

Note: This tutorial assumes that you have completed the previous tutorials: Installation et Configuration de Votre Environnement ROS (/fr/ROS/Tutorials/InstallingandConfiguringROSEnvironment).

💡 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.

Navigation dans le Système de fichier ROS

Description: Ce tutoriel introduit les concepts du système de fichiers ROS, et couvre l'usage des commandes de roscd, rosls, et rospack (/rospack)

Tutorial Level: BEGINNER

Next Tutorial: Creation d'un package ROS (/fr/ROS/Tutorials/CreatingPackage)

catkin

roscd

Sommaire

1. Aperçu rapide des concepts du système de fichiers
2. Outils système de fichiers
 1. Utiliser rospack
 2. Utiliser roscd
 1. Les sous-répertoires
 3. Cas particuliers de roscd
 1. roscd sans argument
 2. roscd log
 4. Utiliser rosls
 5. Tab Complétion
3. En résumé

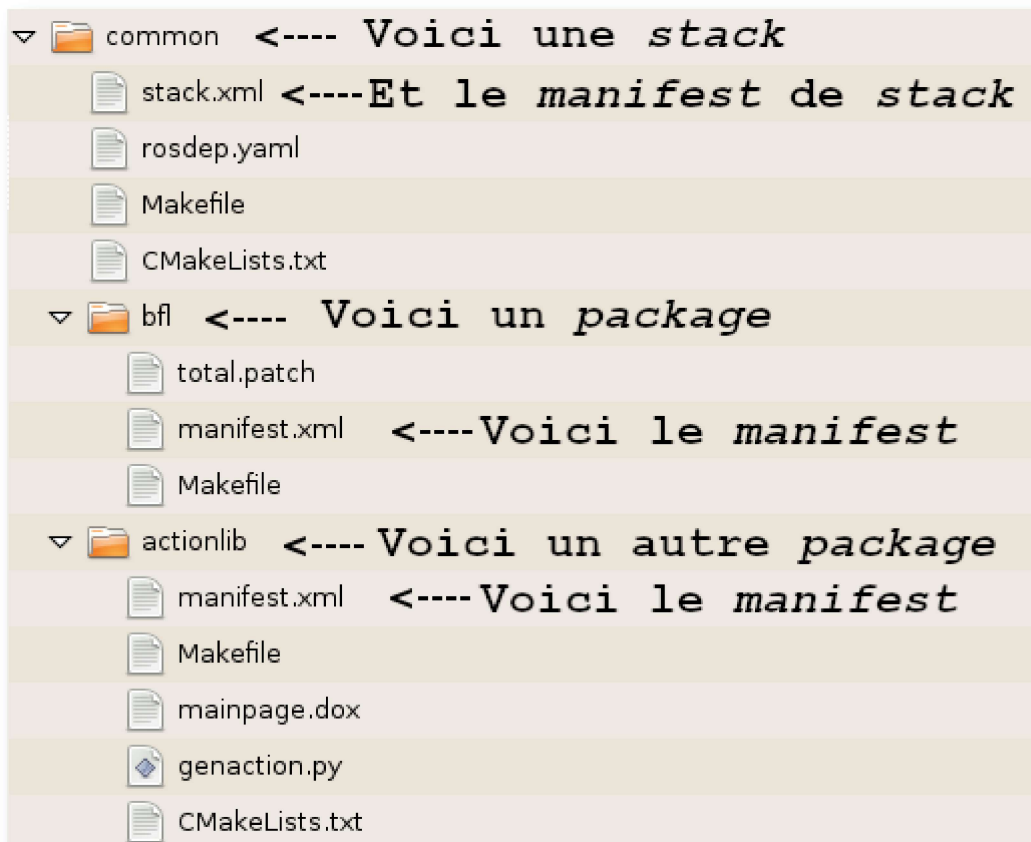
1. Aperçu rapide des concepts du système de fichiers

- **Package :** Le package (ou "paquet") est l'unité de base du code ROS. Chaque package peut contenir des bibliothèques, des exécutables, des scripts ou d'autres objets.

- **Manifest** (package.xml): Un manifeste est une description d'un package. Cela sert à définir les dépendances entre packages et à capturer des méta-informations sur le package comme la version, le gestionnaire, licence, etc ..
- **Stacks** : Une stack (ou "pile") est une collection de packages qui forment ainsi une librairie.
- **Stack Manifest** : Un manifest mais pour une stack 😊

Dans un système de fichiers, il est très facile de reconnaître un *package* ou une *stack*:



- Un package est constitué d'un répertoire avec un fichier manifest.xml.
- Une stack est constitué d'un répertoire avec un fichier stack.xml.



2. Outils système de fichiers

Le code est réparti entre de nombreux packages ROS. La navigation à l'aide des outils en ligne de commande comme ls et cd peut être très fastidieux et c'est pourquoi ROS fournit des outils pour vous aider.

2.1 Utiliser rospack

 `rospack` (<http://wiki.ros.org/rospack>) et  `rostack` (<http://wiki.ros.org/rostack>) vous permet d'obtenir des informations sur les packages et les stacks. Dans ce tutoriel, nous allons seulement parler de l'option `find`, qui renvoie le chemin du package. Utilisation:

```
# rospack find [package_name]
# rostack find [stack_name]
```

Exemple:

```
rospack find roscpp
```


Devrait vous renvoyer :

```
LE_CHEMIN_DE_VOTRE_INSTALLATION/share/roscpp
```

Si vous avez installé ROS Groovy avec `apt` sur Ubuntu Linux, vous verriez exactement ceci:

```
/opt/ros/groovy/share/roscpp
```

2.2 Utiliser `roscd`

`roscd` fait partie de la suite  `roscd` (<http://wiki.ros.org/roscd>) Il vous permet de changer de répertoire (`cd`) pour se placer directement dans le répertoire d'un package ou d'une stack.

Utilisation:

```
#roscd [locationname[/subdir]]
```

Exécuter cet exemple:


```
roscd roscpp
```

Pour vérifier que nous pointons maintenant sur le répertoire du package `roscpp`, nous allons afficher le répertoire dans lequel nous sommes en utilisant la commande Unix `pwd` :

```
pwd
```

Vous devriez avoir:

```
LE_CHEMIN_DE_VOTRE_INSTALLATION/share/roscpp
```

Vous pouvez voir que `LE_CHEMIN_DE_VOTRE_INSTALLATION/share/roscpp` est le même chemin que celui trouvé par `rospack` dans l'exemple précédent. Notez que  `ROS_PACKAGE_PATH` (http://wiki.ros.org/ROS/EnvironmentVariables#ROS_PACKAGE_PATH) tapez:

```
echo $ROS_PACKAGE_PATH
```

Votre `ROS_PACKAGE_PATH` doit contenir une liste de répertoires où vous avez des packages ROS, séparés par des caractères `:`. Un `ROS_PACKAGE_PATH` typique pourrait ressembler à ceci:

```
/opt/ros/hydro/base/install/share:/opt/ros/hydro/base/install/stacks
```

De même que pour d'autres variables d'environnement de type `PATH`, vous pouvez ajouter des répertoires supplémentaires à votre `ROS_PACKAGE_PATH`, chaque chemin séparés par deux points `«:»`.

2.2.1 Les sous-répertoires

`roscd` peut également vous placer dans le sous-répertoire d'un package ou d'une stack. Essayez:

```
roscd roscpp/cmake
pwd
```

Vous devriez voir:

```
LE_CHEMIN_DE_VOTRE_INSTALLATION/share/roscpp/cmake
```

2.3 Cas particuliers de `roscd`

Outre les packages et les stacks, il y a d'autres emplacements spécifiques accessibles via `roscd`.

2.3.1 `roscd` sans argument

`roscd` sans argument vous placera directement dans `$ROS_WORKSPACE`. Essayez ceci:

```
$ roscd
$ pwd
```

Et vous devriez avoir:

```
/home/user/roscd_workspace
```

Note: Pour les versions antérieures à Fuerte, `roscd` vous menait à `$ROS_ROOT`.

2.3.2 `roscd log`

`roscd log` vous emmènera dans le dossier où ROS stocke les fichiers log. Notez que si vous n'avez pas encore exécuté de programmes ROS, cela va donner un message d'erreur indiquant que le répertoire n'existe pas encore.

Si vous avez déjà exécuté un programme ROS avant, essayez:

```
roscd log
```

2.4 Utiliser `rosls`

rosls fait partie de la suite  rosbash (<http://wiki.ros.org/rosbash>) Il vous permet d'exécuter **ls** directement dans un package par son nom plutôt que par un chemin absolu. Utilisation:

```
# rosls [package_name[/subdir]]
```

Exemple:

```
rosls roscpp_tutorials
```

devrait vous retourner :

```
cmake package.xml srv
```

2.5 Tab Complétion

Il peut être fastidieux de taper le nom complet d'un package. Dans l'exemple précédent, roscpp_tutorials est un nom assez long. Heureusement, certains outils ROS permettent la complétion avec la touche TAB.

Commencez par taper:

```
# roscd roscpp_tut<<< maintenant appuyez sur la touche TAB >>>
```

Après avoir appuyé sur la touche TAB, la ligne de commande doit remplir le reste:

```
roscd roscpp_tutorials/
```

Cela fonctionne parce que roscpp_tutorials est actuellement le seul package ROS qui commence par roscpp_tut . Maintenant, essayez de taper:

```
# roscd tur<<< maintenant appuyez sur la touche TAB >>>
```

Après avoir appuyé sur la touche TAB, la ligne de commande doit se remplir autant que possible:

```
roscd turtle
```

Toutefois, dans ce cas, il y a plusieurs packages qui commencent par turtle. Essayez de taper TAB une seconde fois. Ceci devrait afficher tous les packages ROS qui commencent par turtle :

```
turtle_actionlib/  turtlesim/          turtle_tf/
```

Sur la ligne de commande, vous devriez toujours avoir:

```
roscd turtle
```

Maintenant, tapez un s après turtle , puis appuyez sur TAB :

```
# roscd turtles<<< maintenant appuyez sur la touche TAB >>>
```

Comme il n'y a qu'un seul package qui commence par 'turtles', vous devriez voir:

```
roscd turtlesim/
```

3. En résumé

Vous avez sûrement repéré la convention de nommage des outils ROS:

- rospack = ros + pack (age)
- roscd = ros + cd
- rosls = ros + ls

Ce schéma de nommage est valable pour la plupart des outils ROS.

Maintenant que vous savez vous déplacer dans ROS, nous allons voir comment créer un package .

 créer un package (<http://wiki.ros.org/fr/ROS/Tutorials/CreatingPackage>).

Wiki: fr/ROS/Tutorials/NavigatingTheFilesystem (dernière édition le 2014-01-15 13:58:32 par  PascalRey (<mailto:psc.rey@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/>)