ROS入门学习指导

作 者：周友军

版权申明：本文仅作为个人学习使用，不承担任何商用和法律责任。

**关 键 字**：

**摘 要**：

**缩 略 语**：

|  |  |  |
| --- | --- | --- |
| **缩略语** | **英文全拼** | **中文释义** |
| **OSFR** | **O**pen **S**ource **R**obotics **F**oundation. | 开源机器技术基金，一个非营利性的组织进行维护，号召有能力的人对该基金进行捐款。 |
| **ROS** | **R**obot **O**perating **S**ystem. | 机器人操作系统。 |
|  |  |  |

# 前言

# ROS Tutorial快速入门

本章节主要引导初学者快速入门，包括系统的安装、开发环境配置、创建工作台、了解和掌握ROS文件系统结构等等。通过这一系列的指引和了解，初学者能够快速的部署ROS环境，并在ROS上进行软件开发。

## 版本历史

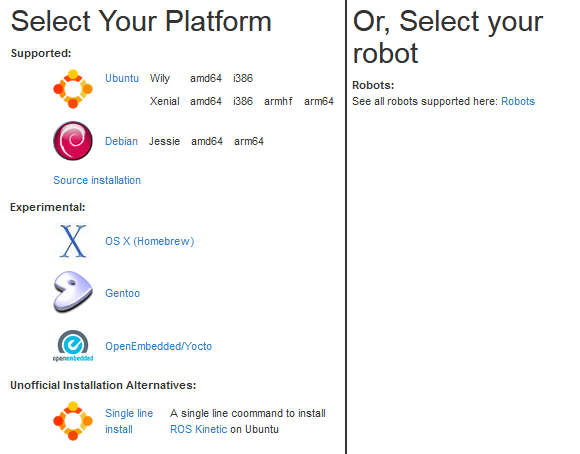
1. ROS历史版本列表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Distribution** | **Release Date** | **Poster** | **Tuturtle, turtle in tutorial** | **EOL Date** |
| Melodic Morenia | May, 2018 | TBD | TBD | May, 2023 |
| Lunar Loggerhead | May 23rd, 2017 | Lunar Loggerhead | Lunar Loggerhead | May, 2019 |
| Kinetic Kame | May 23rd, 2016 | Kinetic Kame | Kinetic Kame | April, 2021 |
| Jade Turtle | May 23rd, 2015 | Jade Turtle | Jade Turtle | May, 2017 |
| Indigo lgloo | July 22nd, 2014 | I-turtle | I-Turtle | April, 2019 |
| Hydro Medusa | September 4th, 2013 | H-turtle | H-Turtle | May, 2015 |
| Groovy Galapagos | December 31st, 2012 | G-turtle | G-Turtle | July, 2014 |
| Fuerte Turtle | April 23rd, 2012 | F-turtle | F-Turtle | -- |
| Electric Emys | August 30th, 2011 | E-turtle | E-Turtle | -- |
| Diamondback | March 2nd, 2011 | D-turtle | D-Turtle | -- |
| C Trutle | August 2nd, 2010 | C-turtle | C-Turtle | -- |
| Box Turtle | March 2nd, 2010 | B-turtle | B-Turtle | -- |

## 安装指导

ROS基维百科上面已经有了很详细的系统安装说明指导，请用户根据自己的开发环境选择适当的版本进行安装，参考网址为<http://wiki.ros.org/ROS/Installation>。

比如Kinetic版本，支持多种发行版本安装（比如Ubuntu），同时也支持使用源代码进行安装。



1. Kinetic全局安装索引图示

这里做一下特别的说明，如果用户系统选用的是Ubuntu作为主机开发系统，在安装ROS及其依赖组件的时候，可以很方便的直接使用”sudo apt-get install ros-desktop-full”的方式来安装，目前仅支持ros-desktop-full。但是这种安装方式作者目前阶段暂时不推荐，因为通过这种安装方式存在着下面两个主要的问题和风险：

1） **rosdep并不能够解决ROS包的依赖关系**。用户根据安装指引从packages.ros.org安装时，可以使用rosdistro工具resdep来完成相关依赖包的安装，但是在Ubuntu使用apt-get直接安装的方式下，不能使用resdep来安装ROS依赖包。

2） **软件包集合并不完整**。目前不是所有的ROS相关软件包都upstream了，当前只有desktop-full可用，后续会陆续的添加其它软件包的支持。

推荐用户根据ROS基维百科的指引进行系统安装。

## 环境配置

为了进行ROS代码开发，开发者需要运行环境变量设置脚本。如果在Ubuntu通过APT命令安装的ROS，可以在“/opt/ros/<distro>”目录发现setup.\*sh文件，如果安装的是kinetic版本，则可以通过下面的命令导入ROS环境变量：

$source /opt/ros/kinetic/setup.bash

为了便于开发，可以将此命令添加到~/.bashrc文件中，这样用户开启命令行终端时实现自动导入。

## 文件系统工具

ROS由一系列的组件包组成，每一个组件包可以包含库文件、可执行文件、脚本或者其它；同时有一个名单文件package.xml，它描述了该软件包的一些基础信息，比如版本号、维护者、许可证信息等。

ROS提供了一系列的文件系统工具，使用这些工具，可以像浏览基本文件系统一样，浏览ROS各个组件，比如rospack, roscd, rosls等基本文件系统工具：

rospack list 能够列出所有的组件；

rospack find <package\_name> 列出组件对应的目录；

roscd <package\_name> 进入组件目录，通cd命令；

rosls <package\_name> 列出组件根目录下的所有文件及文件夹信息，通ls命令；

还有诸如其他ros前缀的文件系统工具，比如roscat等，用户可以在使用中熟悉和使用，它们能够让你更加快速和方面的在ros组件之间进行文件浏览和操作。

## 建立Catkin工作台

用户可以通过rosbuild和catkin来组织和构建自己的ROS代码。由于rosbuild已经不再维护，虽然合法但是不建议使用，推荐使用catkin。文档中就仅仅针对catkin来介绍如何创建工作台环境。

$mkdir -p ~/catkin\_ws/src

$cd ~/catkin\_ws/

$catkin\_make

命令catkin\_make会自动的创建一个CMakeLists.txt文件链接到“src”目录，并且同时生成“build”和“devel”目录。查看“devel”目录可以看到里面有一些setup.\*sh文件，使用source命令将setup.\*sh导入到你当前的环境变量中，通过echo命令查看当前工作台已经包含到了你的环境变量中。

$source devel/setup.bash

$echo $ROS\_PACKAGE\_PATH

/home/youruser/catkin\_ws/src:/opt/ros/kinetic/share

## 构建新Catkin Package

本小节介绍基于Catkin Workspace构建一个全新的Package组件。一个Catkin Workspace可以支持多个Package组件开发，基本目录结构具有下面的特征：

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

**步骤一**：进入Catkin Workspace的src目录，执行“catkin\_create\_pkg <packagename> [depend1] [depend2] [depend3]”命令来创建一个新的Package。如下面所示的例子，其中zhouyoujun\_ros\_tutorial表示我创建的Package名称，std\_msgs、rospy、roscpp为依赖组件包名称：

$cd ~/catkin\_ws/src

$catkin\_create\_pkg zhouyoujun\_ros\_tutorial std\_msgs rospy roscpp

当catkin\_create\_pkg执行完毕后，需要将新建的Package编译到Catkin Workspace中，在编译工作结束后，可以发现devel目录下生成了类似“/opt/ros/kinetic”的目录结构。

$cd ~/catkin\_ws

$catkin\_make

再一次的，将工作台的环境变量导入到ROS开发环境中，这样，使用rospack命令，可以找到我们刚刚新加入的组件包zhouyoujun\_ros\_tutorial。

$source ~/catkin\_ws/devel/setup.bash

$rospack find zhouyoujun\_ros\_tutorial

/home/zhouyoujun/catkin\_ws/src/zhouyoujun\_ros\_tutorial

**步骤二**：查看包依赖关系。在使用catkin\_create\_pkg命令的时候指定了的直接包依赖关系，可以通过“rospack depends1 zhouyoujun\_ros\_tutorial”命令查看。同时，被直接依赖的包本身也存在包依赖，我们称之为间接依赖。所有的直接依赖、间接依赖，都可以通过“rospack depends zhouyoujun\_ros\_tutorial”命令实现完成的组件依赖关系查看。

**步骤三**：Customizing The New Package，用户修改和定制新生成的Package软件包。修改和定制也分为几个步骤，请参考下面的步骤描述。

1. 修改package.xml文件

1.1 修改description tag

<description>The zhouyoujun\_ros\_tutorial package</description>

1.2 修改maintainer tags

<!-- One maintainer tag required, multiple allowed, one person per tag -->

<!-- Example: -->

<!-- <maintainer email="jane.doe@example.com">Jane Doe</maintainer> -->

<maintainer email="zhouyoujun@todo.todo">user</maintainer>

前面三行是注释可以直接删除，将第四行内容改为自己maintainer的内容即可。

<maintainer email="zhouyoujun@huawei.com">zhouyoujun</maintainer>

1.3 修改license tags

<!-- 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>

前面三行是注释可以直接删除，将第四行内容根据实际的情况修改。

<license>BSD</license>

1.4 修改dependencies tags

<buildtool\_depend>catkin</buildtool\_depend>

<build\_depend>roscpp</build\_depend>

<build\_depend>rospy</build\_depend>

<build\_depend>std\_msgs</build\_depend>

<build\_export\_depend>roscpp</build\_depend>

<build\_export\_depend>rospy</build\_depend>

<build\_export\_depend>std\_msgs</build\_depend>

<exec\_depend>roscpp</exec\_depend>

<exec\_depend>rospy</exec\_depend>

<exec\_depend>std\_msgs</exec\_depend>

这里主要由四种依赖关系，buildtool\_depend、build\_depend、build\_export\_depend、exec\_depend，根据实际情况添加。

2. 修改CMakeLists.txt文件

后续再补充…

未完待续…

# ROS Graph Conception整体认知

**Client Libraries** (rospy, roscpp)

**Host Linux Operation System**

**Topics**

TCP/UDP MSGAPI

Nodelet API

**主机2**

**Package/App**

**Package/App**

**主机1**

data/messages

data/messages

**publish**

**subscribe**

**subscribe**

1. ROS Graph Concepts

**Nodes**：执行单元，可以和ROS其它Node进行消息/数据通信。

**Master**：ROS Name Service，比如帮助节点相互进行查找，类似于网络中的路由功能。

**Topics**：Nodes节点可以通过publish message到一个topic，其它节点通过subscribe topic从而获取发布的message。

**Client Libraries**：允许使用不同语言编写的程序节点间进行通信，比如C++、Python应用程序中的Node进行通信。如果通信节点在本地机器或者设备，则消息或者数据使用Nodelet API本地共享；如果通信节点分布在不同的机器或者设备之间，则消息或者数据使用标准的TCP/UDP协议进行传输。

# ROS Node基础知识

ROS Nodes使用Client Library和系统中的其他Nodes进行通信。通过Node能够向一个Topic发布消息，或者订阅该Topic。

当运行ROS的时候，第一件事就是运行roscore，roscore包含master、rosout（等价于stdout/stderr）、以及parameter service（ps在后续章节有专门的介绍）等。

$roscore

留意roscore是否执行成功，如果执行失败，可以尝试确认网络配置是否有问题；另外如果上报权限错误，可以检测~/.ros文件夹是否被root账号拥有，否则改回当前用户所有。

一旦roscore成功执行后，它会一直占有当前开启的终端。我们可以重新开启一个新的终端，执行“rosnode list”命令，来查看roscode运行了那些nodes。

$rosnode list

/rosout

可以看到当前就仅有rosout一个节点在运行，可以使用“rosnode info <nodes>”命令来查看节点信息。

$rosnode info /rosout

------------------------------------------------------------------------

Node [/rosout]

Publications:

\* /rosout\_agg [rosgraph\_msgs/Log]

Subscriptions:

\* /rosout [unknown type]

Services:

\* /rosout/set\_logger\_level

\* /rosout/get\_loggers

contacting node http://machine\_name:54614/ ...

Pid: 25979

使用“rosrun <package\_name> <node\_name>”命令来直接运行某个package里面的一个node。

$rosrun turtlesim turtlesim\_node

执行这条命令会弹出一个画有小乌龟的窗口，不要关闭它，再另起一个终端查看node信息：

$rosnode list

/rosout

/turtlesim

另外我们可以给节点从新命名，只需要在执行rosrun命令的时候添加\_\_name参数，比如：

$rosrun turtlesim turtlesim\_node \_\_name:=zyj\_turtle

启动后我们查看节点信息将不再是“/turtlesim”，而是变成了“/zyj\_turtle”节点，然后我们可以使用“rosnode ping <node\_name>”来检测该节点运行是否正常。

$rosnode ping zyj\_turtle

rosnode: node is [/zyj\_turtle]

pinging /zyj\_turtle with a timeout of 3.0s

xmlrpc reply from http://ubuntu:39730/ time=0.417948ms

# ROS Topics基础知识

接着上一章节，我们运行了turtlesim\_node显示一个小乌龟的窗口，现在我们运行另外一个节点turtle\_teleop\_key来和turtlesim\_node进行通信。其中turtle\_teleop\_key节点向一个topic发布方向键按键消息，turtlesim\_node从同一个topic订阅接收方向按键消息，并根据按键驱动窗口的小乌龟运动。（备注，如果小乌龟不动，请使用鼠标点击turtle\_teleop\_key的终端窗口，然后按键输入）

使用工具rqt\_graph来统计和显示topic状态信息，它提供了一个动态的显示界面来跟踪当前的系统运行状态。如果没有安装rqt，需要先安装rqt之后才能使用：

$sudo apt-get install ros-kinetic-rqt

$sudo apt-get install ros-kinetic-rqt-cmmon-plugins

# ROS Services and Parameters基础知识

# ROS-II学习和了解

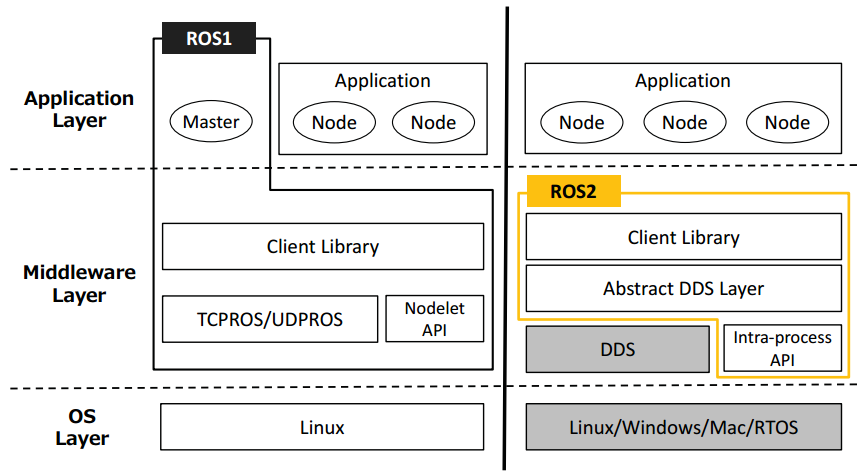
ROS2使用了更加先进的分布式架构，具备更高的可靠性，以及对实时性的优化，甚至于嵌入式设备的支持。和ROS最大的不同之处在于，ROS2采用了数据分发服务（Data-Distribution-Service）。同时ROS2意图解决ROS一直以来的架构局限，比如：

* 多个机器人组成的集群，ROS单Master结构不能很好的满足。
* 小型嵌入式平台的支撑，减少对Ubuntu的依赖负重。
* 提供进程间和跨机器建的通信实时性。
* 减小网络延迟，ROS存在打的延时，甚至在断网重新连接的时候会挂死。
* 提供可靠性，满足生产制造需求。

从具体的技术角度来看，ROS2新增的技术包括：

* Zeroconf.
* Protocol Buffers.
* Zero MQ and the other MQs.
* Redis.
* Data Distribution Service.
* WebSockets.

## ROS2 vs ROS整体架构对比



1. ROS2 vs ROS整体架构对比图

操作系统层面上，ROS主要构建与Linux系统之上，ROS2除了Linux，还支持Windows、Mac、RTOS、甚至Bare Metal。

通信系统，ROS通信系统基于TCPROS/UDPROS，强依赖于Master节点的处理，如果Master节点出问题，会波及到整个系统。而ROS2的架构中，不再存在Master节点，它的通信系统都是基于DDS实现，同时在ROS2内部提供了DDS的抽象层实现，有了这个抽象层，用户就可以不去关注底层的DDS使用了那个商家的API。

应用层，ROS的基本概念就是就发布/订阅的模型实现节点使用，可以让开发者并行开发低耦合的功能模块，并且便于进行二次复用。ROS2 DDS加入后，发布/订阅模型也会发生相应的改变，后面会针对这块进行专门的分析。

另外还有两个独立的模块**Nodelet** for ROS and **Intra-Process** for ROS2，它们都是为同一个进程的多个节点提供更加优化的数据传输方式。它们独立于TCPROS/UDPROS或者DDS，详情请参考关于它们的独立章节分析。

## ROS2整体框架

## Inter-Process API

## 子标题

## 子标题

# 参考文献

[1] <https://github.com/ros>

[2] <http://wiki.ros.org> 从这里出发，开始了解ROS世界。

[3]