ROS机器人操作系统

·WHAT：ROS是什么？

ROS (Robot Operating System)是2007年诞生于斯坦福大学的一个机器人操作系统，原生测试系统为Linux Ubuntu。作为机器人软件的元操作系统，ROS主要提供了四个方面的支持，通信机制、开发工具、应用软件以及生态系统。

·WHY：为什么要用ROS？

·HOW：怎么使用ROS？

这是这一篇文章需要解决的问题。

·ROS & ROS2：如何选择？

目录

[1. 系统安装 Installation 2](#_Toc156818362)

[1.1. 版本选择 2](#_Toc156818363)

[1.2. 下载安装 2](#_Toc156818364)

[1.3. 主文件访问 3](#_Toc156818365)

[1.4. 卸载 3](#_Toc156818366)

[2. 基本概念 Basic Concepts 3](#_Toc156818367)

[2.1. 节点管理器Master 4](#_Toc156818371)

[2.2. 参数服务器Parameter Server 4](#_Toc156818372)

[2.3. 节点Node 4](#_Toc156818373)

[2.4. 话题与消息Topic & Message(msg) 5](#_Toc156818374)

[2.5. 服务 Service 5](#_Toc156818375)

[3. 编程与开发 Coding and Developing 5](#_Toc156818376)

[3.1. 工作空间：创建与初始化 6](#_Toc156818378)

[3.2. 创建软件包 6](#_Toc156818379)

[3.3. 编译C++文件 7](#_Toc156818380)

[3.4. Python文件 7](#_Toc156818381)

[4. 可视化 Visualization 7](#_Toc156818382)

[4.1. Rqt 7](#_Toc156818384)

[4.2. Rviz 7](#_Toc156818385)

[5. 常用命令 Common Commands 7](#_Toc156818386)

[5.1. 文件访问 7](#_Toc156818388)

[1) roscd：访问指定包的路径 7](#_Toc156818389)

[2) rosls：列出指定包文件夹内的对象 8](#_Toc156818390)

[3) rosed：编译指定软件包中的指定文件 8](#_Toc156818391)

[4) rospack：列出软件包的信息 8](#_Toc156818392)

[5.2. 启动与运行 8](#_Toc156818393)

[1) roscore：启动ROS系统 8](#_Toc156818394)

[2) rosrun：运行指定软件包中的指定节点 8](#_Toc156818395)

[3) roslaunch：运行指定软件包中的指定.launch文件 8](#_Toc156818396)

[5.3. 参数服务器 9](#_Toc156818397)

[1) rosparam：参数服务器的操作 9](#_Toc156818398)

[5.4. 节点 9](#_Toc156818399)

[1) rosnode：运行中节点检索与操作 9](#_Toc156818400)

[5.5. 话题与消息 10](#_Toc156818401)

[1) rostopic：活跃话题检索与操作 10](#_Toc156818402)

[2) rosmsg：消息类型查询 11](#_Toc156818403)

[5.6. 服务 11](#_Toc156818404)

[1) rosservice：活跃服务检索与操作 11](#_Toc156818405)

[2) rossrv：服务类型查询 11](#_Toc156818406)

[6. 仿真 Simulation 11](#_Toc156818407)

[7. 无人机控制——MAVROS 11](#_Toc156818408)

[参考资料 13](#_Toc156818409)

# 系统安装 Installation

## 版本选择

ROS的发行版本与Ubuntu的版本一一对应，与Ubuntu系统一样，每两年发布一个长期支持版（Long-term Support），安装时，ROS的版本一定要与Ubuntu系统的版本对应。

由于目前（2024年）正处于ROS与ROS2的过渡时期，运行于Ubuntu 20.04的Noetic已是第一代ROS系统的最终发行版，Ubuntu 22.04以及未来的系统将搭配ROS2，不再支持ROS。不过目前ROS2的社区和软件支持还未成熟，大多数的开发者还在使用第一代ROS。

目前开发常用的ROS版本如下：

1. Ubuntu 18.04 – ROS Melodic - <https://wiki.ros.org/melodic/Installation/Ubuntu>
2. Ubuntu 20.04 – ROS Noetic - <https://wiki.ros.org/noetic/Installation/Ubuntu>
3. Ubuntu 22.04 – ROS2 Humble - <https://docs.ros.org/en/humble/Installation.html>

## 下载安装

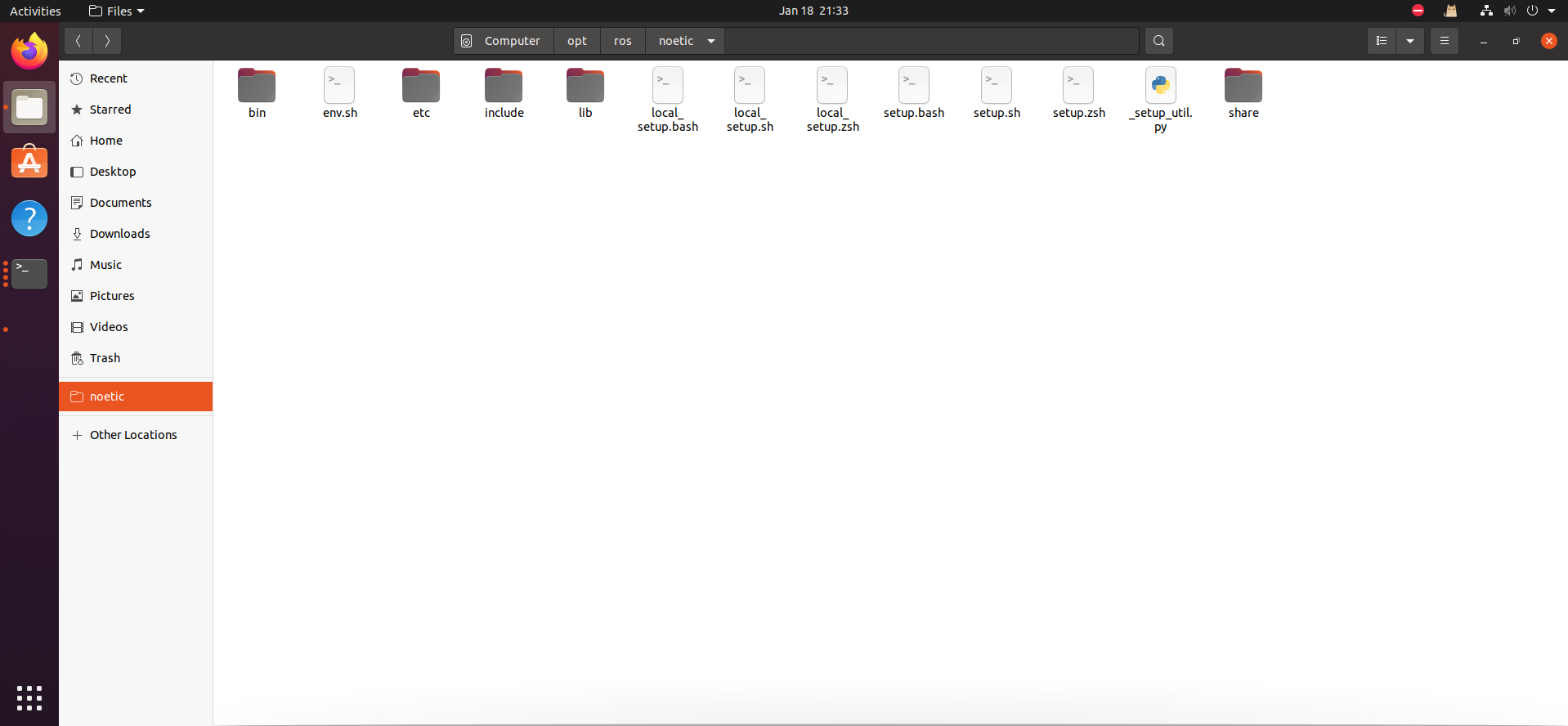
ROS与ROS2的下载步骤有一定区别，具体参考对应版本下载页面上的教程。下面是基本通用的安装步骤说明：

·ROS：添加下载源 >> 下载密钥 >> 更新apt源 >> apt下载 >> 配置环境变量

## 主文件访问

安装成功后，ROS的主文件夹位于以下路径：

|  |
| --- |
| $ cd /opt/ros/<distros>  #<distros>代表当前下载的发行版 |



图：ROS主文件夹

主文件夹下的share文件夹中可以找到下载的软件包，ROS系统也提供了一系列命令用于直接访问软件包所在路径，参见[文件访问](#_文件访问)。

## 卸载

有时候ROS出现一些难以排查的问题，我们希望卸载重装整个ROS系统，卸载的命令如下：

|  |
| --- |
| $ sudo apt remove ros-\*  $ sudo apt autoremove  #第一行自动卸载所有ROS相关的软件包；第二行卸载所有额外的依赖项 |

# 基本概念 Basic Concepts

·官方文档：<https://wiki.ros.org/ROS/Concepts>

ROS中最重要的三个概念是节点Node、话题Topic、服务Service，节点是机器人系统运行的单元，消息与服务则是节点之间建立通讯的基础。在一个庞大的机器人系统中，节点、话题、服务组成类似计算图Graph的结构，可以用ROS自带的rqt\_graph工具查看当前系统的计算图。

|  |
| --- |
| $ rosrun rqt\_graph rqt\_graph  $ rqt\_graph  #以上两种都行 |

在第一代ROS中，系统的节点注册Registration、通讯连接依靠的是一个统筹全局的节点管理器Master，在ROS2则不再有Master的概念。



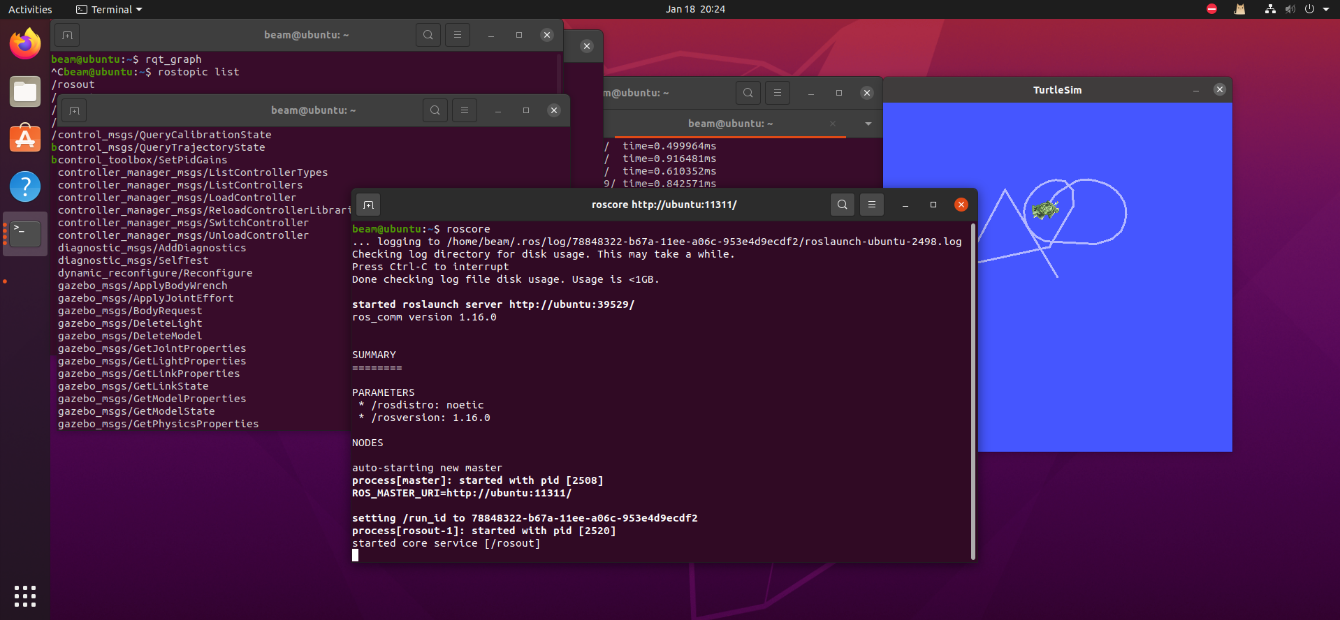
## 节点管理器Master

·官方文档：<https://wiki.ros.org/Master>

·相关命令：[roscore](#_roscore：启动ROS系统)

·关键词：注册Registration

Master是ROS中最核心的进程，它对ROS系统中启用的节点进行注册Registration，为它们提供一个系统中唯一的名字，并在各节点之间建立起话题和服务的通信。在开启ROS系统时运行的roscore命令其中一个功能就是启动Master。

****

图：Master运行的命令行

## 参数服务器Parameter Server

·官方文档：<http://wiki.ros.org/Parameter%20Server>

·相关命令：[rosparam](#_rosparam：参数服务器的操作)、[roscore](#_roscore：启动ROS系统)

·关键词：静态参数Static Parameters

参数服务器Parameter Server随roscore命令执行而启动，是一个供各节点保存静态参数（不需要经常改变的参数），参数服务器上的参数对全局开放，任何节点都可以读写（命令为获取get、设置set）上面保存的参数。

需要注意的是，参数的读写并不支持实时动态更新，这意味着，如果参数<param1>在被节点<node1>读取后更新了值，<node1>并不知道参数<param1>的更新，除非<node1>重新读取<param1>。

## 节点Node

·官方文档：<http://wiki.ros.org/Nodes>

·相关命令：[rosnode](#_rosnode：)

·编程实现：

·关键词：进程Process、可执行文件Executable

节点是ROS系统中运行的单个进程，在ROS的软件包中表现为一个可执行文件Executable。作为ROS系统的运行单元，节点可以发布或者订阅话题、提供或者使用服务。

## 话题与消息Topic & Message(msg)

·官方文档：<http://wiki.ros.org/Topics>（话题）& <http://wiki.ros.org/Messages>（消息）

·相关命令：[rostopic](#_rostopic：活跃话题检索与操作)、[rosmsg](#_rosmsg：消息查询)

·关键词：发布者Publisher、订阅者Subscriber

话题是ROS系统提供的一种单向传输的通信方式，节点可以发布话题，也可以订阅话题，信息单向地从发布者流向订阅者，不存在反馈，通俗来讲，话题相当于现实中的广播。发布者、订阅者是相对具体话题而言的概念，同一个节点可以同时发布多个话题、接收多个话题。

多个节点同时向一个话题中发布消息是可能的，但并不推荐，因为这种做法容易引起话题信息的混乱，类似无线电串频。

话题的类型Type称为消息，消息是由软件包中消息文件.msg所定义的数据格式，描述了话题传输的数据类型Type。话题的发布需要指定所使用的消息类型。

## 服务 Service

·官方文档：<https://wiki.ros.org/Services>

·相关命令：[rosservice](#_rosservice：)、[rossrv](#_rossrv：)

·关键词：服务器Server、客户端Client、请求Request、回答Reply、服务类型Service Type(srv)

服务是一种双向传播的通讯方式，客户端Client节点向服务器Server节点发送请求Request信息，并接受服务端发送的回答Reply信息。与话题类似，同一个节点可以同时充当服务器和客户端。

服务类型文件.srv定义了一种服务的请求和回答信息数据格式。

# 编程与开发 Coding and Developing

为了让用户可以编写自己的机器人控制程序，ROS提供了两种主流语言C++和Python的编程接口roscpp和rospy，这种接口被称为客户端库Client Library（注意！和服务通讯的客户端没有任何关系）。用户的自主开发一般在自创的“工作空间”文件夹中进行。

工作空间Workspace是由你自己定义的，用于存放包Packages、编译源代码的文件夹，运行catkin\_make之后，工作空间就成为一个标准的catkin空间。在一个定义规范的工作空间中进行开发是良好的习惯。



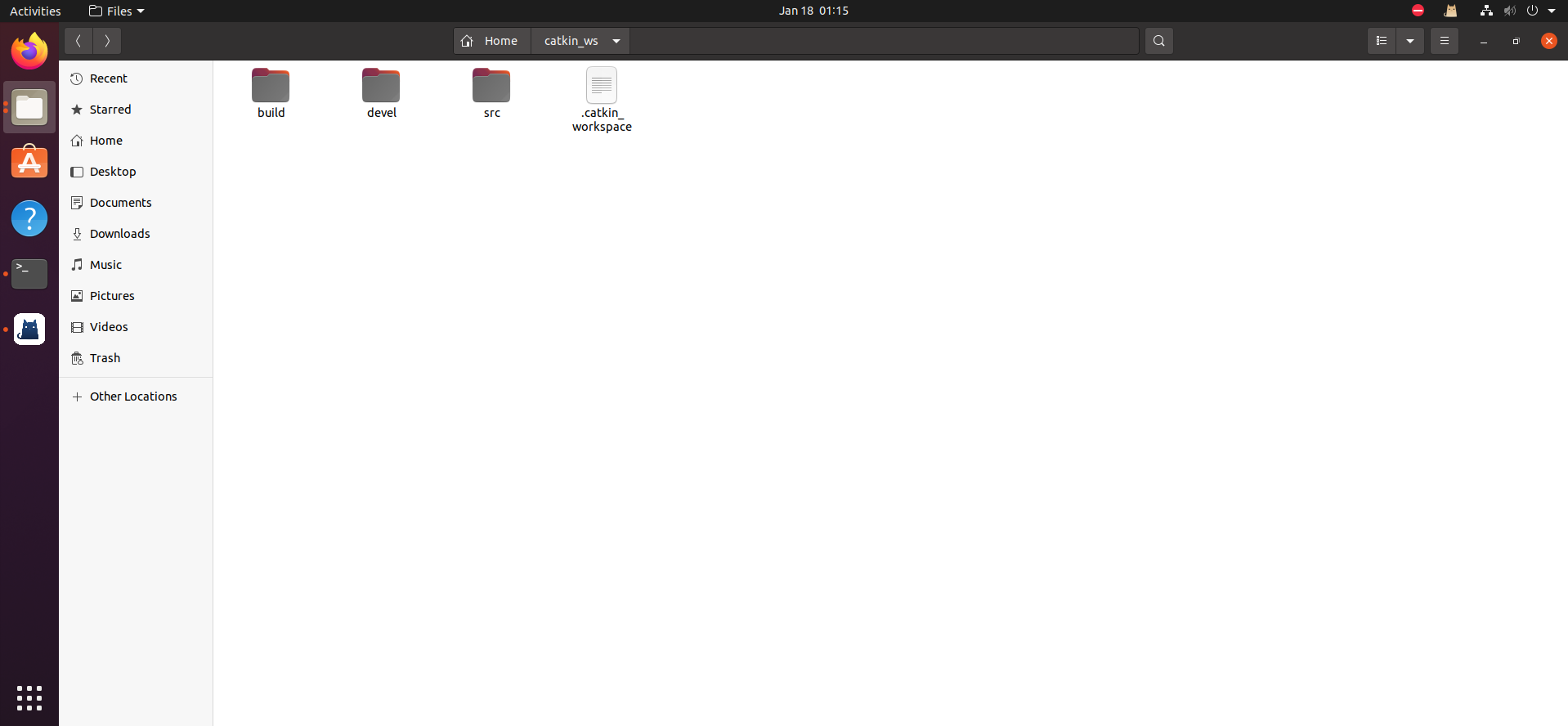
## 工作空间：创建与初始化

工作空间一般存放在Home文件夹中，创建名为catkin\_ws的工作空间命令如下，catkin\_ws可以改成别的名字，下级的src不能改名字。

|  |
| --- |
| $ mkdir -p ~/catkin\_ws/src  $ cd ~/catkin\_ws/  $ catkin\_make  $ catkin\_make install  #命令mkdir创建新文件夹，cd进入新创建的catkin\_ws工作空间，catkin\_make将工作空间初始化为可用于编译的标准catkin工作空间，catkin\_make install（非必须）用于创建install文件夹。 |

完成后，catkin\_ws中有以下文件和文件夹。

1. 编译文件夹“build”：编译过程中产生的文件，一般不怎么用到；
2. 开发文件夹“devel”： 可执行文件以及一些库文件；
3. 安装文件夹“install”：编译的成果——可执行文件和bash脚本；
4. 文件夹“src”：我们定义的包和源代码文件、编译参数配置文件；
5. “.catkin\_workspace”：标记当前文件夹是catkin文件夹。



图：工作空间“catkin\_ws”文件夹

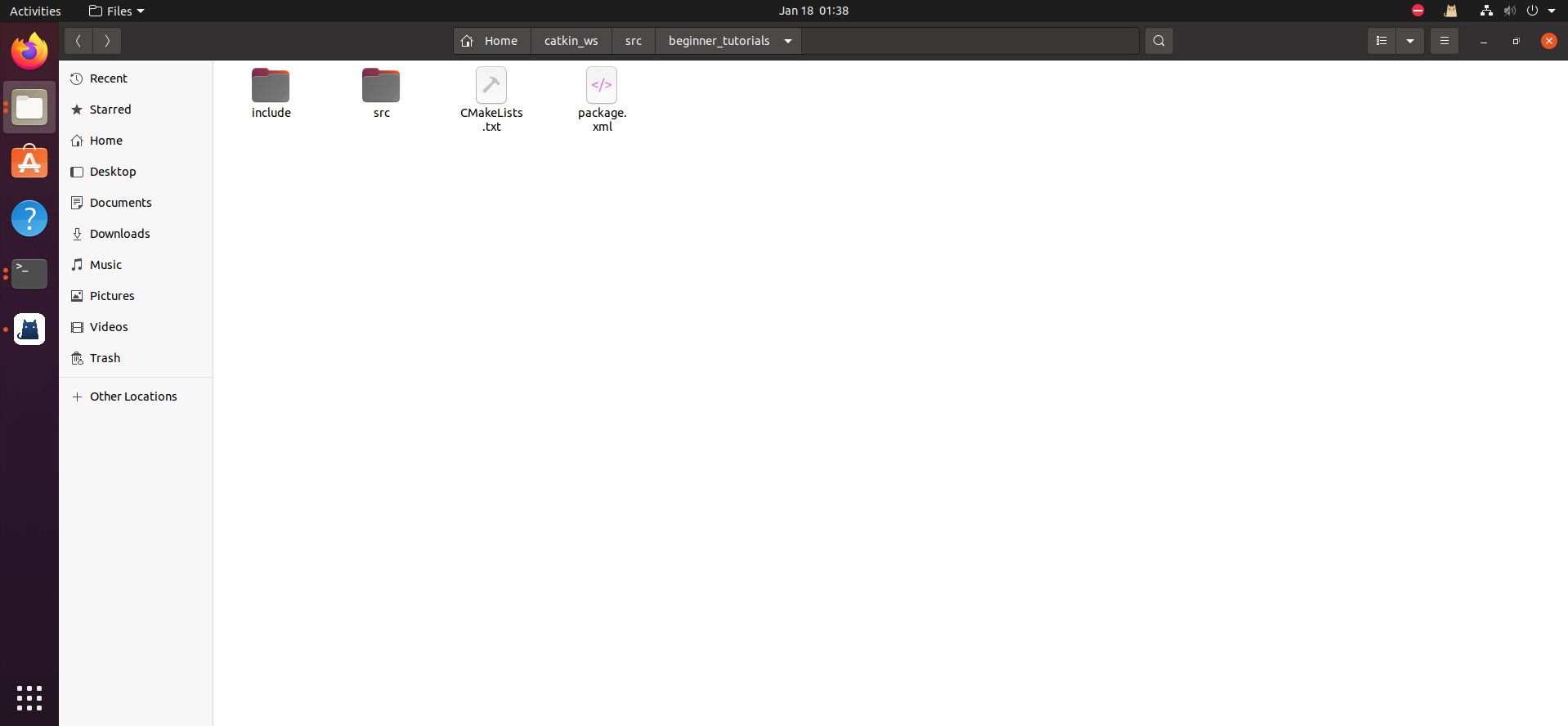
## 创建软件包

进入src文件夹中，输入命令catkin\_create\_pkg，后面的第一个参数beginner\_tutorials是新创建包的名字，其他参数是包的多个依赖项。

|  |
| --- |
| $ catkin\_create\_pkg beginner\_tutorials std\_msgs rospy roscpp |

在新创建的beginner\_tutorials包的文件夹中，有以下文件和文件夹：

1. 文件夹“include”：存放源代码的头文件
2. 文件夹“src”：存放源代码的主体部分
3. “CmakeLists.txt”：包的编译配置文件
4. “packages.xml”：记录包的依赖项



图：工作空间“catkin\_ws”中的“beginner\_tutorials”软件包文件夹

## 编译C++文件

假设现在已经在beginner\_tutorials包的src文件夹下创建好名为test\_code.cpp的C++源码文件，并且已经在里面写好程序，下一步操作就是编译这个C++文件。

点击进入beginner\_tutorials包的CmakeLists.txt文件，写入以下内容并保存：

|  |
| --- |
| $ |

## Python文件

# 可视化 Visualization



ROS系统中有自带的可视化程序，主要包括Rqt系列和Rviz

## Rqt

## Rviz

# 常用命令 Common Commands



## 文件访问

ROS的原生软件包rosbash引入了一系统接受软件包名称作为路径参数的命令，可用于高效地访问、查找、修改ROS软件包中的文件。下面所列举的文件操作命令并不是全部位于rosbash包中。

·官方文档：<https://wiki.ros.org/rosbash>

### roscd：访问指定包的路径

·官方文档：<https://wiki.ros.org/rosbash#roscd>

|  |  |
| --- | --- |
| roscd | cd到ROS主文件夹/opt/ros/<distros> |
| roscd roscpp | cd到roscpp包所在的文件夹 |

### rosls：列出指定包文件夹内的对象

·官方文档：<https://wiki.ros.org/rosbash#rosls>

|  |  |
| --- | --- |
| rosls roscpp -a | 列出roscpp包所在的文件夹中的所有对象 |

### rosed：编译指定软件包中的指定文件

·官方文档：<https://wiki.ros.org/rosbash#rosed>

|  |  |
| --- | --- |
| rosed test\_pkg test\_code.cpp | 使用vim编辑test\_pkg软件包中的test\_code.cpp |

### rospack：列出软件包的信息

·官方文档：<https://docs.ros.org/en/independent/api/rospkg/html/rospack.html>

## 启动与运行

### roscore：启动ROS系统

·官方文档：<https://wiki.ros.org/roscore>

每次使用ROS前要运行这个命令，它实际上执行三个任务：开启节点管理器Master、开启参数服务器Parameter Server、开启rosout日志节点。

### rosrun：运行指定软件包中的指定节点

·官方文档：<https://wiki.ros.org/rosbash#rosrun>

|  |  |
| --- | --- |
| rosrun test\_pkg test\_node | 运行test\_pkg软件包中的test\_node节点（可执行文件） |

### roslaunch：运行指定软件包中的指定.launch文件

·官方文档：<http://wiki.ros.org/roslaunch>

特别的，若roslaunch检测到roscore指令未运行，会自动运行roscore。

|  |  |
| --- | --- |
| rosrun test\_pkg test.launch | 运行test\_pkg软件包中的test.launch文件 |

## 参数服务器

### rosparam：参数服务器的操作

·官方文档：<http://wiki.ros.org/rosparam>

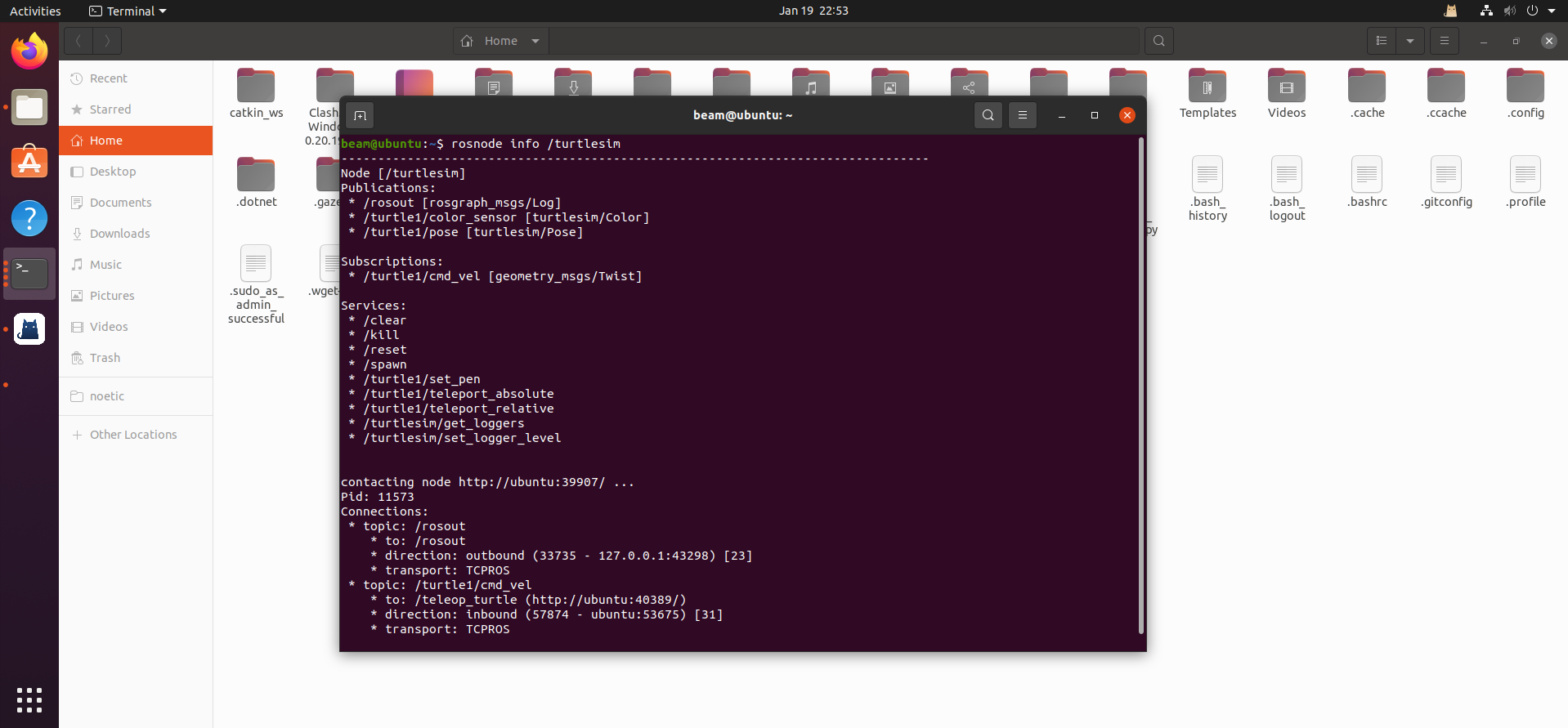
|  |  |
| --- | --- |
| rosparam list | 列出参数服务器中的所有参数 |
| rosparam get /turtlesim/background\_g | 查看/turtlesim/background\_g参数的值 |
| rosparam set /turtlesim/background\_g 9 | 设置/turtlesim/background\_g参数的值为9 |
| rosparam dump | 显示所有参数以及具体的值 |
| rosparam dump > save.txt | 将当前所有参数以YAML格式写入save.txt文件中 |
| rosparam load save.txt | 从save.txt中读取参数设置值 |
| rosparam delete /turtlesim/background\_g | 从参数服务器中删除参数/turtlesim/background\_g |

## 节点

### rosnode：运行中节点检索与操作

·官方文档：<http://wiki.ros.org/rosnode>

|  |  |
| --- | --- |
| rosnode list | 列出所有运行中的节点 |
| rosnode info /turtlesim | 查看/turtlesim节点的详细信息，包括节点名称、发布的消息、订阅的消息、提供的服务、进程编号PID、输入输出端口及协议 |
| rosnode ping /turtlesim | 测试能否连接到/turtlesim节点 |
| rosnode cleanup | 清除当前无法连接节点Unreachable Node的注册信息（无法连接的节点常为在命令行已经结束运行，但注册信息残存的节点） |
| rosnode kill /turtlesim | 关闭节点/turtlesim |
| rosnode machine my\_laptop | 查看计算机my\_laptop的中运行节点，常用于多机器ROS系统 |



图：运行rosnode info /turtlesim输出的节点详细信息

## 话题与消息

### rostopic：活跃话题检索与操作

·官方文档：<http://wiki.ros.org/rostopic>

|  |  |
| --- | --- |
| rostopic list | 列出系统中所有活跃话题 |
| rostopic info /turtle1/cmd\_vel | 查看/turtle1/cmd\_vel话题的详细信息，包括话题消息类型、发布者、订阅者 |
| rostopic type /turtle1/cmd\_vel | 查看/turtle1/cmd\_vel话题的消息类型 |
| rostopic find geometry\_msgs/Twist | 搜索使用了geometry\_msgs/Twist消息类型的活跃话题 |
| rostopic echo /turtle1/pose | 在命令行输出/turtle1/pose话题的消息（该命令会新创建一个订阅/turtle1/pose话题的节点） |
| rostopic hz /turtle1/pose | 输出/turtle1/pose话题的消息发布频率（该命令会新创建一个订阅/turtle1/pose话题的节点） |
| rostopic pub -r 2 /turtle1/cmd\_vel geometry\_msgs/Twist -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]' | 向/turtle1/pose话题发布指定参数的geometry\_msgs/Twist消息，且发布频率为2Hz（消息的类型必须与话题匹配） |

### rosmsg：消息类型查询

·官方文档： <http://wiki.ros.org/rosmsg>

|  |  |
| --- | --- |
| rosmsg list | 列出所有软件包中所有定义的消息 |
| rosmsg packages | 列出所有定义了消息（含有.msg）的软件包 |
| rosmsg info/show geometry\_msgs/Twist | 输出geometry\_msgs/Twist消息的具体定义，用info和show的效果是一样的 |
| rosmsg package mavros\_msgs | 列出mavros\_msgs软件包中定义的所有消息 |
| rosmsg md5 mavros\_msgs | （高级功能）显示mavros\_msgs的md5sum码 |

## 服务

### rosservice：活跃服务检索与操作

### rossrv：服务类型查询

# 仿真 Simulation

# 无人机控制——MAVROS

Mavros包是无人机通讯协议Mavlink的ROS接口，承担起ROS与Mavlink的桥梁，通过其提供的节点、命令和插件，就可以用ROS系统上运行的控制程序控制无人机的飞行。

# 参考资料

1. ROS官方教程：

<http://wiki.ros.org/ROS/Tutorials>

1. 古月居ROS入门21讲：

<https://www.bilibili.com/video/BV1zt411G7Vn/?vd_source=d208713af4296ed6230d640d411a0ebf>