

Note: This tutorial assumes that you have completed the previous tutorials: Construire un package ROS (/fr/ROS/Tutorials/BuildingPackages).

💡 Please ask about problems and questions regarding this tutorial on answers.ros.org (<http://answers.ros.org>). Don't forget to include in your question the link to this page, the versions of your OS & ROS, and also add appropriate tags.

Comprendre les 'nodes' ROS

Description: Ce tutoriel introduit les concepts de ROS graph et l'utilisation des outils en ligne de commande roscore (/roscore), rosnode (/rosnode), et rosrun (/rosrn).

Tutorial Level: BEGINNER

Next Tutorial: Comprendre les topics ROS (/fr/ROS/Tutorials/UnderstandingTopics)

Sommaire

1. Prérequis
2. Revue rapide des concepts de Graph sous ROS
3. Nodes
4. Bibliothèques Client
5. roscore
6. Utiliser rosnode
7. Utilisation de rosrn
8. Compte rendu

0.1 Prérequis

Pour ce tutoriel, nous allons devoir utiliser un simulateur 'léger', veuillez l'installer avec la commande suivante:

```
$ sudo apt-get install ros-<distro>-ros-tutorials
```

Remplacer '<distro>' par le nom de votre distribution ROS (e.g. hydro, groovy, electric, fuerte etc.)

0.1 Revue rapide des concepts de Graph sous ROS

- Nodes (/Nodes): Un node est un exécutable qui utilise ROS afin de communiquer avec d'autres nodes.
- Messages (/Messages): Un type de donnée ROS, utilisé lors de la souscription ou publication vers un topic.

- Topics (/Topics): Les Nodes peuvent *publier des* messages vers un topic mais également *souscrire* à un topic afin de recevoir les messages émis par celui-ci.
- Master (/Master): Name service for ROS (i.e. helps nodes find each other)
- rosout (/rosout): l'équivalent sous ROS des stdout/stderr
- roscore (/roscore): Master + rosout + parameter server (parameter server will be introduced later)

0.1 Nodes

Un node n'est rien de plus qu'un exécutable dans un paquet ROS. Les nodes ROS utilisent une bibliothèque 'client' ROS afin de communiquer avec les autres nodes. Un node peut publier ou souscrire à un Topic. Un node peut également fournir ou utiliser un Service.

0.1 Bibliothèques Client

Les bibliothèques client ROS permettent à des nodes écrits dans des langages de programmation différents, de communiquer:

- rospy = bibliothèque client python
- roscpp = bibliothèque client c++

0.1 roscore

roscore est la première chose que vous devriez lancer lorsque vous désirez utiliser ROS.

Exécutez:

```
$ roscore
```

Vous devriez voir apparaître quelque chose similaire à ceci:

```

... logging to ~/.ros/log/9cf88ce4-b14d-11df-8a75-00251148e8cf/roslaunch-machine_name-13039.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://machine_name:33919/
ros_comm version 1.4.7

SUMMARY
=====

PARAMETERS
* /rosversion
* /rostdistro

NODES

auto-starting new master
process[master]: started with pid [13054]
ROS_MASTER_URI=http://machine_name:11311/

setting /run_id to 9cf88ce4-b14d-11df-8a75-00251148e8cf
process[rosout-1]: started with pid [13067]
started core service [/rosout]

```

Si roscore ne s'initialise pas, vous avez sans doute un problème de configuration réseau: 🌐

Network Setup - Single Machine Configuration

(http://www.ros.org/wiki/ROS/NetworkSetup#Single_machine_configuration)

Si roscore ne s'initialise pas et renvoie un message à propos de permissions insuffisantes, alors le répertoire ~/.ros appartient probablement à root.

Vous pouvez modifier les permissions/propriétés de ce répertoire avec la commande suivante:

```
$ sudo chown -R <your_username> ~/.ros
```

0.1 Utiliser rosnodet

Ouvrons une nouvelle fenêtre **terminal**, et commençons à utiliser **rosnodet** pour visualiser ce que fait roscore...

Note: lorsque vous ouvrez un nouveau terminal, vos paramètres d'environnement sont réinitialisés et le fichier ~/.bashrc est exécuté avec la commande 'source'. Si vous rencontrez des problèmes lors de l'utilisation de commandes telles que 'rosnodet', vous devrez probablement ajouter quelques

variables d'environnement supplémentaires dans votre fichier '~/.bashrc' ou 'sourcer' celles-ci manuellement (via la commande `source` suivi du fichier de configuration, généralement un fichier 'setup.*sh')

la commande `roscat` affiche les informations à propos des nodes ROS en cours de fonctionnement. la commande `roscat list` affiche ces nodes actifs:

```
$ roscat list
```

Vous devriez obtenir le résultat suivant (au minimum):

```
/rosout
```

Cela signifie qu'il y a actuellement un seul node actif: `roscat` (/rosout). Ce node est en permanence actif et collecte et enregistre les sorties de débogage des autres nodes.

La commande `roscat info` renvoie les informations concernant le node indiqué en paramètre.

```
$ roscat info /rosout
```

La commande ci-dessus nous donne quelques informations supplémentaires à propos du node `roscat`, telles que le fait qu'il 'publie' /rosout_agg.

```
-----
--
Node [/rosout]
Publications:
  * /rosout_agg [roscat_msgs/Log]

Subscriptions:
  * /rosout [unknown type]

Services:
  * /rosout/set_logger_level
  * /rosout/get_loggers

contacting node http://machine_name:54614/ ...
Pid: 5092
```

Voyons quelques autres nodes. Nous allons pour cela utiliser la commande `roslaunch` pour lancer d'autres nodes.

0.1 Utilisation de `roslaunch`

`roslaunch` va vous permettre d'utiliser le nom du package afin de lancer directement un node présent dans un package (sans qu'il soit nécessaire de connaître le chemin du package en question).

Utilisation:

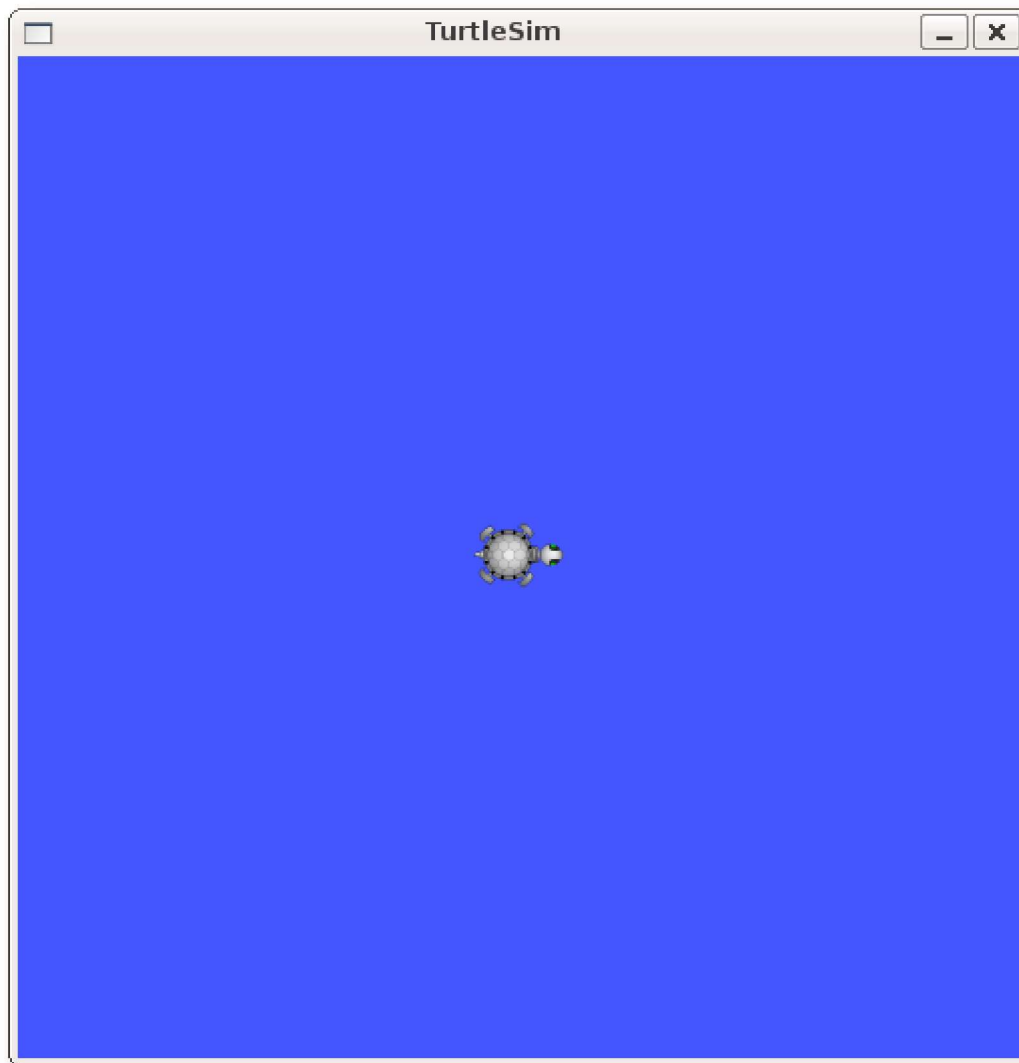
```
$ rosrun [package_name] [node_name]
```

Nous allons donc pouvoir lancer le node 'turtlesim_node' inclut dans le package turtlesim.

Dans un nouveau **terminal**:

```
$ rosrun turtlesim turtlesim_node
```

Vous devriez désormais voir apparaître à l'écran la fenêtre turtlesim:



NOTE: Votre tortue peut avoir un look différent dans votre propre fenêtre. Ne vous en faites pas, il y a beaucoup de races différentes de tortues ([/Distributions#Current_Distribution_Releases](#)), la vôtre sera une surprise!

Dans un nouveau **terminal**, entrez la commande suivante:

```
$ rosnode list
```

Vous devriez obtenir un résultat similaire à:

```
/rosout  
/turtlesim
```

L'une des caractéristiques importante de ROS est que vous pouvez réassignez les Noms via la ligne de commande.

Fermez la fenêtre turtlesim afin de stopper le node (ou retourner sur la fenêtre terminal où vous avez lancé la commande `roslaunch turtlesim`, et utilisez la commande `ctrl-C`). Relançons maintenant ce node, mais cette fois en utilisant le Remapping Argument (/Remapping%20Arguments) pour modifier le nom du node:

```
$ roslaunch turtlesim turtlesim_node __name:=my_turtle
```



Si nous revenons sur la commande `roslaunch`:

```
$ roslaunch
```

Vous devriez désormais voir quelque chose de similaire à ceci:

```
/rosout  
/my_turtle
```

Note: Si vous avez toujours le node `/turtlesim` dans la liste des nodes actifs, cela peut signifier que vous avez fermé le node via le terminal en utilisant la commande `ctrl-C` au lieu de fermer la fenêtre, ou que la variable d'environnement `$ROS_HOSTNAME` n'est pas définie, comme décrit ici: [Network Setup - Single Machine Configuration](http://wiki.ros.org/ROS/NetworkSetup#Single_machine_configuration) (http://www.ros.org/wiki/ROS/NetworkSetup#Single_machine_configuration). Vous pouvez tenter de nettoyer la liste des nodes actifs en utilisant la commande suivante: `$ roslaunch cleanup`

Nous avons donc notre node `/my_turtle`. Utilisons maintenant une nouvelle commande `roslaunch ping`, afin de vérifier que le node est réellement actif:

```
$ roslaunch ping my_turtle
```


```
roslaunch: node is [/my_turtle]  
pinging /my_turtle with a timeout of 3.0s  
xmlrpc reply from http://aqy:42235/      time=1.152992ms  
xmlrpc reply from http://aqy:42235/      time=1.120090ms  
xmlrpc reply from http://aqy:42235/      time=1.700878ms  
xmlrpc reply from http://aqy:42235/      time=1.127958ms
```

0.1 Compte rendu

Ce qui a été couvert dans ce tutoriel:

- roscore = ros+core : master (fourni un serveur de nom pour ROS) + rosout (stdout/stderr) + parameter server (le parameter server sera étudié dans un prochain tuto)
- rosnodetool = ros+node : outil ROS permettant d'obtenir des informations sur un node.
- roslaunch = ros+run : lance un node depuis un package.

Vous comprenez désormais comment fonctionnent les nodes sous ROS, voyons maintenant comment les topics ROS fonctionnent (/fr/ROS/Tutorials/UnderstandingTopics). Ne vous gênez pas pour presser Ctrl-C afin de stopper turtlesim_node.

Wiki: fr/ROS/Tutorials/UnderstandingNodes (dernière édition le 2017-06-08 07:40:52 par  ArnaudMarot (mailto:rikuo.net@gmail.com))

Except where otherwise noted, the ROS wiki is licensed under the Creative Commons Attribution 3.0 (<http://creativecommons.org/licenses/by/3.0/>)

Brought to you by:  Open Robotics

(<https://www.openrobotics.org/>)