Projet De Stijl 2.0 : Plateforme pour robots mobiles

Programmation et conception de systémes temps réel – 4éme année AE/IR

Institut National des Sciences Appliquées de Toulouse

Guide des outils de développement

Version 1.0.1 (23 février 2020)

Référent pédagogique : P.-E. Hladik (`pehladik@insa-toulouse.fr`)

Référents plateforme : S. Di Mercurio (`dimercur@insa-toulouse.fr`)

---

1 Configurer Netbeans pour C/C++

Depuis la version 11 de Netbeans, il est nécessaire que chaque utilisateur configure son environnement avec le plugin C/CC++. Pour cela, il faut :

- ouvrir le menu « Tools > Plugins »
- aller dans l'onglet « Settings »,
- cocher « Netbeans 8.2 plugins Portal »,
- aller dans l'onglet « Available » et rafraichir,
- puis choisir C/C++ et suivre le guide...

2 Code initial du projet

Le code du projet est disponible sur un dépôt git hébergé sur GitHub. Pour le récupérer, placer vous le répertoire cible et exécuter la commande

```
git clone https://github.com/INSA-GEI/dumber.git
```

Tout le code relatif au projet est disponible, cependant vous n'aurez besoin que des codes présents dans les répertoires :

- `./dumber/software/raspberry/superviseur-robot` : ce répertoire contient un projet Netbeans avec l'ensemble du code source pour le projet initial,
- `./dumber/software/monitor/monitor` : ce répertoire contient un projet Mono contenant l'ensemble du code source pour le moniteur.

Le répertoire `superviseur-robot` est constitué des fichiers suivants :

- `/destijl_init/main.cpp` qui contient le main de l'application et lance la création des objets et leur exécution,

— `/destijl_init/src/tasks.h` qui contient l'entête des différentes fonctions,
— `/destijl_init/src/tasks.cpp` qui contient l'implémentation des fonctions de création des objets (tâches, sémaphores, mutex, etc.) ainsi que les fonctions de traitement.

3 Compilation d'une application distante

L'application étant sur une Raspberry Pi, il vous faut compiler le programme pour cette architecture. Nous vous proposons d'utiliser Netbeans pour écrire votre code et faire la compilation distante. Cela signifie que votre code est stocké sur votre compte INSA, que vous éditez le code sur la machine de TP, que ce code est ensuite automatiquement chargé sur la Raspberry Pi puis compilé (Netbeans permet de faire tout cela).

Pour commencer, lancez Netbeans et ouvrez le projet `superviseur-robot`.

Avant de compiler, il vous faut configurer la cible sur laquelle la compilation va se faire :

1. Cliquez droit sur le projet et choisir `Properties`,
2. Allez dans l'onglet `Build`,
3. Choisir `...` pour `Build Host`,
4. Cliquez sur `Add`,
5. Remplir `Hostname` avec `10.105.1.x` l'adresse de la Raspberry Pi,
6. Cliquez `Next`,
7. Saisir `pi` pour le `login`,
8. Cliquez `Next`,
9. Attendre l'ouverture de la communication,
10. Saisir `insa` comme mot de passe,
11. Sélectionner `SFTP` pour `Access project files via`,
12. Cliquez `Finish`,
13. Cliquez `OK`,
14. Cliquez `Apply`,
15. Cliquez `OK`.

Pour compiler, il suffit ensuite de cliquer sur l'icône en forme de marteau. Vous pourrez voir dans le terminal les étapes de compilation qui commencent par le transfert des fichiers suivi de la compilation proprement dite.

Remarque : la première compilation est un peu longue, mais devrait ensuite se fluidifier avec la compilation incrémentale.

4 Exécution du superviseur

Pour exécuter l'application sur le superviseur, il faut avoir mis en place un terminal distant (voir ci-dessous) puis démarrer l'exécution avec la commande `sudo ./path/app` où `path` est le chemin du répertoire où se trouve l'application et `app` est le nom de votre application. Attention les droits `sudo` sont nécessaires pour des questions d'accès à certains services Xenomai de gestion de la mémoire.

Si vous utilisez Netbeans, le répertoire dans lequel est compilée l'application se trouve dans l'arboressence commençant par `.netbeans` (attention au "." au début). Il faut descendre dans cette arboressence jusqu'à une bifurcation, puis choisir le répertoire `dist` et aller jusqu'au bout. L'application se trouve au bout de cette branche.

5 Mise en place d'un terminal distant avec la Raspberry Pi

Pour se connecter à la Raspberry Pi, vous aurez besoin de créer un accès via ssh. Pour cela, depuis un PC d'une salle informatique utilisez la commande :

```
ssh pi@10.105.1.x
```

avec `x` le numéro sur le boitier de la Raspberry Pi.

Le mot de passe est : insa.

6 Exécution du moniteur

Pour exécuter le moniteur, il suffit depuis votre PC de travail de se placer dans le répertoire `./dumber/software/monitor/monitor` et de lancer `./monitor`.

Attention : avant de lancer la première fois l'application, il vous faut ouvrir l'environnement de développement MonoDevelop et ouvrir le projet `./dumber/software/monitor/monitor` et l'exécuter (cela recompile le projet et permet de faire les liens avec les librairies de votre PC). Cette étape ne sera plus à faire par la suite.

7 Comprendre la structuration du code

L'ensemble du code du projet initial est contenu dans le répertorie `superviseur-robot`. Les bibliothèques de traitement sont disponibles dans le repertoire `lib`. Vous n'aurez pas à modifier ces bibliothèques, par contre vous devrez les utiliser. La documentation est directement écrite dans le code source.

Les seuls fichiers que vous allez avoir à modifier sont les fichiers `tasks.h` et `tasks.cpp`. La classe `Tasks` a pour attributs privés l'ensemble des structures constituant l'application (tâches, mutex, sémarphores, ressources, etc.). Les fonctions de traitement associées aux tâches sont déclarées comme des méthodes privées de `Tasks`.

La classe `Tasks` propose quatre méthodes publiques :
— `Init()` qui crée les structures de l'application,
— `Run()` qui lance les tâches,
— `Stop()` qui termine les tâches,
— `Join()` qui suspend l'exécution du `main`.

Pour ajouter des structures à l'application, il suffit de les déclarer dans le fichier `tasks.h` puis de s'assurer qu'elles sont bien instanciées dans les méthodes adéquates de `tasks.cpp`. Pour une tâche, il faut en plus déclarer une nouvelle méthode privée dans tasks.h et fournir son implémentation dans tasks.cpp. Cette nouvelle méthode sera le point d'entrée de la tâche.

Toute la documentation de Xenomai se trouve en ligne (attention c'est Xenomai 3) et nous utilisons l'API alchemy.