# 实验一：ROS环境配置与实践

## 实验目的

* 熟悉Ubuntu系统的基本操作
* 熟悉Ubuntu系统的安装
* 了解ROS的基本功能
* 熟悉ROS的环境配置

## 实验原理

### Ubuntu系统简介



1. Ubuntu系统

Ubuntu是基于Debian的桌面型为主的Linux版本，包含了桌面版、服务器版、IoT版、云镜像四个版本，其主要应用场景为物联网设备和机器人。Ubuntu是建立在Debian的架构和基础设施之上的，包括Linux服务器、桌面和已停产的手机和平板电脑操作系统版本Ubuntu可以预见每六个月发布更新版本，长期支持(LTS)每两年发布一次。截至2021年4月22日，最新的长期支持版本是20.04(“Focal Fossa”)，公众支持到2025年，公司支持到2030年，注册的个人免费。Ubuntu的安装，支持在计算机上单独安装、和Windows系统进行双系统安装、在VMware等虚拟机上安装。

Ubuntu在使用过程中，用户程序以低权限运行，不会破坏操作系统或其他用户的文件。为了提高安全性，sudo工具用于为执行管理任务分配临时特权，这允许root帐户保持锁定，并帮助防止缺乏经验的用户不经意地做出灾难性的系统更改或打开安全漏洞Polkit也被广泛地应用到桌面中。

Ubuntu版本列表

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 代号 | 发布时间 | 维护截止时间 |
| 4.1 | Warty Warthog | 2004/10/20 | 2006/4/30 |
| 5.04 | Hoary Hedgehog | 2005/4/8 | 2006/10/31 |
| 5.1 | Breezy Badger | 2005/10/13 | 2007/4/13 |
| 6.06 LTS | Dapper Drake | 2006/6/1 | 2009/7/14 |
| 6.1 | Edgy Eft | 2006/10/26 | 2008/4/25 |
| 7.04 | Feisty Fawn | 2007/4/19 | 2008/10/19 |
| 7.1 | Gutsy Gibbon | 2007/10/18 | 2009/4/18 |
| 8.04 LTS | Hardy Heron | 2008/4/24 | 2011/5/12 |
| 8.1 | Intrepid Ibex | 2008/10/30 | 2010/4/30 |
| 9.04 | Jaunty Jackalope | 2009/4/23 | 2010/10/23 |
| 9.1 | Karmic Koala | 2009/10/29 | 2011/4/30 |
| 10.04 LTS | Lucid Lynx | 2010/4/29 | 2013/5/9 |
| 10.1 | Maverick Meerkat | 2010/10/10 | 2012/4/10 |
| 11.04 | Natty Narwhal | 2011/4/28 | 2012/10/28 |
| 11.1 | Oneiric Ocelot | 2011/10/13 | 2013/5/9 |
| 12.04 LTS | Precise Pangolin | 2012/4/26 | 2019/4/26 |
| 12.1 | Quantal Quetzal | 2012/10/18 | 2014/5/16 |
| 13.04 | Raring Ringtail | 2013/4/25 | 2014/1/27 |
| 13.1 | Saucy Salamander | 2013/10/17 | 2014/7/17 |
| 14.04 LTS | Trusty Tahr | 2014/4/17 | 2024/4/25 |
| 14.1 | Utopic Unicorn | 2014/10/23 | 2015/7/23 |
| 15.04 | Vivid Vervet | 2015/4/23 | 2016/2/4 |
| 15.1 | Wily Werewolf | 2015/10/22 | 2016/7/28 |
| 16.04 LTS | Xenial Xerus | 2016/4/21 | 2026/4/30 |
| 16.1 | Yakkety Yak | 2016/10/13 | 2017/7/20 |
| 17.04 | Zesty Zapus | 2017/4/13 | 2018/1/13 |
| 17.1 | Artful Aardvark | 2017/10/19 | 2018/7/19 |
| 18.04 LTS | Bionic Beaver | 2018/4/26 | 2028/4/26 |
| 18.1 | Cosmic Cuttlefish | 2018/10/18 | 2019/7/18 |
| 19.04 | Disco Dingo | 2019/4/18 | 2020/1/23 |
| 19.1 | Eoan Ermine | 2019/10/17 | 2020/7/17 |
| 20.04 LTS | Focal Fossa | 2020/4/23 | 2030/4/23 |
| 20.1 | Groovy Gorilla | 2020/10/22 | 2021/7/22 |
| 21.04 | Hirsute Hippo | 2021/4/22 | 2022/1/20 |
| 21.1 | Impish Indri | 2021/10/14 | 2022/7/14 |
| 22.04 LTS | Jammy Jellyfish | 2022/4/21 | 2032/4/21 |

### Ubuntu系统常用操作

**终端（Terminal）**

终端常常被称为命令行或者shell。过去，这就是人机交互的方式。即便现在，Linux 用户还是觉得shell比图形方式更快并且仍有一些优势。接下来，将学习如何使用终端。终端最初用作文件浏览器，即使现在，遇上图形环境失效时，它仍被用作文件浏览器。可以将终端作为一个文件浏览器，用来浏览自己的文件和撤销曾做过的改动。

**启动终端（Terminal）**

* 在Ubuntu下启动终端，可以在应用程序下搜索终端
* 快捷方式：Ctrl+Alt+T（新打开一个终端），Ctrl+Shift+T（并列打开一个终端）

**常用命令**

* 查看目录：ls

ls (List) 用不同颜色、经过排列的文本列出目录下的文件。

* 创建目录： mkdir (目录名)

mkdir (MakeDiRectory) 命令可以创建目录。

* 切换目录： cd (directory/location)

cd (ChangeDirectory) 命令可以从您的当前目录切换到您指定的任意目录。

* 复制文件/目录： cp (源文件或目录名) (目标目录或文件名)

cp (copy) 命令会拷贝您指定的任意文件。cp -r 命令则可以拷贝您指定的任意目录（注：包括该目录里的文件和子目录）。

* 删除文件/目录： rm （文件或目录名）

rm (remove) 可以删除您指定的任意文件。rm -rf 命令则可以删除您指定的任意目录（注：包括该目录里的文件和子目录）。

* 重命名文件/目录： mv （文件或目录名）

mv (Move) 命令可以重命名/移动您指定的任意文件或目录。

* 查找文件/目录：locate（文件或目录名）

1. 切换到 root 用户 ，输入 “sudo -i ”或“sudo su -”, 退出 “exit”

2. pwd 显示当前目录， pwd = print working directory

3. ls 列出目录下当前文件

4. cp 复制文件/目录 cp (源文件或目录) (目标文件或目录)

cp -r 复制文件夹 包括子目录和文件

5. rm 删除文件/目录 可以删除文件

rm -rf 删除目录包含子目录和文件

rmdir 删除空文件夹

6. mv 移动或重命名 文件

7. cd 进入目录

cd / 进入根目录

cd 或 cd ~ 进入用户的 home 目录

cd - 进入上次访问的目录 (相当于 back)

cd .. 进入上级目录

8. man 显示某个命令的 manual

9. df 显示文件系统空间信息

df -h 用 M 和 G 做单位显示文件系统空间信息 -h 意思是 human-readable

10. du 显示目录的空间使用信息

du -sh /media/floppy

-s 意思 summary -h 意思 human-readable

11. ifconfig 显示系统的网络

12. locate 命令会在您的计算机里搜索您指定的任意文件。它使用您系统中的文件索引以便进行快速查找：运行命令 updatedb 可以更新该索引。每天您一开机，该命令便会（在合适的时机）自动运行。运行该命令需要具备管理员权限 (参见 第1.3.3节 ― root 用户和 sudo 命令)。

**文本编辑**

gedit是Ubuntu的默认文本编辑器，通过点击桌面菜单系统的应用程序总搜索文本编辑器来启动它，也可以使用终端中输入gedit来启动。有时为了修改配置文件，会从命令行里使用sudo来运行gedit，这样就能以管理权限来运行gedit。

**添加、删除和更新应用程序**

为了在 Ubuntu 中添加或删除应用程序，您需要使用 软件包管理器。通过将软件处理成为 Ubuntu 优化的预配置软件包，软件包管理器工具可以容易地安装和删除这些应用程序。

APT (Advanced Package Tool,高级软件包工具) 是一个强大的包管理系统，而那些图形化程序如 添加/删除 应用程序 和 Synaptic 都是建立在它的基础之上的。APT 自动处理依赖关系并在系统软件包执行其他操作以便安装所要的软件包。 运行 APT 要求管理权限 (参见 第1.3.3节 ― root 用户和sudo命令)。可以被 APT 使用的一些常用命令：

* 安装软件包：

sudo apt-get install packagename

* 删除软件包：

sudo apt-get remove packagename

* 获取新的软件包列表：

sudo apt-get update

* 升级有可用更新的系统：

sudo apt-get upgrade

* 列出更多命令和选项：

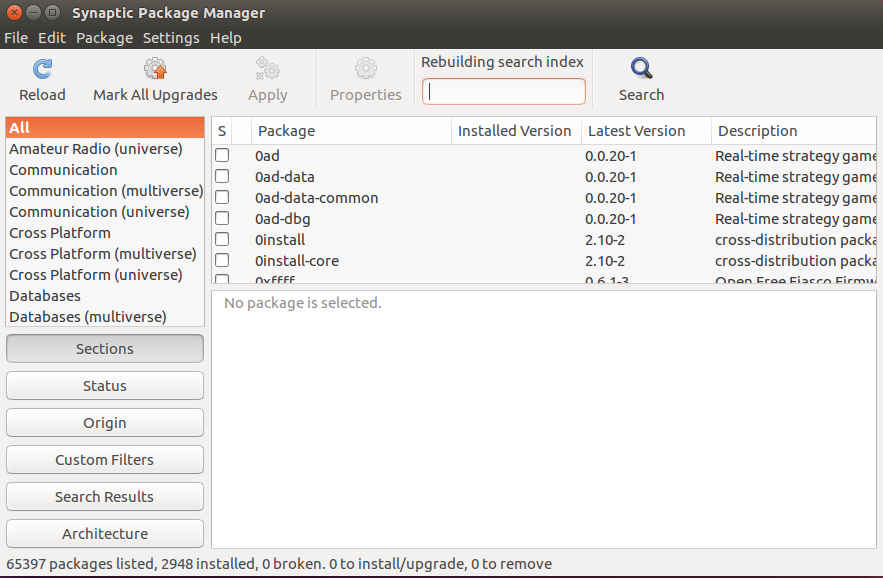
apt-get help

**安装新立得软件包管理器**

打开终端，输入以下命令：

sudo apt-get install synaptic

安装成功后，如下如所示：



安装synaptic后，启动软件没有看到上图中红色线框中的快速搜索框，安装apt-xapian-index插件，在终端中输入

sudo apt-get install apt-xapian-index

### 机器人控制系统简介

**机器人控制系统特点**

多数机器人的结构是一个空间开链结构，各个关节的运动是相互独立的，为了实现机器人末端执行器的运动，需要多关节协调运动，因此，机器人控制系统与普通的控制系统比较，要复杂一些。具体来讲，机器人控制系统主要具有以下特点。

(1)机器人控制系统是一个多变量控制系统，即使简单的工业机器人也有3～5个自由度，比较复杂的机器人有十几个自由度，甚至几十个自由度，每个自由度一般包含一个伺服机构，多个独立的伺服系统必须有机地协调起来。例如，机器人的手部运动是所有关节的合成运动，要使手部按照一定的轨迹运动，就必须控制各关节协调运动，包括运动轨迹、动作时序等多方面的协调。

(2)运动描述复杂，机器人的控制与机构运动学及动力学密切相关。描述机器人状态和运动的数学模型是一个非线性模型，随着状态的变化，其参数也在变化，各变量之间还存在耦合。因此，仅仅考虑位置闭环是不够的，还要考虑速度闭环，甚至加速度闭环。在控制过程中，根据给定的任务，应当选择不同的基准坐标系，并做适当的坐标变换，求解机器人运动学正问题和逆问题。此外，还要考虑各关节之间惯性力、哥氏力等的耦合作用和重力负载的影响，因此，系统中还经常采用一些控制策略，如重力补偿、前馈、解耦或自适应控制等。

(3)具有较高的重复定位精度，系统刚性好。除直角坐标机器人外，机器人关节上的位置检测元件不能安装在末端执行器上，而应安装在各自的驱动轴上，构成位置半闭环系统。但机器人的重复定位精度较高，一般为±0.1 mm。此外，由于机器人运行时要求运动平稳，不受外力干扰，为此系统应具有较好的刚性。

(4)信息运算量大。机器人的动作住往可以通过不同的方式和路径来完成，因此存在一个最优的问题，较高级的机器人可以采用人工智能的方法，用计算机建立起庞大的信息库，借助信息库进行控制、决策管理和操作。根据传感器和模式识别的方法获得对象及环境的工况，按照给定的指标要求，自动选择最佳的控制规律。

(5)需采用加（减）速控制。过大的加（减）速度会影响机器人运动的平稳性，甚至使机器人发生抖动，因此在机器人起动或停止时采取加（减）速控制策略。通常采用匀加（减）速运动指令来实现。此外，机器人不允许有位置超调，否则将可能与工件发生碰撞。因此，要求控制系统位置无超调，动态响应尽量快。

(6)工业机器人还有一种特有的控制方式——示教再现控制方式。当要工业机器人完成某作业时，可预先移动工业机器人的手臂来示教该作业顺序、位置及其他信息，在此过程中把相关的作业信息存储在内存中，在执行任务时，依靠工业机器人的动作再现功能，可重复进行该作业。此外，从操作的角度来看，要求控制系统具有良好的人机界面，尽量降低对操作者的要求。因此，多数情况要求控制器的设计人员不仅要完成底层伺服控制器的设计，还要完成规划算法的编程。

总之，机器人控制系统是一个与运动学和动力学密切相关的、紧耦合的、非线性的多变量控制系统。随着实际工作情况的不同，可以采用各种不同的控制方式。

**机器人控制系统功能**

机器人控制系统是机器人的主要组成部分，用于控制操作机来完成特定的工作任务，其基本功能有示教再现功能、坐标设置功能、与外围设备的联系功能、位置伺服功能。

(1)示教-再现功能。机器人控制系统可实现离线编程、在线示教及间接示教等功能，在线示教又包括示教盒示教和导引示教两种情况。在示教过程中，可存储作业顺序、运动路径、运动方式、运动速度及与生产工艺有关的信息，在再现过程中，能控制机器人按照示教的加工信息执行特定的作业。

(2)坐标设置功能。一般的工业机器人控制器设置有关节坐标、绝对坐标、工具坐标及用户坐标4种坐标系，用户可根据作业要求选用不同的坐标系并进行坐标系之间的转换。

(3)与外围设备的联系功能。机器人控制器设置有输入/输出接口、通信接口、网络接口和同步接口，并具有示教盒、操作面板及显示屏等人机接口。此外，还具有多种传感器接口，如视觉、触觉、接近觉、听觉、力觉（力矩）传感器等多种传感器接口。

(4)位置伺服功能。机器人控制系统可实现多轴联动、运动控制、速度和加速度控制、力控制及动态补偿等功能。在运动过程中，还可以实现状态监测、故障诊断下的安全保护和故障自诊断等功能。

**机器人的控制方式**

（1）点到点控制方式

点到点控制方式用于实现点的位置控制，其运动是由一个给定点到另一个给定点，而点与点之间的轨迹却无关紧要。因此，这种控制方式的特点是只控制工业机器人末端执行器在作业空间中某些规定的离散点上的位姿。控制时只要求工业机器人快速、准确地实现相邻各点之间的运动，而对达到目标点的运动轨迹则不做任何标记，如自动插件机，在贴片机上安插元件、点焊、搬运、装配等作业。这种控制方式的主要技术指标是定位精度和运动所需的时间，控制方式比较简单，但要达到较高的定位精度则较难。

（2）连续轨迹控制方式

连续轨迹控制方式用于指定点与点之间的运动轨迹所要求的曲线，如直线或圆弧。这种控制方式的特点是连续地控制工业机器人末端执行器在作业空间中的位姿，使其严格按照预先设定的轨迹和速度在一定的精度要求内运动，速度可控，轨迹光滑，运动平稳，以完成作业任务。工业机器人各关节连续、同步地进行相应的运动，其末端执行器可形成连续的轨迹。这种控制方式的主要技术指标是机器人末端执行器的轨迹跟踪精度及平稳性。在用机器人进行弧焊、喷漆、切割等作业时，应选用连续轨迹控制方式。

（3）速度控制方式

对机器人的运动控制来说，在位置控制的同时，还要进行速度控制，及对于机器人的行程要求遵循一定的速度变化曲线。例如，在连续轨迹控制方式下，机器人按照预设的指令，控制运动部件的速度，实现加、减速，以满足运动平稳、定位精确的要求。由于工业机器人是一种工作情况（行程负载）多变、惯性负载大的运动机械，控制过程中必须处理好快速与平稳的矛盾，必须注意起动后的加速和停止前的减速这两个过度运动阶段。

（4）力（力矩）控制方式

在进行抓放操作、去毛刺、研磨和组装等作业时，除了要求准确定位之外，还要求使用特定的力或力矩传感器对末端执行器施加在对象上的力进行控制。这种控制方式的原理与位置伺服控制原理基本相同，但输入量和输出量不是位置信号，而是力（力矩）信号，因此系统中必须有力（力矩）传感器。

（5）智能控制方式

在不确定或位置条件下作业，机器人需要通过传感器获得周围环境信息，根据自己内部的知识库做出决策，今儿对个执行机构进行控制，自主完成给定任务。若采用智能控制技术，机器人会具有较强的环境适应性及自学习能力。智能控制方法与人工智能的发展密切相关。

### 机器人软件控制平台

机器人的运动控制是通过包含运动控制算法的程序来实现的，而这些控制程序通常是通过机器人软件控制平台来开发的。一般来说，机器人软件控制平台是给多机器人设备进行程序开的软件包。机器人软件控制平台包含下列内容：

* 统一的编程环境
* 统一的编译执行环境
* 可重用的组建库
* 完备的调试/仿真环境
* 对多种机器人硬件设备的“驱动”程序支持
* 通用的常用功能控制组件，例如计算机视觉技术、导航技术和机械手臂控制等。

下面介绍一下目前主流的机器人软件控制平台

对于机器人控制程序的开发为什么会选择在机器人控制软件平台上？

机器人控制程序的开发是机器人项目成本中很大的一部分，机器人在应用到实际场景中，所涉及到的绝大部分工作都是编写控制程序完成应用场景中的任务。所以，一个好的机器人软件控制平台对于开发人员起到了事倍功半的作用，一个好的平台会起到简化机器人控制程序开发的流程，降低软件开发的难度。

机器人软件控制平台可以有效地解决机器人整个控制系统的运动控制协调的问题，可以有效地将机器人控制中所用到的运动学、逆运动学、动力学、轨迹规划、传感器反馈等结合起来。随着机器学习领域应用不断深化，在机器人方面，机器学习中很多领域得到了广泛的应用，所以将机器人 软件开发所涉及到的方面集成第一个统一的平台上，这样极大地简化开发人员的开发难度，可以有效地提高机器人项目开发的效率。

**Robotmaster**

Robotmaster是一款专门用于工业机器人CAD/CAM无缝集成离线编程、仿真和代码生成的平台。通过该平台可以完成快速、无差错的工业机器人运动控制程序，可以有效地提高机器人的编程速度，其具有以下特点：

* 准确地创建无示教点的机器人轨迹
* “点击拖曳”交互式仿真/编辑环境
* 机器人动作优化
* 兼容不同的机器人制造商

奇点，校准，碰撞，位置限制等方面对于机器人系统来说是非常复杂的。Robotmaster的交互式仿真环境（RISE）为快速，轻松地编程，可视化和优化流程/事件提供了有效地机器人软件开发平台。 Robotmaster可快速显示问题和机会，轻松获得最佳机器人程序。其完整的交互性使其成为找到最佳解决方案的综合工具，无需逐点干预，即便是通过狭窄通道。

通过实时可视化和验证，点击并拖动简单的来解决机器人运动范围和工件定位问题，对于实际应用中，这样可以方便配置机器人各关节的位置和定义工件最佳定位。

Robotmaster可轻松转换编写的运动控制程序，最大限度地缩短操作之间的循环时间。

强大的集成式6轴轮廓编程使开发人员能够创建优化的程序，而无需学习复杂的CAD / CAM和复杂的机器人仿真技能。

Robotmaster可以配件在内的整个机器人单元作为一个集成系统进行管理，控制所有轴的运动，包括轨迹和旋转，以充分优化工件加工过程，可以达到对复杂工件的加工。

Robotmaster简化了编程语言，特定于应用程序的用户界面，允许开发人员设置和管理他的最佳参数。

**Microsoft Robotics Studio**

Microsoft Robotics Studio 是微软的机器人平台，是一款非常好的机器人开发工具。其是Windows环境下用来开发机器人控制程序和机器人情境模拟的集成开发平台。功能包括：[可视化编程](https://en.wikipedia.org/wiki/Visual_programming)工具，用于创建和调试机器人应用程序的[Microsoft Visual Programming Language](https://en.wikipedia.org/wiki/Microsoft_Visual_Programming_Language)，[基于Web和基于](https://en.wikipedia.org/wiki/Internet" \o "互联网) Windows的界面，[3D模拟](https://en.wikipedia.org/wiki/3D_computer_graphics" \o "3D计算机图形学)（包括[硬件加速](https://en.wikipedia.org/wiki/Hardware_acceleration)），可轻松访问机器人的[传感器](https://en.wikipedia.org/wiki/Sensor" \o "传感器)和[执行器](https://en.wikipedia.org/wiki/Actuator)。主要的编程语言是[C＃](https://en.wikipedia.org/wiki/Visual_C_Sharp)。

Microsoft Robotics Developer Studio包括对包以向套件添加其他服务的包的支持。目前可用的包括微软的足球模拟和相扑比赛，以及社区开发的迷宫模拟器，一个可以通过虚拟[机器人](https://en.wikipedia.org/wiki/Robot" \o "机器人)探索墙壁的世界，以及一套[OpenCV](https://en.wikipedia.org/wiki/OpenCV)服务。RDS有四个主要组件：

* CCR（并发和协调运行时）
* DSS（分散软件服务）
* VPL（可视化编程语言）
* VSE（可视化仿真环境）

允许开发MRDS应用程序的工具包含图形环境（Microsoft Visual Programming Language：VPL）命令行工具，允许您在C＃和3D模拟工具中处理Visual Studio项目（VS Express版本已足够）。可视化编程语言是一种使用服务和活动目录的图形开发环境。它们可以以图形方式交互，服务或活动由具有输入和输出的块表示，该输入和输出只需要从目录拖动到图表中。可以使用鼠标进行链接，它允许您定义信号是否同时，允许您对传输的执行操作。VPL还允许您从用户创建的图表生成新的“宏”服务代码。在VPL中可以轻松地为不同的硬件元素定制服务。RDS 3D模拟环境允许您使用包含高级物理的NVIDIA PhysX技术（最由Ageia编写的3D引擎）模拟虚拟世界中机器人的行为。

**Orocos**

Orocos (Open Robot Control Software) 是一个用来构建实时控制软件的C++框架，适合开发机器人或者机器的控制软件。Orocos 实时工具包提供了一个基础框架，以快速开发可运行在实时操作系统的应用，如RTAI和Xenomai ，当然它也支持 Linux 系统。这个计划的目的是要开发一种通用的，免费的模块化架构，用于机器人控制。Orocos计划用四个C++库组成：实时工具集，运动学与动力学算法集，贝叶斯过滤库及 Orocos组件库。如下图，orocostoolchain 包含了实时工具集和Orocos组件库。[1]

Orocos 是工业领域一个非常通用的软件架构，相当于一个企业的架构师.首先，Orocos兼容Xenomai，实时性能非常好，所以也支持Ether CAT 在其软件框架上开发。第二，Orocos 组件与组件之间的通信非常好，避免使用全局变量，这样就可以减小各个功能模块之间的耦合，方便每个开发人员开发自己的模块，最后融合时也很快。第三，Orocos 是一个非常成熟的软件框架，在欧洲很常用，并且有自己的社区，有很多开发人员会将自己设计好的算法组件发布到社区上，这样就大大减小了开发难度，因为有很多算法可以在 Orocos 社区找到，直接作为组件加载进入自己的软件。第四，Orocos 的初衷就是为了方便大家对机器人进行开发，包含了自己的运动学和动力学算法集，所以本论文采用 Orocos 可以节省很多时间和精力，在对多自由度机器人的运动学分析时，可以直接采用 Orocos 的运动学组件。

RTT：Orocos 实时工具集，为软件开发人员构建机器人应用时提供了基础机能，在实时功能、在线交互以及基于组件式的应用中发挥极大作用。

OCL：Orocos 组件库，由开发人员进行开发，一个独特的功能就是一个组件，Orocos  软件框架为 OCL 提供了通信功能、调试功能，很方便开发人员对其 OCL 进行开发。

KDL：机器人运动与动力学组件，为运动学提供了实时的动力学约束计算，这个组件非常有用，有了这个组件，很多机器人开发者可以快速地开发机器人算法。

BFL：贝叶斯过滤库，是由动态贝叶斯网络理论所推导出的。这个理论可以做递归信息处理及基于贝叶斯规则的算法评估，例如卡尔曼滤波，粒子滤波算法等。

**Webots**

Webots是Cyberbotics公司出品的便携式机器人仿真平台，可运行在windows，Mac和Linux上，内建3D编辑器,可构建3D机器人模型。应用C++或JAVA或者MATLAB编辑机器人程序可模拟机器人的动作。也就是说，它支持多种[编程语言](https://www.xuebuyuan.com/" \o "编程语言)，包括C, C++, JAVA, Python, Matlab, VRML97。

Webots使用ODE（[开放动力学引擎](https://en.wikipedia.org/wiki/Open_Dynamics_Engine)）来检测碰撞并模拟刚体动力学。ODE库允许人们准确地模拟物体的物理属性，例如速度，惯性和摩擦力。

它提供一个机器人快速建模的软件环境，让用户能够虚拟的3D环境和物理属性，比如 质量、关节、摩擦系数 等。用户能够加入简单的被动或者主动的机器人。这些机器人能够有不同的运动方案，如轮式机器人、足式机器人、飞行机器人。更多的是，这些机器人身上还能加上一系列传感器和电机，例如距离传感器、驱动轮、摄像头、舵机、触觉传感器、发射器、接收器。

最后，用户能够给每个机器人编程去展示想要的行为和动作。Webots已经有许多机器人模型和控制器程序例程帮助用户使用。Webots还有许多和实际移动机器人的接口，你能够按照你所想的去仿真机器人，并且将仿真的控制程序转移到实际的机器人上。比如e-puck、DARwIn-OP、Nao。

Webots提供了拍摄[PNG](https://en.wikipedia.org/wiki/Portable_Network_Graphics" \o "便携式网络图形)屏幕截图并将模拟记录为[MPEG](https://en.wikipedia.org/wiki/MPEG" \o "MPEG)（Mac / Linux）和[AVI](https://en.wikipedia.org/wiki/Audio_Video_Interleave" \o "音频视频交错)（Windows）电影的可能性。Webots世界存储在跨平台的.wbt文件中，这些文件的格式基于[VRML](https://en.wikipedia.org/wiki/VRML" \o "VRML)语言。也可以以VRML格式导入和导出Webots世界或对象。另一个有用的特征是用户可以随时与正在运行的模拟进行交互，即，可以用鼠标移动机器人和其他对象。Webots可以使用[WebGL](https://en.wikipedia.org/wiki/WebGL" \o "WebGL的)在Web浏览器上传输模拟。

Webots用于几个在线机器人编程竞赛。Robotstadium比赛是对[RoboCup](https://en.wikipedia.org/wiki/RoboCup" \o "机器人世界杯)标准平台联盟的模拟。在这次模拟中，两队[Nao](https://en.wikipedia.org/wiki/Nao_(robot))踢足球的规则与常规足球相似。机器人使用模拟摄像机，超声波和压力传感器。在Rat's Life比赛中，两个模拟[电子球](https://en.wikipedia.org/wiki/E-puck" \o "E-冰球)机器人在[乐高](https://en.wikipedia.org/wiki/Lego)迷宫中争夺能源资源。匹配每天运行，结果可以在在线视频中观看。

**ABB Robotstudio**

RobotStudio是ABB机器人的离线仿真调试软件。 RobotStudio基于ABB VirtualController构建，是运行机器人生产的真实软件的精确副本。这允许使用与车间使用的真实机器人程序和配置文件相同的真实机器人程序和配置文件来执行非常逼真的模拟。 支持所有ABB机器人模型以及变位机，导轨等。 完全和现场实际应用一样的示教器操作，机器人运动仿真与真实一致，真正可以做到在RobotStudio里所见即真实环境所得。 丰富的离线轨迹自动生成功能，支持多种数模导入，机器人轨迹自动生成，免去人工现场调试带来的繁重重复工作。 简洁高效的代码调试环境。丰富的powerpac（RobotStudio插件）功能，针对不同行业，快速解决机器人轨迹生成，编程问题等。比如Machining Powerpac，针对如下轨迹应用快捷方便。

**V-REP**

V-REP是一个强大的机器人3D集成开发环境，其具有几个通用的计算模块（逆运动学，物理/动力学，碰撞检测，最小距离计算，路径规划等），分布式控制架构（无限数量的控制脚本，线程或非线程），以及几个扩展机制（插件，客户端应用程序等），它提供了许多功能，可以通过详的API和脚本功能轻松集成和组合。 V-REP有5种不同的版本，大多数版本都有不同的许可选项。V-REP是快速原型验证，远程监控，快速算法开发，机器人相关教育和工厂自动化系统仿真的完美工具。 V-REP 是机器人仿真器里的“瑞士军刀”：你不会发现一个比它拥有更多功能，特色或是更详尽应用编程接口的机器人仿真器:

•跨平台 (Windows、MacOS、Linux)

•六种编程方法 (嵌入式脚本、插件、附加组件、ROS节点、远程客户端应用编程接口、或自定义的解决方案)

•七种编程语言 (C/C++、Python、Java、Lua、Matlab、Octave、和 Urbi)

•超过400种不同的应用编程接口函数

•100项ROS服务、30个发布类型、25个ROS订户类型、可拓展

•4个物理引擎 (ODE, Bullet, Vortex, Newton)

•Integrated ray-tracer (POV-Ray)

•完整的运动学解算器 (对于任何机构的逆运动学和正运动学)

•Mesh, octree, point cloud-网孔干扰检测

•Mesh, octree, point cloud-网孔最短距离计算

•路径规划 (在2到6维中的完整约束、对于车式车辆的非完整约束)

•嵌入图像处理的视觉传感器 (完全可拓展)

•现实的接近传感器 (在检测区域中的最短距离计算)

•嵌入式的定制用户接口、包括编辑器

•完全集成的第四类Reflexxes运动库 + RRS-1 interface specifications

•表面切削仿真

•数据记录与可视化 (时距图、X/Y图或三维曲线)

•整合图形编辑模式

•支持水/气体喷射的动态颗粒仿真

•带有拖放功能的模型浏览器 (在仿真中依旧可行)

•多层 取消/重做、影像记录、油漆的仿真、详尽的文档等

**OpenRAVE**

Openrave是一个开源的机器人开发平台，可以上面进行机器人的运动规划。 OpenRAVE由Rosen Diankov在[卡内基梅隆大学](https://en.wikipedia.org/wiki/Carnegie_Mellon_University" \o "卡内基·梅隆大学) [机器人研究所](https://en.wikipedia.org/wiki/Robotics_Institute)的生活质量技术中心创立。[[3]](https://en.wikipedia.org/wiki/OpenRAVE#cite_note-3)它的灵感来自RAVE模拟器[James Kuffner](https://en.wikipedia.org/wiki/James_J._Kuffner_Jr." \o "James J. Kuffner Jr.)在1995年开始开发并用于他的大量实验。OpenRAVE项目始于2006年，最初是对RAVE的完全重写，以支持插件。它迅速分化为自己的架构概念，并开始得到全世界许多机器人研究人员的支持。在2010年8月从[机器人研究所](https://en.wikipedia.org/wiki/Robotics_Institute" \o "机器人研究所)获得博士学位后，Rosen Diankov成为[东京大学](https://en.wikipedia.org/wiki/University_of_Tokyo" \o "东京大学) JSK机器人实验室的博士后，目前正在维护OpenRAVE。

OpenRAVE为在真实的机器人应用程序中测试、开发和部署运动规划算法提供了一个环境。主要研究与运动规划相关的运动学和几何信息的仿真和分析。OpenRAVE的独立特性允许很容易地集成到现有的机器人系统中。它提供了许多与机器人和计划器一起工作的命令行工具，并且运行时内核足够小，可以在控制器和更大的框架中使用。一个重要的目标应用是工业机器人自动化。 OpenRAVE提供的最具影响力的技术可能是名为IKFast的工具，即机器人运动学编译器。与大多数逆运动学求解器不同，IKFast可以解析地求解任何复杂运动学链的运动学方程，并生成特定于语言的文件（如[C ++](https://en.wikipedia.org/wiki/C%2B%2B" \o "C ++)）供以后使用。最终结果是非常稳定的解决方案，在最近的处理器上可以快速运行5 [微秒](https://en.wikipedia.org/wiki/Microsecond)。OpenRAVE支持[COLLADA](https://en.wikipedia.org/wiki/COLLADA" \o "COLLADA) 1.5文件格式，用于指定机器人并添加自己的机器人特定扩展集。机器人扩展包括：机械手、传感器、计划特定的参数。OpenRAVE设计的核心是提供运动规划算法的接口和实现。大多数规划算法实现用于[机器人手臂](https://en.wikipedia.org/wiki/Robot_arm" \o "机器人手臂)并使用采样来探索任务配置空间。

**ROS**

ROS---(Robot Operating System）是一个机器人软件平台，它能为异质[计算机集群](https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA%E9%9B%86%E7%BE%A4)提供类似操作系统的功能。ROS的前身是[斯坦福](https://baike.baidu.com/item/%E6%96%AF%E5%9D%A6%E7%A6%8F)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，主要由威楼加拉吉继续该项目的研发。

ROS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是基于一种图状架构，从而不同节点的进程能接受，发布，聚合各种信息（例如传感，控制，状态，规划等等）。目前ROS主要支持Ubuntu。

ROS可以分成两层，低层是上面描述的操作系统层，高层则是广大用户群贡献的实现不同功能的各种软件包，例如定位绘图，行动规划，感知，模拟等等。ROS（低层）使用BSD许可证，所有是开源软件，并能免费用于研究和商业用途。而高层的用户提供的包则可以使用很多种不同的许可证。

### ROS概述

**（1）ROS简介**

随着机器人领域的快速发展和复杂化，代码的复用性和模块化的需求原来越强烈，而已有的开源机器人系统又不能很好的适应需求。 ROS为Robot Operating System（机器人操作系统）的简写，是一个面向机器人的开源元操作系统（open-source，meta-operating system）。它能够提供类似传统操作系统的诸多功能，如硬件抽象、底层设备控制、进程间消息传递和程序包管理等。此外，它还提供相关工具和库，用于获取、编译、编辑代码以及在多个计算机之间运行程序完成分布式计算。ROS的运行架构是一种使用ROS通信模块实现模块间P2P的松耦合的网络连接的处理架构，它执行若干种类型的通讯，包括基于服务的同步RPC（远程过程调用）通讯、基于Topic的异步数据流通讯，还有参数服务器上的数据存储。2010年Willow Garage公司发布了开源机器人操作系统ROS（robot operating system），很快在机器人研究领域展开了学习和使用ROS的热潮。

ROS系统最初起源于2007年斯坦福大学人工智能实验室的项目与机器人技术公司Willow Garage的个人机器人项目（Personal。Robots Program）合作开发的，而2008年之后就由Willow Garage来负责进一步的推广和维护。但这并不意味着ROS是封闭的系统，相反，它是由众多学校及科研机构联合开发及维护的，这种联合开发模式也为ROS系统生态的构建与壮大带来有力的促进。2013年，WillowGarage公司被SuitableTechnologies公司收购，但是在2012年，ROS的开发和维护管理工作被移交给了新成立的非营利性组织——开源基金会（Open Source Robotics Foundation），此后该组织负责维护和更新ROS，并为机器人社区提供相应的支持和开源工具。

**ROS是什么**

ROS（Robot Operating System，下文简称“ROS”）是一个适用于机器人的开源的元操作系统。它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。它也提供用于获取、编译、编写、和跨计算机运行代码所需的工具和库函数。

ROS 的主要目标是为机器人研究和开发提供代码复用的支持。ROS是一个分布式的进程（也就是“节点”）框架，这些进程被封装在易于被分享和发布的程序包和功能包中。ROS也支持一种类似于代码储存库的联合系统，这个系统也可以实现工程的协作及发布。这个设计可以使一个工程的开发和实现从文件系统到用户接口完全独立决策（不受ROS限制）。同时，所有的工程都可以被ROS的基础工具整合在一起。

**（2）ROS的优势**

ROS被称作为机器人操作系统，其更多的承担类似于桥梁的中间通讯的角色，通常我们是在ROS现有的基础上开发出一整套针对于某一种机器人的控制系统架构， 值得一提的是，ROS的设计者考虑到各开发者使用的开发语言不同，因此ROS的开发语言独立，支持C++，python等多种开发语言。因此，除了官方提供的功能包之外，ROS还聚合了全世界开发者实现的大量开源功能包。

ROS 的主要目标是为机器人研究和开发提供代码复用的支持。ROS提供了一种分布式处理架构，在ROS中称之为节点（Node），这使每个可执行文件能被单独设计，并且ROS在运行时可以由多个松散耦合进程组成。 在实际使用时，这种松耦合的结构设计可以让开发者根据机器人所需功能灵活添加各个功能模块。 ROS具有优良的开源基因，松耦合设计方法，可扩展的软件结构，功能复用思想，极大降低了机器人领域的进入门槛，让开发者无需像前人一样走过众多弯路，掌握多种知识后才能开始实现其机器人设计的梦想。开发者可以用ROS的基础框架配合选定的功能包快速实现系统原型，从而让开发人员将更多时间用于核心算法的开发改进上。用户也可以单独选用某些功能包，将其集成在已有产品中，实现特定功能。

ROS是一个1）众多的机器人软件辅助开发工具、2）基于统一通讯协议的数据可视化，调试，管理工具、3）众多的机器人，传感器，执行器、4）众多的机器人领域一流的先进算法实现过程，gmapping、LSD-slam、OMPL、PCL，全球的社区服务支持

开发者：快速先进算法的服用以及开发效率的大幅提升，快速原型验证项目可行性

研究者：机器人系统整体搭建的便利，更加专注于某领域的研究

学习者：门槛降低的机器人学习平台，高质量的优秀源代码获取平台

ROS的缺点：作为一个开源项目，ROS也存在一些问题。例如，ROS因为将功能分布在各个节点之中，节点间基于消息机制通信，通讯部分消耗了很多系统资源。尤其是当所有节点位于同一个处理器时，ROS仍然一直执行相应的消息分发，节点间的数据传递通过内存复制，大量的系统资源都浪费在通讯上，使得系统必须选用高性能的处理器和存储系统以弥补损耗。换句话说，利用ROS来实现SLAM，需要配备性能优越的硬件设备，这对于一些小型化嵌入式平台，尤其是实际的机器人产品里，其对计算资源、存储空间的消耗会使成本大幅上升。

**（3）主要特点**

ROS的运行[架构](http://lib.csdn.net/base/16" \o "大型网站架构知识库)是一种使用ROS通信模块实现模块间P2P的松耦合的网络连接的处理架构，它执行若干种类型的通讯，包括基于服务的同步RPC（远程过程调用）通讯、基于Topic的异步数据流通讯，还有参数服务器上的数据存储。但是ROS本身并没有实时性。

ROS的主要特点可以归纳为以下几条：

1）点对点设计

2）多语言支持

3）精简与集成

4）工具包丰富

5）免费且开源

**1）点对点设计**

一个使用ROS的系统包括一系列进程，这些进程存在于多个不同的主机并且在运行过程中通过端对端的拓扑结构进行联系。虽然基于中心服务器的那些软件框架也可以实现多进程和多主机的优势，但是在这些框架中，当各电脑通过不同的网络进行连接时，中心数据服务器就会发生问题。ROS的点对点设计以及服务和节点管理器等机制可以分散由计算机视觉和语音识别等功能带来的实时计算压力，能够适应多机器人遇到的挑战。

**2）多语言支持**

在写代码的时候，许多编程者会比较偏向某一些编程语言。这些偏好是个人在每种语言的编程时间、调试效果、语法、执行效率以及各种技术和文化的原因导致的结果。为了解决这些问题，我们将ROS设计成了语言中立性的框架结构。ROS现在支持许多种不同的语言，例如C++、[Python](http://lib.csdn.net/base/11" \o "Python知识库)、Octave和LISP，也包含其他语言的多种接口实现。

ROS的特殊性主要体现在消息通讯层，而不是更深的层次。端对端的连接和配置利用XML-RPC机制进行实现，XML-RPC也包含了大多数主要语言的合理实现描述。我们希望ROS能够利用各种语言实现的更加自然，更符合各种语言的语法约定，而不是基于C语言给各种其他语言提供实现接口。然而，在某些情况下利用已经存在的库封装后支持更多新的语言是很方便的，比如Octave的客户端就是通过C++的封装库进行实现的。

为了支持交叉语言，ROS利用了简单的、语言无关的接口定义语言去描述模块之间的消息传送。接口定义语言使用了简短的文本去描述每条消息的结构，也允许消息的合成，例如下图就是利用接口定义语言描述的一个点的消息：

每种语言的代码产生器就会产生类似本种语言目标文件，在消息传递和接收的过程中通过ROS自动连续并行的实现。这就节省了重要的编程时间，也避免了错误：之前3行的接口定义文件自动的扩展成137行的C++代码，96行的Python代码，81行的Lisp代码和99行的Octave代码。因为消息是从各种简单的文本文件中自动生成的，所以很容易列举出新的消息类型。在编写的时候，已知的基于ROS的代码库包含超过四百种消息类型，这些消息从传感器传送数据，使得物体检测到了周围的环境。

**3）精简与集成**

大多数已经存在的机器人软件工程都包含了可以在工程外重复使用的驱动和算法，不幸的是，由于多方面的原因，大部分代码的中间层都过于混乱，以至于很困难提取出它的功能，也很难把它们从原型中提取出来应用到其他方面。

为了应对这种趋势，我们鼓励将所有的驱动和算法逐渐发展成为和ROS没有依赖性单独的库。ROS建立的系统具有模块化的特点，各模块中的代码可以单独编译，而且编译使用的CMake工具使它很容易的就实现精简的理念。ROS基本将复杂的代码封装在库里，只是创建了一些小的应用程序为ROS显示库的功能，就允许了对简单的代码超越原型进行移植和重新使用。作为一种新加入的有优势，单元测试当代码在库中分散后也变得非常的容易，一个单独的测试程序可以测试库中很多的特点。

ROS利用了很多现在已经存在的开源项目的代码，比如说从Player项目中借鉴了驱动、运动控制和仿真方面的代码，从OpenCV中借鉴了视觉算法方面的代码，从OpenRAVE借鉴了规划算法的内容，还有很多其他的项目。在每一个实例中，ROS都用来显示多种多样的配置选项以及和各软件之间进行数据通信，也同时对它们进行微小的包装和改动。ROS可以不断的从社区维护中进行升级，包括从其他的软件库、应用补丁中升级ROS的源代码。

**4）工具包丰富**

为了管理复杂的ROS软件框架，我们利用了大量的小工具去编译和运行多种多样的ROS组建，从而设计成了内核，而不是构建一个庞大的开发和运行环境。

这些工具担任了各种各样的任务，例如，组织源代码的结构，获取和设置配置参数，形象化端对端的拓扑连接，测量频带使用宽度，生动的描绘信息数据，自动生成文档等等。尽管我们已经测试通过像全局时钟和控制器模块的记录器的核心服务，但是我们还是希望能把所有的代码模块化。我们相信在效率上的损失远远是稳定性和管理的复杂性上无法弥补的。

**5）免费并且开源**

ROS所有的源代码都是公开发布的。我们相信这将必定促进ROS软件各层次的调试，不断的改正错误。虽然像Microsoft Robotics Studio和Webots这样的非开源软件也有很多值得赞美的属性，但是我们认为一个开源的平台也是无可为替代的。当硬件和各层次的软件同时设计和调试的时候这一点是尤其真实的。

ROS以分布式的关系遵循这BSD许可，也就是说允许各种商业和非商业的工程进行开发。ROS通过内部处理的通讯系统进行数据的传递，不要求各模块在同样的可执行功能上连接在一起。如此，利用ROS构建的系统可以很好的使用他们丰富的组件：个别的模块可以包含被各种协议保护的软件，这些协议从GPL到BSD，但是许可的一些“污染物”将在模块的分解上就完全消灭掉。

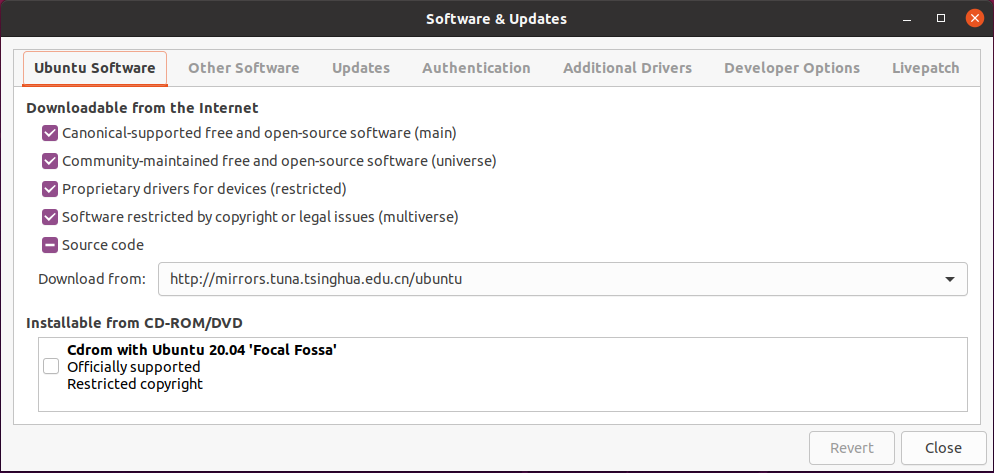
**ROS发行版本**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本名称** | **发布日期** | **版本生命周期** | **操作系统平台** |
| ROS Noetic Ninjemys | 2020年5月 | 2025年5月 | Ubuntu 20.04 |
| ROS Melodic Morenia | 2018年5月23日 | 2023年5月 | Ubuntu 17.10, Ubuntu 18.04, Debian 9, Windows 10 |
| ROS Lunar Loggerhead | 2017年5月23日 | 2019年5月 | Ubuntu 16.04, Ubuntu 16.10, Ubuntu 17.04,Debian 9 |
| ROS Kinetic Kame | 2016年5月23日 | 2021年4月 | Ubuntu 15.10, Ubuntu 16.04, Debian 8 |
| ROS Jade Turtle | 2015年5月23日 | 2017年5月 | Ubuntu 14.04, Ubuntu 14.10, Ubuntu 15.04 |
| ROS Indigo Igloo | 2014年7月22日 | 2019年4月 | Ubuntu 13.04, Ubuntu 14.04 |
| ROS Hydro Medusa | 2013年9月4日 | 2015年5月 | Ubuntu 12.04, Ubuntu 12.10, Ubuntu 13.04 |
| ROS Groovy Galapagos | 2012年12月31日 | 2014年7月 | Ubuntu 11.10, Ubuntu 12.04, Ubuntu 12.10 |
| ROS Fuerte Turtle | 2012年4月23日 | -- | Ubuntu 10.04, Ubuntu 11.10, Ubuntu 12.04 |
| ROS Electric Emys | 2011年8月30日 | -- | Ubuntu 10.04, Ubuntu 10.10, Ubuntu 11.04, Ubuntu 11.10 |
| ROS Diamondback | 2011年3月2日 | -- | Ubuntu 10.04, Ubuntu 10.10, Ubuntu 11.04 |
| ROS C Turtle | 2010年8月2日 | -- | Ubuntu 9.04, Ubuntu 9.10, Ubuntu 10.04, Ubuntu 10.10 |
| ROS Box Turtle | 2010年3月2日 | -- | Ubuntu 8.04, Ubuntu 9.04, Ubuntu 9.10, Ubuntu 10.04 |

### ROS安装

**软件仓库配置**

* 打开Software & Update
* 确保勾选restricted,universe 和 multiverse.



* 关闭后，会提示重新加载

**设置软件源**

打开terminal终端，输入下方代码

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

**设置秘钥**

sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654

如果使用代理服务器，可以使用curl替代apt-key命令，具体内容如下：

sudo apt install curl # 如果没有安装curl，运行此命令安装curl

curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -

成功后返回

gpg: key F42ED6FBABXXXXXX: public key "Open Robotics <info@osrfoundation.org>" imported

gpg: Total number processed: 1

gpg: imported: 1

**ROS安装**

安装前，确保Debian包索引到最新

sudo apt-get update

下面就是选择安装多大的ROS包

* 桌面完整版(官方推荐):包含2D/3D仿真和感知包

sudo apt install ros-noetic-desktop-full

* 桌面版:基本的ROS功能并包含rqt和rviz等图形化工具

sudo apt install ros-noetic-desktop

* 基础版:包含基础的ROS包、编译工具和通信库

sudo apt install ros-noetic-ros-base

* 如果需要安装其他包文件，可以选择相应的包直接安装

sudo apt install ros-noetic-PACKAGE

其中，PACKAGE对应你所要安装的包，详细可参考安装[包列表](https://index.ros.org/packages/page/1/time/" \l "noetic)。

**环境配置**

如果需要使用ROS，在每个bash终端中需更新ros的配置脚本，具体命令如下：

source /opt/ros/noetic/setup.bash

也可以设置每次打开终端时，自动更新ROS的配置脚本

echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc

source ~/.bashrc

**编译依赖包**

* 安装rosdep

如果未安装rosdep工具，输入下述命令：

$ sudo apt install python-rosdep

如果系统默认只包含python3,相应命令修改如下：

$ sudo apt install python3-rosdep

* 依赖初始化

$ sudo rosdep init

返回下述结果说明初始化成功

Wrote /etc/ros/rosdep/sources.list.d/20-default.list

Recommended: please run

rosdep update

* 更新依赖

$ rosdep update

返回下述结果说明更新成功

eading in sources list data from /etc/ros/rosdep/sources.list.d

Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/osx-homebrew.yaml

Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/base.yaml

Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/python.yaml

Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/ruby.yaml

Hit https://raw.githubusercontent.com/ros/rosdistro/master/releases/fuerte.yaml

Query rosdistro index https://raw.githubusercontent.com/ros/rosdistro/master/index-v4.yaml

Skip end-of-life distro "ardent"

Skip end-of-life distro "bouncy"

Skip end-of-life distro "crystal"

Add distro "dashing"

Add distro "eloquent"

Add distro "foxy"

Skip end-of-life distro "groovy"

Skip end-of-life distro "hydro"

Skip end-of-life distro "indigo"

Skip end-of-life distro "jade"

Add distro "kinetic"

Skip end-of-life distro "lunar"

Add distro "melodic"

Add distro "noetic"

Add distro "rolling"

updated cache in /home/zgh/.ros/rosdep/sources.cache

**验证**

输入下述命令，验证ROS是否安装成功。

roscore

命令行返回如下结果，就表明ROS已安装成功。

Checking log directory for disk usage. This may take a while.

Press Ctrl-C to interrupt

Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://zgh-VirtualBox:36577/

ros\_comm version 1.15.7

SUMMARY

========

PARAMETERS

\* /rosdistro: noetic

\* /rosversion: 1.15.7

NODES

auto-starting new master

process[master]: started with pid [29713]

ROS\_MASTER\_URI=http://zgh-VirtualBox:11311/

setting /run\_id to 70500dfa-cbc5-11ea-8c25-9b0cacf5f716

process[rosout-1]: started with pid [29723]

started core service [/rosout]

^C[rosout-1] killing on exit

[master] killing on exit

shutting down processing monitor...

... shutting down processing monitor complete

done

## 实验内容

### 1、Ubuntu安装与环境配置

（1）使用虚拟机或双系统的方式完成Ubuntu20.04的安装与环境配置

### 2、ROS安装与环境配置

（1）在Ubuntu20.04下完成ROS安装与环境配置

## 注意事项

1. 在进行Ubuntu20.04的安装过程中，注意BIOS的设置，通常要关闭电脑的安全模式，根据不同电脑型号，具体操作有所不同。
2. 谨慎进行Win10+Ubuntu20.04的安装，初次安装建议在Win10下安装虚拟机，在虚拟机下安装Ubuntu20.04

## 参考文献

1. https://en.jinzhao.wiki/wiki/Ubuntu
2. <https://cn.ubuntu.com>
3. <http://wiki.ros.org/>
4. <http://wiki.ros.org/noetic/Installation/Ubuntu>