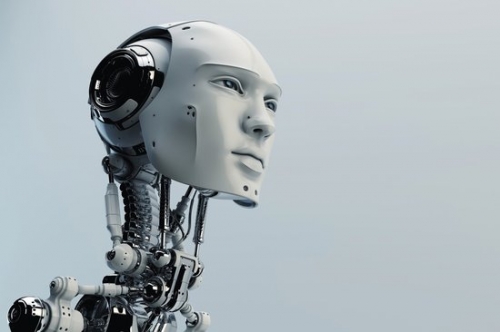
ROS入门



中国大学MOOC

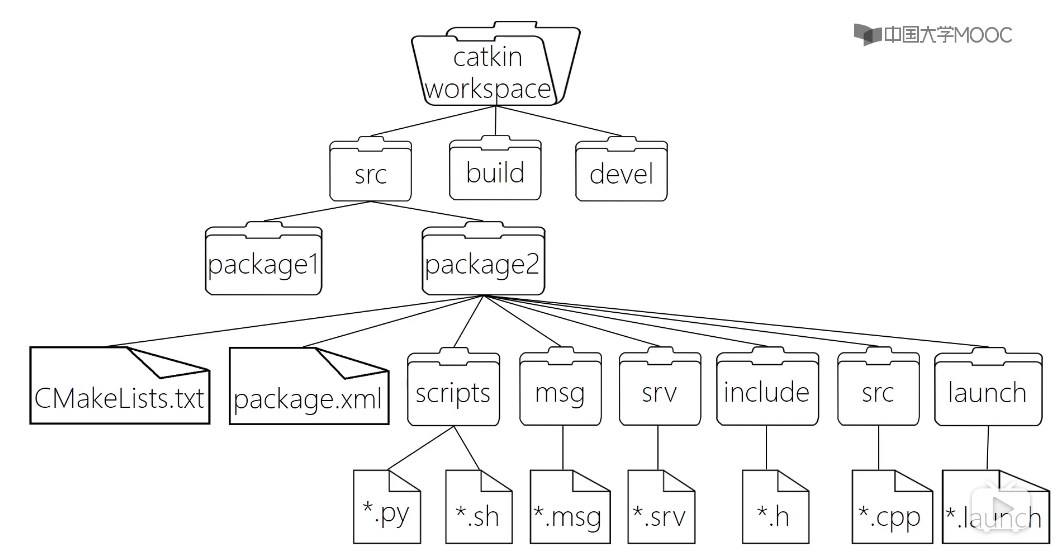
2019.3

* 1. Welcome

AI is the new electricity! --Andrew Ng

* 1. 什么是ROS

1. ROS=框架+工具+功能+社区
2. 一个Node就是一个进程。
   1. 机器人与ROS演示
   2. ROS的安装与配置
   3. catkin工作空间与编译系统
3. 一个典型的ROS工程结构：



1. 建立工作空间：

$ mkdir -p ~/catkin\_ws/src

$ cd ~/catkin\_ws/

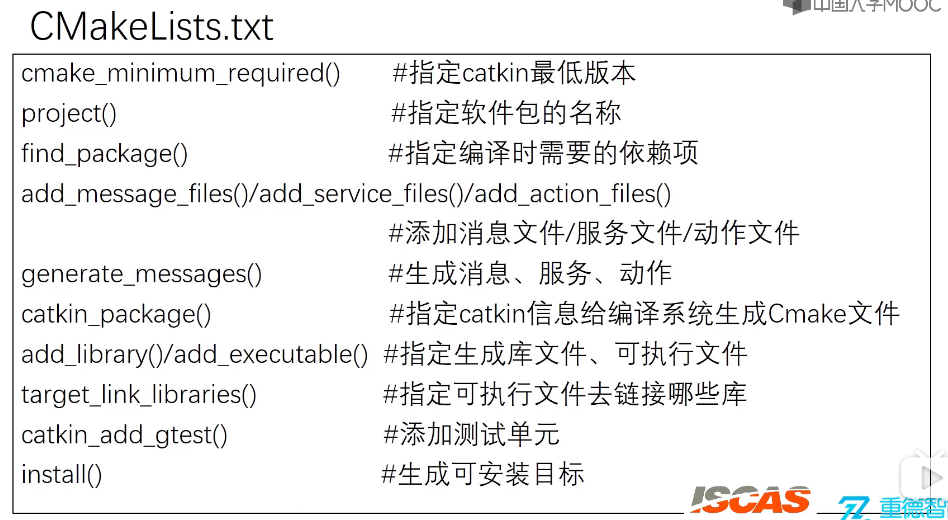
$ catkin\_make

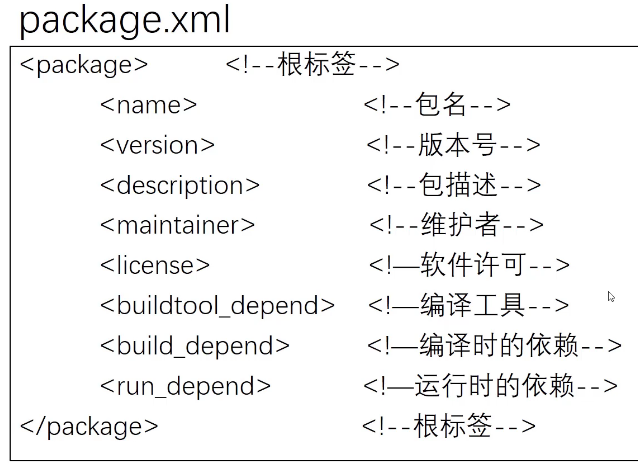
编译：

$ cd ~/catkin\_ws #必须先回到工作空间才能进行catkin\_make操作

$ catkin\_make

$ source ~/catkin\_ws/devel/setup.bash #编译完成后要source刷新环境

1. Package是catkin编译的基本单元。
   1. package组成
2. 一个package至少要包括一个CMakeLists.txt和一个package.xml，缺一不可。
3. CMakeLists.txt规定catkin编译的规则；举例如下：

Package.xml定义package的属性信息；举例如下：

1. Package下的scripts文件夹用来放可执行脚本(Python,shell);include文件夹和src文件夹分别用来放C++的头文件和源文件.但src文件夹下也可以放python的module(可import).
2. Package下除了上述内容,还可以放自定义的通信内容,比如消息(msg),服务(srv),动作(action).
3. 通常,package里还有launch文件和配置文件(yaml),launch文件用来一次启动多个可执行文件.
4. 几个常用指令:

$ rospack find package\_name #查找某个package的地址

$ rospack list #列出本地所有package

$ roscd package\_name #跳转到某个package路径下

$ rosls package\_name #列出某个package下的文件信息

$ rosed package\_name file\_name #编辑某个package下的文件

$ catkin\_create\_pkg <package\_name> [deps] #创建一个package

$ rosdep install [package\_name] #安装某个package所需的依赖

* 1. 操作演示:catkin操作空间探索
  2. Metapackage

1. Metapackage本身没有什么实质性的内容,相当于一个”虚包”,但是依赖了很多的包,安装ROS时的ros-kinetic-desktop-full就是一个metapackage.
2. ROS中常见的metapackage有:

Navigation 导航相关的功能包集

Moveit 运动规划相关的功能包集(主要是机械臂)

Image\_pipeline 图像获取和处理相关的功能包集

Vision\_opencv ROS与OpenCV交互的功能包集

Turtlebot turtlebot机器人相关的功能包集

Pr2\_robot pr2机器人驱动功能包集

* 1. master和Node

1. Master称为节点管理器，每个node启动时都要向master注册，master负责管理各个node间的通信。
2. 启动master的指令：$ roscore 此指令除了启动master外，还会启动rosout（日志输出）和parameter server（参数服务器）。
3. 和node有关的几个常用指令：

$ rosrun [package\_name] [node\_name] #启动一个node

$ rosnode list #列出当前运行的node信息

$ rosnode info [node\_name] #显示某个node的详细信息

$ rosnode kill [node\_name] #结束某个node

$ roslaunch [pkg\_name] [file\_name.launch] #启动master和多个node

* 1. 操作演示
  2. topic和msg

1. Topic：ROS中的异步通信方式，node间通过publish-subscribe机制通信。
2. 一个topic可以有多个subscriber和多个publisher。
3. Message：topic内容的数据类型，定义在file\_name.msg文件中。
4. 和topic、message有关的常用指令：

$ rostopic list #列出当前所有topic

$ rostopic info /topic\_name #显示某个topic的具体信息

$ rostopic echo /topic\_name #显示某个topic的内容

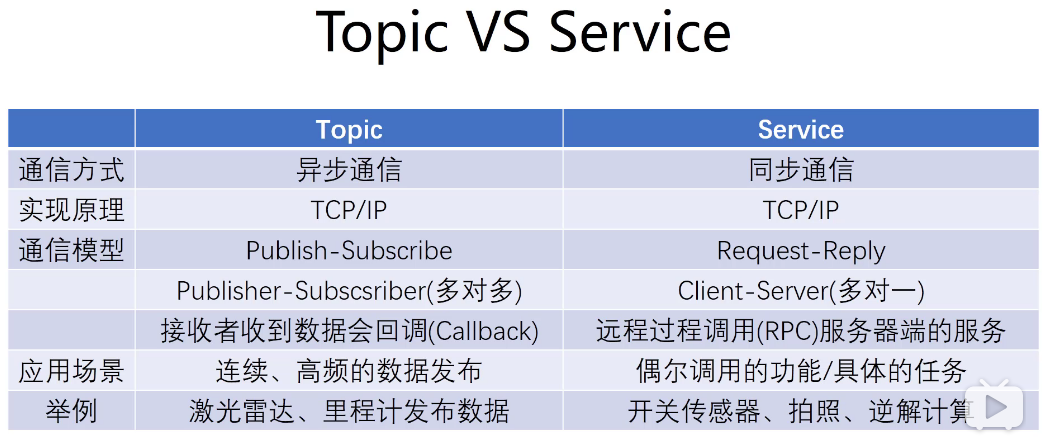
$ rostopic pub /topic\_name ... #向某个topic发布内容

$ rosmsg list #列出系统上所有message

$ rosmsg show /msg\_name #显示某个message的内容

* 1. 操作演示
  2. Service和srv

1. Service：ROS中的同步通信方式，node间通过request-reply方式通信。



1. Srv：Service通信的数据格式，定义在file\_name.srv文件中。
2. 几个常用的指令：

$ rosservice list #列出当前活跃的所有survice

$ rosservice info service\_name #显示某个service的具体信息

$ rosservice call service\_name args #调用某个service

$ rossrv list #列出系统上所有srv

$ rossrv show srv\_name #显示某个srv的内容

* 1. Parameter Server

1. Parameter server：存储各种参数的字典，可用命令行、launch文件和node（API）读写。
2. 用命令行对Parameter Server进行操作的相关指令：

$ rosparam list #列出当前所有参数

$ rosparam get param\_key #显示某个参数的值

$ rosparam set param\_key param\_value #设置某个参数的值

$ rosparam dump file\_name #保存参数到文件

$ rosparam load file\_name #从文件读取参数

$ rosparam delete param\_key #删除某个参数

1. Parameter Server在launch文件中的配置方法：利用param和rosparam两个标签。
   1. 操作演示
   2. Action
2. Action：类似于Service，但是是一种带有状态反馈的通信方式，通常用在长时间、可抢占的任务中。
3. action:Action通信的数据格式，定义在file\_name.action中。

5.1 Gazebo

1. Gazebo：仿真

5.2 RViz

1. RViz: The Robot Visualization Tool，可视化工具，方便监控和调试。

5.3 rqt

1. rqt：可视化工具，常用rqt\_graph、rqt\_plot、rqt\_console

2. rqt\_graph：显示通信架构；rqt\_plot：绘制曲线；rqt\_console：查看日志。

5.4 bag

1. rosbag：ROS命令行工具，记录和回放数据流。可以监听指定的topic，给数据打上时间戳，保存到一个file\_name.bag文件中，有了这个文件就可以回放机器人运行过程中监听数据的变化情况。

2. 指令：

$ rosbag record <topic\_name> #记录指定topic到bag中

$ rosbag record -a #记录所有topic到bag中

$ rosbag play <bag\_file> #回放bag

* 1. Client Library与roscpp

1. Client Library：类似于API，提供ROS编程的库（接口），例如：建立node、发布消息、调用服务...
2. Roscpp常用功能：

Ros::init() : 解析传入的ROS参数，使用roscpp第一步需要用到的函数

Ros::NodeHandle : 和topic、Service、param等交互的公共接口

Ros::master : 包含从master查询信息的数据

Ros::this\_node : 包含查询这个进程（node）的函数

Ros::service ： 包含查询服务的函数

Ros::param : 包含查询参数服务器的函数，而不需要用到NodeHandle

Ros::names : 包含处理ROS图资源名称的函数

* 1. topic\_demo上
  2. topic\_demo下
  3. Service\_demo