ROS Cheat Sheet

3iL INGENIEURS
</les sciences informatiques>

Gestion des packages

catkin_create_pkg nom_pkg Permet de créer et préconfigurer le package **nom_pkg**. Commande à exécuter dans le répertoire src du workspace.
catkin_make Compile les noeuds de tous les packages du workspace. Commande à exécuter à la racine du workspace.
roscd nom_pkg Se rend dans le package **nom_pkg**.

Gestion des noeuds

roscore Lance les éléments nécessaires à tout écosystème ROS. **rosrun nom_pkg nom_noeud** Lance le noeud **nom_noeud** du package **nom_pkg**.
rosnode list Dresse la liste des noeuds existants.
rosnode kill Tue le noeud spécifié. Le paramètre -a permet de tuer tous les noeuds lancés.
rosnode ping Teste la connectivité au noeud spécifié. Le paramètre –all permet de tester la connectivité à chaque noeud lancé.
roslaunch fichier.launch Exécute le fichier launch **fichier.launch** se trouvant dans le répertoire courant.
roslaunch nom_pkg fichier.launch Exécute le fichier launch **fichier.launch** se trouvant dans le répertoire **launch** du package **nom_pkg**.

Gestion des topics

rostopic list Affiche la liste des topics existants dans l'écosystème ROS.
rostopic type nom_topic Précise le type de message transitant dans le topic **nom_topic**
rostopic hz nom_topic Affiche la fréquence d'envoi des messages sur le topic **nom_topic**.
rostopic bw nom_topic Affiche la bande passante consommée par le topic **nom_topic**.
rostopic echo nom_topic Affiche les messages transitant sur le topic **nom_topic**.
rostopic pub nom_topic ... Publie un message sur le topic **nom_topic**. Le message est spécifié en utilisant l'autocomplétion pour remplacer les **...** puis en complétant les différents champs.
rosmsg show nom_msg Affiche la structure du message **nom_msg**.

Gestion des services

rosservice list Affiche la liste des services disponibles dans l'écosystème ROS.
rosservice args nom_srv Affiche les paramètres attendus pour appeler le service **nom_srv**.
rosservice call nom_srv ... Entraine l'exécution du service **nom_srv**. Les paramètres sont spécifiés en utilisant l'autocomplétion pour remplacer les **...** puis en affectant les valeurs souhaitées.

Gestion des paramètres

rosparam list Affiche la liste de tous les paramètres existant dans l'écosystème ROS.
rosparam list /ns Affiche la liste de tous les paramètres existant dans l'écosystème ROS et appartenant au namespace **ns**.
rosparam delete param Supprime le paramètre **param** de l'écosystèe ROS.
rosparam set param valeur Affecte au paramètre **param** la valeur **valeur**.
rosparam get param Affiche la valeur du paramètre **param**.
rosparam dump fichier.yaml Enregistre tous les paramètres existants et leurs valeurs dans le fichier **fichier.yaml**.
rosparam dump fichier.yaml /ns Enregistre tous les paramètres du namespace **ns** dans le fichier **fichier.yaml**
rosparam load fichier.yaml Charge les paramètres stockés dans **fichier.yaml** dans l'écosystème ROS.

Visualisation / Déboggage

rviz Visualisation 3D des topics compatibles.
rqt IHM formée de plugins, configurable et sauvegardable.
rqt_graph Visualisation sous forme de graphe des connexions topics / noeuds de l'écosystème ROS.
rqt_console Console permettant l'affichage des logs en fonction de leur degré d'importance (**DEBUG,INFO,WARN et ERROR**.
rqt_plot Permet de tracer les courbes d'évolution des valeurs numériques transitant par les topics.

Roscpp

#include <ros/ros.h> Include indispensable pour utiliser les fonctionnalités ROS dans un fichier .cpp
ros::init(argc,argv,"nom_noeud") Initialise le noeud en lui affectant un nom (qui doit être unique), et en redirigeant les paramètres **argc** et **argv** pour éventuelle utilisation.
ros::NodeHandle nh Lance le noeud (appel caché de ros::start()) et lui donne une référence pour la gestion des topics.
ros::Rate loop_rate(freq) Initie un cadenceur garantissant l'exécution de la boucle infinie du noeud à une fréquence de **freq**Hz
nh.advertise<type>("tpc",queue) Initialise un publisher envoyant des messages de type **type** sur le topic **tpc**. **queue** désigne le nombre de messages pouvant être envoyés simultanément.
nh.subscribe("tpc", queue, &callback) Initialise un subscriber recevant des messages en provenance du topic **tpc**, et exécutant la fonction callback à chaque fois que c'est le cas. **queue** désigne le nombre de messages pouvant être reçus simultanément.
while(ros::ok(){...} Boucle infinie qui définit le coeur d'exécution du noeud.
ros::spinOnce() Interroge le master et déclenche l'exécution des callbacks lorsque des messages sont arrivés sur les topics souscrits.

CMakeLists.txt

find_package(catkin REQUIRED COMPONENTS roscpp) Indique que le package utilise la librairie **roscpp**.
include_directories(rep1 rep2) Précise où trouver les fichiers headers (.h) utilisés par les codes sources. Par exemple, **${catkin_INCLUDE_DIRS}** désigne où sont stockés tous les fichiers .h de ROS.
add_executable(nom_noeud fichier1.cpp) Ajoute le noeud **nom_noeud** au package, compilé à partir du code source stocké dans **fichier1.cpp**.
target_link_libraries(nom_noeud lib) Précise que le noeud **nom_noeud** utilise la librairie **lib**. Par exemple, **${catkin_LIBRARIES}** correspond à la librairie principale de ROS.