



智能无人机技术设计实践

--ROS基础

联系方式: nics-efc@tsinghua.edu.cn

时间: 2021年12月20日 Monday





ROS 基础

- 1. 为什么要用ROS
- 2. ROS 的核心思想
- 3. ROS 的基础组件
- 4. ROS 常用工具（备查询）



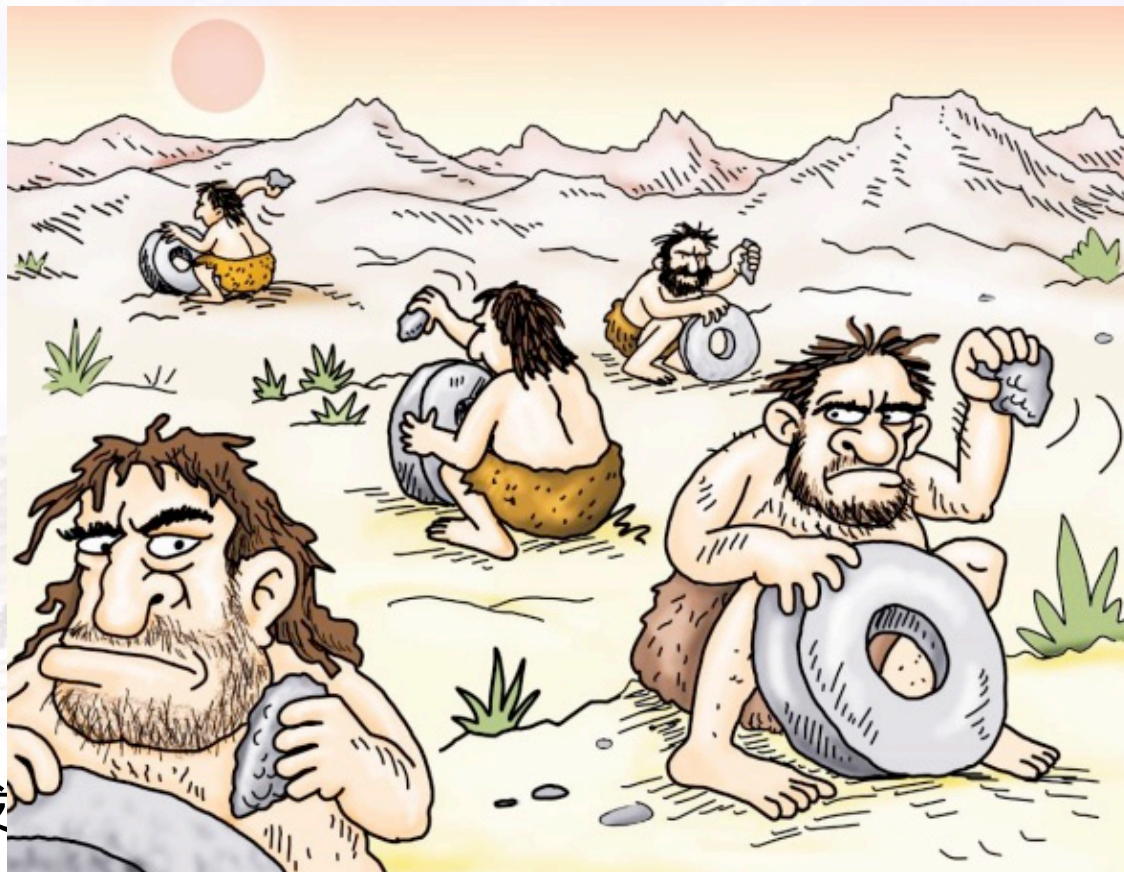
为什么要用ROS

➤ 不要重新造轮子：

- 感知模块
- 决策模块
- 控制模块
- 传感器模块
- 电源管理
- 可视化
- 调试接口
- ...

➤ 这么多功能如何开发

- 一套统一的框架





ROS 的起源

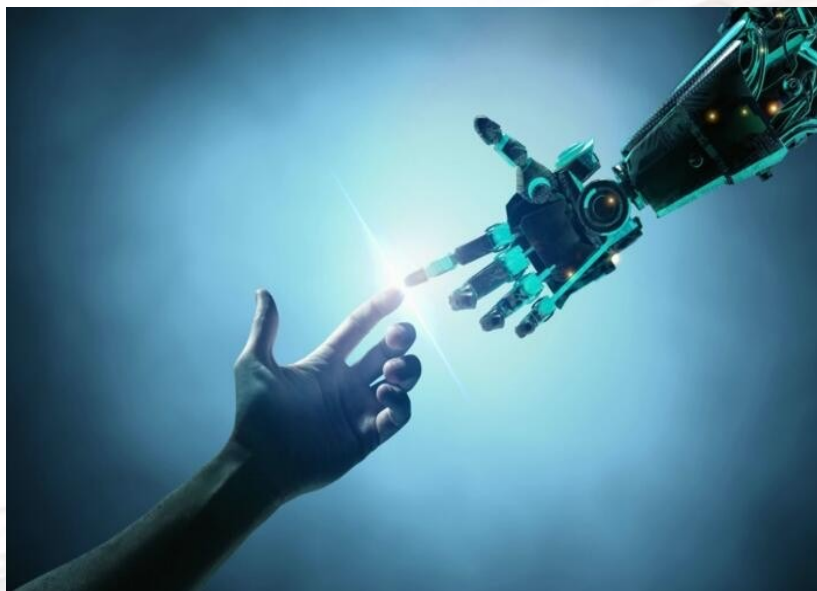


本世纪初，人工智能的研究热潮席卷全球，各种不同功能、不同结构的AI项目大量的出现。**人们需要一个有完整标准，统一接口和协议的AI搭建平台来便捷的开发不同的AI系统。**

斯坦福大学人工智能实验室创立了STAIR (Stanford Artificial Intelligence Robot) 项目，并组创建了灵活的、动态的软件系统的原型，用于机器人技术。

在2007年，机器人公司Willow Garage和该项目组合作，他们十分具有前瞻性的，提供了大量资源进一步扩展了这些概念，经过具体的研究测试实现之后，大量的研究人员将他们的专业性研究贡献到ROS核心概念和其基础软件包，为ROS丰富的功能打下了基础。

ROS软件的开发自始至终采用开放的BSD协议，在机器人技术研究领域逐渐成为一个被广泛使用的平台。



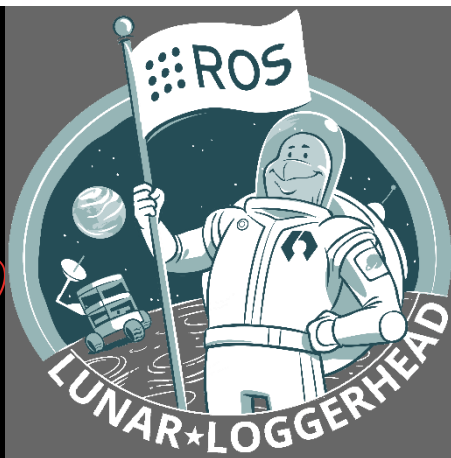


什么是ROS

ROS.org

Robot Operating System

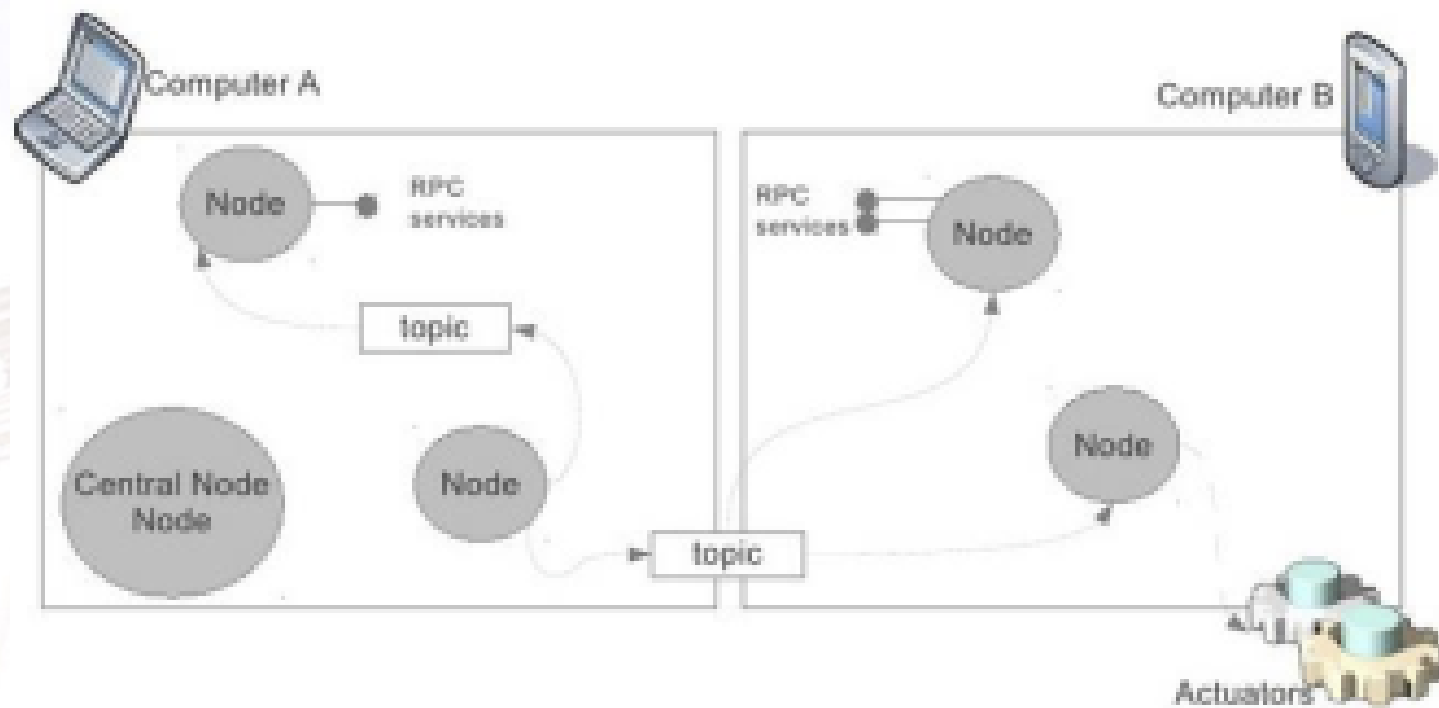
- 机器人的开发是一个庞杂的系统工程，涉及到机械、电子、控制、通信、软件等等诸多领域，个人独自开发面临着很多挑战。
- 随着机器人产业分工的细化，各个部件如底盘、电机、摄像头等分由不同的厂家进行生产，而各个部件需要一个统一的控制平台来完成集成。ROS操作系统就提供了这样一个平台来方便的进行机器人的操作。
- **ROS作为一个机器人编程框架或者说机器人的“操作系统”**，为各个零散的部件提供了通信的架构和标准，使机器人的感知、决策、控制算法可以更便捷、更精确的进行。
- ROS具有分布式点对点的特点，让每一个**进程**可以以独立的形式存在并运行，便于**模块化操作**下的功能实现和修改。





ROS 的基础组件

- Nodes
- Messages and Topics
- Services
- Parameters





ROS 的基础组件

➤ Nodes：简单理解就是一个可执行程序

- 节点是各自独立的可执行文件，能够通过话题、服务或参数服务器与其他进程（节点）通信。
- ROS通过使用节点将代码和功能解耦，提高了系统容错能力和可维护性，使系统简化。同时，节点允许了ROS系统能够布置在任意多个机器上并同时运行。
- 节点在系统中必须有唯一的名称。
- 节点可以使用不同的库进行编写，如roscpp和rospy。
- roscpp基于C++
- rospy基于Python。



ROS 的基础组件

➤ Topics : 不同程序之间通过Topic进行通信

- 话题是节点间用来传输数据的总线。
- 1-to-N Publish/Subscribe 模式: 同一个话题也可以有很多个订阅者。
- 使用TCP/IP传输, 称为TCPROS, ROS默认。
- 使用UDP传输, 称为UDPROS, 适合于远程操控任务。(低延迟高效率的传输方式, 但可能产生数据丢失)。



ROS 的基础组件

➤ Messages : 话题的具体传输的信息

- 一个节点通过向特定话题发布消息。
- 消息具有一定的类型和数据结构：包括 ROS 提供的标准类型和用户自定义类型。
- 消息的类型标准命名方式进行约定：功能包名称
msg 文件名称
- 常用数据类型：
 - String
 - int32
 - 更多的消息类型可以上ROS官网查询（比如查询几何变换相关的消息） http://wiki.ros.org/geometry_msgs



ROS 的基础组件

➤ **Services**：话题/消息发出去不见得有回应，
Service调用需要有应答

- 1 to 1 Service/Client 模型：当你需要直接与节点通信并获得应答时，将无法通过话题实现，这时需要使用服务。
- 服务需要由用户开发，节点并不提供标准服务。
- 本课程只需要调用Service，不需要掌握Service维护方法。



ROS 的基础组件

- Parameters : 对机器人的参数 (传感器、算法) 进行设置
 - 参数设置在参数服务器中
 - 用户在使用需要用到参数时可从参数服务器中获取 , 也可以手动更改参数服务器中的参数



. ROS 常用工具（备查询）

- 查询打印不同基础组件的信息

- rostopic(Topics)
- rosservice(Services)
- rosnode(Nodes)
- rosparam(Parameters)
- rosmmsg(Messages)
- rossrv(Services)

- 可视化调试

- rqt_graph（节点和消息）
- rqt_tf_tree（查看tf树）
- rrvz（可视化消息）



2 ROS文件系统



2.1 catkin编译及管理

◆ catkin编译系统

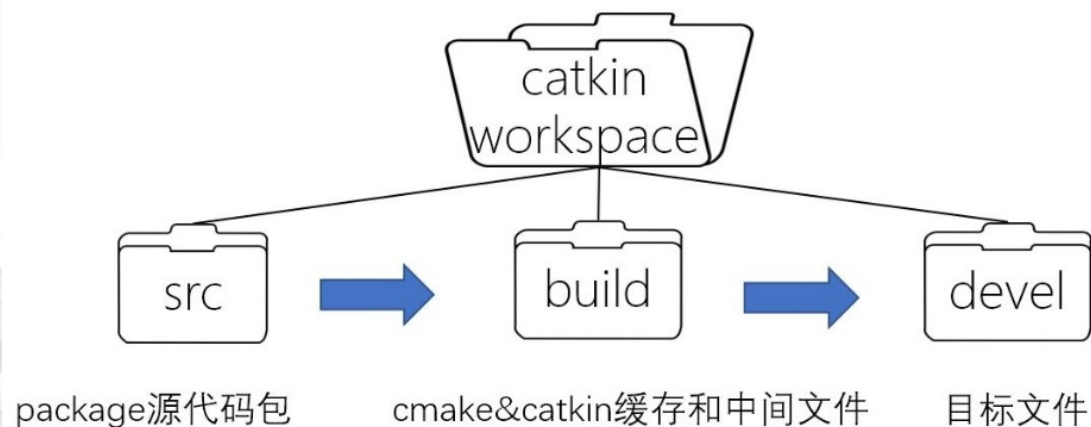
- ROS系统支持的编程语言主要是C/C++和Python两种。
- 源代码包需要经过编译才能被ROS所理解。
- ROS选用了CMake这一编译工具，并对其进行了扩展，形成了ROS使用的catkin编译系统。
- 一个catkin系统的软件包（package）必须要包含两个文件：
 - package.xml：包含了package的描述，例如包名, 功能描述, 版本号, 包作者等
 - CMakeLists.txt：构建package所需的CMake文件
- catkin封装后CMake编译工具操作简单，一个package编译过后可重复使用，非常方便。



2.1 catkin编译及管理

◆ catkin工作空间

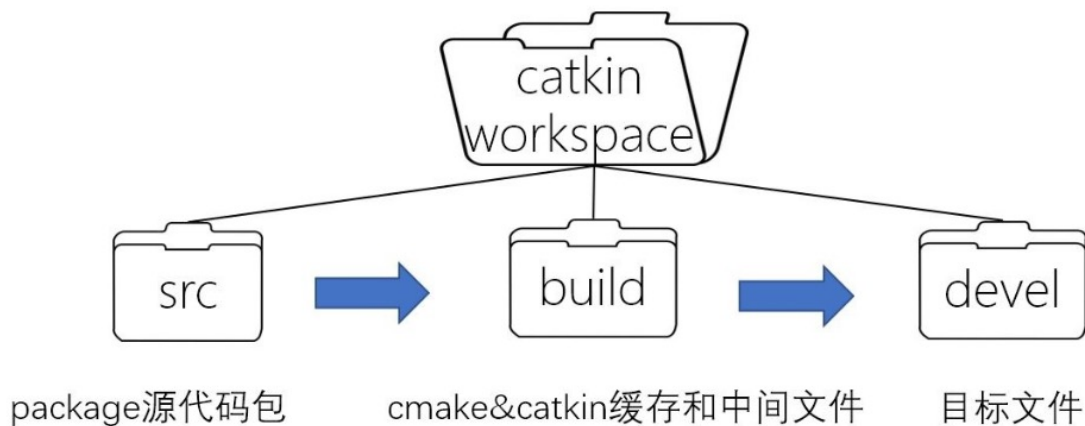
- catkin工作空间是创建、修改、编译catkin软件包的目录。它相当于一个仓库，装载着ROS的各种项目工程，便于系统组织管理调用。在Linux下catkin工作空间是一个文件夹，所有的ROS工程都以package的形式放在这个文件夹里。
- catkin工作空间的结构如下：



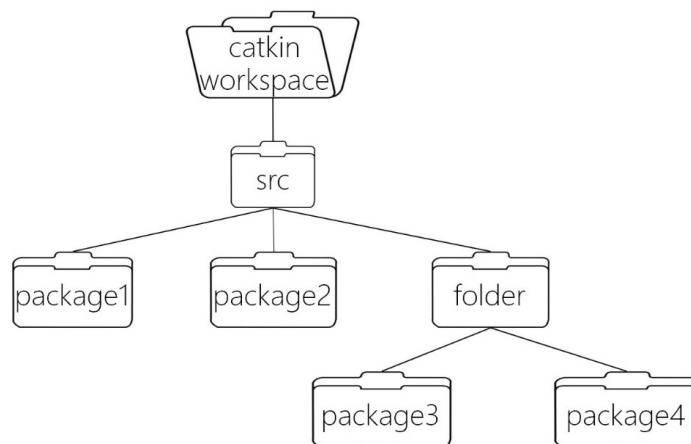
- ✓ /src：储存package软件包，是源代码的仓库。
- ✓ /build：CMake的缓存信息和中间文件。
- ✓ /devel：编译生成的环境变量和目标文件（头文件、动态链接库、静态链接库、可执行文件等）。



2.1 catkin编译及管理



- /build和/devel两个路径由catkin编译自动生成并管理，通常不需要涉及。
- 写好的软件包以package文件夹为单位储存在/src里，每次编译catkin会自动递归查找并编译所有在/src里的package，因此也可以把几个package放在同一个文件夹下，如：





2.2 package软件包

◆ package软件包

package软件包是catkin编译的基本单元，任何ROS程序都需要组织成package才可以进行编译。ROS的源代码就储存在package里，一个package可以视为作为一个完整的工程组，就像C++里的一个Project。

package中的文件、路径如下：

- CMakeLists.txt：定义package的编译规则，必需成分；
- package.xml：package的描述文件，必需成分；
- src/：存放.cpp和.py源代码的路径；
- include/：存放C++源码对应的头文件；
- scripts/：存放可执行脚本的路径，例如.sh和.py；
- msg/：存放自定义格式的消息.msg；
- srv/：存放自定义格式的服务.srv；
- launch/：存放launch文件xxx.launch。



2.2 package软件包

◆ package相关的操作命令

1. rospack指令

rospack是对package管理的工具，用法如下表：

rospack指令	作用
rospack help	显示rospack用法
rospack list	列出本机所有package
rospack depends [package]	显示package的依赖包
rospack find [package]	定位某个package
rospack profile	刷新所有package的位置

2. roscd指令

roscd [package]：直接cd到该package的路径下

3. rosls指令

rosls [package]：直接列出该package中的内容



2.2 package软件包

◆ package相关的操作命令

4. rosdep指令

rosdep是package管理依赖项的工具，用法如下表：

rosdep指令	作用
rosdep check [package]	检查package的依赖是否满足
rosdep install [package]	安装package的依赖
rosdep db	生成和显示依赖数据库
rosdep init	初始化/rosdep中的源
rosdep keys	检查package的依赖是否满足
rosdep update	更新本地的rosdep数据库

注：常使用rosdep install --from-paths src --ignore-src --rosdistro=kinectic -y 来安装工作空间中src/路径下所有package的依赖项。依赖项由每个package的package.xml指定。



2.3 CMakeLists.txt与package.xml文件

◆ CMakeLists.txt文件 <http://wiki.ros.org/catkin/CMakeLists.txt>

CMakeLists.txt原本是CMake编译系统的规则文件，catkin编译系统是在CMake编译系统上扩展得到的，故直接保留了这个规则文件，只是针对ROS工程添加了一些定义。

切换行号显示

```
1  # Get the information about this package's buildtime dependencies
2  find_package(catkin REQUIRED
3    COMPONENTS message_generation std_msgs sensor_msgs)
4
5  # Declare the message files to be built
6  add_message_files(FILES
7    MyMessage1.msg
8    MyMessage2.msg
9  )
10
11 # Declare the service files to be built
12 add_service_files(FILES
13   MyService.srv
14 )
15
16 # Actually generate the language-specific message and service files
17 generate_messages(DEPENDENCIES std_msgs sensor_msgs)
18
19 # Declare that this catkin package's runtime dependencies
20 catkin_package(
21   CATKIN_DEPENDS message_runtime std_msgs sensor_msgs
22 )
23
24 # define executable using MyMessage1 etc.
25 add_executable(message_program src/main.cpp)
26 add_dependencies(message_program ${PROJECT_NAME}_EXPORTED_TARGETS)
27
28 # define executable not using any messages/services provided by this package
29 add_executable(does_not_use_local_messages_program src/main.cpp)
30 add_dependencies(does_not_use_local_messages_program ${catkin_EXPORTED_T
```




2.3 CMakeLists.txt与package.xml文件

◆ package.xml文件 <http://wiki.ros.org/catkin/package.xml>

package.xml也是一个catkin的package必备文件，它是这个软件包的描述文件，包含了package的名称、版本号、内容描述、维护人员、软件许可、编译构建工具、编译依赖、运行依赖等信息。

```
<package>
  <name>foo_core</name>
  <version>1.2.4</version>
  <description>
    This package provides foo capability.
  </description>
  <maintainer email="ivana@willowgarage.com">Ivana Bildbotz</maintainer>
  <license>BSD</license>

  <url>http://ros.org/wiki/foo_core</url>
  <author>Ivana Bildbotz</author>

  <buildtool_depend>catkin</buildtool_depend>

  <build_depend>message_generation</build_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>std_msgs</build_depend>

  <run_depend>message_runtime</run_depend>
  <run_depend>roscpp</run_depend>
  <run_depend>rospy</run_depend>
  <run_depend>std_msgs</run_depend>

  <test_depend>python-mock</test_depend>
</package>
```



2.4 其他文件

◆ launch文件

launch文件一般以.launch或.xml结尾，它对ROS需要运行程序进行了打包，通过一句命令来启动。一般launch文件中会指定要启动哪些package下的哪些可执行程序，指定以什么参数启动，以及一些管理控制的命令。launch文件通常放在软件包的launch/路径中。

◆ yaml文件

yaml文件一般存储了ROS需要加载的参数信息，一些属性的配置。通常在launch文件或程序中读取.yaml文件，把参数加载到参数服务器上。通常我们会把yaml文件存放在param/路径下。



谢谢！

答疑地点：双清大厦2号楼502