

ROS简介

什么是ROS?

- ROS 是一个适用于机器人的开源的元操作系统
- 提供了操作系统应有的服务，包括硬件抽象、底层设备控制、常用函数的实现、进程间消息传递、以及包管理
- 也提供用于获取、编译、编写和跨计算机运行代码所需的工具和库函数
- 在某些方面ROS相当于一种“机器人框架（robot frameworks）”类似的“机器人框架”有：Player, YARP, Orocosp, CARMEN, Orca, MOOS和 Microsoft Robotics Studio。

起源

- 始于2007年，斯坦福大学，Switchyard，
- 2008，Willow Garage初创公司开发
- 2013，开源机器人基金会OSRF

ROS生态

- 完全开源



<https://www.ros.org/blog/ecosystem/>

管道

- ROS的核心是提供一个消息传递系统，通常称为“中间件”或“管道”
- ROS的内置且经过良好测试的消息传递系统通过匿名发布/订阅模式管理分布式节点之间的通信细节，从而节省了时间。
- 这种方法鼓励软件开发中的良好实践，包括故障隔离、关注点分离和清晰的接口。
- 使用ROS可以使系统更容易维护、贡献和重用。

工具

- 作为高效的开发工具，ROS具有丰富的功能，包括：启动、自省、调试、可视化、绘图、日志记录和回放。这些工具可以加快开发团队的进度，并且可以在发布产品时将其包含在产品中

能力

- ROS生态系统具有丰富的机器人软件。无论是GPS的设备驱动程序，四足动物的行走和平衡控制器，还是移动机器人的地图系统，ROS都能满足需求。从驱动程序到算法，再到用户界面，借助ROS提供的构建块，使开发者能够专注于应用程序
- ROS项目的目标是不断降低构建机器人应用程序的门槛。任何人只要有一个有用的(或有趣的)机器人的好想法，都应该能够使这个想法成为现实，而不必了解底层硬件和软件的所有内容。

社区

- ROS社区是一个庞大、多样和全球性的社区。从学生和业余爱好者到跨国公司和政府机构，各行各业的个人和组织都在推动ROS项目的发展。
- Open Robotics是该项目的社区中心和中立管家，它托管共享的在线服务(如ros.org)，创建和管理发行版(包括安装的二进制包)，并开发和维护ROS中的大部分核心软件。Open Robotics还提供与ROS相关的工程服务。

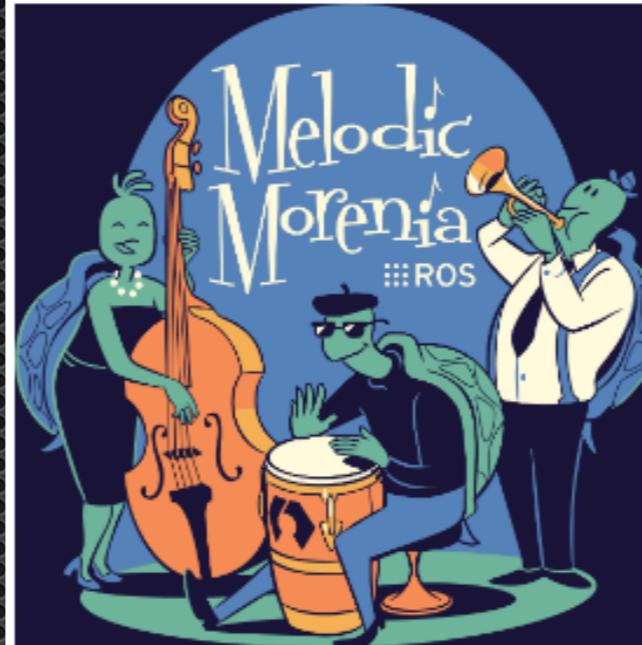
ROS发行版

ROS Melodic Morenia

发布于2018年5月

LTS长期支持到2023年5月

建议在Ubuntu 18.04上安装



ROS Noetic Ninjemys

发布于2020年5月

最新的LTS长期支持到2025年5月

建议在Ubuntu 20.04上安装



Get ROS Noetic
Ninjemys on Ubuntu
Linux

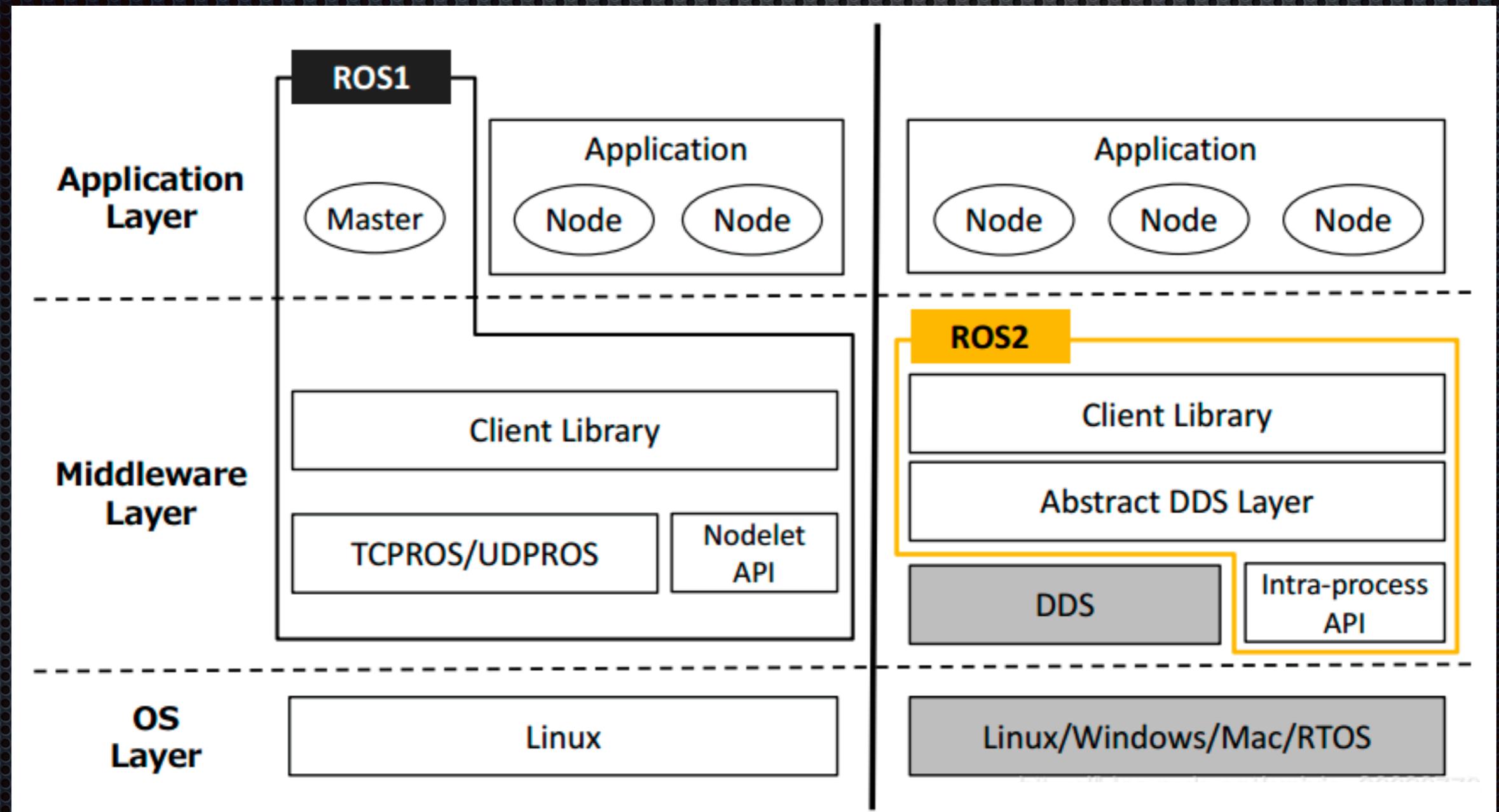


Get Humble Hawksbill on
Ubuntu Linux 22.04,
Windows 10



Get Iron Irwini on
Ubuntu Linux 22.04,
Windows 10

ROS1与ROS2



ROS1版本

Ubuntu	ROS1	Release date	End of Life
18.04 LTS	Melodic Morenia	May 23rd, 2018	May, 2023
20.04 LTS	Noetic Ninjemys (Recommended)	May 23rd, 2020	May, 2025

<https://www.ros.org/reps/rep-2000.html>

ROS2

- Tier 1 Platforms: 积极支持的平台（定期维护），兼容性最好，经过良好测试，提供二进制安装包。
- Tier 2 Platforms: 支持的平台（按要求维护），未进行积极测试且不定期维护。因此，发布的产品可能不会在这些平台上开箱即用，但报告的问题将在客户分析/报告/修复/重新测试时得到修复/维护。不提供二进制安装包，必须从源码编译。
- Tier 3 Platforms: 已知可用的平台，未得到官方支持或测试。

ROS2与操作系统对应关系

ROS2

Foxy Fitzroy

Galactic
Geochelone

Humble Hawksbill
(Recommended)

Iron Irwini
(Recommended)

Tier 1 Platforms

Ubuntu 20.04
(Focal): amd64 and arm64
Mac macOS 10.14 (Mojave)
Windows 10 (Visual Studio 2019)

Ubuntu 20.04
(Focal): amd64 and arm64
Windows 10 (Visual Studio
2019): amd64

Ubuntu 22.04
(Jammy): amd64 and arm64
Windows 10 (Visual Studio
2019): amd64

Ubuntu 22.04
(Jammy): amd64 and arm64
Windows 10 (Visual Studio
2019): amd64

Tier 2
Platforms

RHEL 8: amd64

RHEL 8: amd64

RHEL 9: amd64

Tier 3 Platforms

Ubuntu 20.04 (Focal): arm32
Debian Buster (10): amd64, arm64 and arm32
OpenEmbedded Thud (2.6) / webOS
OSE: arm32 and x86

Ubuntu 20.04 (Focal): arm32
Debian Bullseye (11): amd64, arm64 and arm32
OpenEmbedded Thud (2.6) / webOS OSE: arm32
and arm64
Mac macOS 10.14 (Mojave): amd64

Ubuntu 20.04 (Focal): amd64
macOS: amd64
Debian Bullseye: amd64

macOS: amd64
Debian Bullseye: amd64

支持的机器人

Find a robot by category:

[Aerial](#)



[Component](#)



[Ground](#)



[Manipulator](#)



[Marine](#)



ROS支持的传感器

- 2D range finders
- 3D Sensors
- Pose Estimation (GPS + IMU)
- Cameras
- Sensor Interfaces

<https://wiki.ros.org/Sensors>

支持的执行器

- 电机控制器
- 伺服控制器

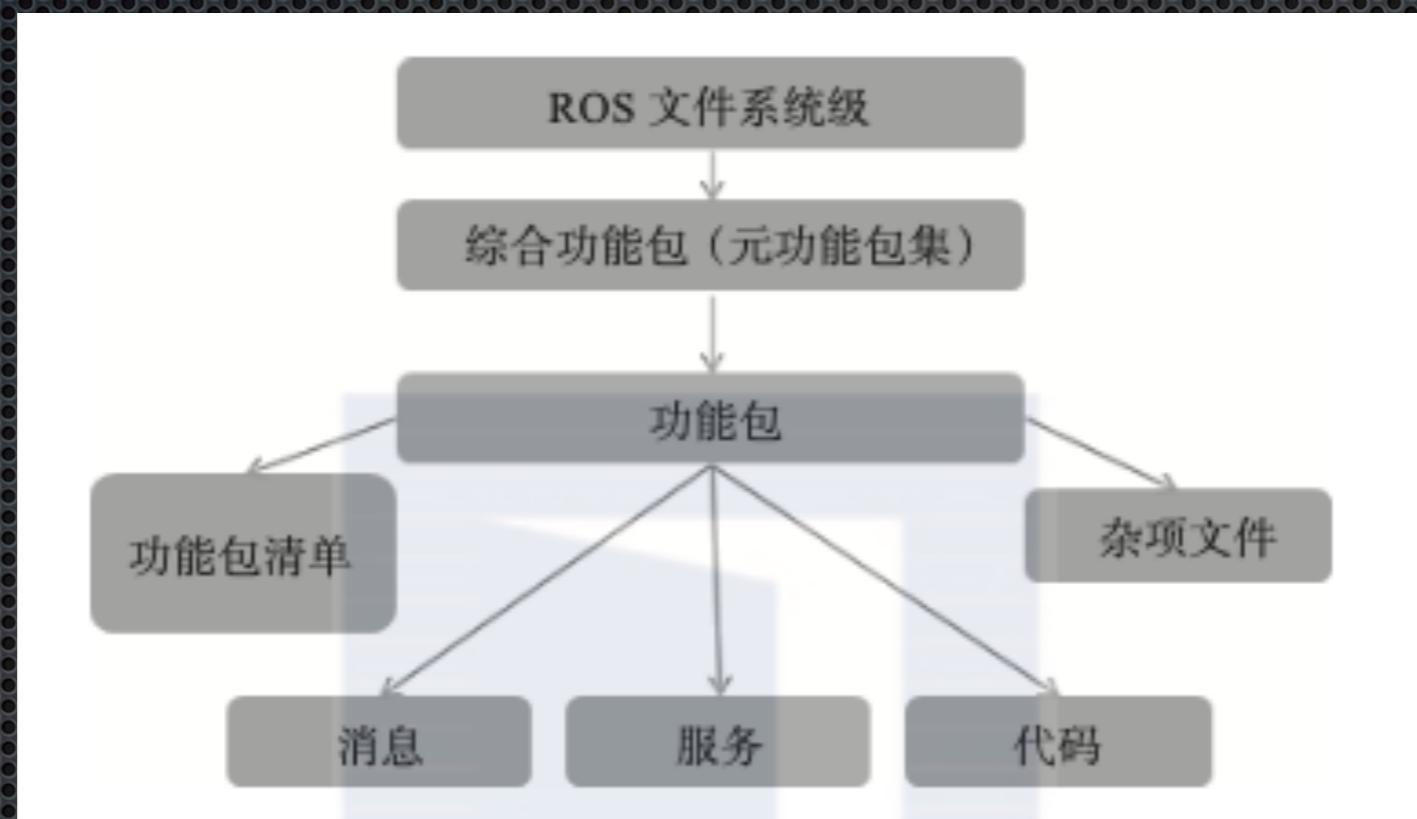
<https://wiki.ros.org/Motor%20Controller%20Drivers>

为什么选择ROS

- 协作开发：开源的，可以自由地用于工业和研究领域。开发人员可以通过添加功能包来扩展 ROS 的功能。几乎所有的 ROS 包都在硬件抽象层上工作，可以轻松地为其他机器人重用。
- 语言支持：ROS 通信框架可以轻松地以任何现代编程语言实现。已经支持流行的语言，如 C++、Python 和 Lisp，还有 Java 和 Lua 的实验库。
- 库集成：ROS 具有许多第三方机器人库的接口，如开源计算机视觉(Open Source Computer Vision, Open-CV)、点云库(Point Cloud Library, PCL)、Open-NI、Open-Rave 和 Orocos。
- 集成化仿真器：ROS 与开源仿真器(如 Gazebo)有着紧密的联系，并且具有堪比专业仿真器的良好界面，如 Webots 和 V-REP。
- 代码测试：ROS 提供了一个名为 rostest 的内置测试框架来发现代码错误，提高代码质量。
- 可扩展性：ROS 框架的宗旨就是可扩展性。我们可以通过在云端或异构集群上让使用 ROS 的机器人执行大量的计算任务。
- 可定制性：正如前面所述，ROS 是完全开放源代码且免费的，可以根据机器人的要求自定义框架。如果只想使用 ROS 消息传递平台，可以删除所有其他组件，甚至可以为特定机器人定制 ROS 以获得更好的性能。
- 社区：ROS 是社区驱动的项目，主要由 OSRF 领导。

ROS文件系统级

- 文件系统级解释了ROS文件是如何在硬盘上组织的

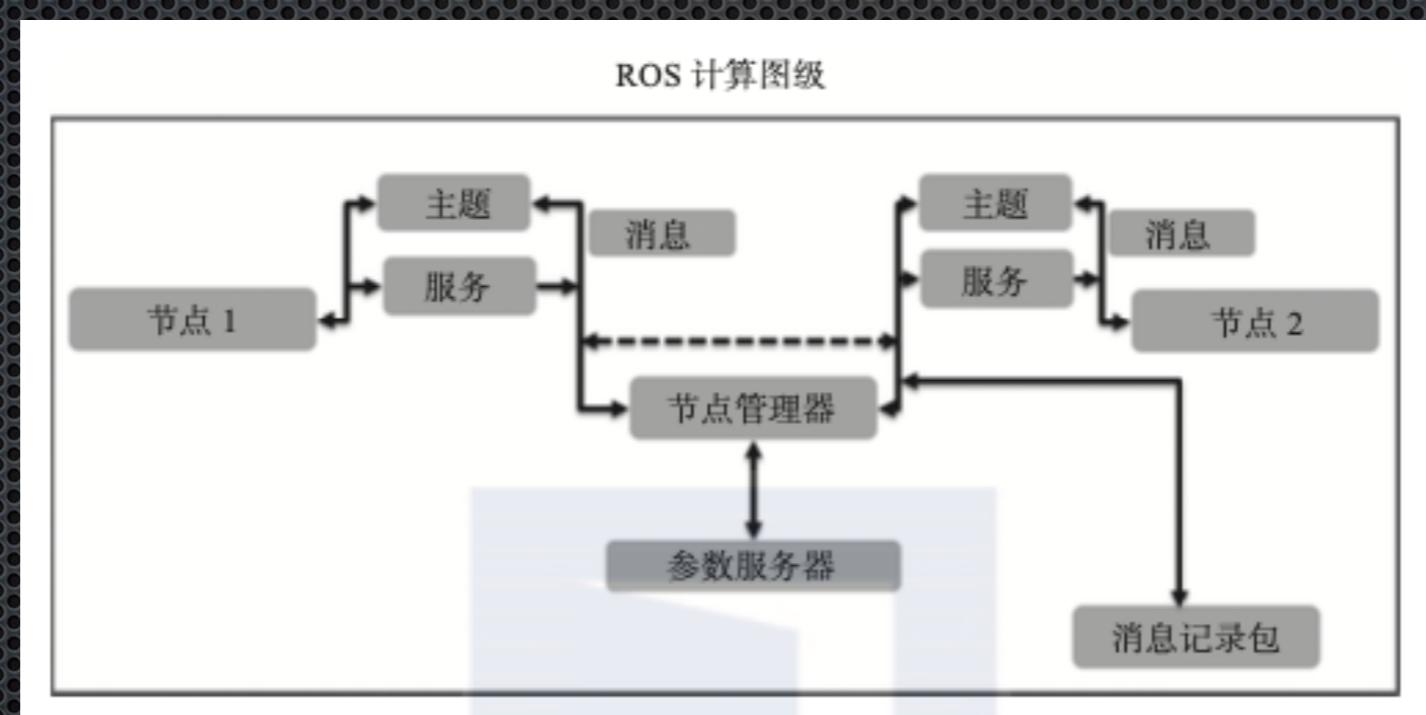


ROS文件系统级

- **综合功能包:** 综合功能包将特定应用程序包的列表组合在一起。例如，在ROS中，有一个称为导航(navigation)的综合功能包用于移动机器人导航。它可以保存相关功能包的信息，并有助于在自己的安装过程中安装这些功能包。
- **功能包:** ROS 软件主要以ROS功能包的形式组织。可以说ROS功能包是ROS的元构建单元。一个功能包的单个模块中可能包括ROS节点/进程、数据集和配置文件。
- **功能包清单:** 每个包都有一个名为package.xml的清单文件。该文件包括包的名称、版本、作者、许可证和依赖项等信息。综合功能包的package.xml文件由相关功能包的名称组成。
- **消息(msg):** ROS通过发送ROS消息进行通信。可以在具有.msg扩展名的文件中定义消息数据的类型。这些文件称为消息文件。按照约定，消息文件保存在our_package/msg/message_files.msg下。
- **服务(srv):** 计算图级概念之一是服务。与ROS消息类似，约定服务定义放在our_package/srv/service_files.srv下。

ROS计算图级

- ROS 计算图是ROS进程的对等网络，并能处理数据。ROS 计算图概念包括节点、主题、消息、主节点(节点管理器)、参数服务器、服务和消息记录包。



概念描述

- 节点:ROS 节点是一个使用 ROS API 进行通信的进程。机器人可能有许多节点来执行计算。
- 节点管理器:ROS 节点管理器作为一个中间节点，用于在不同 ROS 节点之间建立连接。节点管理器具有 ROS 环境中运行的所有节点的全部细节。它将与另一个节点交换细节，建立它们之间的连接。交换信息后，这两个 ROS 节点将开始通信。
- 参数服务器:参数服务器在 ROS 中非常有用。节点可以在参数服务器中存储变量，并设置其私有属性。如果参数具有全局域，则可以由其他所有节点访问。ROS 参数与 ROS 节点管理器一起运行。
- 消息:ROS 节点可以通过多种方式相互通信。在所有方法中，节点以 ROS 消息的形式发送和接收数据。ROS 消息是 ROS 节点用于交换数据的数据结构。
- 主题:ROS 主题是两个 ROS 节点之间通信和交换 ROS 消息的方法之一。主题是命名的总线，使用 ROS 消息交换数据。每个主题都有一个特定的名称，一个节点会将数据发布到主题，另一个节点可以通过订阅从主题中读取。
- 服务:服务是不同于主题的另一种通信方式。主题使用发布或订阅通信，但在服务中，使用的是请求或回复的方法。一个节点将作为服务提供者，运行服务例程，客户端节点可从服务器节点请求服务。服务器将执行服务程序并将结果发送给客户端。客户端节点等待服务器响应结果。
- 消息记录包:消息记录包是 ROS 中用于录制和播放 ROS 主题的实用工具。在机器人工作时，可能会出现某些情况，我们需要在没有实际硬件的情况下工作。使用 rosbag，可以记录传感器数据，并可以将消息记录包文件复制到其他计算机，通过回放来检查数据。

社区级

- ROS 社区级包括 ROS 软件和知识等共享资源
 - 发行版:ROS 发行版是 ROS 包的版本集合，如同 Linux 发行版。
 - 软件库:与 ROS 相关的功能包和文件取决于版本控制系统(Version-Control System, VCS)，例如 Git、SVN 和 Mercurial，使用来自世界各地的开发人员贡献的功能包。
 - ROS 维基:ROS 社区维基是 ROS 的知识中心，任何人都可以为其创建文档。可以从 ROS 维基找到有关 ROS 的标准文档和教程。
 - 邮件列表:订阅 ROS 邮件列表使用户能够获得关于 ROS 功能包的更新推送，同时他们也可以在此提问有关 ROS 的问题(<http://wiki.ros.org/Mailing%20Lists>)。
 - ROS 问答:ROS 问答网站是 ROS 的问题反馈(Stack Overflow)。用户可以询问有关 ROS 和相关领域的问题(<http://answers.ros.org/questions/>)。
 - 博客:ROS 博客通过照片和视频等定期更新有关 ROS 社区的最新动态(<http://www.ros.org/news>)。

catkin软件包组成

- 一个包要想称为catkin软件包，必须符合以下要求：
 - 这个包必须有一个符合catkin规范的package.xml文件
 - 这个package.xml文件提供有关该软件包的元信息
 - 这个包必须有一个catkin版本的CMakeLists.txt文件
 - 如果它是个Catkin元包的话，则需要有一个CMakeList.txt文件的相关样板
 - 每个包必须有自己的目录
 - 这意味着在同一个目录下不能有嵌套的或者多个软件包存在

- 最简单的软件包看起来就像这样：

```
my_package/  
CMakeLists.txt  
package.xml
```

catkin工作空间中的软件包

- 开发catkin软件包的推荐方法是使用catkin工作空间，但是你也可以单独开发catkin软件包。一个简单的工作空间如下所示：

```
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
```

Catkin工作空间

```
workspace_folder/      -- WORKSPACE
src/                  -- SOURCE SPACE
CMakeLists.txt        -- The 'toplevel' CMake file
package_1/
    CMakeLists.txt
    package.xml
...
package_n/
    CATKIN_IGNORE   -- Optional empty file to exclude package_n from being processed
    CMakeLists.txt
    package.xml
...
build/                -- BUILD SPACE
    CATKIN_IGNORE   -- Keeps catkin from walking this directory
devel/                -- DEVELOPMENT SPACE (set by CATKIN_DEVEL_PREFIX)
    bin/
    etc/
    include/
    lib/
    share/
    .catkin
    env.bash
    setup.bash
    setup.sh
...
install/              -- INSTALL SPACE (set by CMAKE_INSTALL_PREFIX)
    bin/
    etc/
    include/
    lib/
    share/
    .catkin
    env.bash
    setup.bash
```

创建catkin软件包

- 本部分教程将演示如何使用catkin_create_pkg脚本来创建一个新的catkin软件包，以及创建之后都能做些什么。
- 首先切换到刚才创建的空白catkin工作空间中的源文件空间目录：
 - \$ cd ~/catkin_ws/src
- 现在使用catkin_create_pkg命令创建一个名为beginner_tutorials的新软件包，这个软件包依赖于std_msgs、roscpp和rospy：
 - \$ catkin_create_pkg beginner_tutorials std_msgs rospy roscpp
- 这将会创建一个名为beginner_tutorials的文件夹，这个文件夹里面包含一个package.xml文件和一个CMakeLists.txt文件，这两个文件都已经部分填写了在执行catkin_create_pkg命令时提供的信息。
- catkin_create_pkg命令会要求你输入package_name，如有需要还可以在后面添加一些需要依赖的其它软件包：
 - # This is an example, do not try to run this
 - # catkin_create_pkg <package_name> [depend1] [depend2] [depend3]
- catkin_create_pkg命令也有更多的高级功能，这些功能在catkin/commands/catkin_create_pkg中有描述。

构建一个catkin工作区并生效配置文件

- 现在需要在catkin工作区中构建软件包：
 - \$ cd ~/catkin_ws
 - \$ catkin_make
- 工作空间构建完成后，在devel子目录下创建了一个与通常在 /opt/ros/\$ROSDISTRO_NAME下看到的目录结构类似的结构。
- 要将这个工作空间添加到ROS环境中，需要source一下生成的配置文件：
 - \$. ~/catkin_ws/devel/setup.bash

参考文献

- [http://wiki.ros.org/cn/ROS/Tutorials/
UnderstandingNodes](http://wiki.ros.org/cn/ROS/Tutorials/UnderstandingNodes)
- <http://wiki.ros.org/cn/ROS/Tutorials/BuildingPackages>