

mkdir -p 自定义主目录/src

cd 自定义空间目录。

catkin_make

cd src

catkin_create_pkg 循环定义包名 rosdep rospy std_msgs

cd 自定义包名

catkin_make → 需要在相同目录

roscore 必要步骤

新建终端

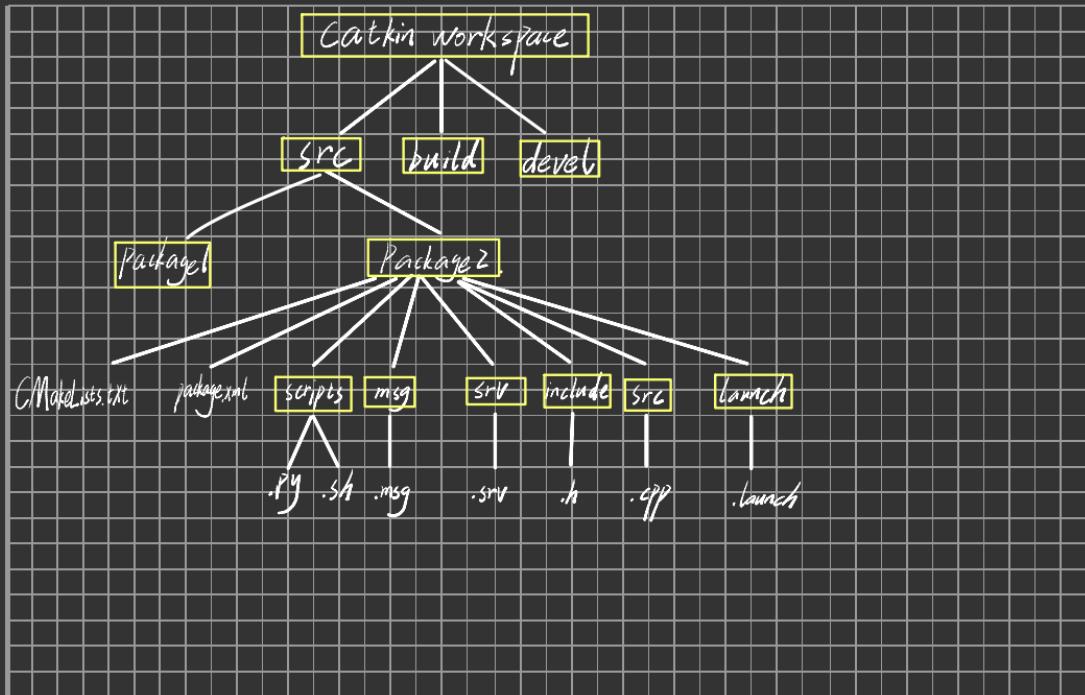
rosrun turtlesim turtlesim_node

新建终端

source 工程目录/develop/setup.bash

rosrun 取包名 命令行

Kos 文件体系、统一



WorkSpace --- 自定义的工作空间

|- build: 编译空间, 用于存放CMake和catkin的缓存信息、配置信息和其他中间文件。

|- devel: 开发空间, 用于存放编译后生成的目标文件。包括头文件、动态&静态链接库、可执行文件等。

|- src: 源码

-- package: 功能包(ROS基本单元)包含多个节点、库与配置文件。包名所有字母小写, 只能由字母、数字与下划线组成

|- CMakeLists.txt 配置编译规则, 比如源文件、依赖项、目标文件

|- package.xml 包信息, 比如包名、版本、作者、依赖项...(以前版本是 manifest.xml)

|- scripts 存储python文件

|- src 存储C++源文件

|- include 头文件

|- msg 消息通信格式文件

|- srv 服务通信格式文件

|- action 动作格式文件

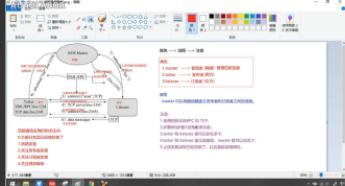
|- launch 可一次性运行多个节点

|- config 配置信息

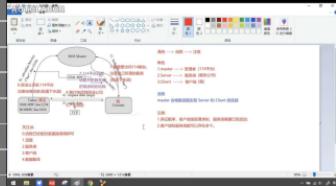
-- CMakeLists.txt: 编译的基本配置

通信机制

连接通信



服务通信



参数服务体系

rosnode ping	测试到节点的连接状态
rosnode list	列出活动节点
rosnode info	打印节点信息
rosnode machine	列出指定设备上节点
rosnode kill	杀死某个节点
rosnode cleanup	I 清除不可连接的节点

rosmsg show	显示消息描述
rosmsg info	显示消息信息
rosmsg list	列出所有消息
rosmsg md5	显示 md5 加密后的消息
rosmsg package	显示某个功能包下的所有消息
rosmsg packages	列出包含消息的功能包

rosparam set	设置参数
rosparam get	获取参数
rosparam load	从外部文件加载参数
rosparam dump	将参数写出到外部文件
rosparam delete	删除参数
rosparam list	列出所有参数

更多操作命令

rostopic bw	显示主题使用的带宽
rostopic delay	显示带有 header 的主题延迟
rostopic echo	打印消息到屏幕
rostopic find	根据类型查找主题
rostopic hz	显示主题的发布频率
rostopic info	显示主题相关信息
rostopic list	显示所有活动状态下的主题
rostopic pub	将数据发布到主题
rostopic type	打印主题类型

rossrv show	显示服务消息详情
rossrv info	显示服务消息相关信息
rossrv list	列出所有服务信息
rossrv md5	显示 md5 加密后的服务消息
rossrv package	显示某个包下所有服务消息
rossrv packages	显示包含服务消息的所有包

资源操作

rosservice args	打印服务参数
rosservice call	使用提供的参数调用服务
rosservice find	按照服务类型查找服务
rosservice info	打印有关服务的信息
rosservice list	列出所有活动的服务
rosservice type	打印服务类型
rosservice uri	打印服务的 ROSRPC uri

匀速线速度 $\rightarrow x$

角速度 $\rightarrow z$

话题 $\rightarrow w$

消息 $\rightarrow u$

API

操作ros节点

`ros::init(argc, argv, "node_name")`

调用ROS相关函数

参数：
argc 封装实参数个数(1+1)

argv 封装参数的数组

name 为节点命名(唯一性)

options 命名参数选项(选择使用)

使用

`argc, argv`:

按照特定格式输入参数，可以加以利用 for instance 设置空字符串，给予之重命名
options:

因为节点称唯一性，同一个节点不能重复启动

若需要一个节点多次启动且正常运行，设置启动项 `ros::init_options::AnonymousName`
(创建ros结时会自动产生随机名字并返回给启动项)

创建发布者对象

模板：被发布的消息类型

参数：
主题名称

序列长度

batch(可选)

spinOnce()

处理一轮回调

通常运用在循环体

spin()

进入循环不会退出

表现差异

spinOnce()之后可实现

spin() 之后不可实现

rosbag play /path/to/your/rosbag/bagfile.bag

rostopic list

话题列表

rostopic info 话题名 (小字)

消息

rosmsg info 消息名 (大字)

了解具体消息

ROS | ← Launch 文件 一次性启动多个节点

node Pkg →

包含某节点的功能包

type →

被运行的节点

name →
为节点名

output →
设置节点的输出目标

功能包中添加 Launch 文件夹

在文件夹中添加 .launch 文件

< launch >

```
< node pkg="." type=" " name=" " />
```

</ launch >

roslaunch 包名 .launch

(Output = " ")
↓

终端输出
screen (终端)

roslaunch 能保证
按照 node 的声明顺序来
启动节点

注释掉

<!-- -->

<launch>根标签
子标签

<launch deprecated=""> 不再使用该 launch 文件输出的内容

node 以脚本形式

output="log/screen" 一般的输出到 log 软件 / screen 屏幕，默認為 log。

args="" 将参数传递给节点

machine="主机名" 在指定机器上启动节点

respawn="true/false" 如果节点退出，是否能恢复

respawn-delay="N" 如果 respawn 为 true，并且延迟进入秒后启动节点

required="true/false" 该节点在必要的时候被调用，当该节点退出时将自动挂起

ns="xxx" 在指定命名空间 xxx 中启动节点 避免重名

clear-params="true/false" 在启动前删除节点的重命名的所有参数，慎用

</node>

include - 使用其他 launch 文件

<include file="\$find(已知)/xxx/xxx.launch"/> 等待的文件插槽 ns="xxx" (可选) 在指定命名空间启动

remap > 话题重命名

```
<remap from=" " to=" " />
```

param 主要用于参数服务器上设置参数，被派到执行节点通过value指定，也可通过树的方式加载，在nodes树签中相当子属性

```
<param name=" " type=" " value=" " /> (节点 launch 下 node 属性)
```

```
<param name=" " type=" " value=" " /> (节点, node 属性)
```

name = 节点算子参数名

value = "xxx" (可选) 一定义参数值，此时源码必须指定外部文件作为参数原

type = "str/int/double/bool/float" (可选) float 是一个文件

一指定参数类型，如果未指定，noLaunch 会尝试确定参数类型，规则如下：

如果“包含”单精度浮点数字符串，识别为 float “非单精度”是双精度（double） 其他类型

在 node.js 中使用参数

`rosparam` 一可以以从 YAML 文件导入参数，或将参数导出到 YAML，也可以删除参数

```
<rosparam command="load" file="$(find $(rospack find rosparam))/xxx.yaml"/> (添加到 .yaml 文件中)
<rosparam command="dump" param=" " />
command = load/dump/delete (加载/导出/删除)
↳ 导出参数时 launch 文件会自动将参数写入文件
param = 参数名称 ns = 动作空间 (可选)
```

`group` 一对括号分组具有属性，可以让其归属于某个命名空间

ns = 命名空间 (可选)

clar, params = command (可选) 一般向宏函数传递的都是对象参数 指定参数

参数传递 读取参数

```
new Parameter()
  .add("param1", "float", "Parameter type float", "param1", "0.1")
  .add("param2", "float", "Parameter type float", "param2", "0.2")
  .add("param3", "float", "Parameter type float", "param3", "0.3")
```

`arg` 一用于动态地给类赋予参数的参数

name = 参数名称 default = 默认值 (可选) value = 值 (可选)一个静态 default 属性 doc = 描述 - 参数说明

```
new Argument()
  .add("car_length", "float", "Car length", "car_length", "0.33")
  .add("car_length", "float", "Car length", "car_length", "0.33")
  .add("car_length", "float", "Car length", "car_length", "0.33") ...
```