

Notions de workspace catkin, de paquets, compilation

Catkin (<http://wiki.ros.org/catkin>) est l'outil de compilation système de ROS, il permet de compiler tout type de paquets ROS.

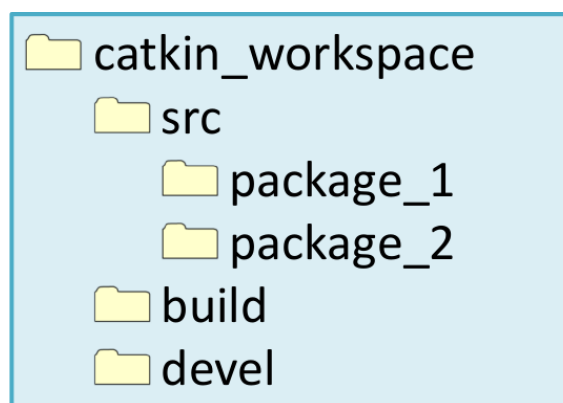
- est basé sur l'outil CMAKE
- est multiplate-forme (Ubuntu , Windows, linux embarqué)
- remplace l'ancien outil de compilation 'roscpp'

Catkin est par défaut inclus dans l'installation de ROS. Mais il peut aussi être installé séparément par les paquets ubuntu via la commande 'apt-get' :

```
sudo apt-get install ros-indigo-ros
```

Catkin nécessite une structure de répertoire particulière :

- chaque projet catkin dispose de son propre 'catkin workspace'
- dans un 'catkin workspace' il peut y avoir plusieurs paquets qui constituent les parties/modules du projet



- Une fois la compilation du 'catkin workspace' terminée, les fichiers temporaires compilés se trouveront dans le dossier 'build' et le résultat dans le dossier 'devel'. Ces dossiers sont automatiquement créés.

Créer un 'catkin workspace' nommé 'my_catkin_ws'

1- Ouvrir une fenêtre terminal et créer un répertoire 'my_catkin_ws' et son sous dossier 'src'.
`mkdir -p my_catkin_ws/src`

2- Initialiser le workspace en lançant la commande 'catkin_init_workspace' dans le dossier 'src'.
`cd my_catkin_ws/src`
`catkin_init_workspace`

Ceci va créer un fichier 'CMakeLists.txt' qui est en fait un lien vers le fichier 'cmake' ROS
'/opt/ros/indigo/share/catkin/cmake/toplevel.cmake'
Ce fichier contenant toute l'architecture système nécessaire à la compilation du 'catkin workspace'.

3- Le workspace est vide, il n'y a pas encore de paquets dedans, mais on peut quand même lancer un ordre de compilation en utilisant l'outil spécifique 'catkin_make' à la racine du workspace.

```
cd ..  
catkin_make
```

Une fois la compilation terminée on peut observer que les dossiers 'build' et 'devel' ont été créés.

4- Rendre visible au système ROS le nouveau workspace créé en lançant le fichier bash 'setup.bash' qui se trouve dans le dossier 'devel'. Cela positionnera dans votre fenêtre terminal des variables d'environnement nécessaires à ROS, dont la variable 'ROS_PACKAGE_PATH'.

```
env | grep ROS_PACKAGE_PATH  
ROS_PACKAGE_PATH=/opt/ros/indigo/share:/opt/ros/indigo/stacks  
  
source devel/setup.bash  
  
env | grep ROS_PACKAGE_PATH  
ROS_PACKAGE_PATH=../../my_catkin_ws/src:/opt/ros/indigo/share:/opt/ros/indigo/stacks
```

Créer un paquet ROS dans le workspace

Un paquet ROS est en fait un composant ROS pouvant se compiler et s'exécuter séparément. Il doit être forcément créé dans un 'catkin workspace'. Physiquement parlant, un paquet est un répertoire dans lequel il faut au minimum 2 fichiers, un 'CMakeLists.txt' pour les ordres de compilation, un fichier 'package.xml' contenant les 'metadata' du paquet (nom, auteur, description, dépendances avec d'autres paquets).

1- Créer un paquet ROS nommé 'my_first_package' en utilisant la commande 'catkin-create-pkg' dans le dossier 'src' du workspace.

```
cd my_catkin_ws/src  
catkin-create-pkg my_first_package roscpp
```

Ici on a ajouté un 3ième paramètre à la commande, permettant de spécifier une dépendance au paquet '**roscpp**' (<http://wiki.ros.org/roscpp>) du paquet créé. Ce paquet permettant de s'interfacer en C++ avec des outils ROS comme des 'topics', 'services', 'paramètres' que l'on étudiera plus tard.

```
ls  
CMakeLists.txt my_first_package  
cd my_first_package  
ls  
CMakeLists.txt include package.xml src
```

Le nouveau paquet créé contient les fichiers 'CMakeLists.txt' et 'package.xml' mais aussi deux dossiers 'src' et 'include' dans lesquels on va mettre le code source C++.

2- Ouvrir avec votre éditeur favori le fichier 'package.xml' afin de compléter la partie 'auteur –

description – licence'

```
1 <?xml version="1.0"?>
2 <package>
3   <name>my_first_package</name>
4   <version>0.0.0</version>
5   <description>The my_first_package package</description>
6
7   <!-- One maintainer tag required, multiple allowed, one person per tag -->
8   <!-- Example: -->
9   <!-- <maintainer email="jane.doe@example.com">Jane Doe</maintainer> -->
10  <maintainer email="lequievre.lequievre@univ-bpclermont.fr">Laurent LEQUIEVRE</maintainer>
11
12
13  <!-- One license tag required, multiple allowed, one license per tag -->
14  <!-- Commonly used license strings: -->
15  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1, LGPLv3 -->
16  <license>BSD</license>
17
```

3- Observer la deuxième partie du fichier 'package.xml' qui contient les informations :

```
43 <buildtool_depend>catkin</buildtool_depend>
44 <build_depend>roscpp</build_depend>
45 <run_depend>roscpp</run_depend>
46
47
48 <!-- The export tag contains other, unspecified, tags -->
49 <export>
50   <!-- Other tools can request additional information be placed here -->
51
52 </export>
53 </package>
```

La dépendance au paquet 'roscpp' pour la compilation du paquet 'my_first_package' s'est traduit par l'ajout d'une balise XML '<build_depend>'. De même la balise '<run_depend>' exprime une dépendance pour l'exécution. Enfin la balise '<buildtool_depend>' permet de spécifier le type d'outil pour compiler le paquet, ici 'catkin'.

4- Observer le début du fichier 'CMakeLists.txt' du paquet créé :

```
1 cmake_minimum_required(VERSION 2.8.3)
2 project(my_first_package)
3
4 ## Find catkin macros and libraries
5 ## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS xyz)
6 ## is used, also find other catkin packages
7 find_package(catkin REQUIRED COMPONENTS
8   roscpp
9 )
10
```

La macro 'find_package' permet de définir et de trouver les paquets ROS nécessaires à la compilation, ici 'roscpp'. Si le paquet est trouvé, des variables d'environnement cmake seront positionnées et pourront être utiles dans la suite du fichier cmake :

<NAME>_FOUND (NAME = nom du paquet) si le paquet est trouvé
<NAME>_INCLUDE_DIRS indique le dossier où se trouve le paquet

<NAME>_LIBRARIES indique le nom et dossier où se trouvent les bibliothèques du paquet

5- Modifier le fichier 'CMakeLists.txt' en ajoutant après 'find_package' le code cmake ci-dessous :

```
if(${roscpp_FOUND})
    message(STATUS "The package roscpp is found !")
    message(STATUS "include dir = ${roscpp_INCLUDE_DIRS}")
    message(STATUS "libraries = ${roscpp_LIBRARIES}")
else(${roscpp_FOUND})
    message(STATUS "The package roscpp is not found !")
endif(${roscpp_FOUND})
```

6- Observer le reste du fichier 'CMakeLists.txt' du paquet créé :

```
107 #####
108 ## Build ##
109 #####
110
111 ## Specify additional locations of header files
112 ## Your package locations should be listed before other locations
113 # include_directories(include)
114 include_directories(
115     ${catkin_INCLUDE_DIRS}
116 )
117
```

Ici est indiqué via la macro 'include_directories' d'inclure dans la compilation du paquet les dossiers 'include' de ROS catkin :

catkin_INCLUDE_DIRS = /opt/ros/indigo/include;/usr/include

7- Compiler le workspace et observer l'affichage des 'message cmake'.

```
cd my_catkin_ws
catkin_make
```

```
-- ~~~~ traversing 1 packages in topological order:
-- ~~~~ - my_first_package
-- ~~~~
-- +++ processing catkin package: 'my_first_package'
-- ==> add_subdirectory(my_first_package)
-- The package roscpp is found !
-- include dir = /opt/ros/indigo/include;/usr/include
-- libraries = /opt/ros/indigo/lib/libroscpp.so;/usr/lib/x86_64-linux-gnu/libbo
```

Il est à noter que 'catkin' indique les paquets à compiler du workspace et lance leur compilation.

8- Utiliser 'roscd' pour changer de dossier.

'roscd' fait partie de 'rosbash' (<http://wiki.ros.org/rosbash>) contenant les outils nécessaires pour utiliser ros sous 'bash'.

'roscd' suivi du nom du paquet permet de se positionner directement dans le dossier du paquet.

```
cd my_catkin_ws
roscd my_first_package
.../my_catkin_ws/src/my_first_package$
```

9- Utiliser 'rospack' (<http://wiki.ros.org/rospack>) pour obtenir des infos sur un paquet

'rospack' fait partie des 'ROS package management tool' et peut être utilisé avec différents paramètres.

→ trouver le chemin absolu où se trouve le paquet :

```
rospack find my_first_package
.../my_catkin_ws/src/my_first_package
```

Pour faire cela 'rospack' recherche dans le dossier contenu dans la variable d'environnement 'ROS_ROOT' mais aussi dans tous les dossiers contenus dans la variable 'ROS_PACKAGE_PATH'.

→ afficher les dépendances (paquets) d'un paquet :

```
rospack depends my_first_package
cpp_common
...
roscpp
```

→ afficher tous les paquets installés dans le système

```
rospack list
```