|  |  |  |
| --- | --- | --- |
| Logo Marcel-Callo Nov 2010 |  |  |
| **Créer un PACKAGE avec source C++** |

Principe de la structure d’un work\_space

1. Créer un répertoire de travail par vos soins, sous home.
2. Créer un dossier « src » dans ce répertoire.
3. Se placer dans ce dossier « src »
4. Pour créer un nouveau package utiliser la commande :

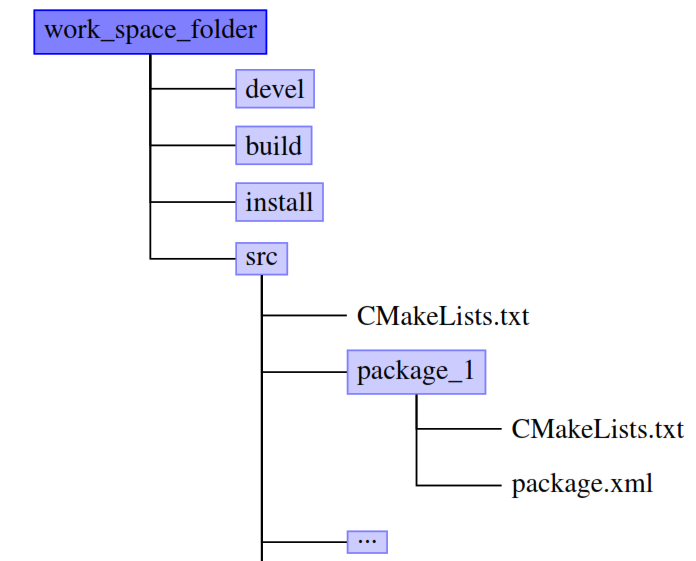
catkin\_create\_pkg package\_1 std\_msgs roscpp

Les dépendances invoquées que l’on retrouve plus tard dans le fichier **package.xml**

Le nom du package

La commande pour créer le package

le répertoire de travail crée par nos soins



le dossier src créé par nos soins

src

Structure créée pour le package par la commande

catkin\_create\_pkg

Dossiers et fichiers crées par la compilation du projet plus tard.

include

On répète les packages autant de fois que nécessaire



catkin\_create\_pkg

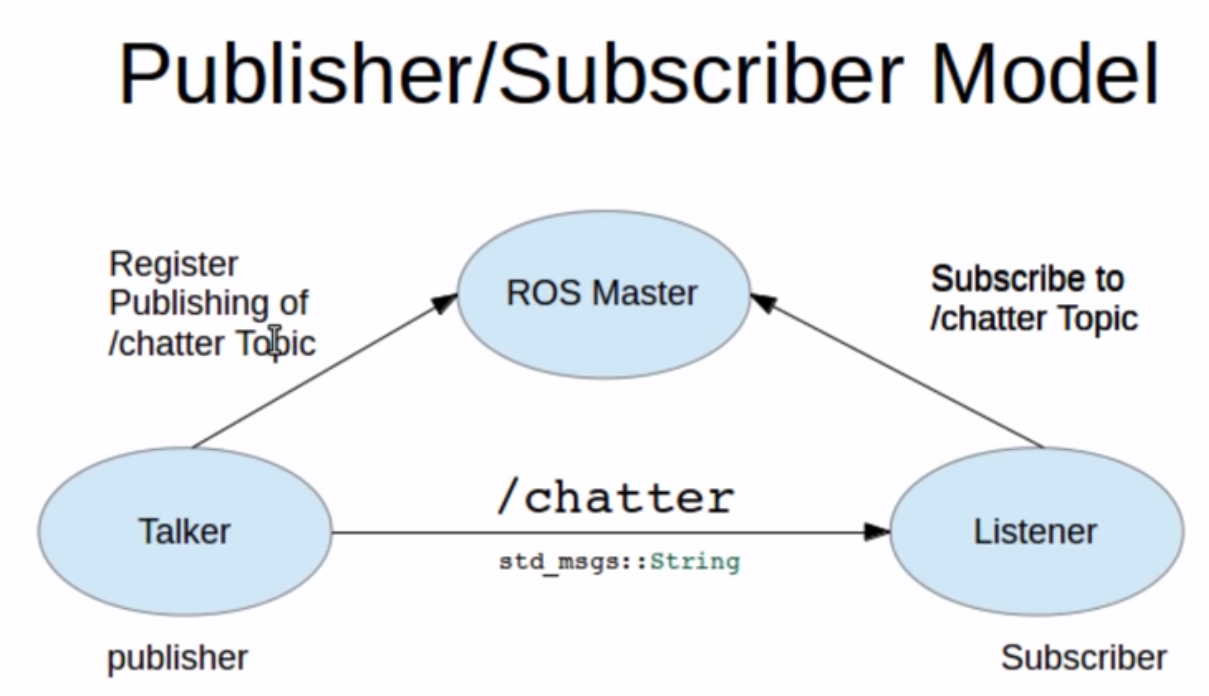
créé **include** car présence de roscpp dans les dépendances

 Si vous utilisez des programmes cpp, il faudra toujours compiler le code.

Suivre tuto #7 P1 à P2 de Shawn Chen

Exemple du package Hello\_world, en C++ : **avec deux fichiers exécutables séparés dans le src**

1. **Principe du projet :**



<https://github.com/methylDragon/coding-notes/blob/master/Robot%20Operating%20System%20(ROS)/ROS/02%20ROS%20-%20Writing%20Nodes%20(Pub-Sub).md#2.9>

Créer un work\_space portant le nom de TP2\_21.

 ne pas créer votre work space en **super user** 🡺 cela verrouille les fichiers

Créer le package suivant:

catkin\_create\_pkg TP2\_21 std\_msgs roscpp

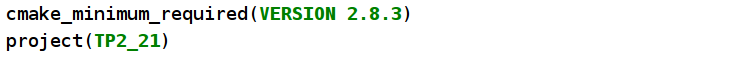
Dans le dossier src de package TP2\_21, créer les fichiers talker.cpp et listner.cpp avec les contenus de codes donnés dans les fichiers .cpp

 Etant donné la dépendance **roscpp** 🡺 il est nécessaire de compiler le projet et de bien configurer les fichiers **CMakeLists.txt** et **package.xml** pour renseigner **catkin** sur les fichiers à prendre en compte, pour une compilation sans problème.

1. **Préparation à la compilation**

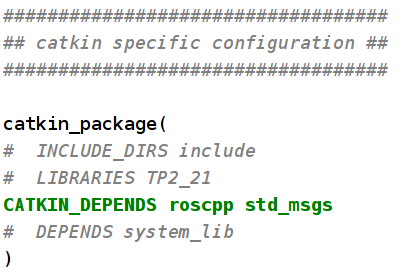
Dans le **CMakeLists.txt**, compléter le code avec le fichier donné

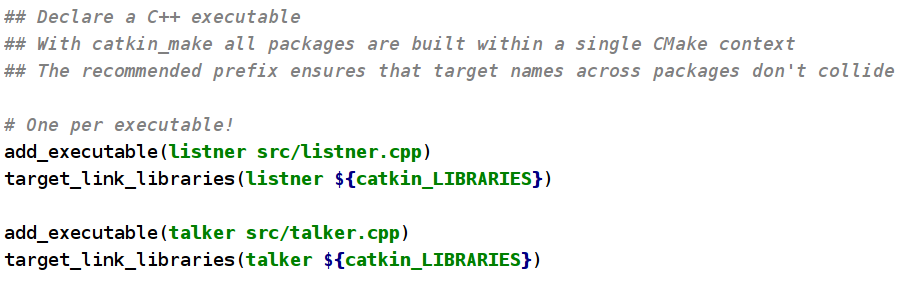
nom de package.





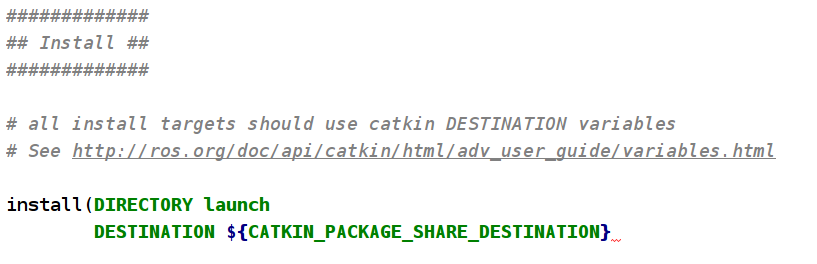
le liens aux dépendances.



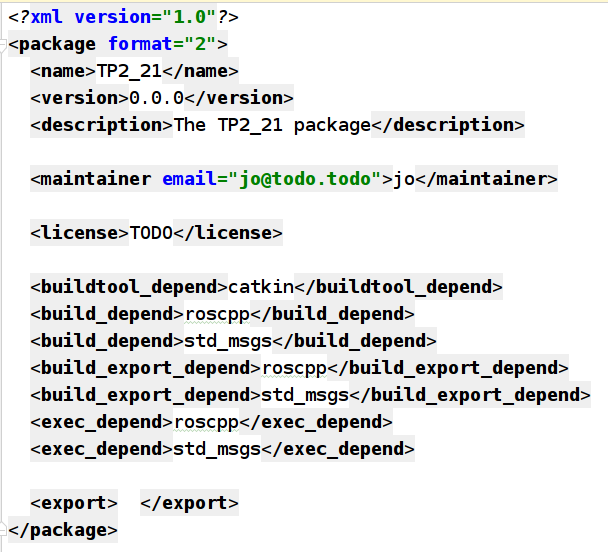


Les différents fichiers à compiler.

On peut rajouter des dépendances supplémentaires pour renforcer la robustesse (voir wikiROS).



Dans le fichier **package.xml,** copier le code si dessous donné dans le fichier



Les dépendances.

Des informations générales

Le nom du package

1. **Test du package**

Lancer la compilation sous le répertoire du work\_space (à la racine) :

catkin\_make

La compilation se fait et doit conduire à zéro erreur.

Ouvrir un autre terminal et lancer **roscore**

roscore

Revenir dans la console de la work\_space sous la racine  (pas nécecessaire si intégré au .bashrc)

source devel/setup.bash

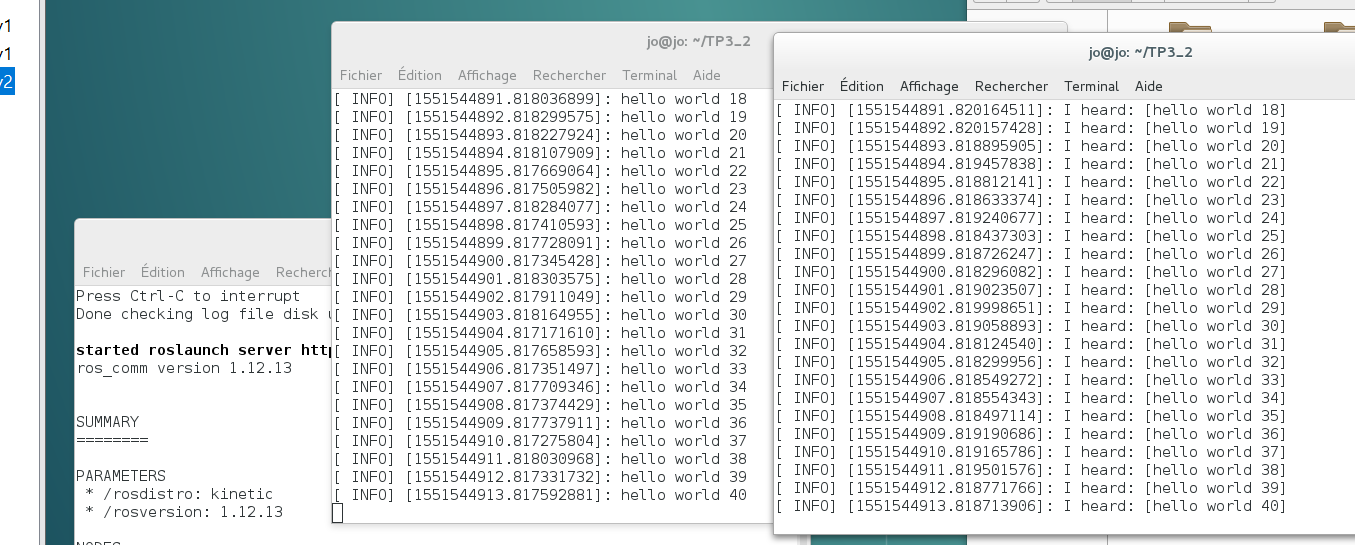
Cela donne accès à l’environnement du work\_space :

Dans une autre console, lancer depuis la racine du work\_space l’exécutable talker du package package\_1  :

rosrun TP2\_21 talker

Dans une autre console, lancer depuis la racine du work\_space l’exécutable listner du package package\_1 :

rosrun TP2\_21 listner



talker

listner

Roscore

On voit que les programme fonctionnent bien.

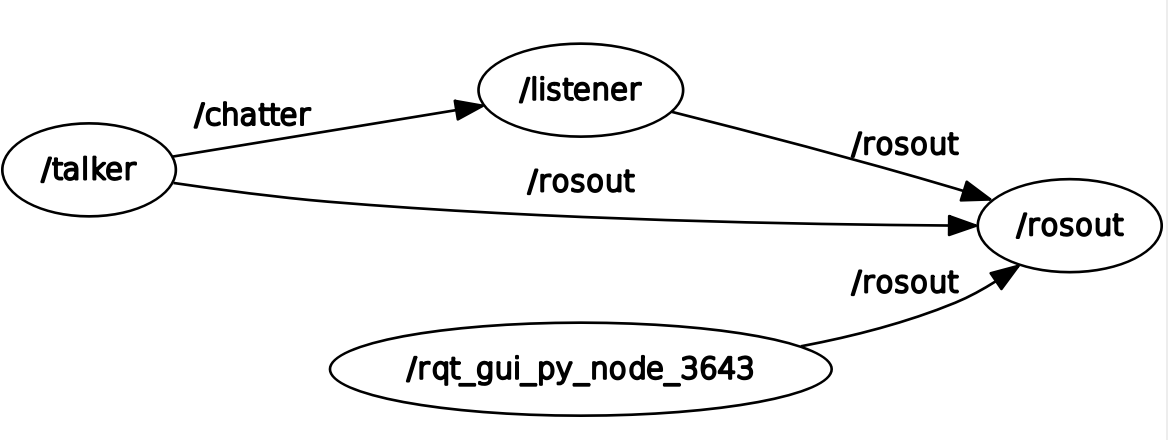
1. **Autres spécificités intéressantes**

Lancer dans une autre console :

rqt\_graph

Cela donne le graphe ci-dessous qui est conforme au principe exposé au départ.

Ici, ROSMaster publie à travers cette node le topic chatter

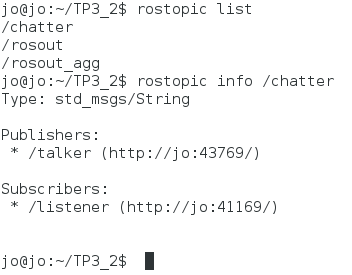


en utilisant les commandes de rostopic on a :

Nous donne le type du topic chatter

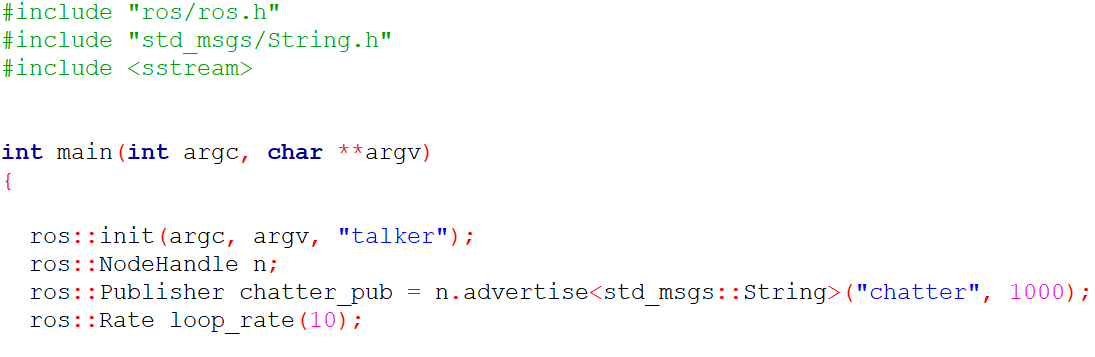
Donc le topic chatter est de type std\_msg, et la data est une string

et nous informe des adresses utilisées sur le réseau.



**Explications des lignes de codes pour le talker.**

Les classiques includes qui permettent de travailler avec des strings et des flux

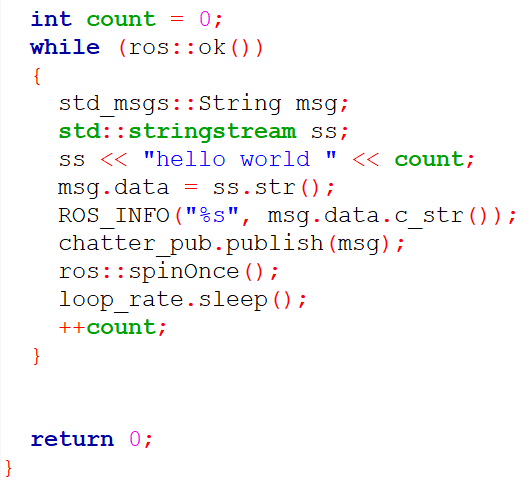


Déclaration de l'objet chatter\_pub pour notre topic chatter qui va publier un message string de type std\_msg au travers la méthode advertise (publier) pour l’objet **n**

Déclaration de **n** qui est l’objet nodeHandle qui va transporter les informations dans le programme

Déclaration de la **node** talker dans le réseau ROS

Déclaration d’un taux de 10 messages par seconde.



publication du message

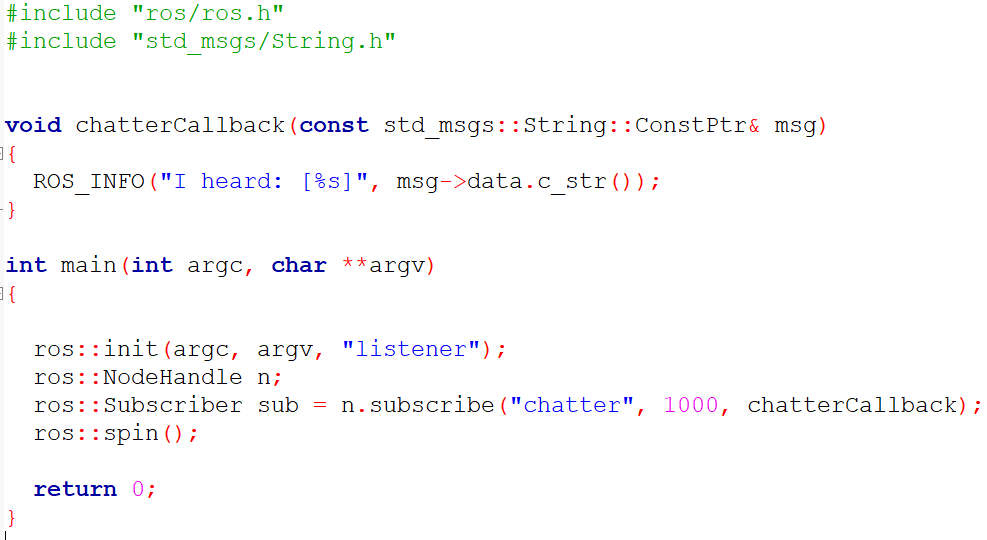
Déclaration de **msg** (pour le programme) = objet String de type **std\_msgs** , qui est impérativement de même type que le paramètre à faire passer dans le topic chatter et **ss** qui est une **stringstream** (un flux de strings)

ROS\_INFO 🡺 information pour la console pour le debuguage.

Récupère la string "hello world" concaténé avec count depuis l'objet ss et l'affecte à data de msg  data, qui est le paramètre à déposer plus tard pour notre topic chatter.

Tous les callbacks sont scrutés une fois

**Explications des lignes de codes pour le listner.**



tous les callbacks sont scrutés de manière infinie

Ici, on déclare une node listener et un objet NodeHandle n qui souscrit à travers sub au topic chatter, et à chaque message reçu, on pointe vers un callback qui s'occupe de renvoyer le message dans la console.