements qui surviendront dans le code. On dit que notre code va être ajouté à l'index.

git add MONFICHIER

Ensuite, nous allons valider les modifications apportées à l'index

git commit –m "Description du commit"

Enfin, nous finissons en envoyant notre code sur le serveur distant

git push

Vous pouvez maintenant vous attaquer à votre première tâche, qui sera de réaliser la classe "Server". Cette classe doit nous permettre de recevoir des données envoyer par l'ensemble des capteurs, puis de les stocker dans des fichiers de logs ou encore de les visualiser dans la console.

Pour cela, vous allez commencer par implémenter la forme canonique de Coplien de la classe. Une fois cela fait, vous écrirez deux fonctions:

- "consoleWrite" afin de visualiser les données arrivantes dans la console.
- "fileWrite" pour stocker les données des capteurs dans des fichiers de logs (chaques capteur devra disposer de son fichier de logs).

Travail en autonomie.

À la fin de cette première séance, vous aurez normalement fini d'implémenter votre classe "Server" à travers son Coplien ainsi que ces fonctions.
Cependant, pour que vous ne restiez pas sans pratiquer jusqu'à la prochaine séance, il vous est demandé d'implémenter deux opérateurs "=" & ">>", que vous allez devoir Surcharger.

Pour les deux opérateurs, la surcharge se définit comme suivant:

- "=" : La surcharge doit permettre de copier le contenu d'un objet dans l'objet courant.
- ">>" : Deux surcharges vous sont demandées pour l'opérateur>>.
  - L'élément que l'on souhaite afficher doit être redirigé vers la console.
  - L'élément que l'on souhaite afficher doit être redirigé vers un fichier de log.

De plus, vous aurez à explorer les diverses possibilités qu'offre Git. Pour cela, vous trouverez un tuto à l'adresse suivante:
https://try.github.io/, se nommant "Learn Git branching".

De plus, vous utiliserez l'outil "Visualizing Git" pour tester et comprendre les commandes suivantes:

- Git config
- Git init
- Git status
- Git merge
- Git diff
- Git blame

Vous devrez rendre pour la prochaine séance un petit rapport, détaillant le rôle des commandes ci-dessus.

3 Deuxième séance

Lors de cette deuxième séance, vous allez mettre l'architecture de votre projet en place. Vous commencerez votre séance par une introduction à l'architecture logiciel et une mise en pratique.

Il vous sera ensuite distribué un schéma que vous aurez à compléter, de manière individuel. Une fois terminé, vous rendrez ce document à votre enseignant, qui notera une appréciation qui comptera pour l'évaluation de votre projet.

Cela fait, il vous sera distribué une version finale de l'architecture du projet, qui vous servira de patron pour le développement.

Vous allez maintenant créer les deux nouvelles classes indiqué sur votre diagramme de classe; Le diagramme de classe vous indiquant les fonctions que vous allez être amené à coder.

3.1 Sensor

La classe "Sensor" représente la fonction mère sur laquelle vous allez baser les classes de vos capteurs.

Dans un premier temps, tous vos capteurs vous retournerons le même type de données.

Pour votre rendu final, vous devrez pouvoir générer des données de plusieurs types différents, en fonction des capteurs.

- Temperature, Humidity : FLOAT
- Sound : INTEGER
- Light : BOOLEAN

3.2 Scheduler

Dans cette classe, vous allez devoir implémenter un algorithme que vous allez "inventer", qui va vous permettre de récupérer les informations mesurés par chacun des capteurs et que vous aurez à retourner au serveur.

La contrainte final demander, sera que les informations de chaque capteurs pourrons être récupérer et transmise à des intervalles de temps différents.

3.3 Server

Le code de cette classe représente la première partie du projet.

4 Synthèse

Voici la liste des logiciels que vous avez pu mettre en place pour votre projet:

- Git
- GitHub
- g++
- man
- make pour faire des makefile (compilation automatisé)
- gdb (débeuger)
- valgrind (détection fuite mémoire)
- éditeur de code: VIM, Gedit, atom, nano, imax, sublimeText
- éditeur Architecture logiciel (UML): le seul et unique "ASTAH"