

Nota: Este tutorial pressupõe que você concluiu os tutoriais anteriores: criando um pacote ROS (/ROS/Tutorials/CreatingPackage) .

💡 Por favor, pergunte sobre problemas e perguntas relacionadas a este tutorial em [Answers.ros.org](http://answers.ros.org) (<http://answers.ros.org>) . Não se esqueça de incluir na sua pergunta o link para esta página, as versões do seu SO e ROS, e também adicionar as tags apropriadas.

Construindo um pacote ROS

Descrição: Este tutorial cobre o conjunto de ferramentas para construir um pacote.

Nível do Tutorial: INICIANTE

Próximo Tutorial: Compreendendo os nós ROS (/ROS/Tutorials/UnderstandingNodes)

gatinho

robuild

Conteúdo

1. Construindo Pacotes
 1. Usando catkin_make
 2. Construindo Seu Pacote

1. Construindo Pacotes

Contanto que todas as dependências do sistema do seu pacote estejam instaladas, agora podemos construir seu novo pacote.

Nota: Se você instalou o ROS usando o `apt` ou algum outro gerenciador de pacotes, você já deverá ter todas as suas dependências.

Antes de continuar, lembre-se de fornecer o arquivo de configuração do ambiente, caso ainda não o tenha feito. No Ubuntu seria algo assim:

```
# fonte /opt/ros/%YOUR_ROS_DISTRO%/setup.bash
$ source /opt/ros/kinetic/setup.bash # Para Kinetic, por exemplo
```

1.1 Usando catkin_make

`catkin_make` (/catkin/commands/catkin_make) é uma ferramenta de linha de comando que adiciona alguma conveniência ao fluxo de trabalho padrão do catkin. Você pode imaginar que `catkin_make` (/catkin/commands/catkin_make) combina as chamadas para `cmake` e `make` no fluxo de trabalho padrão do CMake.

Uso:

```
# Em um espaço de trabalho catkin
$ catkin_make [make_targets] [-DCMAKE_VARIABLES=...]
```

Para pessoas que não estão familiarizadas com o fluxo de trabalho padrão do CMake, ele se divide da seguinte forma:

Nota: Se você executar os comandos abaixo não funcionará, pois este é apenas um exemplo de como o CMake geralmente funciona.

```
# In a CMake project
$ mkdir build
$ cd build
$ cmake ..
$ make
$ make install # (optionally)
```

This process is run for each CMake project. In contrast catkin projects can be built together in workspaces. Building zero to many catkin packages in a workspace follows this work flow:

```
# In a catkin workspace
$ catkin_make
$ catkin_make install # (optionally)
```

The above commands will build any catkin projects found in the `src` folder. This follows the recommendations set by REP128 (<http://ros.org/reps/rep-0128.html>). If your source code is in a different place, say `my_src` then you would call `catkin_make` like this:

Note: If you run the below commands it will not work, as the directory `my_src` does not exist.

```
# In a catkin workspace
$ catkin_make --source my_src
$ catkin_make install --source my_src # (optionally)
```

For more advanced uses of `catkin_make` (`/catkin/commands/catkin_make`) see the documentation: `catkin/commands/catkin_make` (`/catkin/commands/catkin_make`)

1.2 Building Your Package

If you are using this page to build your own code, please also take a look at the later tutorials (C++) (`/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29`)/(Python) (`/ROS/Tutorials/WritingPublisherSubscriber%28python%29`) since you may need to modify `CMakeLists.txt`.

You should already have a catkin workspace (`/catkin/Tutorials/create_a_workspace`) and a new catkin package called `beginner_tutorials` from the previous tutorial, `Creating a Package` (`/ROS/Tutorials/CreatingPackage`). Go into the catkin workspace if you are not already there and look in the `src` folder:

```
$ cd ~/catkin_ws/
$ ls src
```

```
beginner_tutorials/ CMakeLists.txt@
```

You should see that there is a folder called `beginner_tutorials` which you created with `catkin_create_pkg` (`/catkin/commands/catkin_create_pkg`) in the previous tutorial. We can now build that package using `catkin_make` (`/catkin/commands/catkin_make`):

```
$ catkin_make
```

You should see a lot of output from `cmake` and then `make`, which should be similar to this:

```

Base path: /home/user/catkin_ws
Source space: /home/user/catkin_ws/src
Build space: /home/user/catkin_ws/build
Devel space: /home/user/catkin_ws/devel
Install space: /home/user/catkin_ws/install
####
#### Running command: "cmake /home/user/catkin_ws/src
-DCATKIN_DEVEL_PREFIX=/home/user/catkin_ws/devel
-DCMAKE_INSTALL_PREFIX=/home/user/catkin_ws/install" in "/home/user/catki
n_ws/build"
####
-- The C compiler identification is GNU 4.2.1
-- The CXX compiler identification is Clang 4.0.0
-- Checking whether C compiler has -isysroot
-- Checking whether C compiler has -isysroot - yes
-- Checking whether C compiler supports OSX deployment target flag
-- Checking whether C compiler supports OSX deployment target flag - yes
-- Check for working C compiler: /usr/bin/gcc
-- Check for working C compiler: /usr/bin/gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Using CATKIN_DEVEL_PREFIX: /tmp/catkin_ws/devel
-- Using CMAKE_PREFIX_PATH: /opt/ros/kinetic
-- This workspace overlays: /opt/ros/kinetic
-- Found PythonInterp: /usr/bin/python (found version "2.7.1")
-- Found PY_em: /usr/lib/python2.7/dist-packages/em.pyc
-- Found gtest: gtests will be built
-- catkin 0.5.51
-- BUILD_SHARED_LIBS is on
-- ~~~~~
-- ~~ traversing packages in topological order:
-- ~~ - beginner_tutorials
-- ~~~~~
-- +++ add_subdirectory(beginner_tutorials)
-- Configuring done
-- Generating done
-- Build files have been written to: /home/user/catkin_ws/build
####
#### Running command: "make -j4" in "/home/user/catkin_ws/build"
####

```

Note that `catkin_make` (`/catkin/commands/catkin_make`) first displays what paths it is using for each of the 'spaces'. The spaces are described in the [REP128](http://ros.org/reps/rep-0128.html) (<http://ros.org/reps/rep-0128.html>) and by documentation about catkin workspaces on the wiki: `catkin/workspaces` (`/catkin/workspaces`). The important thing to notice is that because of these default values several folders have been created in your catkin workspace. Take a look with `ls`:

```
$ ls
```

```
build  
devel  
src
```

The `build` folder is the default location of the build space (`/catkin/workspaces#Build_Space`) and is where `cmake` and `make` are called to configure and build your packages. The `devel` folder is the default location of the devel space (`/catkin/workspaces#Development_.28Devel.29_Space`), which is where your executables and libraries go before you install your packages.

Now that you have built your ROS package let's talk more about ROS Nodes (</ROS/Tutorials/UnderstandingNodes>).

Except where

otherwise noted, Wiki: ROS/Tutorials/BuildingPackages (última edição 2020-04-18 18:53:46 efectuada por PedroAlcantara (/PedroAlcantara))

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/>)