ROS星火计划

主讲：田博
http://answers.ros.org/users/1587/tianb03/
www.tianb03.blogspot.com
https://www.zhihu.com/people/tianb03

ROS发展背景

- 问题：机器人跨学科特点，系统庞大，工程应用涉及面广，硬件繁多，软件开发困难、低级重复
- 基础：机器人研究相关领域发展迅速：计算机视觉，异构计算加速，室内移动机器人算法的成熟等等。
- 需求：机器人研究过程中需要统一开发/测试平台
- 解决方法：机器人软件中间件

ROS发展历史

- 起源：STAIR PR 项目
- 兴起：Willow garage 与PR2
- 现状：osrf 接手ROS

ROS 8周年
官方宣传视频

- https://www.youtube.com/watch?v=Z70_3wMFO24
- 云盘地址：https://yunpan.cn/cxVh997fn2kmQ

ROS ：优点

- 分布式计算
- 软件复用
- 功能包管理与多语言构建
- 快速原型与测试
- 仿真与实体机器人的无缝转换

ROS ：不足

- 目前基于Ubuntu系统，移植嵌入式系统困难
- 无实时性设计
- 体积较大
- 系统整体运行效率低
- 总的来说，ROS不擅长开发成熟的商业产品。（*？？？不是田师说的！！！*）

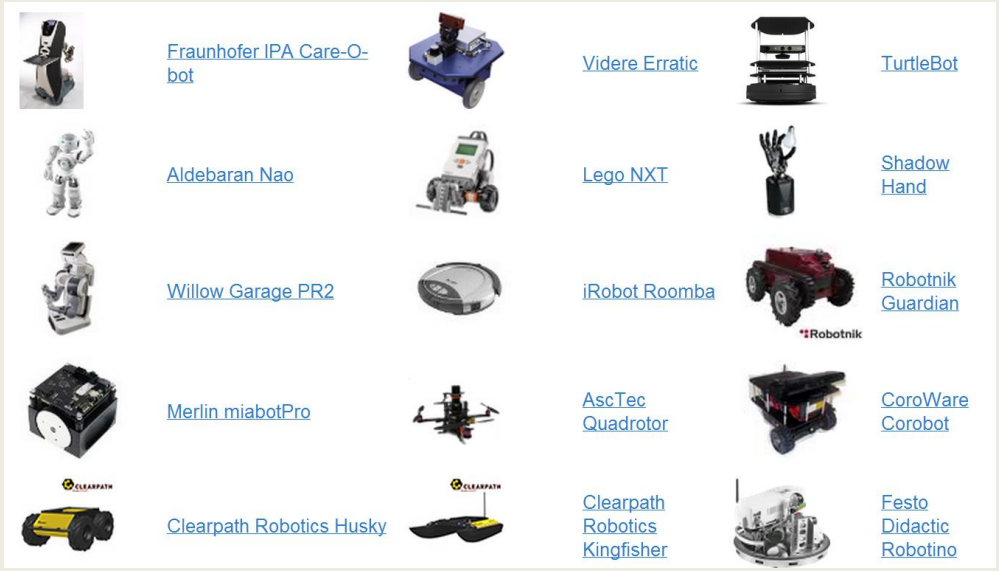
一句话概括ROS

田师的一句话：书同文，车同轨。

- 规定了统一通信接口的松耦合分布式开源软件框架。
- 最大的好处： 即插即用，提供统一软件接口，事实上的机器人研究的统一平台。
- 有了统一的平台，我们才有共同交流的可能，以上也是我们学习ROS的主要原因。

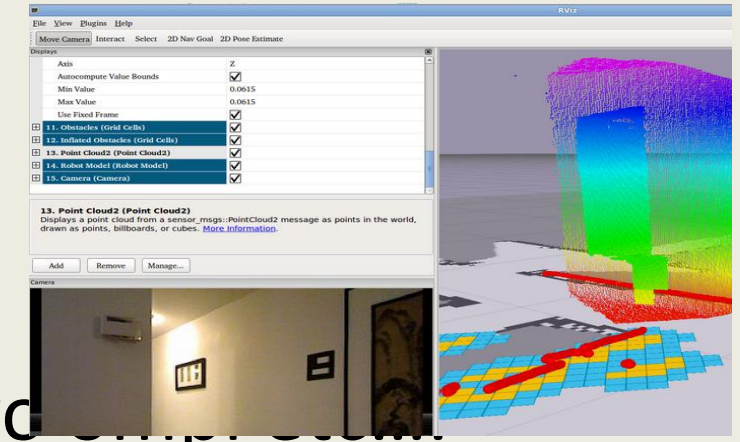
ROS应用场景（1）机器人

- Mobile manipulator ： PR2
- Flight control system ： pixhawk
- Multiple sensor：hokuyo sick pointgrey
- 更多：http://wiki.ros.org/Robots



ROS应用场景（2）先进算法及行业应用

- ros-industrial



- state_of_art algorithm：lsd_slam svo

- Software Architecture ：ros_control moveit

ROS应用场景:总结

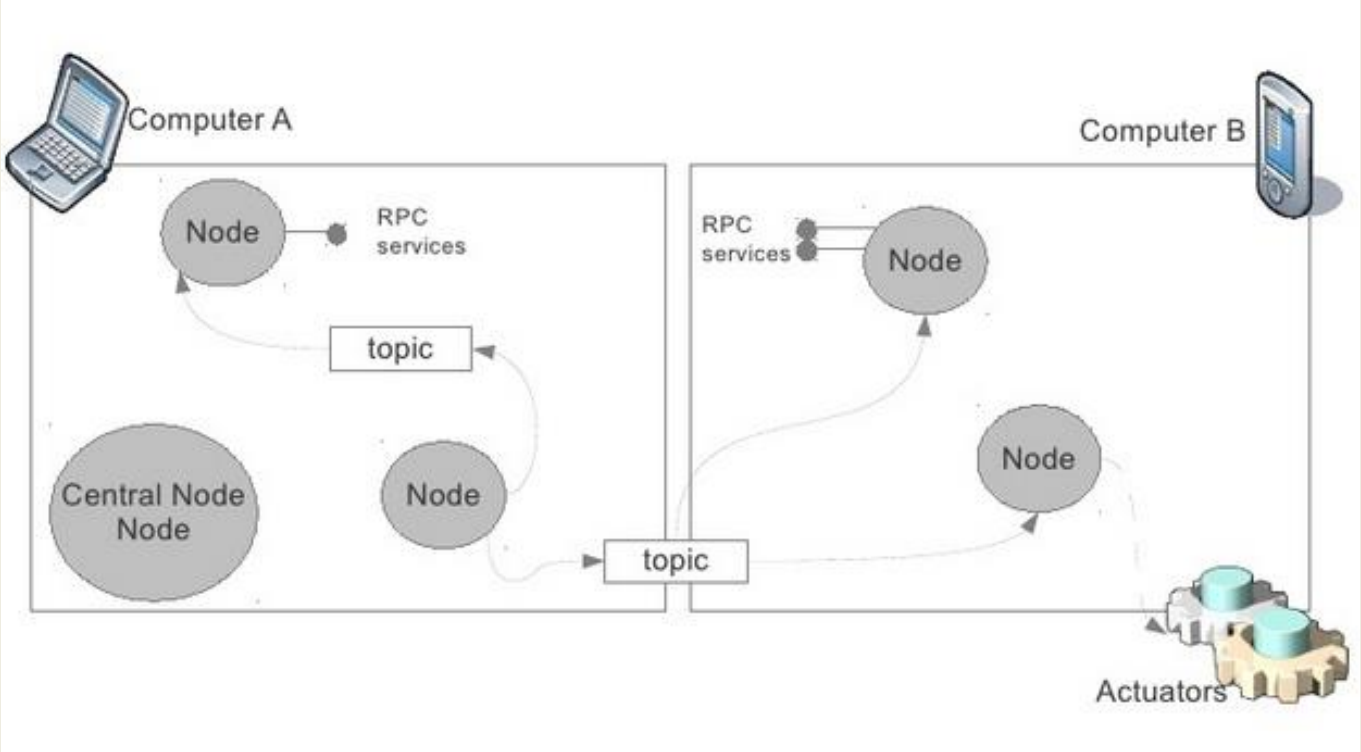
- 商业上的机器人产品一般会给出ROS的软件接口（比如ros-I 基于的工业机器人的moveit接口）商业上很多机器人相关软件也会给出ROS接口（比如webots仿真器，matlab的robot system toolbox），从一方面说明ROS的普及程度。

- ROS现在最广泛的用途：算法与硬件的快速结合，系统集成方案快速构建、评估与验证。

- 现阶段不建议将ROS**整体、直接**用于商业产品。（*我们来好好谈谈。。*）

- ROS 2.0 ：新的架构（取消了Master），新的通信机制（DDS数据分发协议），对实时性的支持，有望实现嵌入式应用。（*用户太少*）

ROS设计思想： 分布式架构

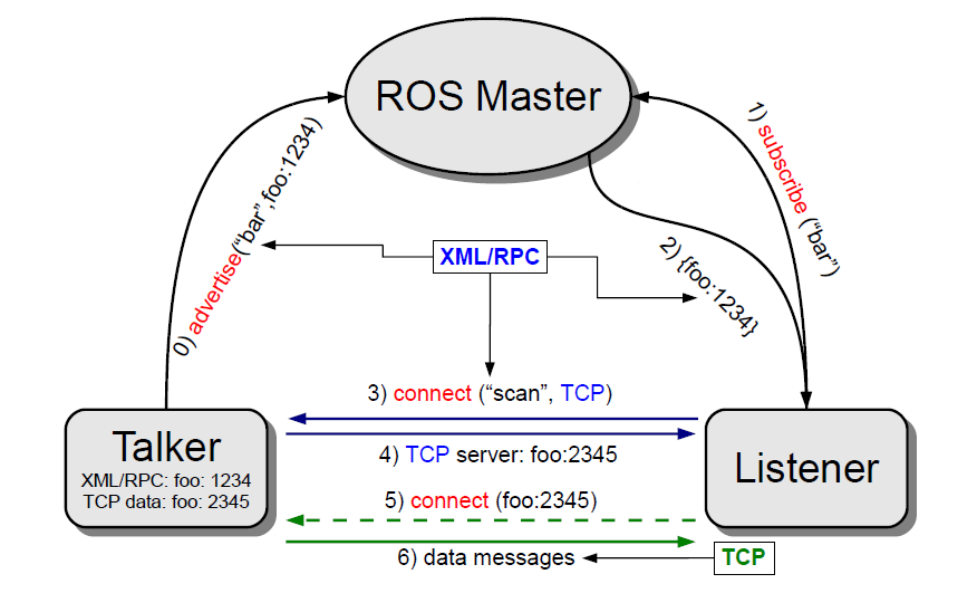


ROS 核心概念

- Nodes 节点
- Messages and Topics 消息与话题（或主题）
- Services 服务
- ROS Master ROS管理器
- Parameters 参数
- Stacks and packages 功能包集与功能包

ROS Master

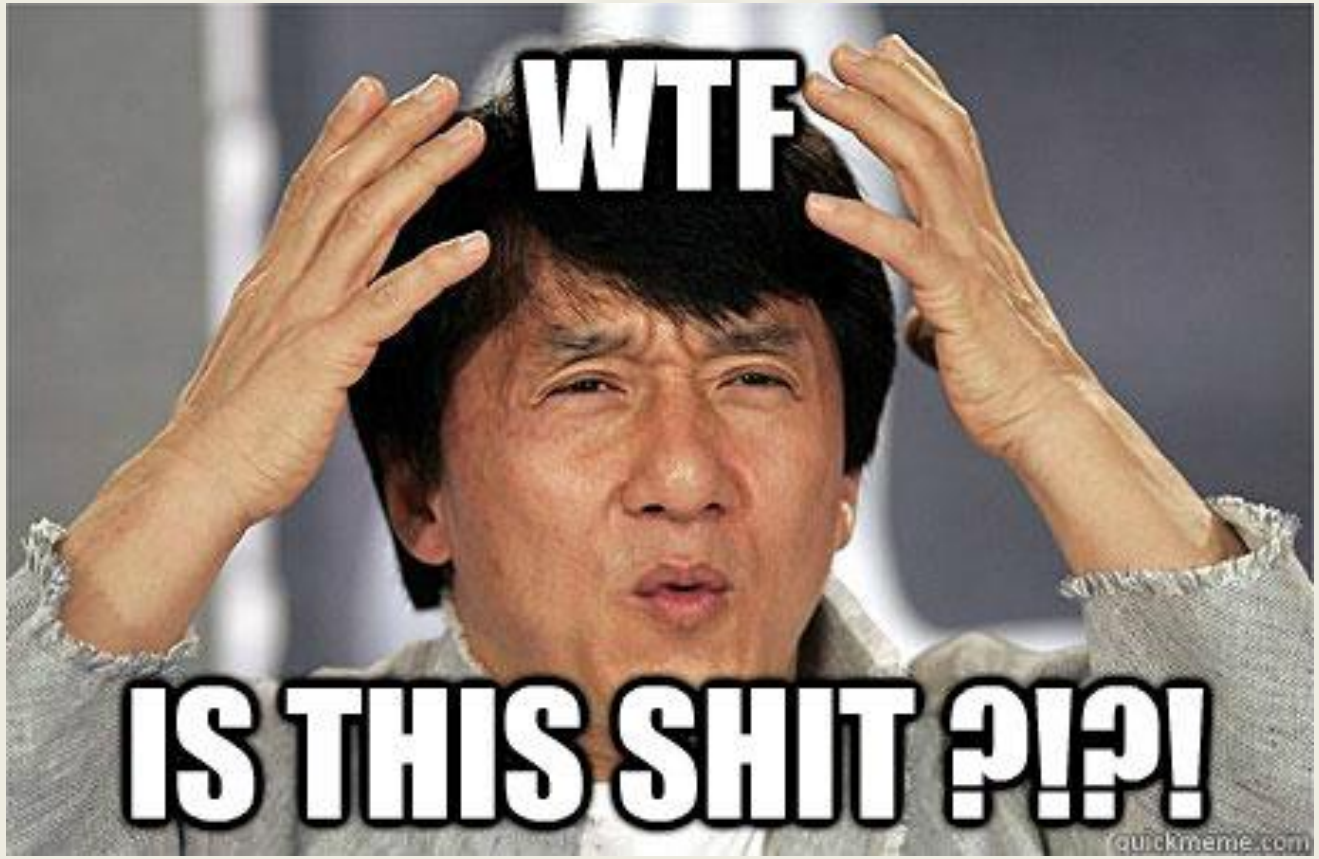
- 向ROS系统中其他节点提供命名和注册服务
- 跟踪和记录话题的发布者和订阅者
  - 使ROS 节点之间能够相互查找。一旦节点找到了彼此，就能建立一种点对点的通信方式。
- 提供参数服务器



ROS 常用命令工具

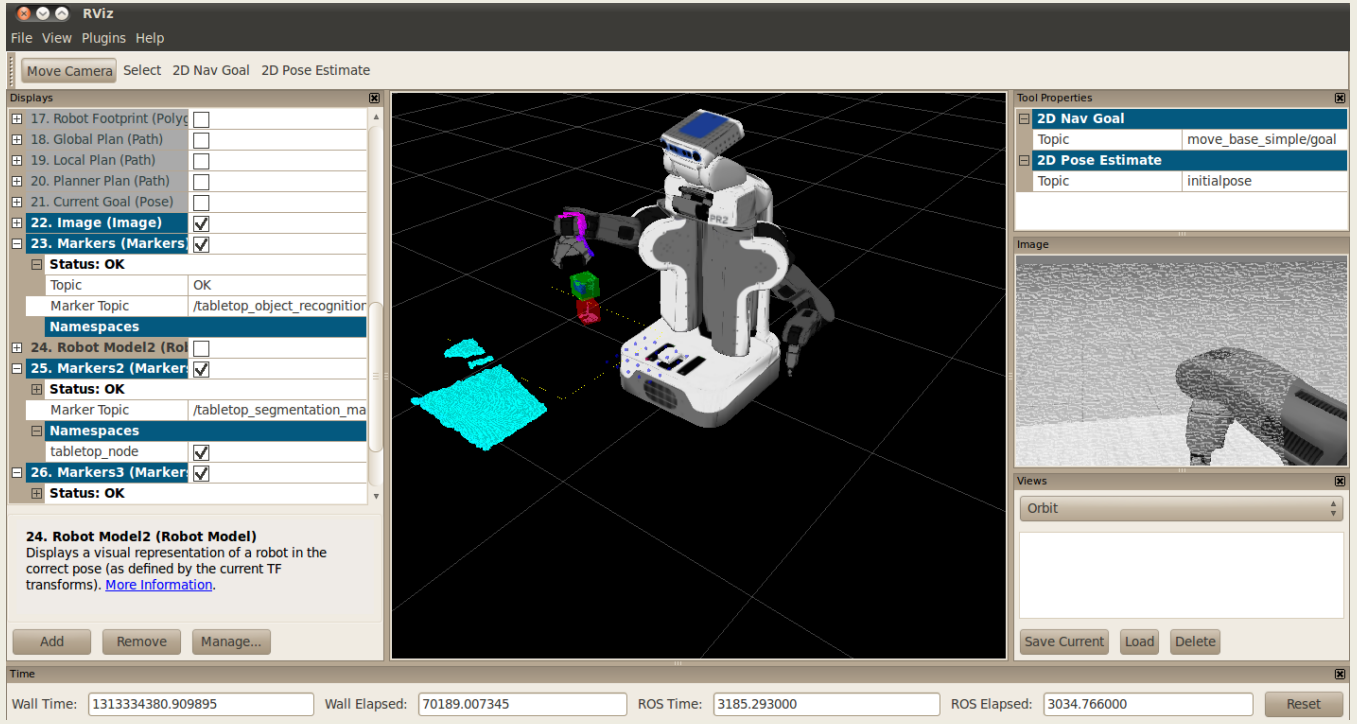
- rostopic (Topics)
- rosservice (Services)
- rosnode (Nodes)
- rosparam (Parameters)
- rosmsg (Messages)
- rossrv (Services)
- roswtf (General debugging)（*不是rosgd？*）

所以说英语要好



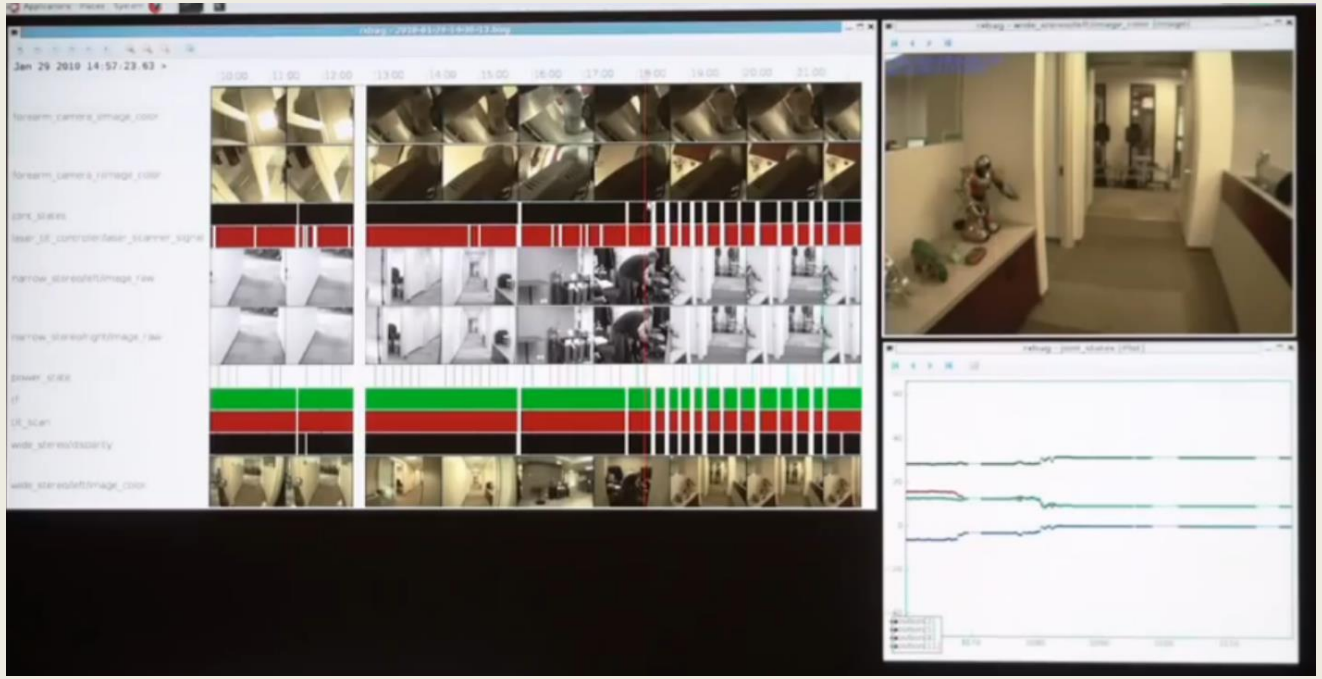
ROS 可视化工具

- 便于机器人仿真与观察
- rqt – 集成图像交互界面
- rviz – 3D 可视化工具



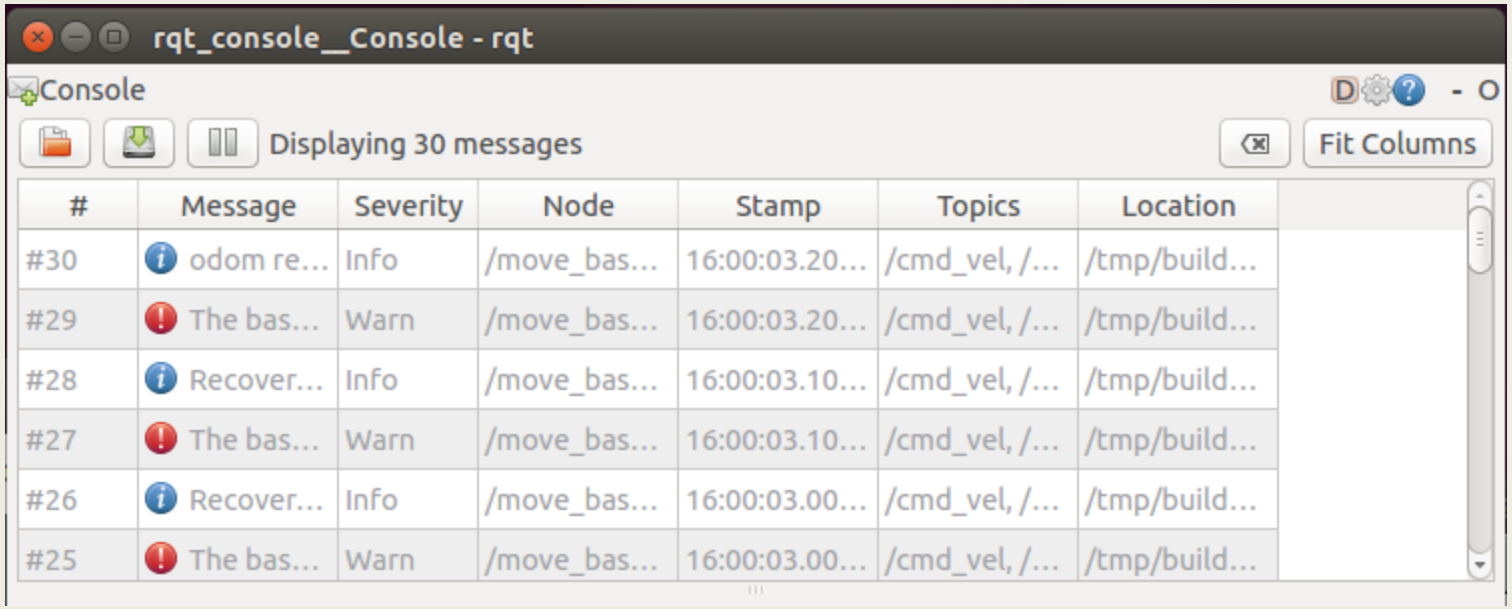
ROS 数据存储/回放

- 思想：使用bag存储topic（例如现实中的传感器数据），以后调用bag的topic数据则不必每次都在现实中运行机器人
- Rosbag:高性能，直接生成topic，避免数据转换



ROS log系统

- 记录软件运行相关信息
- 基于log4cxx
- rosconsole ：提供多个层级的log 信息



运行示例

第一步启动管理器:roscore

```
$ roscore
```

- Roscore将启动:
  - a ROS Master
  - a ROS Parameter Server
  - a rosout logging node

第二步运行功能包：rosrun

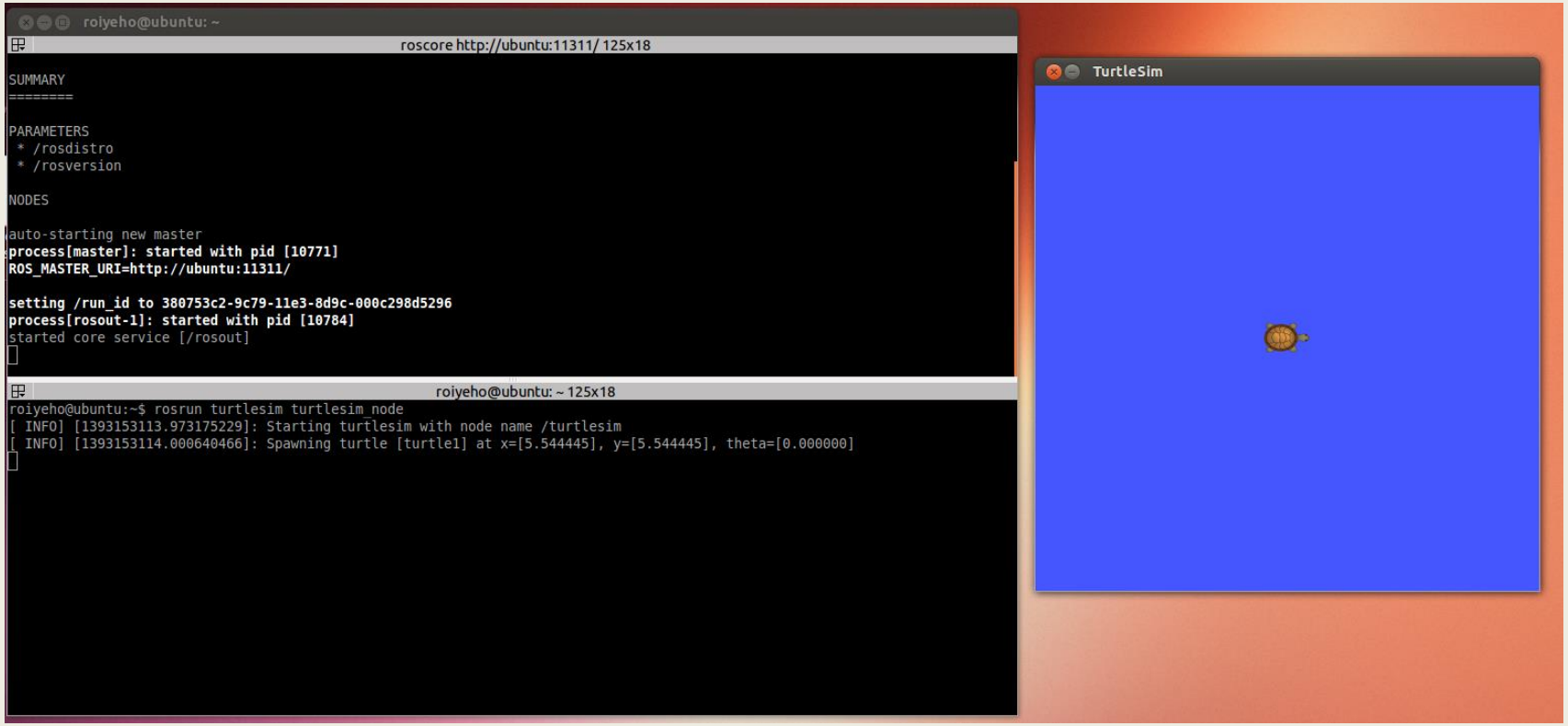
- 语法:

```
$ rosrun <package> <executable>
```

- 例如:

```
$ rosrun turtlesim turtlesim_node
```

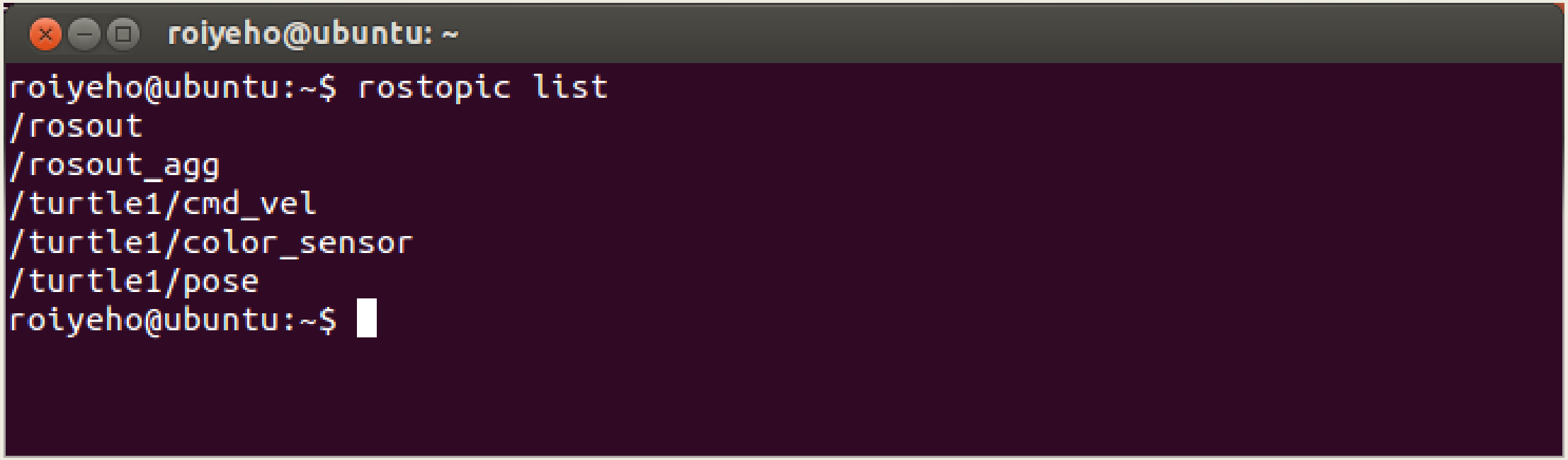
你将会看到：TurtleSim Demo



第三步查看话题：rostopic list

```
$ rostopic list
```

- 显示节点所发布的所有话题



第四步自己手动发送单个话题： **rostopic pub**

- 目的：让海龟以0.2m/s 爬行
- 方法：发布cmd_vel message 给 /turtle1/cmd_vel的x:
  - 完整参数

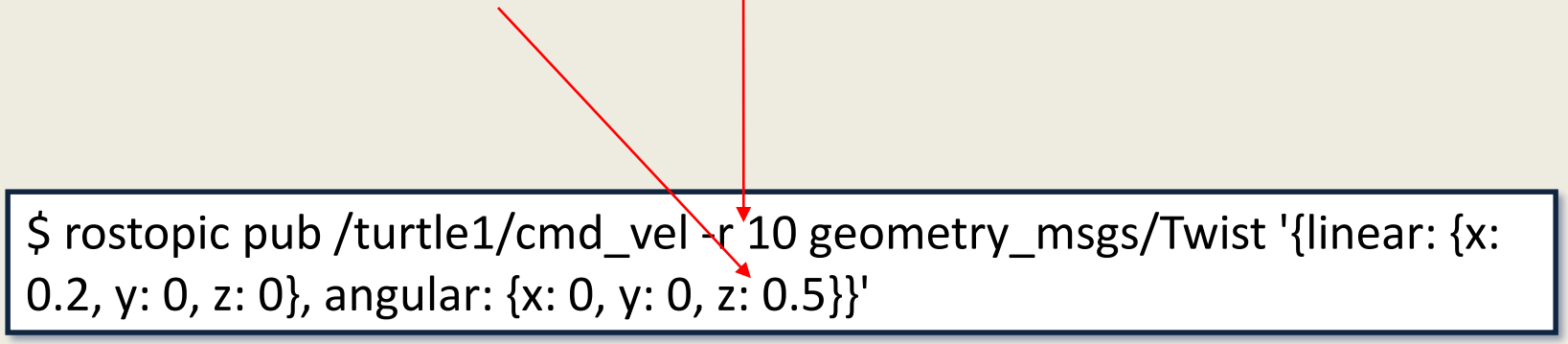
```
$ rostopic pub /turtle1/cmd_vel geometry_msgs/Twist '{linear: {x: 0.2, y: 0, z: 0}, angular: {x: 0, y: 0, z: 0}}'
```

  - 简化参数

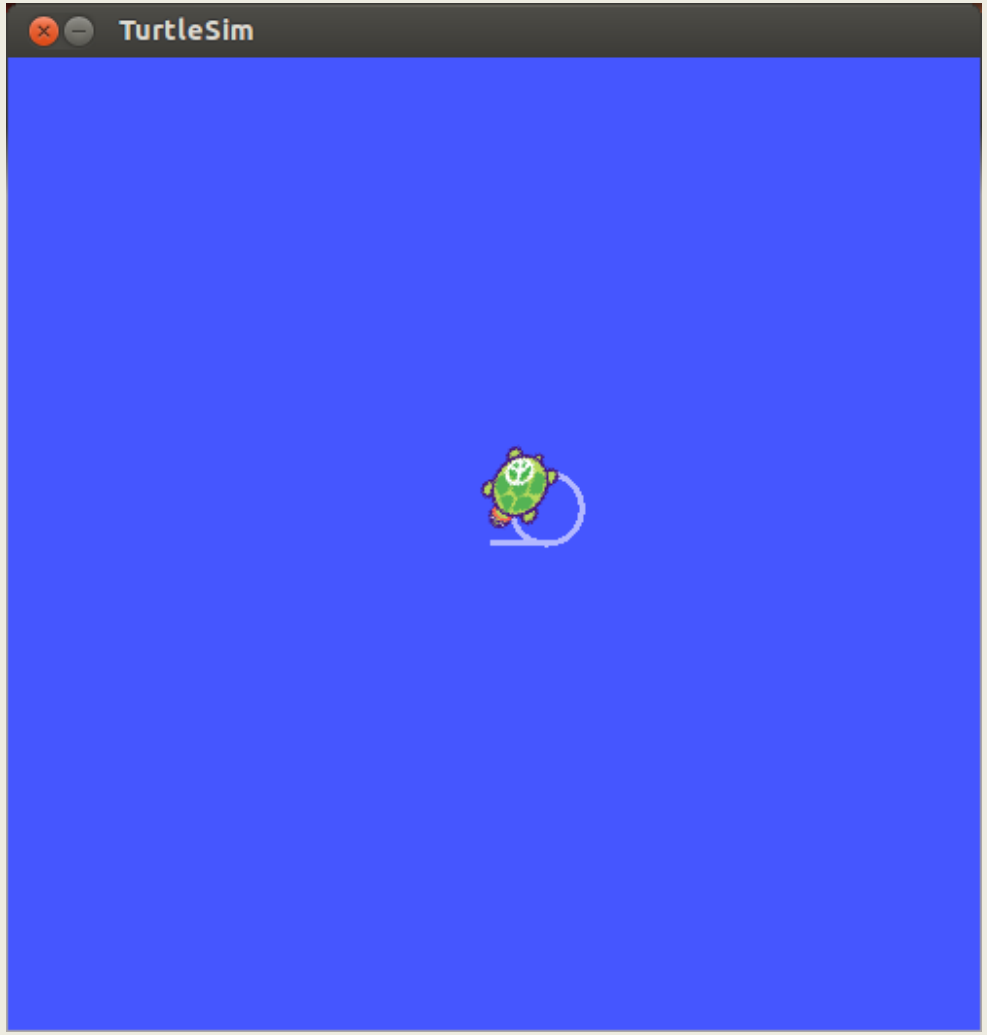
```
$ rostopic pub /turtle1/cmd_vel geometry_msgs/Twist '{linear: {x: 0.2}}'
```

连续自动发送给话题： **rostopic pub**

- 参量：-r 单位Hz (default is 10Hz)
- 例如让海龟连续转圈圈



海龟转起来



第五步记录ROS Topic： rosbag record

- 记录所有topic（ -a ）

```
$ rosbag record –a –O cmd_vel_record
```

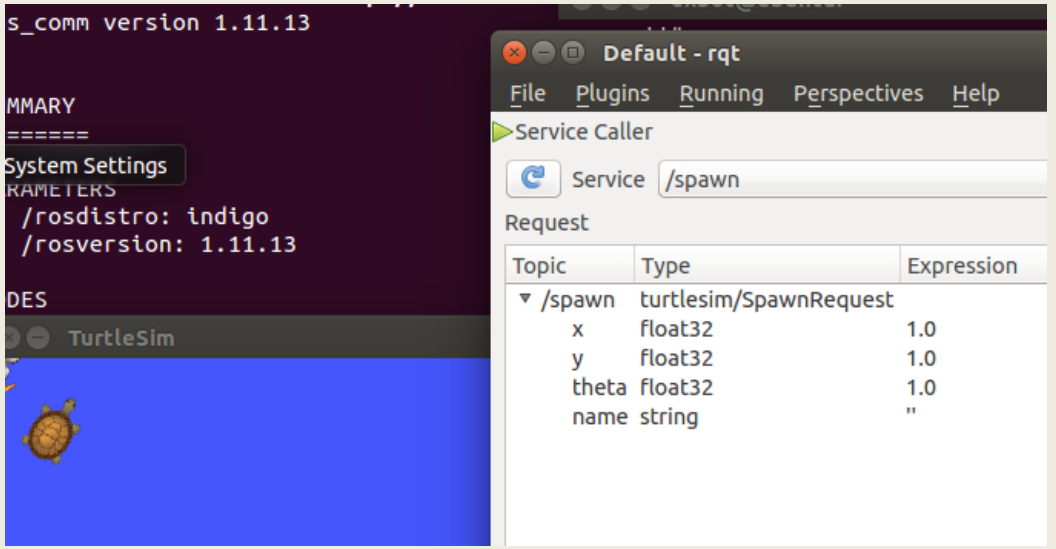
- 回放cmd_vel_record.bag

```
$ rosbag play cmd_vel_record.bag
```

使用ROS service

- 使用ros service 调用一只新的海龟

- $ rosservice call /spawn "x: 1.0
  y: 1.0
  theta: 1.0”



小结

- 第一课我们大致了解了ROS发展的一些历史与发展现状，大致了解了ROS核心的通信层概念，下一课我们将学习ROS的通信及工作原理。

ROS 核心概念补充

- Node：单个可执行程序，ROS中编译运行与管理的基本单元，与其他node进行通信方式为advertise/subscribetopic，或者使用service。
- Message：node间通信使用的强类型数据类型，使用msg文件定义，用相应语言的message-generation工具生成相应代码，提供统一的serialize/deserialize方法。
- Topic : 遵循 subscribe/publish范式的异步通信模式。
- Service : 类似于远程调用的同步通信模式。
- Master : 使用rpc帮助node互相通信的特殊进程，管理node的名字与地址信息，管理topic与service名字与相应通信node的信息，使用rpc传递node之间信息，在ROS通信中非常重要。
- Parameterserver：提供储存/更改/查询node运行参数的服务，运行在master中。具体也是用xml-rpc实现的。
- Name：有关于ROS的计算图模型（ computation-model ），在之后的课程中仔细讲解。
- Bag：在ROS中一个记录/回放ROS消息格式数据的数据格式。 Bag经常用来存放传感器数据，方便算法的离线调试/演示。
- Log：使用log4cxx工具实现的多层次ROS日志管理工具。

ROS 特性补充

ROS has two basic "sides"

- The operating system side, which provides standard opearting system services such as:
  - hardware abstraction
  - low-level device control
  - implementation of commonly used functionality
  - message-passing between processes
  - package management
- A suite of user contributed packages (organized into sets called *stacks*) that implement common robot functionality such as SLAM, planning, perception, simulation etc.

星火计划

NXROBO | Exbot

谢谢！