智能无人机技术设计实践

--ROS通信指导书

于超

联系方式: yc19@mails.tsinghua.edu.cn

手机:15652601306

时间：2019.10.12

目 录

ROS通信架构

◆ **实验1.0（准备）**

① 下载https://github.com/zoeyuchao/ros_base_code，将其中param_demo、pyservice_demo、pytopic_demo三个Package复制到catkin/src/路径下，并在catkin路径下运行catkin_make命令，进行编译。

```
>> cd ~/catkin_ws
>> catkin_make
```

注意:source命令，编译完成后必须刷新一下工作空间的环境，否则可能找不到工作空间。许多时候我们为了打开终端就能够运行工作空间中编译好的ROS程序，我们习惯把:
source ~/tutorial_ws/devel/setup.bash 命令追加到~/.bashrc文件中(tutorial_ws替换为你的工作空间名称)，这样每次打开终端，系统就会刷新工作空间环境。

② 运行下面的命令(之前运行过就不需要了):

```
>> echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

③ 关闭终端，打开一个新的终端，运行下面的命令：

```
>> rospack find pytopic_demo
```

如果出现pytopic_demo的路径，说明source添加成功。

实验 1.1 Node和Master的启动

◆ **实验1.1：Node和Master**

① 在Linux环境下启动ROS，先启动ros master。

```
>> roscore
```

② 新建两个终端，分别运行pytopic_demo中的两个节点（先给pytalker和pylistener赋予可执行权限 sudo chmod +x ）。

```
>> rosrun pytopic_demo pytalker
>> rosrun pytopic_demo pylistener
```

③ 可以看到运行pytalker和pylistener的终端中有GPS信息显示。

④ 新建一个终端，运行以下命令查看正在运行的节点。

```
>> rosnode list
```

⑤ 在前三个终端中分别输入CTRL+C，终止ROS进程。

实验 1.1 Node和Master的启动

- ✓ rosrun 和 rosnode 命令

① rosrun命令的详细用法如下：

```
>> rosrun [--prefix cmd] [--debug] pkg_name node_name [ARG]
```

rosrun将会寻找PACKAGE下的名为EXECUTABLE的可执行程序，将可选参数ARGS传入。

② rosnode命令的详细作用列表如下：（前三个命令常用）

rosnode指令	作用
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node
rosnode ping	测试连接节点
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

实验1.2 Launch文件和roslaunch

◆ **实验1.2：Launch文件和roslaunch**

① 阅读pytopic_demo/launch/pytopic_demo.launch，执行如下命令：

```
>> roslaunch pytopic_demo pytopic_demo.launch
```

② roslaunch命令通过预设好的launch文件来启动ROS系统并启动所有launch文件里运行的节点。

③ 在终端中输入CTRL+C终止进程。

实验1.3 Topic通信模型

◆ **实验1.3：Topic通信模型**

① 阅读pytopic_demo/scripts/pytalker.py 和 pylistener.py代码及注释，理解topic通信模式代码的写法。

② 阅读pytopic_demo/msg/gps.msg，理解msg文件写法。

③ 执行如下命令启动topic模型。

```
>> roslaunch pytopic_demo pytopic_demo.launch
```

④ 新建一个终端，执行如下命令观察现有的topic列表。

```
>> rostopic list
```

⑤ 执行如下命令观察正在传输的topic内容。

```
>> rostopic echo gps_info
```

⑥ 在第一个终端中输入CTRL+C终止进程。

⑦ 阅读pytopic_demo/CMakeLists.txt的51行到54行，观察msg在CMakeLists里的声明方法。

实验1.3 Topic通信模型

- ✓ Topic相关的操作命令

① rostopic指令

rostopic是对topic管理的命令行工具，用法如下表：

rostopic指令	作用
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)

可以通过rostopic help 命令查看具体用法。

实验1.4：Service通信模型

◆实验1.4：Service通信模型

① 阅读pyservice_demo/scripts/pyclient.py 和 pyserver.py代码及注释，理解service通信模式代码的写法。

② 阅读pyservice_demo/srv/Greeting.srv，理解srv文件写法。

③ 执行如下命令启动ROS master。

```
>> roscore
```

④ 新建两个终端，分别执行如下命令启动Service通信模型。

```
>> rosrun pyservice_demo pyserver.py
>> rosrun pyservice_demo pyclient.py
```

⑤ 在两个终端观察Service的信息传递。

⑥ 在前三个终端中输入CTRL+C终止进程。

⑦ 阅读pyservice_demo/CMakeLists.txt的58行到61行，观察srv在CMakeLists里的声明方法。

实验1.4： Service通信模型

- ✓ Service相关的操作命令

① rosservice指令

rosservice是对service管理的命令行工具，用法如下表：

rosservice指令	作用
rosservice list	列出当前所有的topic
rosservice 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)

可以通过rostopic help 命令查看具体用法。

实验1.5：Parameter Server通信模型

◆实验1.5：Parameter Server通信模型

① 阅读pyparam_demo/scripts/pyparamtalker.py 和 pyparamlistener.py代码及注释，观察其中使用到Parameter Server的部分。

② 阅读pyparam_demo/launch/pyparam_demo.launch，观察launch文件中设置Parameter Server中参数的方法。

③ 执行如下命令启动parameter server 模型。

```
>> roslaunch pyparam_demo pyparam_demo.launch
```

④ 新建一个终端，获取参数服务器参数列表。

```
>> rosparam list
```

⑤ 读取参数服务器中参数值。

```
>> rosparam get x
>> rosparam get y
>> rosparam get arrays
>> rosparam get movable
```

⑥ 改变参数服务器中参数值并观察第一个终端中信息的变化。

```
>> rosparam set x 10
```

⑦ 输入CTRL+C终止进程。

实验1.5：Parameter Server通信模型

✓ Parameter Server相关的操作命令

① rosparam指令

rosparam是对parameter server管理的命令行工具，用法如下表：

rosparam指令	作用
rosparam set param_key param_value	设置参数
rosparam get param_key	显示参数
rosparam load file_name	从文件加载参数
rosparam dump file_name	保存参数到文件
rosparam delete	删除参数
rosparam list	列出参数名称

可以通过rosparam help 命令查看具体用法。

附录

✓ 附rospy的常用API

- Node相关

返回值	方法	作用
	rospy.init_node(name, argv=None,anonymous=False)	注册和初始化node
MasterProxy	rospy.get_master()	获取master的句柄
bool	rospy.is_shutdown()	节点是否关闭
	rospy.on_shutdown(fn)	在节点关闭时调用fn函数
str	get_node_uri()	返回节点的URI
str	get_name()	返回本节点的全名
str	get_namespace()	返回本节点的名字空间

附录

- Topic相关

返回值	方法	作用
[[str, str]]	get_published_topics()	返回正在被发布的所有topic名称和类型
Message	wait_for_message(topic,topic_type, time_out=None)	等待某个topic的message
	spin()	触发topic或service的回调/处理函数，会阻塞直到关闭节点。
	rospy.on_shutdown(fn)	在节点关闭时调用fn函数

附录

- Topic相关publisher类

返回值	方法	作用
	init(self, name, data_class, queue_size=None)	构造函数
	publish(self, msg)	发布消息
str	unregister(self)	停止发布

- Topic相关subscriber类

返回值	方法	作用
	init_(self, name, data_class, call_back=None, queue_size=None)	构造函数
	unregister(self, msg)	停止订阅

附录

- Service相关

返回值	方法	作用
	wait_for_service(service, timeout=None)	阻塞直到服务可用

- Server类

返回值	方法	作用
	init(self, name, service_class,handler)	构造函数，handler为处理函数，service_class为srv类型
	shutdown(self)	关闭服务的server

- Client类

返回值	方法	作用
	init(self, name, service_class)	构造函数，创建Client
	call(self, args, *kwds)	发起请求
	close(self)	关闭服务的client

附录

- Param相关

返回值	方法	作用
XmlRpcLegalValue	get_param(param_name,default=_unspecified)	获取参数的值
[str]	get_param_names()	获取参数的名称
	set_param(param_name,param_value)	设置参数的值
	delete_param(param_name)	删除参数
bool	has_param(param_name)	参数是否存在于参数服务器上
str	search_param()	搜索参数

附录

- 时钟相关

返回值	方法	作用
Time	get_rostime()	获取当前时刻的Time对象
float	get_time()	返回当前时间，单位秒
	sleep(duration)	执行挂起

- Time类

返回值	方法	作用
	init(self, secs=0, nsecs=0)	构造函数
Time	now()	静态方法返回当前时刻的Time对象

- Duration类

返回值	方法	作用
	init(self, secs=0, nsecs=0)	构造函数

谢谢！

答疑地点：双清大厦2号楼502