

TP_AP4A

ROMET Pierre

Automne 2020

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 et du TD; il sera également une première mise en contexte concernant le déroulement et les attentes du projet Java que vous réaliserez à la suite de ce tp.

Vous avez jusqu'au jeudi 22 octobre pour me rendre votre tp projet.

1.2 Mise en contexte

Ce mini-projet vise la réalisation d'un simulateur d'environnement IOT.

Vous devrez modéliser un éco-système de capteurs ayant pour but de monitorer l'environnement de la cabine des passagers à bord d'un avion de ligne.

Votre éco-système sera basé sur cinq types de capteurs différents, devant monitorer la température, l'humidité, la lumière ainsi que le niveau sonore dans la cabine. Ces capteurs devront ensuite communiquer avec un serveur sur lequel sera stocké les données qui lui seront transmises.

1.3 Rendu attendu

Pour le 22 octobre , vous devrez me rendre un rapport décrivant la manière dont vous aurez architecturé, pensé, mené le développement de votre projet. L'objectif n'est pas d'expliquer dans le détail les fonctions que vous aurez coder, mais bien la manière dont vous aurez conçu ce tp projet.

Je ferais un point sur cet élément lors de la première réunion Teams le jeudi 17 septembre à 14h.

2 Le simulateur

Lors de la première scéance, vous allez être amené à mettre en pratique les bases de la programmation C++ qui vous ont été introduites lors du cours et du TD.

Vous devrez également mettre en place des outils de programmation (l'IDE sur lequel vous déciderez de travailler), ainsi qu'un outil de versionning et de travail collaboratif.

PS: Vous retrouverez en fin de document, un glossaire des logiciels que vous allez être amené à utiliser, ainsi qu'un coding standard que vous devrez respecter.

2.1 Votre premier programme

Pour commencer, vous allez coder un classique de la programmation, le "Hello World".

Seul contrainte obligatoire pour cette partie, vous devrez éditer votre code dans un fichier text à l'aide d'un éditeur (vim, nano ou autre) et ensuite le compiler et l'exécuter en ligne de commande.

Concernant la compilation, vous utiliserez le compilateur GNU C++, à savoir "g++". De plus, vous utiliserez les options du compilateur pour renommer votre fichié compilé "HELLOWORLD" ainsi qu'afficher les warnings.

De plus, vous vous baserez sur la documentation Linux de la console, le "man" pour rédiger votre commande g++.

2.2 Premier pas dans le simulateur

Nous allons maintenant commencer le développement de votre simulateur. Comme expliqué précédemment, le simulateur est composé d'un serveur ainsi que d'un ensemble de capteurs.

Cependant, avant de commencer à coder, vous allez mettre en place un dépôt GitHub. GitHub est une plate-forme web permettant d'héberger votre travail, de le versionner et de travailler à plusieurs dessus. C'est notamment grâce à ce dépôt, que votre travail sera rendu en milieu de semestre, afin d'être évalué.

Vous allez donc commencer par vous créer un compte sur la plateforme (éviter d'utiliser le mail utbm), puis vous mettrez un commentaire dans le canal teams dédié au tp "Forum TP-Projet" contenant votre identifiant GitHub afin que l'encadrant de tp puisse vous ajouter au dépôt sur lequel vous serez amener à travailler.

Une fois cela fait, 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: **git version 2.28.0**

Sinon, installez le: **apt install git-all**

Vous pouvez maintenant cloner à l’emplacement que vous souhaitez sur votre machine, le repo fourni, avec la commande:

git clone "URL du dépôt fourni par le prof"

L’URL étant: `https://github.com/Aldarme/AP4A_TP_C-.git`

Vous allez maintenant vous déplacer dans le repo que vous venez de cloner avec la commande `cd`:

cd NOMDUDOSSIER

Ensuite, vous aller créer une nouvelle branche personnel sur le repo, qui sera votre espace de travail.

git branch MABRANCH

Puis, vous aller vous définir cette nouvelle branche comme espace de travail par défaut:

git checkout MABRANCH

Pour finir, 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 "helloword" 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 du fichier. On dit que le code est ajouté à l’index.

git add MONFICHIER

Ensuite, vous aller valider les modifications apportées à l’index

git commit -m "Description du commit"

Si probleme d’authentification lors du commit:

git config --global user.email "MONMAILGITHUB"

git config --global user.name "MONNOM"

Enfin, vous pousser votre code sur le serveur distant

git push

Rentrer email, mdp github

2.3 Première classe

Avant toute chose, vous devrez prendre compte et analyser le coding standard qui vous est fourni avec ce sujet de tp. Vous devrez appliquer celui-ci pour coder votre projet.

La classe "Server" doit permettre de recevoir, de stocker et d'afficher les données envoyés provenant des capteurs de la cabine de l'avion.

Vous allez commencer par implémenter la forme canonique de la classe "Server".

Ensuite, vous implémenterez le stockage et l'affichage des mesures provenant des capteurs:

- Concernant le stockage des données, vous devrez logger les mesures reçus dans un fichier de log.
- Pour l'affichage, vous redirez les données des capteurs vers la console.

Travail à rendre pour le lundi 28 septembre

Cette partie de votre tp projet sera à rendre pour le lundi 28 septembre. Vous effectuerez votre rendu sous la forme d'un rapport que vous déposerez sur moodle et vous pousserez votre code sur github.

Vous aurez à implémenter la surcharge des opérateurs "=" et "<<".

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:
 - "<<": Redirection vers la console
 - "<<": Redirection vers un fichier

Pour vous aider, utiliser la doc C++ en ligne:

<https://www.cplusplus.com/reference/>,

Enfin, 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", que vous aller devoir faire.

Vous me rendrez un rapport faisant état de la réussite de ce tuto.

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

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

Vous incluez votre compréhension de ces commandes dans le rapport.

3 Simulateur, la suite

Nous allons maintenant voir comment mettre en place l'architecture logiciel du projet et son implémentation. L'architecture du projet vous est donné, afin que vous puissiez tous travailler sur la même base et que vous ne rencontriez pas de difficulté lié à la conception de l'architecture du projet.

Vous trouverez le fichier illustrant l'architecture du projet en annexe. Attention, le diagramme de classe en annexe ne fourni que le nom des fonctions, mais il ne défini pas le type de retour ainsi que les paramètres possibles. Vous devrez donc faire preuve d'imagination pour implémenter au mieux ces fonctions.

Je vous présenterais ce diagramme de classe à l'aide d'une vidéo que vous trouverez sur le canal de TP.

Après avoir pris connaissance du fichier en annexe, vous pouvez commencer à coder les classes définit par le diagramme de classe (fichier en annexe).

3.1 Classe "Sensor"

La classe "Sensor" représente la classe 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 le rendu final, vous devrez pouvoir générer des données de types différents, en fonction du type des capteurs.

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

3.2 Classe "Scheduler"

La classe Schéduler est le coeur de votre simulateur. C'est lui qui va gérer la fréquence à laquelle il va récupérer les données des différents capteurs pour ensuite les transmettre au serveur.

Dans un premier temps, vous récupérerez les mesures de tous les capteurs en même temps. La contrainte final sera que chaque type de capteur devra être récupéré à des fréquences différentes, avant d'être remonté sur le serveur.

3.3 Classe "Server"

La classe "Server" doit permettre de recevoir, de stocker et d'afficher les données envoyés provenant des capteurs de la cabine de l'avion.

Concernant le stockage des données, vous devrez logger les mesures envoyés par les capteurs dans des fichiers de log. Chaque type de capteur aura son fichier de log dédié. De plus, vous devrez pouvoir activer ou non, l'affichage des données ou le stockage de ces dernières.

4 Cheat Sheet

Voici une liste de logiciels qui pourrons vous être utile pour le projet:

- Editeur Architecture logiciel (UML): le seul et unique "ASTAH"
- Le man
- éditeur de code: VIM, EMACS, Gedit, atom, nano, imax, sublimeText
- IDE: Eclipse, VisualStudio, Jetbrain, etc...
- Compilateur: g++
- Debugger: gdb
- makefile (compilation automatisé)
- valgrind (détection fuite mémoire)
- Git
- GitHub

5 Guide de programmation

Vous trouverez de plus, un coding standard joint au sujet de TP, que vous aurez à respecter pour la programmation de votre TP projet.

