Note: This tutorial assumes that you have completed the previous tutorials: navigating the ROS filesystem (/ROS/Tutorials/NavigatingTheFilesystem).

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.

Creating a ROS Package

Description: This tutorial covers using roscreate-pkg (/roscreate) or catkin (/catkin) to create a new package, and rospack (/rospack) to list package dependencies.

Tutorial Level: BEGINNER

Next Tutorial: Building a ROS package (/ROS/Tutorials/BuildingPackages)

catkin | rosbuild

Contents

1. What makes up a catkin Package?

For a package to be considered a catkin package it must meet a few requirements:

- The package must contain a catkin compliant package.xml (/catkin/package.xml) file.
    - That package.xml file provides meta information about the package.
- The package must contain a CMakeLists.txt which uses catkin (/catkin/CMakeLists.txt).
    - If it is a catkin metapackage (/catkin/package.xml#Metapackages) it must have the relevant boilerplate CMakeLists.txt file.
- Each package must have its own folder
    - This means no nested packages nor multiple packages sharing the same directory.

The simplest possible package might have a structure which looks like this:

```
my_package/
  CMakeLists.txt
  package.xml
```

2. Packages in a catkin Workspace

The recommended method of working with catkin packages is using a catkin workspace (/catkin/workspaces), but you can also build catkin packages standalone. A trivial workspace might look like this:

```
workspace_folder/        -- WORKSPACE
  src/                   -- SOURCE SPACE
    CMakeLists.txt       -- 'Toplevel' CMake file, provided by catkin
    package_1/
      CMakeLists.txt     -- CMakeLists.txt file for package_1
      package.xml        -- Package manifest for package_1
    ...
    package_n/
      CMakeLists.txt     -- CMakeLists.txt file for package_n
      package.xml        -- Package manifest for package_n
```

Before continuing with this tutorial create an empty catkin workspace by following the Creating a workspace for catkin (/catkin/Tutorials/create_a_workspace) tutorial.

3. Creating a catkin Package

This tutorial will demonstrate how to use the catkin_create_pkg (/catkin/commands/catkin_create_pkg) script to create a new catkin package, and what you can do with it after it has been created.

First change to the source space directory of the catkin workspace you created in the Creating a Workspace for catkin tutorial (/catkin/Tutorials/create_a_workspace):

```
# You should have created this in the Creating a Workspace Tutorial
$ cd ~/catkin_ws/src
```

Now use the `catkin_create_pkg` script to create a new package called 'beginner_tutorials' which depends on std_msgs, roscpp, and rospy:

```
$ catkin_create_pkg beginner_tutorials std_msgs rospy roscpp
```

This will create a `beginner_tutorials` folder which contains a package.xml (/catkin/package.xml) and a CMakeLists.txt (/catkin/CMakeLists.txt), which have been partially filled out with the information you gave `catkin_create_pkg`.

`catkin_create_pkg` requires that you give it a `package_name` and optionally a list of dependencies on which that package depends:

```
# This is an example, do not try to run this
# catkin_create_pkg <package_name> [depend1] [depend2] [depend3]
```

`catkin_create_pkg` also has more advanced functionalities which are described in catkin/commands/catkin_create_pkg (/catkin/commands/catkin_create_pkg).

4. Building a catkin workspace and sourcing the setup file

Now you need to build the packages in the catkin workspace:

```
$ cd ~/catkin_ws
$ catkin_make
```

After the workspace has been built it has created a similar structure in the `devel` subfolder as you usually find under `/opt/ros/$ROSDISTRO_NAME`.

To add the workspace to your ROS environment you need to source the generated setup file:

```
$ . ~/catkin_ws/devel/setup.bash
```

5. package dependencies

5.1 First-order dependencies

When using catkin_create_pkg (/catkin/commands/catkin_create_pkg) earlier, a few package dependencies were provided. These **first-order** dependencies can now be reviewed with the `rospack` tool.

```
$ rospack depends1 beginner_tutorials
```

```
roscpp
rospy
std_msgs
```

As you can see, `rospack` lists the same dependencies that were used as arguments when running `catkin_create_pkg`. These dependencies for a package are stored in the **package.xml** file:

```
$ roscd beginner_tutorials
$ cat package.xml
```

```
<package format="2">
...
  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>rospy</build_depend>
  <build_depend>std_msgs</build_depend>
...
</package>
```

5.2 Indirect dependencies

In many cases, a dependency will also have its own dependencies. For instance, `rospy` has other dependencies.

```
$ rospack depends1 rospy
```

```
genpy
roscpp
rosgraph
rosgraph_msgs
roslib
std_msgs
```

A package can have quite a few indirect dependencies. Luckily `rospack` can recursively determine all nested dependencies.

```
$ rospack depends beginner_tutorials
cpp_common
rostime
roscpp_traits
roscpp_serialization
catkin
genmsg
genpy
message_runtime
gencpp
geneus
gennodejs
genlisp
message_generation
rosbuild
rosconsole
std_msgs
rosgraph_msgs
xmlrpcpp
roscpp
rosgraph
ros_environment
rospack
roslib
rospy
```

6. Customizing Your Package

This part of the tutorial will look at each file generated by catkin_create_pkg (/catkin/commands/catkin_create_pkg) and describe, line by line, each component of those files and how you can customize them for your package.

6.1 Customizing the package.xml

The generated package.xml (/catkin/package.xml) should be in your new package. Now lets go through the new package.xml (/catkin/package.xml) and touch up any elements that need your attention.

6.1.1 description tag

First update the description tag:

Toggle line numbers

```
<description>The beginner_tutorials package</description>
```

Change the description to anything you like, but by convention the first sentence should be short while covering the scope of the package. If it is hard to describe the package in a single sentence then it might need to be broken up.

6.1.2 maintainer tags

Next comes the maintainer tag:

Toggle line numbers

```
<!-- One maintainer tag required, multiple allowed, one person per tag -->
<!-- Example:  -->
<!-- <maintainer email="jane.doe@example.com">Jane Doe</maintainer> -->
<maintainer email="user@todo.todo">user</maintainer>
```

This is a required and important tag for the package.xml (/catkin/package.xml) because it lets others know who to contact about the package. At least one maintainer is required, but you can have many if you like. The name of the maintainer goes into the body of the tag, but there is also an email attribute that should be filled out:

Toggle line numbers

```
<maintainer email="you@yourdomain.tld">Your Name</maintainer>
```

6.1.3 license tags

Next is the license tag, which is also required:

Toggle line numbers

```
<!-- One license tag required, multiple allowed, one license per tag -->
<!-- Commonly used license strings: -->
<!--   BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1, LGPLv3 -->
<license>TODO</license>
```

You should choose a license and fill it in here. Some common open source licenses are BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1, and LGPLv3. You can read about several of these at the Open Source Initiative (http://opensource.org/licenses/alphabetical). For this tutorial we'll use the BSD license because the rest of the core ROS components use it already:

Toggle line numbers

```
<license>BSD</license>
```

6.1.4 dependencies tags

The next set of tags describe the dependencies of your package. The dependencies are split into `build_depend`, `buildtool_depend`, `exec_depend`, `test_depend`. For a more detailed explanation of these tags see the documentation about Catkin Dependencies (/catkin/package.xml#Build.2C_Run.2C_and_Test_Dependencies). Since we passed `std_msgs`, `roscpp`, and `rospy` as arguments to catkin_create_pkg (/catkin/commands/catkin_create_pkg), the dependencies will look like this:

Toggle line numbers

```
<!-- The *_depend tags are used to specify dependencies -->
<!-- Dependencies can be catkin packages or system dependencies -->
<!-- Examples: -->
<!-- Use build_depend for packages you need at compile time: -->
<!--   <build_depend>genmsg</build_depend> -->
<!-- Use buildtool_depend for build tool packages: -->
<!--   <buildtool_depend>catkin</buildtool_depend> -->
<!-- Use exec_depend for packages you need at runtime: -->
<!--   <exec_depend>python-yaml</exec_depend> -->
<!-- Use test_depend for packages you need only for testing: -->
<!--   <test_depend>gtest</test_depend> -->
<buildtool_depend>catkin</buildtool_depend>
<build_depend>roscpp</build_depend>
<build_depend>rospy</build_depend>
<build_depend>std_msgs</build_depend>
```

All of our listed dependencies have been added as a `build_depend` for us, in addition to the default `buildtool_depend` on catkin. In this case we want all of our specified dependencies to be available at build and run time, so we'll add a `exec_depend` tag for each of them as well:

Toggle line numbers

```
<buildtool_depend>catkin</buildtool_depend>

<build_depend>roscpp</build_depend>
<build_depend>rospy</build_depend>
<build_depend>std_msgs</build_depend>

<exec_depend>roscpp</exec_depend>
<exec_depend>rospy</exec_depend>
<exec_depend>std_msgs</exec_depend>
```

6.1.5 Final package.xml

As you can see the final package.xml (/catkin/package.xml), without comments and unused tags, is much more concise:

Toggle line numbers

```
<?xml version="1.0"?>
<package format="2">
  <name>beginner_tutorials</name>
  <version>0.1.0</version>
  <description>The beginner_tutorials package</description>

  <maintainer email="you@yourdomain.tld">Your Name</maintainer>
  <license>BSD</license>
  <url type="website">http://wiki.ros.org/beginner_tutorials</url>
  <author email="you@yourdomain.tld">Jane Doe</author>

  <buildtool_depend>catkin</buildtool_depend>

  <build_depend>roscpp</build_depend>
  <build_depend>rospy</build_depend>
  <build_depend>std_msgs</build_depend>

  <exec_depend>roscpp</exec_depend>
  <exec_depend>rospy</exec_depend>
  <exec_depend>std_msgs</exec_depend>

</package>
```

6.2 Customizing the CMakeLists.txt

Now that the package.xml (/catkin/package.xml), which contains meta information, has been tailored to your package, you are ready to move on in the tutorials. The CMakeLists.txt (/catkin/CMakeLists.txt) file created by catkin_create_pkg (/catkin/commands/catkin_create_pkg) will be covered in the later tutorials about building ROS code.
Now that you've made a new ROS package, let's build our ROS package (/ROS/Tutorials/BuildingPackages).

7. Video Demonstration

Watch the video below to have more explanation on Custom Workspace and Package Creation with step by step guide .