第八课 ROS 文件系统介绍

1. 预备工作

本教程中我们将会用到 ros-tutorials 程序包，请先安装：

```
$ sudo apt-get install ros-kinetic-ros-tutorials
```

2. 快速了解文件系统概念

- **Packages:** 软件包，是 ROS 应用程序代码的组织单元，每个软件包都可以包含程序库、可执行文件、脚本或者其它手动创建的东西。
- **Manifest** (package.xml)：清单，是对于'软件包'相关信息的描述，用于定义软件包相关元信息之间的依赖关系，这些信息包括版本、维护者和许可协议等。

3. 文件系统工具

程序代码是分布在众多 ROS 软件包当中，当使用命令行工具（比如 ls 和 cd）来浏览时会非常繁琐，因此 ROS 提供了专门的命令工具来简化这些操作。

4. 使用 rospack

rospack 允许你获取软件包的有关信息。在本教程中，我们只涉及到 rospack 中 find 参数选项，该选项可以返回软件包的路径信息。

```
用法： # rospack find [包名称]
```

示例：

```
$ rospack find roscpp
```

应输出：

```
YOUR_INSTALL_PATH/share/roscpp
```

如果你是在 Ubuntu Linux 操作系统上通过 apt 来安装 ROS，你应该会准确地看到：

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

5. 使用 roscd

roscd 是 rosbash 命令集中的一部分，它允许你直接切换(cd)工作目录到某个软件包或者软件包集当中。

用法：

```
# roscd [本地包名称[/子目录]]
```

示例：

```
$ roscd roscpp
```

为了验证我们已经切换到了 roscpp 软件包目录下，现在我们可以使用 Unix 命令 pwd 来输出当前工作目录：

```
$ pwd
```

你应该会看到：

```
YOUR_INSTALL_PATH/share/roscpp
```

你可以看到 YOUR_INSTALL_PATH/share/roscpp 和之前使用 rospack find 得到的路径名称是一样的。

注意，就像 ROS 中的其它工具一样，roscd 只能切换到那些路径已经包含在 ROS_PACKAGE_PATH 环境变量中的软件包，要查看 ROS_PACKAGE_PATH 中包含的路径可以输入：

```
$ echo $ROS_PACKAGE_PATH
```

你的 ROS_PACKAGE_PATH 环境变量应该包含那些保存有 ROS 软件包的路径，并且每个路径之间用冒号分隔开来。一个典型的 ROS_PACKAGE_PATH 环境变量如下：

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

跟其他路径环境变量类似，你可以在 ROS_PACKAGE_PATH 中添加更多其它路径，每条路径使用冒号':'分隔。

6. 子目录

使用 roscd 也可以切换到一个软件包或软件包集的子目录中。

执行：

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

应该会看到：

```
YOUR_INSTALL_PATH/share/roscpp/cmake
```

7. roscd log

使用 roscd log 可以切换到 ROS 保存日记文件的目录下。需要注意的是，如果你没有执行过任何 ROS 程序，系统会报错说该目录不存在。

如果你已经运行过 ROS 程序，那么可以尝试：

```
$ roscd log
```

8. 使用 rosls

rosls 是 rosbash 命令集中的一部分，它允许你直接按软件包的名称而不是绝对路径执行 ls 命令（罗列目录）。

用法：

```
# rosls [本地包名称[/子目录]]
```

示例：

```
$ rosls roscpp_tutorials
```

应输出：

```
cmake    package.xml    srv
```

9. Tab 自动完成输入

当要输入一个完整的软件包名称时会变得比较繁琐。在之前的例子中 roscpp tutorials 是个相当长的名称，幸运的是，一些 ROS 工具支持 TAB 自动完成输入的功能。

输入：

```
# roscd roscpp_tut<<< 现在请按 TAB 键 >>>
```

当按 **TAB** 键后，命令行中应该会自动补充剩余部分：

```
$ roscd roscpp_tutorials/
```

这应该有用，因为 roscpp tutorials 是当前唯一一个名称以 roscpp tut 作为开头的 ROS 软件包。现在尝试输入：

```
# roscd tur<<< 现在请按 TAB 键 >>>
```

按 **TAB** 键后，命令应该会尽可能地自动补充完整：

```
$ roscd turtle
```

但是，在这种情况下有多个软件包是以 turtle 开头，当再次按 **TAB** 键后应该会列出所有以 turtle 开头的 ROS 软件包：

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

这时在命令行中你应该仍然只看到：

```
$ roscd turtle
```

现在在 turtle 后面输入 s 然后按 **TAB** 键：

```
# roscd turtles<<< 请按 TAB 键 >>>
```

因为只有一个软件包的名称以 turtles 开头，所以你应该会看到：

```
$ roscd turtlesim/
```

10. 总结

你也许已经注意到了 ROS 命令工具的的命名方式：

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

这种命名方式在许多 ROS 命令工具中都会用到。