

Nota: Este tutorial pressupõe que você tenha concluído os tutoriais anteriores: Instalando e configurando seu ambiente ROS (/ROS/Tutorials/InstallingandConfiguringROSEnvironment) .

💡 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.

Navegando no sistema de arquivos ROS

Descrição: Este tutorial apresenta os conceitos do sistema de arquivos ROS e aborda o uso das ferramentas de linha de comando roscd, rosls e rospack (/rospack) .

Nível do Tutorial: INICIANTE

Próximo Tutorial: Criando um pacote ROS (/ROS/Tutorials/CreatingPackage)

Selecione " **catkin** " ou " **rosworld** " logo abaixo. O novo sistema de compilação do ROS é "catkin", enquanto "rosworld" é o antigo sistema de compilação do ROS que foi substituído pelo catkin. Se você é novo no ROS, escolha **catkin** . Aqui estão mais algumas informações. sobre cada um:

1. gatinho (/catkin)
2. catkin/conceitual_overview (/catkin/conceitual_overview)
3. catkin_or_rosworld (/catkin_or_rosworld)
4. rosworld (/rosworld)

gatinho

rosworld

Conteúdo

1. Pré-requisitos
2. Visão geral rápida dos conceitos do sistema de arquivos
3. Ferramentas do sistema de arquivos
 1. Usando rospack
 2. Usando roscd
 1. Subdiretórios
 3. registro roscd
 4. Usando rosls
 5. Conclusão da guia
4. Análise

1. Pré-requisitos

Para este tutorial iremos inspecionar um pacote em ros-tutorials, instale-o usando

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

Substitua '<distro>' (incluindo '<>') pelo nome da sua distribuição ROS (/Distributions) (por exemplo, índigo, cinético, lunar etc.)

2. Visão geral rápida dos conceitos do sistema de arquivos

- **Pacotes:** Pacotes são a unidade de organização de software do código ROS. Cada pacote pode conter bibliotecas, executáveis, scripts ou outros artefatos.
- **Manifestos (`package.xml` (/catkin/package.xml)):** Um manifesto é uma descrição de um *pacote* . Serve para definir dependências entre *pacotes* e capturar meta informações sobre o *pacote* como versão, mantenedor, licença, etc...

Mostrar () Nota sobre pilhas ()

3. Ferramentas do sistema de arquivos

O código está espalhado por muitos pacotes ROS. Navegar com ferramentas de linha de comando como `ls` e `cd` pode ser muito entediante e é por isso que o ROS fornece ferramentas para ajudá-lo.

3.1 Usando rospack

`rospack` (/rospack) permite que você obtenha informações sobre pacotes. Neste tutorial, abordaremos apenas a `find` , que retorna o caminho para o pacote. opção

Uso:

```
$ rospack encontrar [nome_do_pacote]
```

Exemplo:

```
$ rospack encontrar roscpp
```


retornaria:

```
SEU_INSTALL_PATH/share/roscpp
```

Se você instalou o ROS Kinetic do `apt` no Ubuntu Linux, você veria exatamente:

```
/opt/ros/kinetic/share/roscpp
```

3.2 Usando roscd


`roscd` (/rosbash#roscd) faz parte do conjunto `rosbash` (/rosbash) . Ele permite que você altere o diretório ( `cd` (<http://ss64.com/bash/cd.html>)) diretamente para um pacote ou pilha.

Uso:

```
$ roscd <pacote-ou-pilha>[/subdir]
```

Para verificar se mudamos para o diretório do pacote roscpp, execute este exemplo:

```
$ roscd roscpp
```

Agora vamos imprimir o diretório de trabalho usando o comando Unix  pwd (<http://ss64.com/bash/pwd.html>) :

```
$ pwd
```

Você deveria ver:

```
SEU_INSTALL_PATH/share/roscpp
```

Você pode ver que YOUR_INSTALL_PATH/share/roscpp é o mesmo caminho que rospack find deu no exemplo anterior.

Observe que roscd (/roscd) , como outras ferramentas ROS, encontrará *apenas* pacotes ROS que estejam nos diretórios listados em seu ROS_PACKAGE_PATH (/ROS/EnvironmentVariables#ROS_PACKAGE_PATH) . Para ver o que está em seu ROS_PACKAGE_PATH (/ROS/EnvironmentVariables#ROS_PACKAGE_PATH) , digite:

```
$ echo $ ROS_PACKAGE_PATH
```

Seu ROS_PACKAGE_PATH (/ROS/EnvironmentVariables#ROS_PACKAGE_PATH) deve conter uma lista de diretórios onde você possui pacotes ROS separados por dois pontos. Um ROS_PACKAGE_PATH (/ROS/EnvironmentVariables#ROS_PACKAGE_PATH) típico pode ser assim:

```
/opt/ros/kinetic/base/install/share
```

Da mesma forma que outros caminhos de ambiente, você pode adicionar diretórios adicionais ao seu ROS_PACKAGE_PATH (/ROS/EnvironmentVariables#ROS_PACKAGE_PATH) , com cada caminho separado por dois pontos ':'.

3.2.1 Subdiretórios

roscd (/roscd) também pode mover para um subdiretório de um pacote ou pilha.

Tentar:

```
$ roscd roscpp/cmake
$ pwd
```

Você deveria ver:

```
SEU_INSTALL_PATH/share/roscpp/cmake
```

3.3 registro roscd


roscd log will take you to the folder where ROS stores log files. Note that if you have not run any

ROS programs yet, this will yield an error saying that it does not yet exist.

If you have run some ROS program before, try:

```
$ roscd log
```

3.4 Using rosls

rosls (/rosbash#rosls) is part of the rosbash (/rosbash) suite. It allows you to  ls (<http://ss64.com/bash/ls.html>) directly in a package by name rather than by absolute path.

Usage:

```
$ rosls <package-or-stack>[/subdir]
```


Example:

```
$ rosls roscpp_tutorials
```

would return:

```
cmake launch package.xml  srv
```

3.5 Tab Completion

It can get tedious to type out an entire package name. In the previous example, `roscpp_tutorials` is a fairly long name. Luckily, some ROS tools support  TAB completion (http://en.wikipedia.org/wiki/Command_line_completion).

Start by typing:

```
$ roscd roscpp_tut<<< now push the TAB key >>>
```

After pushing the **TAB** key, the command line should fill out the rest:

```
$ roscd roscpp_tutorials/
```

This works because `roscpp_tutorials` is currently the only ROS package that starts with `roscpp_tut`.

Now try typing:

```
$ roscd tur<<< now push the TAB key >>>
```

After pushing the **TAB** key, the command line should fill out as much as possible:

```
$ roscd turtle
```

However, in this case there are multiple packages that begin with `turtle`. Try typing **TAB** another time. This should display all the ROS packages that begin with `turtle`:

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

On the command line you should still have:

```
$ roscd turtle
```

Now type an `s` after `turtle` and then push **TAB**:

```
$ roscd turtles<<< now push the TAB key >>>
```

Since there is only one package that starts with `turtles`, you should see:

```
$ roscd turtlesim/
```

If you want to see a list of all currently installed packages, you can use tab completion for that as well:


```
$ rosls <<< now push the TAB key twice >>>
```

4. Review

You may have noticed a pattern with the naming of the ROS tools:

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

This naming pattern holds for many of the ROS tools.

Now that you can get around in ROS, let's  create a package (<http://wiki.ros.org/ROS/Tutorials/CreatingPackage>).

Except where

otherwise noted, Wiki: ROS/Tutorials/NavigatingTheFilesystem (última edição 2020-09-21 17:34:03 efectuada por Himanshu (/Himanshu))

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/>)