**第三章 ROS通信架构（一）**

**学习要求：**

掌握node, master, topic, msg的概念以及 等命令。

章节简介：

前面的章节在介绍ROS的时候已经提到过，ROS的一个理念是点到点，ROS是由大量互相连接并不断交换消息的小计算机程序组成，这些消息直接从一个程序传递到另一个程序，在ROS中，我们把这样的小程序叫做节点（nodes），ROS正是由许多个nodes组成的。不同的nodes通过通信进行数据的传递。

ROS的通信架构包括各种数据的处理、进程的运行、消息的传递等等，是ROS中最重要的一个环节，没有通信，系统当中的不同的部分就无法组合在一起，整个系统便没有联系，若同一盘散沙。

本章会重点介绍ROS通信架构中的基本方式和相关概念和命令，并通过实例说明如何进行通信。ROS中的通信方式有四种,主题、服务、参数服务器、动作库。每个通信方式都有自己的特点，本章首先介绍话题通信方式--topic。

**3.1 nodes & master**

3.1.1 nodes

在软件包当中有许多可执行文件，这些可执行文件运行之后就成了一个进程（process），也就是节点（node）。从程序角度来说，node就是一个可执行文件（通常为C++编译生成的可执行文件、Python脚本）被执行，加载到了内存之中；从功能角度来说，通常一个节点负责机器人的某一个单独的功能。由于机器人的功能模块非常复杂，我们往往不会把所有功能都集中到一个节点上，例如，一个节点控制一个激光测距仪，一个节点控制机器人的车轮电机，一个节点执行定位，一个节点执行路径规划，一个节点提供系统的图形视图，等等。

在ROS中使用节点对整个系统有几个好处：当崩溃被单独的节点隔离时，系统的其他部分仍旧能够正常工作。与单片系统相比，代码的复杂性降低了，开发一个复杂系统也更为方便。实现细节也隐藏得很好，因为节点向图的其余部分公开了一个最小的API，而编程语言也很容易被替换。

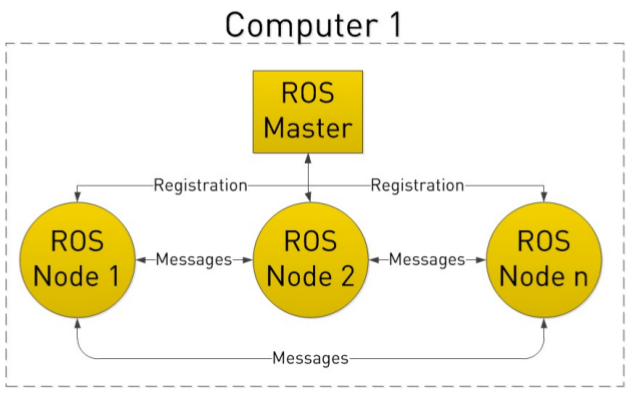
3.1.2 master

由于ROS当中有许多节点，就需要有节点管理器master来管理各个节点。

ROS Master向ROS系统中的其他节点提供命名和注册服务。它跟踪发布者（publisher）和订阅者（subscriber）的主题（topic）和服务（service）。Master的作用是使每个ROS节点能够彼此定位、建立连接，同时还为系统提供参数服务器，管理全局参数。

node首先在master处进行注册，之后master会将该node纳入整个ROS程序中。节点之间的通信也是先由master进行“牵线”，才能两两的进行点对点通信。当ROS程序启动时，第一步首先启动master，由节点管理器处理依次启动node。

Master、node之间和node之间的关系如下图所示：



3.1.3 启动和命令

（1）roscore

roscore是一个向节点提供连接信息，以便节点间可以互相传递消息的服务程序。每个节点都在启动时连接到roscore并注册该节点发布和订阅的消息（这部分也就是本章的重点，后面会讲到，在这里先做大体了解）。当一个新的节点出现时，roscore向它提供与其他发布并订阅相同消息主题的节点建立点对点连接的必要信息。每一个ROS都需要一个roscore，没有它，节点无法相互找到彼此。

当一个ROS节点启动时，该进程（节点即是进程）假设存在一个名为ROS\_MASTER\_URI的环境变量。这个变量包含了一个格式为http://hostname:11311/的字符串，表示roscore有一个运行的实例可以通过主机hostname的11311端口进行访问。（这部分目前只做了解，后面的章节会讲到。）

已知roscore在网络中的位置后，节点在roscore中完成注册并请求roscore通过命名找到其他节点以及数据流。每个节点告诉roscore该节点提供和接收何种消息，roscore则提供了相关消息的发布者和接受者的地址。（再次提醒这是本章通讯的主要内容，本章就是讲如何以这种方式进行节点间的通讯。）

roscore同时也提供了一个被ROS节点广泛用于程序配置的参数服务器。参数服务器允许节点存储和获取任意数据结构，例如机器人的描述，算法的参数等等。有一个简单的命令行工具用于与参数服务器交互：rosparam。（会在通讯章节的第二部分讲到，现在提到只是让同学们有一个大致的了解，后来学到这个东西的时候知道是和roscore有关就行。到时候再返回来看这部分。）

启动master之后，节点管理器就开始按照系统的安排协调进行启动具体的节点。节点就是一个进程，只不过在ROS中它被赋予了专用的名字里——node。在第二章我们介绍了ROS的文件系统，我们知道一个package中存放着可执行文件，可执行文件是静态的，当系统执行这些可执行文件，将这些文件加载到内存中，它就成为了动态的node。具体启动node的语句是:

$ rosrun pkg\_name node\_name

下面就具体介绍一下rosrun命令。

（2）rosrun

ROS节点通常是可执行程序，用户可以在文件系统中通过cd命令手动切换到各个位置并启动节点。但是在一个很大的文件系统中，因为节点可能放在层级结构的文件系统中很深的位置，进入长的路径中会比较麻烦。因此ROS提供了一个命令行程序rosrun来寻找程序包中的可执行程序并向这个程序传递任何参数。

具体用法如下：

$ rosrun pkg\_name node\_name

rosrun允许你使用包名直接运行一个包内的节点而不需要知道这个包的路径。rosrun寻找package下名为node\_name的可执行程序，将可选参数传入。

到目前为止，ROS一共发布了11个版本，每一个版本都伴随着一个小乌龟吉祥物。下面我们就可以通过小乌龟例程开始认识ROS。在turtlesim功能包中，核心是节点turtlesim\_node，功能包提供一个可视化的乌龟仿真器，可以实现很多ROS基础功能的测试。

每一个ROS功能包都是一个独立的功能，其中可能包含一个或多个节点，这些功能对外使用topic, service, parameter作为接口（通讯架构章节的重点）。其他开发者在使用这个功能包时，可以不用关注内部的代码实现，只需要知道这些接口的类型和作用，就可以集成到自己的系统中。

（3）rosnode

rosnode是一个命令行工具，用于显示关于ROS节点的调试信息，包括发布、订阅和连接。它还包含一个用于检索节点信息的实验库。本库仅供内部使用。

常用的命令有：

rosnode list 列出当前运行的node信息

rosnode info node\_name 显示出node的详细信息

rosnode kill node\_name 结束某个或多个node

kill 进入交互模式。这使您能够从编号列表中选择要杀 死的节点，这对于杀死匿名节点非常有用。

kill -a/--all 结束所有节点

rosnode ping nodename 测试连接节点

ping -a/--all 测试连接所有节点

ping -c COUNT 测试COUNT次（--all和-c不能同时使用）

rosnode machine 列出在特定机器或列表机器上运行的节点

rosnode cleanup 清除不可到达节点的注册信息

rosnode cleanup是一种临时解决方案，在正常操作中不鼓励使用它。它的好处是美观，也有潜在的未注册的功能节点的缺点。

以上命令中常用的为前三个，在开发调试时经常会需要查看当前node以及node信息，所以请记住这些常用命令。如果你想不起来，也可以通过rosnode help来查看rosnode命令的用法。

在后面的练习当中我们会使用主要的一些命令，使大家直观地了解这些命令的作用。

**3.2 launch文件**

机器人是一个系统工程,通常一个机器人运行操作时要开启很多个node,对于一个复杂的机器人的启动操作应该怎么做呢?当然,我们并不需要每个节点依次进行rosrun,ROS为我们提供了一个命令能一次性启动master和多个node。该命令是:

$ roslaunch pkg\_name file\_name.launch

roslaunch命令首先会自动进行检测系统的roscore有没有运行，也即是确认节点管理器是否在运行状态中，如果master没有启动，那么roslaunch就会首先启动master，然后再按照launch的规则执行。launch文件是描述一组节点以及它们话题重映射和参数的XML文件，里已经配置好了启动的规则。所以roslaunch就像是一个启动工具，能够一次性把多个节点按照我们预先的配置启动起来，减少我们在终端中一条条输入指令的麻烦。

launch文件同样也遵循着xml格式规范,是一种标签文本,它的格式包括以下标签:

<launch> <!--根标签-->

<node> <!--需要启动的node及其参数-->

<include> <!--包含其他launch-->

<machine> <!--指定运行的机器-->

<env-loader> <!--设置环境变量-->

<param> <!--定义参数到参数服务器-->

<rosparam> <!--启动yaml文件参数到参数服务器-->

<arg> <!--定义变量-->

<remap> <!--设定参数映射-->

<group> <!--设定命名空间-->

</launch> <!--根标签-->

**3.3 topic**

3.3.1 topic介绍

节点之间最重要的通信机制就是基于发布（publish）/订阅（subscrib）模型的消息（message）通信。每一个消息都是一种严格的数据结构，支持标准数据类型（整型、浮点型、布尔型等），也支持嵌套结构和数组，还可以根据需求由开发者自主定义。

topic在ROS中使用最为频繁。在ROS中有两个节点，一个是发布者（publisher），一个是订阅者（Subscriber），两个节点分别发布、订阅同一个topic，启动顺序没有强制要求，此处假设publisher先启动，可大致分成以下七个步骤来分析建立通信的详细过程。

1. publisher注册

publisher启动，向ROS Master注册发布者的信息，包含所发布消息的话题名，ROS Master会将节点的注册信息加入到注册列表中。

1. subscriber注册

subscriber启动，同样向ROS Master注册订阅者的消息，包含需要订阅的话题名。

1. ROS Master进行信息匹配

Master根据subscriber的订阅信息从注册列表中查找，如果没有找到匹配的发布者，则等待发布者的加入；如果找到匹配的发布者的信息，则向subscriber发送publisher的地址信息。

1. subscriber发送连接请求

subscriber接收到Master发回的subscriber的地址信息，尝试向publisher发送连接请求，传输订阅的话题名、消息类型以及通信协议。

1. publisher确认连接请求

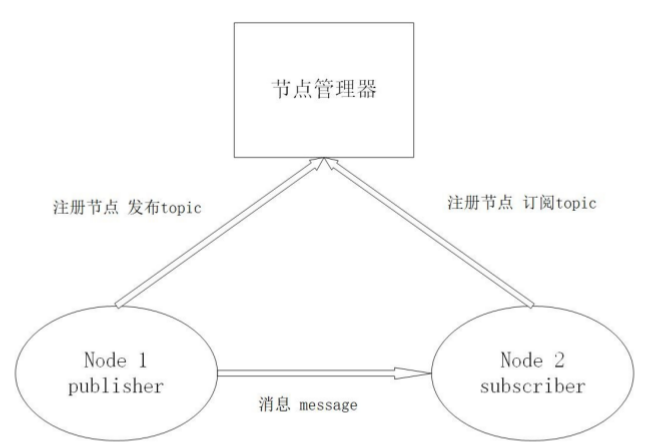
publisher接收到连接请求后，继续向subscriber确认连接信息，其中包含了自身的TCP地址信息。

1. subscriber尝试与publisher建立

publisher接收到确认信息后，使用TCP尝试与subscriber建立网络连接。

1. publisher向subscriber发送数据

成功建立连接后，publisher开始向subscriber发送话题消息数据。



subscriber接收到消息会进行处理，一般这个过程叫做回调（callback）。就是提前定义好了一个处理函数(写在代码中),当有消息来就会触发这个处理函数,函数会对消息进行处理。

总结三点:

1. topic通信方式是异步的,发送时调用publish()方法,发送完成立即返回,不用等待反馈。

2. subscriber通过回调函数的方式来处理消息。

3. topic可以同时有多个subscribers,也可以同时有多个publishers。

3.3.2 rostopic命令（一）

rostopic

rostopic list 列出当前所有的topic

rostopic info topic\_name 显示某个topic的属性信息

rostopic echo topic\_name 显示某个topic的内容

rostopic pub topic\_name ... 向某个topic发布内容

rostopic bw topic\_name 查看某个topic的带宽

rostopic hz topic\_name 查看某个topic的频率

rostopic find topic\_type 查找某个类型的topic

rostopic type topic\_name 查看某个topic的类型(msg)

rostopic list能够列出当前所有订阅和发布的话题。

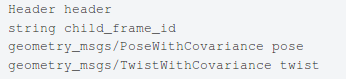
下面我们先学习了解消息msg，然后再回到topic上来。

3.3.3 msg

消息（msg）文件就是一个描述ROS中所使用消息类型的简单文本。它们会被用来生成不同语言的源代码。msg文件存放在package的msg目录下，msg文件其实就是每行声明一个数据类型和变量名，可以使用的数据类型如下：

bool、int8、int16、int32、int64(以及uint)、float、float64、string、time、duration、header、可变长数组array[]、固定长度数组array[C]等等。ROS中还有一个特殊的数据类型：Header，含有时间戳和坐标系信息。在msg文件的第一行经常可以看到Header header的声明。

下面给出一个msg文件的案例：



（1）创建一个msg

（2）rosmsg命令

rosmsg list #列出系统上所有的msg

rosmsg show msg\_name #显示某个msg的内容

我们可以通过rosmsg list命令 查看系统上所有的消息。

3.3.4 rostopic命令（二）

（1）rostopic type

用来查看所发布话题的消息类型。

用法：rostopic type [topic]

（2）rostopic pub

可以把数据发布到当前某个正在广播的话题上。

用法：

rostopic pub [topic] [msg\_type] [args]

（3）rostopic hz

用来查看数据发布的频率。

用法：

rostopic hz [topic]