新工科机器人工程专业规划教材

机器人技术与系统国家重点实验室 组织编写

蔡鹤皋 院士 / 邓宗全 院士 顾问

Analysis and Engineering Application of ROS2

ROS2 源代码分析与工程应用

丁 亮 曲明成 张亚楠 夏科睿 编著

清华大学出版社

新工科机器人工程专业规划教材

Analysis and Engineering Application of ROS2

ROS2 源代码分析与工程应用

丁 亮 曲明成 张亚楠 夏科睿 编著

清华大学出版社
北京

内 容 简 介

本书从源代码层面深入解析 ROS2，并对常用工具库及操控机器人的常用模块进行系统介绍。基于 JuLab 系列机器人，进行具体应用场景的代码和功能介绍。本书共包括 6 章。第 1 章介绍 ROS2 Ardent 版本、ROS2 安装及环境配置、基本命令等。第 2 章分析 ROS2 Ardent 的总体框架，进行源代码概述，开展 ament、Fast-CDR、Fast-RTPS、RMW、robot_model、RCL、RCLcpp 等代码分析。第 3 章介绍 ROS2 常用工具库中的 orocos_kinematics_dynamics、POCO、urdfdom、PCL 以及 MoveIt。第 4 章阐述 SLAM 导航及应用，通过 ROS2 发布里程计信息，基于 JuLab1 机器人完成 SLAM 开发。第 5 章介绍六轴机械臂轨迹规划，基于 JuLab1 机器人完成实例开发。第 6 章介绍 OpenCV 图像和视频基础，图像转换，机器人 3D 视觉技术等。

本书深入浅出地分析了 ROS2 的核心功能，并结合了典型工程案例。既可以作为机器人专业核心教材，又可以作为物联网专业辅助教材，同时也可以作为工程开发人员的指导书。

图书在版编目(CIP)数据

ROS2 源代码分析与工程应用/丁亮等编著. —北京：清华大学出版社，2019
(新工科机器人工程专业规划教材)
ISBN 978-7-302-52745-9

Ⅰ. ①R…　Ⅱ. ①丁…　Ⅲ. ①机器人－程序设计－高等学校－教材　Ⅳ. ①TP242

中国版本图书馆 CIP 数据核字(2019)第 067177 号

责任编辑：许　龙
封面设计：常雪影
责任校对：赵丽敏
责任印制：刘海龙

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：北京鑫海金澳胶印有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：13.75　　**字　　数**：334 千字
版　　次：2019 年 5 月第 1 版　　**印　　次**：2019 年 5 月第1次印刷
定　　价：42.00 元

产品编号：080813-01

前言

FOREWORD

ROS是开源的机器人操作系统软件平台，是机器人技术和人工智能技术的一个令人兴奋的结合点。2007年前后，ROS起源于美国斯坦福大学人工智能实验室与Willow Garage公司的项目合作；2013年，ROS的开发和维护工作被移交给了开源机器人基金会(Open Source Robotics Foundation)。十余年来，ROS在国际学者的大力支持下迅速发展。随着机器人与人工智能热潮的到来，基于ROS的开发与应用近几年在我国也开展得如火如荼。

ROS的主要目标是为不同类型的机器人提供基础中间件和应用软件，降低机器人的开发和应用难度，避免重复开发。ROS定义抽象层，允许软件被多种机器人重用。然而，现有ROS在设计软件框架时未预料到会如此受欢迎，其已有的架构越来越难以满足众多新用途和潜在的市场需求。例如，在ROS的master-slave架构下，当master节点发生问题时，将产生系统通信中断的严重后果，不适合多机器人协作的应用场景。现有ROS的另一个重要瓶颈是实时性能差，难以应用于对实时性能要求很高的领域。例如，运行于复杂动态环境中的高机动移动机器人、高速运行的流水线作业机器人等，要求机器人以极快的速度对指令进行响应，以极高的可靠性完成任务。

为了解决ROS的诸多不足，设计新的软件框架以满足目前及未来市场需求势在必行，ROS2也因此应运而生。在ROS2开发过程中，对原ROS框架进行了重大改进。为了满足多机器人协同工作的高可靠、实时性要求，ROS2摒弃了原来的master-slave架构，使用更加先进的分布式架构，采用了数据分发服务(DDS)技术，节点可以分布于不同主机，可以互为服务器/客户端，方便负载均衡，可降低中心节点失效导致slave节点通信失败的风险。更为重要的是，ROS2提供对实时操作系统和嵌入式设备的支持，使得实时控制可以直接在ROS2上运行，并支持节点间、进程间实时通信。ROS2的第一个正式版本Ardent Apalone于2017年12月8日发布，该版本并未完全脱离ROS，而是继承了ROS的众多优点，如接口与编程语言无关、驱动程序和算法可封装成独立的库、方便移植、支持多语言等。

作为国内机器人操作系统的积极应用者和推动者，研究团队一直关注ROS和ROS2的发展，并积极探索、改进和应用ROS及ROS2，开发了支持ROS和ROS2的JuLab系列机器人通用硬件平台。为了更好地与对ROS2感兴趣的相关人员分享开发与应用ROS2的成果和经验，更深入地展示ROS2的架构优势、运行模式以及应用方法，本书从源代码层面深入解析ROS2，并对常用工具库及操控机器人的常用模块进行了系统介绍。基于JuLab系列机器人，进行了具体应用场景的代码和功能介绍。

本书共包括6章。第1章介绍ROS2 Ardent版本、ROS2安装及环境配置、基本命令等。第2章分析ROS2 Ardent的总体框架，进行源代码概述，开展ament、Fast-CDR、Fast-RTPS、RMW、robot_model、RCL、RCLcpp等代码分析。第3章介绍ROS2常用工具库中的orocos_kinematics_dynamics、POCO、urdfdom、PCL以及MoveIt。第4章阐述SLAM导航及应用，通过ROS2发布里程计信息，基于JuLab1机器人完成SLAM开发。第5章介绍

六轴机械臂轨迹规划，基于JuLab1机器人完成实例开发。第6章介绍OpenCV图像和视频基础，图像转换，机器人3D视觉技术等。本书的电子文档资源同步在www.airtros.com网站发布。本书深入浅出地分析了ROS2的核心功能，并结合了典型工程案例。书中实践案例部分以JuLab机器人为载体，该机器人助力哈尔滨工业大学计算机学院两支队伍夺得了2018年全国大学生物联网大赛总决赛一等奖和二等奖。

本书作者均为从事机器人、操作系统科研和教学工作的人员。第1章由丁亮、曲明成、夏科睿撰写，第2章由曲明成、张亚楠、刘鹏飞撰写，第3章由张亚楠、曲明成、夏科睿撰写，第4章由丁亮、夏科睿、郭龙撰写，第5章由曲明成、夏科睿、丁亮撰写，第6章由张亚楠、夏科睿撰写。全书由丁亮统稿，张亚楠组织编辑和校对。相关研究工作和本书的编著得到了哈工大机器人集团董事长王飞、高级副总裁于振中等领导的大力支持，郭龙、金马、王权、陈伟伟、蒋晨旭、彭超、侯旗、何婷婷、张韬庚、姬鹏鹏等参与了本书案例的开发和编写工作，清华大学出版社的编辑对本书出版给予了大力支持，在此表示衷心的感谢！

本书涉及的研究工作得到了国家自然科学基金优秀青年基金项目(51822502)、安徽省科技重大专项项目、合肥市庐州产业创新团队项目、哈工大机器人(合肥)国际创新研究院立项项目等资助，在此表示感谢！

由于作者水平所限，书中难免存在不足之处，敬请各位读者批评指正。

作　者

2018年秋

目 录

CONTENTS

第 1 章

ROS2简介

本章主要介绍 ROS2 第一个正式发布的版本——ROS2 Ardent 版本，以及 ROS2 Ardent 的安装、环境配置、基本命令。通过本章的学习，读者可以了解 ROS2 的基础知识，了解 ROS2 与 ROS 的联系和区别，理解 ROS2 的改进方向和特点。

1.1 ROS2 Ardent Apalone 概述

ROS(Robot Operating System)是一个机器人软件平台，它能为异质计算机集群提供类似操作系统的功能。ROS 的前身是斯坦福人工智能实验室为了支持斯坦福智能机器人 STAIR 而建立的交换庭(switchyard)项目。到 2008 年，主要由 Willow Garage 继续该项目的研发。ROS 主要的特点是提供了一种发布订阅式的通信框架用以简单、快速地构建分布式计算系统，ROS 定义了节点，允许不同节点的进程能接收、发布、聚合各种信息(如传感、控制、状态、规划等)。提供了大量的工具组合用以配置、启动、自检、调试、可视化、登录、测试、终止分布式计算系统。提供了广泛的库文件实现以机动性、操作控制、感知为主的机器人功能。提供一些标准操作系统服务，例如硬件抽象、底层设备控制、常用功能实现、进程间消息以及数据包管理。

ROS 最早应用于 Willow Garage PR2 机器人开发。ROS 主要目标是为不同的机器人应用提供软件工具，因此，ROS 在定义抽象层(通常是通过消息接口)上付出了大量努力，从而允许许多软件被其他机器人重用。ROS 不仅应用在 PR2 机器人及类似的机器人上，而且应用在大小腿轮式机器人、人形机器人、军事工业机器人、室外地面车辆(包括自动驾驶汽车)、飞行器等众多设备上。除此之外，ROS 正在被大多数学术研究团体以外的领域采纳。以 ROS 为基础的产品正在大量上市，包括制造机器人、农业机器人、商业清洁机器人等。政府机构也在关注 ROS 的使用。例如，美国宇航局预计将运行 ROS 的 Robonaut 2 部署到国际空间站。

现有 ROS 在设计软件框架层面时未考虑到 ROS 会有如此之多的应用，已有的软件架构已经不能胜任众多的新用途和潜在的市场，因此设计新的软件框架，满足市场需求势在必行，ROS2 的开发也就水到渠成。在 ROS2 中，对原 ROS 框架做了重大改进。例如，为了满足多机器人团队协作的需求，ROS2 摒弃了原 ROS 的 master-slave 架构，这种结构下当 master 节点发生问题将产生通信中断的严重后果。现有 ROS 另一个重要的缺陷是实时性能差，不具备应用在对实时性能要求很高的领域。未来 ROS2 将计划支持实时操作系统，使

得实时控制可以直接在 ROS 上运行，并支持节点间、进程间实时通信。

ROS2 Ardent Apalone 是 ROS 的后续版本，其中 Ardent Apalone 是版本名称，该版本代号为 Ardent 。ROS2 的版本命名方式与 ROS 相似，之前的 ROS 版本包括 ROS Lunar Loggerhead、ROS Kinetic Kame、ROS Jade Turtle、ROS Indigo Igloo 等。ROS2 Ardent Apalone 并没有完全放弃 ROS，保留了 ROS 拥有的众多优点，如接口与编程语言无关，第三方库不依赖于 ROS，方便移植，支持多语言。ROS2 采用分布式框架，节点可以分布于不同主机，可以互为服务器/客户端，方便负载均衡。ROS2 针对 ROS 节点间通信机制采用 master-slave 模式的诸多问题，提出使用更先进的分布式架构，具有更高的可靠性，以及对实时性和嵌入式设备的支持。ROS2 最大的不同在于采用了数据分发服务（Data Distribution Service，DDS）技术，DDS 技术更适用于实时分布式嵌入式系统。

ROS2 Ardent 的源代码文件中主要有四个文件夹，包括 ament、eProsima、ros、ros2。ROS2 Ardent 的代码量统计如表 1-1 所示。

表 1-1 ROS2 Ardent 的代码统计

代码统计	代码量
类的数量	6142
代码行数	346060
注释行数	126286
注释与代码比率	0.36
声明语句	118649
执行语句	168295
文件数量	3571
函数数量	27326
预处理行数	30457

1.2 ROS2 安装及环境配置

ROS2 Ardent Apalone 在 2017 年 12 月 8 日发布，这是 ROS2 的第一个正式版本。本节以 ROS2 的 Ardent Apalone 版本为例，介绍 ROS2 的安装与环境配置方法。

1.2.1 安装 ROS2

ROS2 支持 Ubuntu16.04 系统，安装方法和 ROS 类似，可以按照以下步骤进行安装：

1. 添加软件源

在 Ubuntu 命令行终端输入如下命令：

```
$ sudo apt update && sudo apt install curl
$ curl http://repo.ros2.org/repos.key | sudo apt-key add -
$ sudo sh -c 'echo "deb [arch=amd64,arm64] http://repo.ros2.org/ubuntu/main xenial main" >
/etc/apt/sources.list.d/ros2-latest.list'
```

2. 安装 ROS2

```
$ sudo apt - get update
$ sudo apt install 'apt list ros - ardent - * 2 > /dev/null | grep "/" | awk - F/ '{print $ 1}'
| grep - v - e ros - ardent - ros1 - bridge - e ros - ardent - turtlebot2 - | tr "\n" " "'
```

以上安装命令排除了 ros-ardent-ros1-bridge 和 ros-ardent-turtlebot2-* 等功能包，这些功能包需要依赖 ROS，可以在后续单独安装。

3. 设置环境变量

```
$ source /opt/ros/ardent/setup.bash
```

如果安装了 Python 包——argcomplete，还需要设置以下环境变量：

```
$ source /opt/ros/ardent/share/ros2cli/environment/ros2 - argcomplete.bash
```

4. 配置 ROS Middleware

DDS 是 ROS2 中的重要部分，ROS2 默认使用的 ROS Middleware（简称 RMW）是 FastRPTS，也可以通过以下环境变量将默认 RMW 修改为 OpenSplice：

```
RMW_IMPLEMENTATION = rmw_opensplice_cpp
```

注意：按照上述 1～4 步执行，在当前终端可以执行 ROS2 的命令。如果关闭当前终端，再次开启终端，则 ROS2 命令不生效。导致 ROS2 命令失效的原因是 1～4 配置步骤只在当前终端生效，即配置在局部空间，在全局空间不生效。

如果希望对 ROS2 的配置在任意新打开的终端生效，在配置完上述 1～4 步骤后，还需要设置. bashrc 文件。首先介绍. bashrc 文件，. bashrc 文件主要保存个人的一些个性化设置，如命令别名、路径等。. bashrc 文件是隐藏文件，其内容可使用 ls -a 命令来查看。在变量名前加'$'，可获取变量值，例如“echo $PATH”会显示当前设置的 PATH 变量“/usr/bin:/usr/local/bin:/bin”。处理 $PATH 变量要注意：在原有变量的后面增加新的变量，例如，在原有 PATH 变量后面增加“/some/directory”则输入“PATH= $PATH:/some/directory”。如果要将“/some/directory”定义为一个全局变量，使在以后打开的终端中生效，需要将局部变量输出（export），可以用 export 命令：export PATH= $PATH:/some/directory，这里需要注意 export 命令只能改变当前终端及以后运行的终端里的变量，对于已经运行的终端没有作用。

使 ROS2 的配置全局生效，可执行以下两个步骤。

1）编辑环境变量

首先执行 source /opt/ros/ardent/setup.bash，使得更改的环境变量生效。

```
$ sudo gedit ~/.bashrc
```

在 bashrc 文件中增加：

```
export PATH = $ PATH:/some/directory source /opt/ros/ardent/setup.bash
```

如果安装了 Python 包——argcomplete，还需要设置以下环境变量：

```
$ source /opt/ros/ardent/share/ros2cli/environment/ros2 - argcomplete.bash
```

2）配置环境变量

在.bashrc 文件末尾增加：

```
source /opt/ros/ardent/setup.bash
source /opt/ros/ardent/share/ros2cli/environment/ros2 - argcomplete.bash
set RMW_IMPLEMENTATION = $ RMW_IMPLEMENTATION:/rmw_opensplice_cpp
export RMW_IMPLEMENTATION
```

5. 可选装依赖 ROS 的功能包

ROS2 在很长一段时间会和 ROS 并存，所以目前很多 ROS2 中的功能包需要依赖 ROS 中的功能包，ROS2 也提供了与 ROS 之间通信的桥梁——ros1_bridge。在安装这些与 ROS 有依赖关系的功能包之前，需要系统已经成功安装 ROS。下面以 Kinect 版本为例安装 ROS。

1）设置 sources.list

```
sudo sh - c 'echo "deb http://packages.ros.org/ros/ubuntu $ (lsb_release - sc) main" > /etc/
apt/sources.list.d/ros - latest.list'
```

2）设置 key

```
sudo apt - key adv - - keyserver http ://ha. pool. sks - keyservers. net: 80 - - recv - key
421C365BD9FF1F717815A3895523BAEEB01FA116
```

3）安装

执行命令 sudo apt-get update 安装 ROS2。

执行命令 sudo apt-get install ros-kinetic-desktop 安装 ROS 基本库。

6. ROS2 bridge 功能

安装好 ROS 后，才能通过以下命令安装 ROS2 的“bridge”功能包：

```
$ sudo apt update
$ sudo apt install ros - ardent - ros1 - bridge  ros - ardent - turtlebot2 - *
```

按照以上方法安装完成后，就可以使用 ROS2 的命令了。ROS2 的默认安装路径依然是在 Ubuntu 系统的/opt/ros 路径下。

使用如下命令查看 ROS2 命令行工具相关的帮助信息，如图 1-1 所示。

```
$ ros2 - - help
```

```
~ ros2 --help
usage: ros2 [-h] Call `ros2 <command> -h` for more detailed usage. ...

ros2 is an extensible command-line tool for ROS 2.

optional arguments:
  -h, --help            show this help message and exit

Commands:
  daemon    Various daemon related sub-commands
  msg       Various msg related sub-commands
  node      Various node related sub-commands
  pkg       Various package related sub-commands
  run       Run a package specific executable
  security  Various security related sub-commands
  service   Various service related sub-commands
  srv       Various srv related sub-commands
  topic     Various topic related sub-commands

  Call `ros2 <command> -h` for more detailed usage.
```

图 1-1　ROS2 命令行工具相关的帮助信息

1.2.2 运行 talker 和 listener

ROS2 安装完成后，默认带有部分示例，为了验证 ROS2 是否安装成功，可以使用如下命令进行测试：

```
$ ros2 run demo_nodes_cpp talker
$ ros2 run demo_nodes_cpp listener
```

运行成功后，效果与在 ROS 1 中实现的效果类似，如图 1-2 所示。

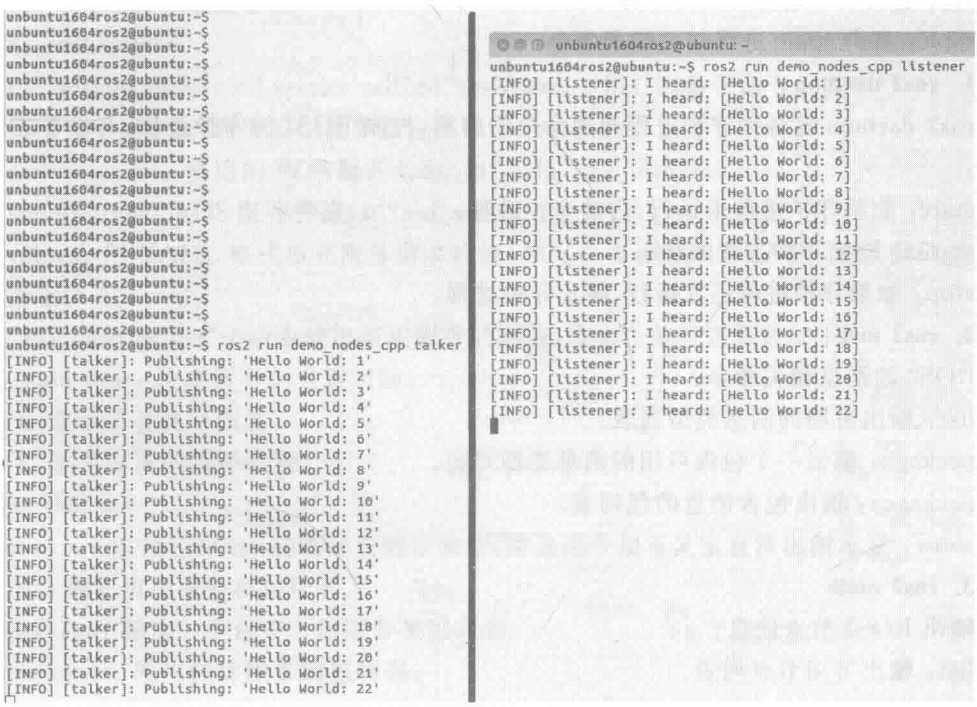

图 1-2 ROS2 节点通信示例

通过这个例程可以看到，在 ROS2 中运行节点时，并不需要启动 ROS Master，两个节点之间建立的通信连接完全依靠节点自身的“Discovery”机制。

1.3 ROS2 的基本命令

ROS2 的基本命令包括两部分：一部分是 ROS2 的核心命令；另外一部分是需要和 ROS 交互时，可能用到的 ROS 命令。

1.3.1 ROS2 核心命令

ROS2 核心命令分为以下几部分。在“$”提示符后输入“ros2-h”，则可以看到以下

ROS2 命令。

daemon：各种守护进程相关的子命令。

msg：各种消息进程相关的子命令。

node：各种 node 进程相关的子命令。

pkg：各种 pkg 进程相关的子命令。

run：运行特定软件包的可执行文件。

security：各种安全进程相关的子命令。

service：各种服务进程相关的子命令。

srv：各种 srv 进程相关的子命令。

topic：各种 topic 进程相关的子命令。

1. ros2 daemon

ros2 daemon 是各种守护进程相关命令的前缀，控制 ROS2 的守护进程，有以下三个子命令：

start：如果守护进程未运行，启动守护进程。

status：输出守护进程的状态。

stop：如果守护进程正在运行，停止守护进程。

2. ros2 msg

ROS2 的消息相关命令。

list：输出可用的消息类型列表。

package：输出一个包内可用的消息类型列表。

packages：输出包含消息的包列表。

show：显示输出消息定义。

3. ros2 node

输出 ROS2 节点信息。

list：输出可用节点列表。

4. ros2 pkg

输出安装包信息。

executables：输出特定于软件包的可执行文件列表。

list：输出可用软件包列表。

prefix：输出包的前缀路径。

5. ros2 run

输出安装包信息。

命令格式：ros2 run [-h] [--prefix PREFIX] package_name executable_name

位置参数如下：

package_name：ROS 包名称。

executable_name：可执行文件名称。

argv：将任意参数传递给可执行文件。

可选参数如下：

-h，--help：显示此帮助消息并退出。

--prefix PREFIX：在可执行文件之前的前缀命令，包含空格的命令必须包含在引号中。

6. ros2 security

ros2 security 包含各种安全进程相关的子命令。

命令格式和参数如下：

create_key：创建秘钥。

create_keystore：创建密钥库。

create_permission：创建许可。

distribute_key：分配密钥。

list_keys：列出表键。

7. ros2 service

ros2service 包含“ros2 service call”和“ros2 service list”两条子命令。其中“ros2 service call”调用一个服务，该命令的位置参数包括以下三个参数：

service_name：调用的 ROS 服务名称(如“/add_two_ints”)。

service_type：ROS 服务类型(如“std_srvs/Empty”)。

values：用 YAML 格式填充服务请求的值(如“{a：1，b：2}”)，否则，将以默认值发布服务请求。

“ros2 service list”子命令输出可用服务的列表，该命令的可选参数如下：

--spin-time：发现服务的自旋时间。

-t：显示服务补充信息。

-c：显示发现的服务数量。

8. ros2 srv

ros2 srv 包含各种 srv 进程相关的子命令，可选命令如下：

list：输出可用服务类型列表。

package：输出一个包中可用服务类型列表。

packages：输出包含服务的包列表。

show：输出服务定义。

9. ros2 topic

ros2 topic 是各种 topic 进程相关的子命令的前缀。

1.3.2 ROS 与 ROS2 交互相关命令

为了更好地从 ROS 过渡到 ROS2，ROS2 保留了与 ROS 交互的相关命令。本节主要介绍 ROS 工作空间、ROS 服务以及 ROS 文件系统的相关命令。

1. ROS 工作空间

执行命令“$ roscore”启动 ROS，执行以下命令，创建工作环境：

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/src
$ catkin_init_workspace
```

执行以下命令，编译 ROS 程序：

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

执行以下命令，添加程序包到全局路径：

```
$ echo "source catkin_ws/devel/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
```

执行以下命令，创建程序包，添加程序包的依赖包，编译新建的程序包：

```
$ cd ~/catkin_ws/src
$ catkin_create_pkg  [depend1] [depend2] [depend3]
$ cd ~/catkin_ws
$ catkin_make
```

执行以下命令查找编译后的程序包：

```
$ rospack find [package name]
```

执行以下命令查看程序包依赖关系：

```
$ rospack depends
$ rospack depends1
```

2. ROS 服务

执行以下命令，查看所有正在运行的节点：

```
$ rosnode list
```

执行以下命令，查看某节点信息：

```
$ rosnode info [node_name]
```

执行以下命令，运行节点：

```
$ rosrun [package_name] [node_name] [__name:=new_name]
```

执行以下命令，查看所有 topic 列表：

```
$ rostopic list
```

执行以下命令，图形化显示 topic：

```
$ rosrun rqt_graph rqt_graph
$ rosrun rqt_plot rqt_plot
```

执行以下命令，查看某个 topic 信息：

```
$ rostopic echo [topic]
```

执行以下命令，查看 topic 消息格式：

```
$ rostopic type [topic]
$ rosmsg show [msg_type]
```

执行以下命令，向 topic 发布消息：

```
$ rostopic pub [-1]  [-r 1] -- [args] [args]
```

Serv 执行以下命令，查看所有 service 操作：

```
$ rosservice -h
```

执行以下命令，查看 service 列表：

```
$ rosservice list
```

执行以下命令，调用 service：

```
$ rosservice call [service] [args]
```

执行以下命令，查看 service 格式并显示数据：

```
$ rosservice type [service] | rossrv show
```

执行以下命令，设置 service parameter：

```
$ rosparam set [parame_name] [args]
```

执行以下命令，获得 parameter：

```
$ rosparam get [parame_name]
```

执行以下命令，加载 parameter：

```
$ rosparam load [file_name] [namespace]
```

执行以下命令，删除 parameter：

```
$ rosparam delete
```

执行以下命令，清除服务执行后留下的数据：

```
rosservice call clear
```

执行以下命令，记录所有 topic 变化：

```
$ rosbag record -a
```

执行以下命令，记录某些 topic：

```
$ rosbag record -O subset
```

执行以下命令，查看记录信息：

```
$ rosbag info
```

执行以下命令，回放记录信息：

```
$ rosbag play (-r 2)
```

3. ROS 文件系统的相关命令

首先介绍 ROS 文件系统中的相关概念，然后介绍 ROS 文件系统工具集。ROS 文件系统主要由功能包(package)、功能包的描述文件组成。功能包是 ROS 中比较基础的软件组

织方式，每个功能包可包含依赖库、可执行文件、脚本文件及其他的相关文件；功能包的描述文件是 Manifest，主要以 XML 形式存在，它主要定义功能包与其他功能包之间的依赖关系，并提供关于功能包的版本信息、维护者信息和许可信息等。

由于 ROS 代码分布在许多功能包中，如果采用 ls 和 cd 等命令将会非常烦琐，因此 ROS 提供一些自定义的工具集辅助操作。

1）rospack 使用介绍

rospack 用于获取功能包的相关信息，下面只涉及其中的 find 参数，它返回功能包的路径信息。

命令格式如下：

```
# rospack find [package_name]
```

示例：

```
$ rospack find roscpp
```

执行结果如下：

```
YOUR_INSTALL_PATH/share/roscpp
```

Advanced Packaging Tool(apt)是 Linux 下的一款安装包管理工具。如果 ROS 是从 apt 安装到 Ubuntu 系统中的，将会显示如下结果：

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

2）roscd 使用介绍

roscd 是 rosbash 套件中的一部分，利用它可以改变路径到指定的功能包或功能包集中。

命令格式如下：

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

示例：

```
$ roscd roscpp
```

为了验证是否改变到 roscpp 功能包的路径下，可采用 UNIX 的命令 pwd 打印当前的路径信息；

```
$ pwd
```

得到如下执行结果：

```
YOUR_INSTALL_PATH/share/roscpp
```

这个路径和之前示例中 rospack find roscpp 查找出的路径是一致的。

注意，roscd 和其他 ROS 命令工具一样，只会查找 ROS_PACKAGE_PATH 变量中包含的 package，通过下面的命令可以查看 ROS_PACKAGE_PATH 包含的信息：

```
$ echo $ ROS_PACKAGE_PATH
```