《ROS机器人开发技术》

实训讲稿

|  |  |  |
| --- | --- | --- |
| 课程名称 | ： | ROS机器人开发技术 |
| 教师姓名 | ： | 刘梦倩 |
| 提交时间 | ： | 2018年7月15日 |

中国大学MOOC制

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 课程  名称 | 《ROS机器人开发技术》 | | 章 | | 第三章 | | | 课程类型 |
| 节 | | 第一节 | | | 授课( )  实训(√ ) |
| 名称 | ROS通讯架构（一） | | | | | | |
| 教师 |  | 课时 | | | | 1课时 | | |
| 参考  资料 | 1. ROS wiki <http://wiki.ros.org/> 2. 《ROS机器人编程实践》 3. 《ROS机器人开发实践》 | | | | | | | |
| 教学  目的  要求 | 掌握：ROS的通信架构之node,master,launch | | | | | | | |
| 教学  重点  难点 |  | | | PPT页面 | | | 时间分配 | |
| 教学重点 | | | | | | | |
| 1. roscore | | | 7~8页 | | | 15分钟 | |
| 1. rosnode | | | 11~19页 | | | 15分钟 | |
| 1. launch文件 | | | 20~25页 | | | 15分钟 | |
| 教学难点 | | | | | | | |
| 1. roscore | | | 7~8页 | | | 15分钟 | |
| 1. rosnode | | | 11~19页 | | | 15分钟 | |
|  | 1. launch文件 | | | 20~25页 | | | 15分钟 | |

|  |  |  |  |
| --- | --- | --- | --- |
| 教学方法 | | | |
| 本授课以课堂讲授为主，与课堂实时练习方式相结合 | | | |
| 教学内容 | | 操作演示 | 知识点 |
| **PPT第1页:**  现在开始，我们进入topic通讯的实训部分。  本节课要讲的是第一部分。 | |  |  |
| **PPT第2页:**  下面我们先来复习一下上节课当中所讲的，在本部分实训的中会用得上的知识点。  首先就是roscore的概念，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。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。  roscore就相当于一个类似于平台的地方，卖家和买家都在上面注册并且发布信息，当卖家卖的东西买家正好需要的时候，就在平台上面联系，但是最后的交钱和交货的部分和平台无关。 | |  |  |
| **PPT第3页:**  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。  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。 | |  |  |
| **PPT第4页:**  下面我们复习一下有关于rosnode命令相关的知识点。其中主要是不同的命令都有什么作用：……  **PPT第5页:**  …… | |  | roscore |
| **PPT第6页:**  复习的第三个知识点是launch文件。  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。  通过roslaunch命令来启动。 | |  |  |
| **PPT第7页:**  本节课我们会完成第一个任务：如何同时打开两个turtlesim程序，使乌龟做相同的运动。  首先进行第一个子任务，很简单，运行ROS的节点管理器——ROS Master。  就只需要一个很简单的小命令：roscore。 | |  |  |
| **PPT第8页:**  每一次我们运行ROS的节点前，都需要把master启动起来，这样才能够让节点启动和注册。  断点1：操作演示**示例代码1**  可以看到显示如图所示。  要记住无论何时都需要先启动roscore才能做ros的其他操作。 | | 操作演示：示例代码1 1.演示目的：roscore基本使用  2. 打开终端。  3.输入roscore    **结论：ROS Master启动。** | roscore |
| **PPT第9页:**  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。图中这些就是其中一些版本的图标。  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。  下面我们通过这个简单的小乌龟例程逐渐认识节点间的topic通讯方式。  **PPT第10页:**  如果没有安装turtlesim功能包，可以通过以下命令安装：  $ sudo apt-get install ros-kinetic-turtlesim  因为老师的电脑里面已经安装完了，在此就不再演示。  断点1：操作演示**示例代码1** | | 操作演示：示例代码1 1.演示目的：安装turtlesim功能包  2. 打开新终端。  3.输入命令：  $ sudo apt-get install ros-kinetic-turtlesim  **结论：安装完毕。** |  |
| **PPT第11页:**  开始第二个子任务：如何 打开turtlesim\_node节点？  可以回忆起rosnode这个命令。  现在我们可以运行turtlesim包中的turtlesim\_node。  之前已经运行了roscore，不要关闭，打开一个新的终端，输入：  $ rosrun turtlesim turtlesim\_node  断点1：操作演示**示例代码1**  **PPT第12页:**  你会看到这样一个窗口：    注意：看到的乌龟可能每个人都不同，因为有不同版本的turtle，可以关闭然后多运行几次看看每次会随机到哪只乌龟。 | | 操作演示：示例代码1 1.演示目的：运行turtlesim包中的turtlesim\_node  2. 打开新终端。  3.输入命令：  $ rosrun turtlesim turtlesim\_node    **结论：安装完毕。** | rosrun |
| **PPT第13页:**  我们已经学习了rosnode的使用，那么我们现在来做一些练习。  开始子任务3：重新配置节点名称。  这个任务比较难，我们先一点点联系rosnode这个命令的用法。  在练习前我们要记住如果忘了具体的命令，可以通过help来查看。  断点1：操作演示**示例代码1** | | 操作演示：示例代码1  1. 演示目的：学习rosnode的使用   2. 打开新终端。  3.输入命令：  $ rosnode help    **结论：看到命令。** | rosnode |
| **PPT第14页:**  下面我们来依次演示这些命令的作用。  断点1：操作演示**示例代码1**  rosout是用来收集和记录节点调试输出信息的，roscore命令后和Master一起启动，因此总是运行的**。** | | 操作演示：示例代码1 1.演示目的：学习rosnode的使用  2输入命令：  $ rosnode list    **结论：看到list。** | rosnode |
| **PPT第15页:**  ROS可以通过命令行重新配置名称，关闭turtlesim窗口停止运行节点（或者回到rosrun turtlesim终端并使用Ctrl+c）。现在我们重新运行它，但是这一次使用remapping argument改变节点名称：  看一下演示：  断点1：操作演示**示例代码1** | | 操作演示：示例代码1 1.演示目的：使用remapping argument改变节点名称  2输入命令：  $ rosrun turtlesim turtlesim\_node \_\_name:=my\_turtle  $ rosnode list    **结论：名称改变。** | rosnode |
| **PPT第16页:**  但是也有人会出现这样的显示：    这是因为使用了Ctrl+c停止节点而非关闭重开窗口，可以通过rosnode cleanup清除无用的节点。  断点1：操作演示1 | | 操作演示：示例代码1 1.演示目的：使用remapping argument改变节点名称2  2输入命令：  $ rosnode cleanup    输入y继续。  完成后重新输入：  $ rosnode list  会看到正常的结果，无效的节点被清除了。    **结论：名称改变。** | rosnode |
| **PPT第17页:**  使用ping指令来测试这个节点：  演示一下：  断点1：操作演示1  得到这样的输出结果，在最后使用Ctrl+c结束ping。 | | 操作演示：示例代码1 1.演示目的：使用ping指令来测试这个节点2输入命令：  $ rosnode ping my\_turtle    **结论：测试成功。** | rosnode |
| **PPT第18页:**  也可以使用设置次数的方式得到有限次数的结果：  比如设置只ping四次。  该如何操作呢？  断点：操作演示1 | | 操作演示：示例代码1  1. 演示目的：使用指定次数的ping指令   2输入命令：  $ rosnode ping -c 4 my\_turtle    **结论：测试成功。** | rosnode |
| **PPT第19页:**  那么当一个节点我们不再需要了，我们就可以用命令关闭它：  断点：操作演示1 | | 操作演示：示例代码1 1.演示目的：关闭节点  2输入命令：  $ rosnode kill /turtlesimrosnode kill /turtlesim    我们再输入list命令：  $ rosnode list    **结论：节点关闭。** | rosnode |
| **PPT第20页:**  关于rosnode的命令我们就演示到这里。  下面开始进行子任务4：  建立launch文件。  先在catkin\_ws工作空间中建立一个package：beginner（这些都是前面讲过的内容，我们顺便复习一下）：  断点：操作演示1 | | 操作演示：示例代码1 1.演示目的：建立一个package  2输入命令：  $ cd ~/catkin\_ws/src #改变目录  $ catkin\_create\_pkg beginner std\_msgs rospy roscpp #创建包，后面三个是依赖  $ cd ~/catkin\_ws #改变目录  $ catkin\_make #开始编译  然后开始创建launch文件：  先转到beginner package目录下：  输入：  $ roscd beginner  然后创建一个launch文件夹：  $ mkdir launch  $ cd launch  **结论：建立一半了。** | launch |
| **PPT第21页:**  然后在这个目录下创建一个名为turtlemimic.launch的launch文件并复制以下内容到该文件里面  断点：操作演示1  下面我们会逐行解释这个文件。 | | 操作演示：示例代码1 1.演示目的：建立一个launch文件  2复制以下内容到该文件里面：  <launch>  <group ns="turtlesim1">  <node pkg="turtlesim" name="sim" type="turtlesim\_node"/>  </group>  <group ns="turtlesim2">  <node pkg="turtlesim" name="sim" type="turtlesim\_node"/>  </group>  <node pkg="turtlesim" name="mimic" type="mimic">  <remap from="input" to="turtlesim1/turtle1"/>  <remap from="output" to="turtlesim2/turtle1"/>  </node>  </launch>  **结论：建立成功。** | launch |
| PPT第22页: ……  #这部分创建了两个节点分组，并以namespace标签来区分，其中一个命名为trutlesim1，一个命名为trutlesim2，两个组里面都使用相同的turtlesim节点并命名为’sim’，这样可以使我们同时启动两个turtlesim模拟器而不产生命名冲突。每个<node> 标签中包含了声明ROS图中节点的名字属性，该节点所在的包名，以及节点的类型，也就是可执行程序的文件名。  其中pkg定义节点所在的功能包的名称，type定义节点的可执行文件名称，name定义节点运行的名称，这三个是最常用的属性。 | |  | launch |
| PPT第23页: #这里我们启动模仿节点，并将所有topic的输入输出分别重命名为trutlesim1和trutlesim2，这样就会使trutlesim2模仿trutlesim1。我们后面会看到这样设定的作用。  …… | |  | launch |
| PPT第24页: 下面就是最后一个子任务：启动launch文件。大家回忆一下这里要用到什么命令？  没错，就是roslaunch。  下面我们通过roslaunch命令来启动launch文件：  断点：操作演示1  现在将会有两个turtlesim被启动，然后我们在一个新终端中使用rostopic命令发送速度设定消息（这部分还没设计到，现在先照着做，后面的章节会讲到）：  断点：操作演示2 PPT第25页: 会看到两个turtlesim同时开始移动，虽然发布命令只是给了trutlesim1 | | 操作演示：示例代码1 1.演示目的：启动launch文件  2输入命令：  $ roslaunch beginner turtlemimic.launch  **结论：两个程序被启动。** 示例代码2 1.演示目的：发送速度设定消息  2输入命令：  $ rostopic pub /turtlesim1/turtle1/cmd\_vel geometry\_msgs/Twist -r 1 -- ‘[2.0, 0.0, 0.0]’ ‘[0.0, 0.0, -1.8]’    **结论：乌龟开始运动。** | roslaunch |
| PPT第26页: 我们也可以通过rqt\_graph来更好地理解在launch文件中所做的事情：  断点：操作演示1  这就是ROS图。通过ROS图可以直观形象地看到各个节点之间的关系。 | | 操作演示：示例代码1 1.演示目的：使用rqt\_graph  2输入命令：  $ rqt\_graph  rosgraph  **结论：显示ros图。** | roslaunch |
| 知识点框图 | | | |
| 结构图 |  | | |