9 ROS编程

方宝富

fangbf@hfut.edu.cn

提纲

- ROS编程简介
- roscpp简介
- roscpp 举例demo
- rospy 简介
- rospy 举例demo
- 机器人运动控制实例

9.1　ROS编程简介

Client Library

Client Library 是ROS官方提供的集成编程库（类似API），用户可以基于Client Library 进行ROS编程（建立node、发布消息、调用服务等操作），而不需要关心程序最底层如何实现。

Client Library 编程库的不同语言版本：

roscpp--常用

rospy –常用

roslisp

…

提纲

- ROS编程简介
- roscpp简介
- roscpp 举例demo
- rospy 简介
- rospy 举例demo
- 机器人运动控制实例

9.2　roscpp简介

roscpp 是C++版本的编程接口库，主要包含几个部分：

（函数）ros::init() : 解析传入的ROS参数并为本node命名，使用roscpp第一步需要用到的函数

（类）ros::NodeHandle 和topic、service、param等交互的公共接口

（命名空间）ros::master : 包含从master查询信息的函数

（命名空间） ros::this_node：包含查询这个进程(node)的函数

（命名空间） ros::service：包含查询服务的函数

（命名空间） ros::param：包含查询参数服务器的函数，而不需要用到NodeHandle

（命名空间） ros::names：包含处理ROS图资源名称的函数

9.2 roscpp简介

ros::Nodehandle nh;

ros::Publisher pub =nh.advertise(…);

pub.publish(msg);

Class ros::NodeHandle，其主要成员函数如下：

//创建话题的publisher

ros::Publisher advertise(const string &topic, uint32_t queue_size); //**ros::Publisher也是一个类**

//创建话题的subscriber

ros::Subscriber subscribe(const string &topic, uint32_t queue_size, void(*)(M));

//创建服务的server

ros::ServiceServer advertiseService(const string &service, bool(*srv_func)(Mreq &, Mres &));

//创建服务的client

ros::ServiceClient serviceClient(const string &service_name, bool persistent=false);

//查询某个参数的值

bool getParam(const string &key, void &val);

//给参数赋值

bool setparam(const string &key, void val);

9.2 roscpp简介

Namespace ros::master 该空间的主要函数有：

ros::master::check();

bool **check();** //检查master是否启动

const string& **getHost ();** //返回master所处的hostname

bool **getNodes(V_string &nodes);** //从master返回已知的node名称列表

bool **getTopics(V_TopicInfo &topics);** //返回所有正在被发布的topic列表

bool **getURI();** //返回到master的URI地址，如http://host:port/

uint32_t **getPort();** //返回master运行在的端口

9.2 roscpp简介

Namespace ros::this_node 该空间的主要函数有：

void **getAdvertisedTopics(V_string &topics);** //返回本node发布的topic

const string &**getName ();** //返回当前node的名称

const string &**getNamespace();** //返回当前node的命名空间

void **getSubscribedTopics (V_string &topics);** //返回当前node订阅的topic

9.2　roscpp简介

Namespace ros::service 该空间的主要函数有：

//调用一个RPC服务

bool **call**(const string &service, Service &service);

//创建一个服务的client

ServiceClient **createClient(**const string& service_name, bool persistent=false, const M_string &header_values=M_string());

//确认服务可调用

bool **exists**(const string &service_name, bool print_failure_reason);

//等待一个服务，直到它可调用

bool **waitForService**(const string &service_name, int32_t timeout);

9.2 roscpp简介

Namespace ros::names 该空间的主要函数有：

string **append**(const std::string &left, const std::string &right); //追加名称

string **clean** (const string &name); //清除图资源名称：删去双斜线、结尾斜线

const M_string &**getRemappings** (); //返回重映射remapping

string **remap** (const string &name); //对名称重映射

string **resolve**(const string &name, bool remap=true); //解析出名称的全名

bool **validate**(const string &name, string &error); //验证名称

提纲

- ROS编程简介
- roscpp简介
- roscpp 举例demo
- rospy 简介
- rospy 举例demo
- 机器人运动控制实例

9.3　roscpp举例demo

本节对roscpp编程进行举例应用，分别实现：

话题基本操作demo—topic_demo；

服务基本操作demo—service_demo；

服务参数器demo　— param_demo；

9.3　roscpp举例demo

topic_demo

功能描述：两个node，一个发布模拟的GPS消息（格式为自定义，包括坐标和工作状态），另一个接收并处理该信息（计算到原点的距离）。

步骤：

①package

②msg

③talker.cpp

④listener.cpp

⑤CMakeList.txt&package.xml

⑥编译

9.3 roscpp举例demo

①package

```
$ cd  ~/catkin_ws/src
$ catkin_create_pkg  topic_demo  roscpp  rospy  std_msgs
```

②msg

```
$ cd  topic_demo/
$ mkdir  msg
$ cd msg
$ gedit  gps.msg
```

gps.msg

```
float32 x
float32 y
string state
```

catkin_make $\rightarrow$ ~/catkin_ws/devel/include/topic_demo/gps.h

```
#include<topic_demo/gps.h>
Topic_demo::gps msg;
```

9.3 roscpp举例demo

③talker.cpp

```
#include <ros/ros.h>
#include <topic_demo/gps.h>
int main(int argc, char** argv){
    ros::init(argc, argv, “talker”);        //解析参数，命名节点talker
    ros::NodeHandle nh;                     //创建句柄，实例化node
    topic_demo::gps msg;                    //创建gps消息
    msg.x = 1.0;
    msg.y = 1.0;
    msg.state = “working”;
    ros::Publisher pub = nh.advertise<topic_demo::gps>(“gps_info”,1);     //创建publisher
    ros::Rate loop_rate(1.0);               //定义循环发布的频率
    while(ros::ok()){
        msg.x = 1.03 * msg.x;               //以指数增长，每隔1s
        msg.y = 1.01 * msg.y;
        ROS_INFO(“Talker: GPS: x = %f, y = %f”, msg.x, msg.y);   //输出当前msg
        pub.publish(msg);                   //发布消息
        loop_rate.sleep(); //根据定义的发布频率，sleep
    }
    return 0;
}
```

ros::Publisher advertise(const string &topic,

uint32_t queue_size);

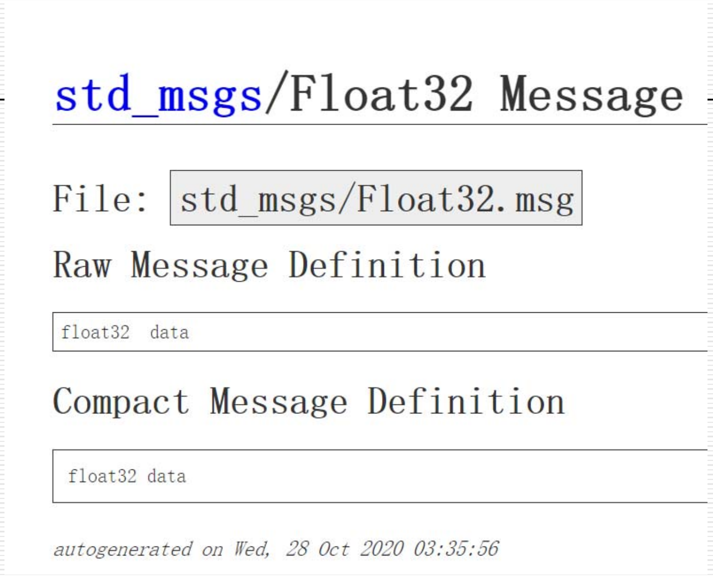
9.3　roscpp举例demo

④listener.cpp

```
#include <ros/ros.h>
#include <topic_demo/gps.h>
#include <std_msgs/Float32.h>

void gpsCallback(const topic_demo::gps::ConstPtr &msg)
{
    std_msgs::Float32 distance;
    distance.data = sqrt(pow(msg->x,2), pow(msg->y,2));
    ROS_INFO("Listener: Distance to origin = %f, state = %s", distanace.data, msg->state.c_str());
}

int main(int argc, char** argv){
    ros::init(argc,argv,"listener");
    ros::NodeHandle n;
    ros::Subscriber sub = n.subscribe("gps_info", 1, gpsCallback);  //创建subscriber
    ros::spin();  //反复调用当前可触发的回调函数，阻塞
    return 0;
}
```



std_msgs/Float32 Message

File: std_msgs/Float32.msg

Raw Message Definition

```
float32  data
```

Compact Message Definition

```
float32 data
```

autogenerated on Wed, 28 Oct 2020 03:35:56

ros::spinOnce();

9. 3　roscpp举例demo

⑤CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)   #CMAKE版本
project(topic_demo)      #项目名称

find_package(catkin REQUIRED COMPONENTS message_generation roscpp rospy std_msgs) #指定依赖
add_message_files(FILES gps.msg)   #添加自定义的msg
generate_messages(DEPENDENCIES std_msgs)   #生成msg对应的头文件

catkin_package(CATKIN_DEPENDS  roscpp rospy std_msgs message_runtime)
#用于配置ROS和pacakge配置文件和Cmake文件

include_directories(include ${catkin_INCLUDE_DIRS})   #指定C/C++的头文件路径

add_executable(talker src/talker.cpp )    #生成可执行目标文件
add_dependencies(talker topic_demo_generate_messages_cpp)   #添加依赖，必须有此局以生成msg
target_link_libraries(talker ${catkin_LIBRARIES})       #链接

add_executable(listener src/listener.cpp )
add_dependencies(listener topic_demo_generate_messages_cpp)
target_link_libraries(listener ${catkin_LIBRARIES})
```

9.3 roscpp举例demo

⑤package.xml

```
<?xml version="1.0"?>
<package>
  <name>topic_demo</name>
  <version>0.0.0</version>
  <description>The publish_subscribe_demo package</description>
  <maintainer email="hanhaomin008@126.com">davidhan</maintainer>
  <license>BSD</license>
  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>message_generation</build_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>rospy</build_depend>
  <build_depend>std_msgs</build_depend>
  <run_depend>roscpp</run_depend>
  <run_depend>rospy</run_depend>
  <run_depend>std_msgs</run_depend>
  <run_depend>message_runtime</run_depend>
  <export>
    <!-- Other tools can request additional information be placed here -->
   </export>
</package>
```

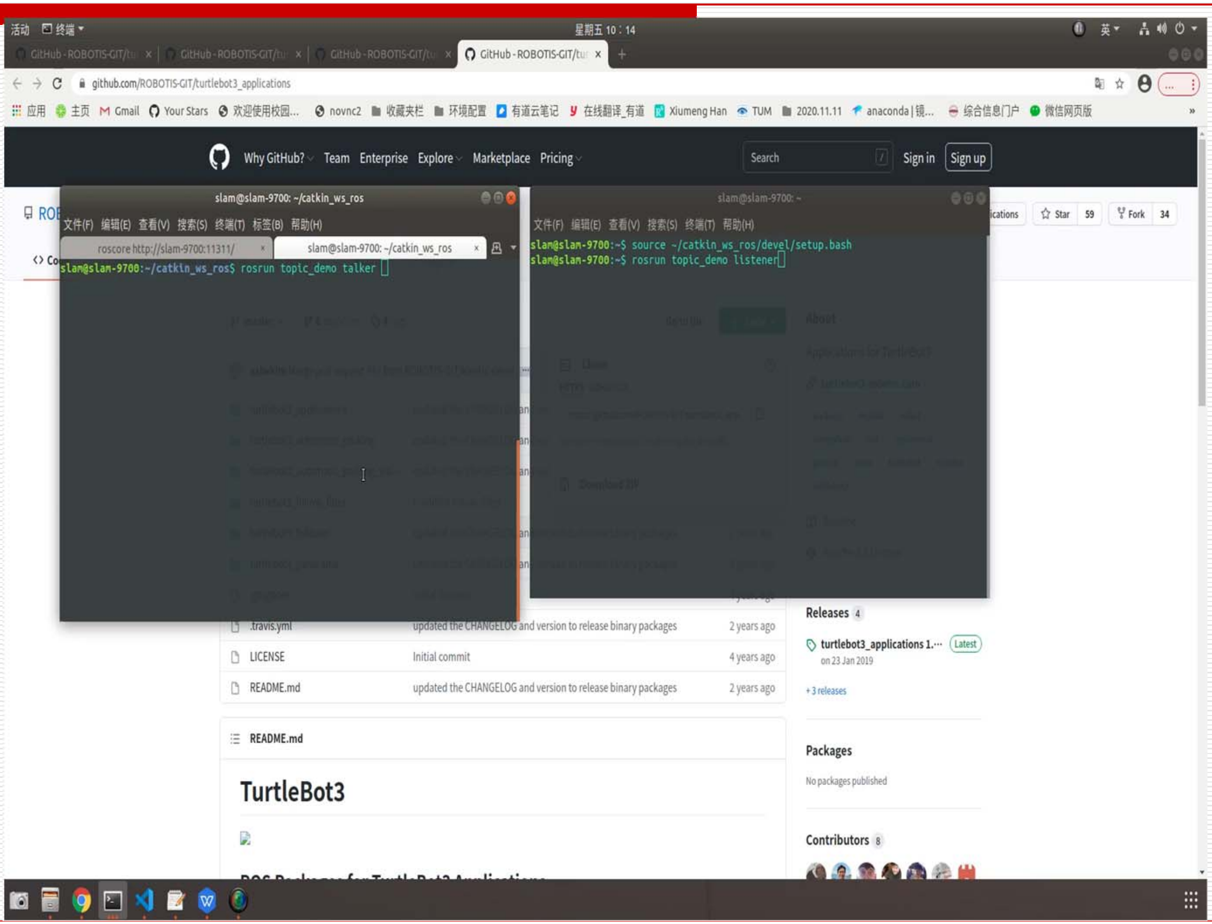
9.3 roscpp举例demo

⑥编译

```
cd ~/catkin_ws
catkin_make
```

9.3 roscpp举例demo

topic_demo


slam@slam-9700: ~/catkin_ws_ros
roscore http://slam-9700:11311/
slam@slam-9700: ~/catkin_ws_ros
slam@slam-9700: ~

```
slam@slam-9700:~/catkin_ws_ros$ rosrun topic_demo talker
```

```
slam@slam-9700:~$ source ~/catkin_ws_ros/devel/setup.bash
slam@slam-9700:~$ rosrun topic_demo listener
```

合肥工业大学 计算机与信息学院

9. 3　roscpp举例demo

service_demo

功能描述：两个node，客户端发布模拟身份信息注册请求（格式自定义，包括姓名、年龄），服务器接收处理该信息，并返回信息。

步骤:

①package

②srv

③server.cpp

④client.cpp

⑤CMakeList.txt&package.xml

⑥编译

9.3 roscpp举例demo

①**package**

```
$ cd  ~/catkin_ws/src
$ catkin_create_pkg  service_demo  roscpp  rospy  std_msgs
```

②**srv**

```
$ cd  service_demo/
$ mkdir  srv
$ vi  Greeeting.srv
```

Greeting.srv

```
string name
int32 age
---
string feedback
```

service_demo::Greeting::Request req;

service_demo::Greeting::Response res;

~/catkin_ws/devel/include/service_demo/Greeting.h

.../GreetingRequest.h

.../GreetingResponse.h

9.3 roscpp举例demo

③ server.cpp

```
#include <ros/ros.h>
#include <service_demo/Greeting.h>

bool handle_function(service_demo::Greeting::Request &req, service_demo::Greeting::Response &res){
   //显示请求信息
   ROS_INFO("Request from %s with age %d", req.name.c_str(), req.age);
   //处理请求，结果写入response
   res.feedback = "Hi " + req.name + ". I'm server!";
   //返回true，正确处理了请求
   return true;
}

int main(int argc, char** argv){
   ros::init(argc, argv, "greetings_server");        //解析参数，命名节点
   ros::NodeHandle nh;                    //创建句柄，实例化node
   ros::ServiceServer service = nh.advertiseService("greetings", handle_function);
   ros::spin();
   return 0;
}
```

9.3 roscpp举例demo

④client.cpp

```
#include <ros/ros.h>
#include <service_demo/Greeting.h>
int main(int argc, char** argv){
   ros::init(argc, argv, “greetings_server”);       //解析参数，命名节点
   ros::NodeHandle nh;                      //创建句柄，实例化node
   ros::ServiceClient client = nh.serviceClient<service_demo::Greeting>(“greetings”);

   service_demo::Greeting srv;
   srv.request.name = “HAN”;
   srv.request.age = “20”;

   if(client.call(srv)){
      ROS_INFO(“Feedback from server: %s.”, srv.response.feedback);
   }
   else{
      ROS_ERROR(“Failed to call service greetings.”);
      return 1;
   }
   return 0;
}
```

9.3 roscpp举例demo

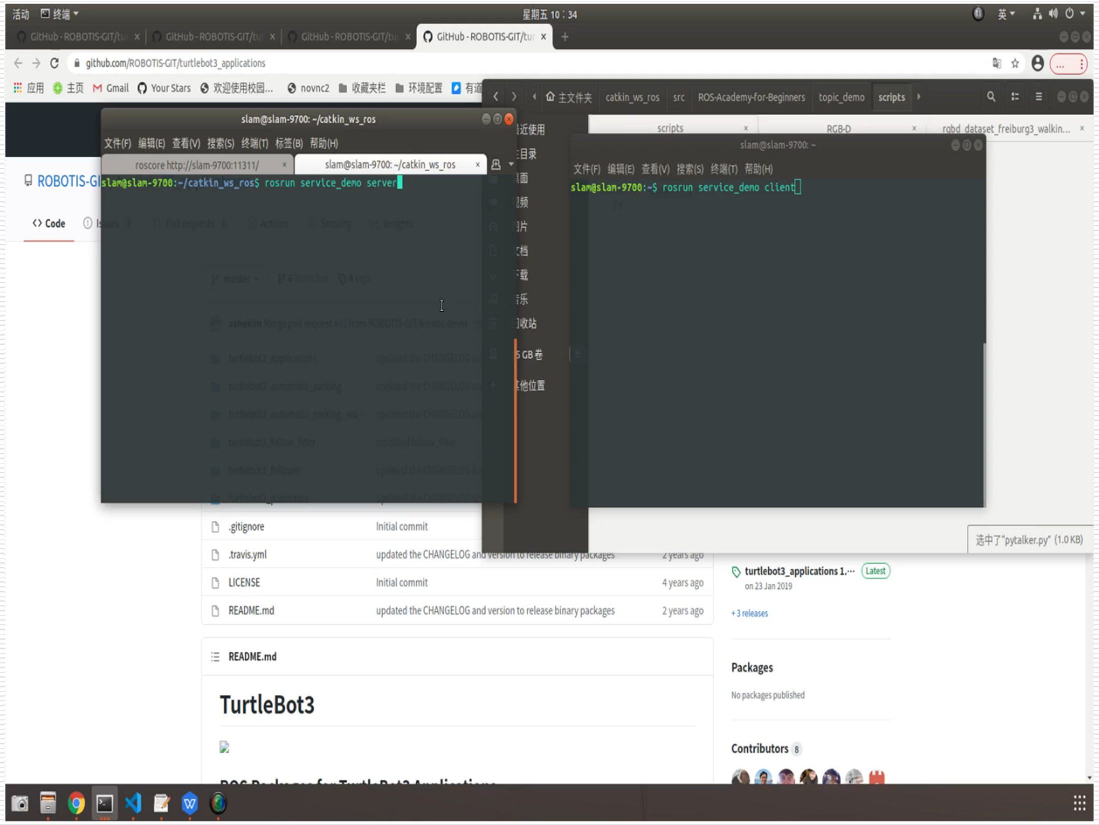
⑤CMakeList.txt&package.xml

⑥编译

同上一节

9.3 roscpp举例demo

service_demo



```
slam@slam-9700:~/catkin_ws_ros$ rosrun service_demo server
```

```
slam@slam-9700:~$ rosrun service_demo client
```

9.3 roscpp举例demo

param_demo

两种API：ros::param和ros::NodeHandle

9.3 roscpp举例demo

param_demo.cpp

```
#include <ros/ros.h>
int main(int argc, char** argv){
   ros::init(argc, argv, “greetings_server”);
   ros::NodeHandle nh;
   int parameter1, parameter2, parameter3, parameter4, parameter5;
   //获取参数
   ros::param::get(“param1”, parameter1);
   nh.getParam(“param2”, parameter2);
   nh.param(“param3”, parameter3, 123);
   //设置参数
   ros::param::set(“param4”, parameter4);
   nh.setParam(“param5”,parameter5);
   //检查参数是否存在
   ros::param::has(“param5”);
   nh.hasParam(“param6”);
   //删除参数
   ros::param::del(“param5”);
   nh.deleteParam(“param6”);
   return 0;
}
```

9. 3　roscpp举例demo

param_demo_cpp.launch

```
<launch>
   <!--Param标签设置单个参数-->
   <param name=“param1” value=“1”/>
   <param name=“param2” value=“2”/>
   <param name=“robot_descrption” command=”$(find xacro)/xacro.py $(find demo)/urdf/robot.urdf”/>

   <!—rosparam标签设置多个参数-->
   <rosparam>
      param3: 3
      param4: 4
      param10: helloworld!
   </rosparam>
   <rosparam file=“$(find param_demo)/config/myparam.yaml” command=“load”/>

   <node pkg=“param_demo” type=“param_demo” name=“param_demo” output=“screen”/>

</launch>
```

提纲

- ROS编程简介
- roscpp简介
- roscpp 举例demo
- rospy 简介
- rospy 举例demo
- 机器人运动控制实例

9.4　rospy简介

rospy 是python版本的编程接口库，部分函数用法与roscpp有所差异

rospy相关函数及类

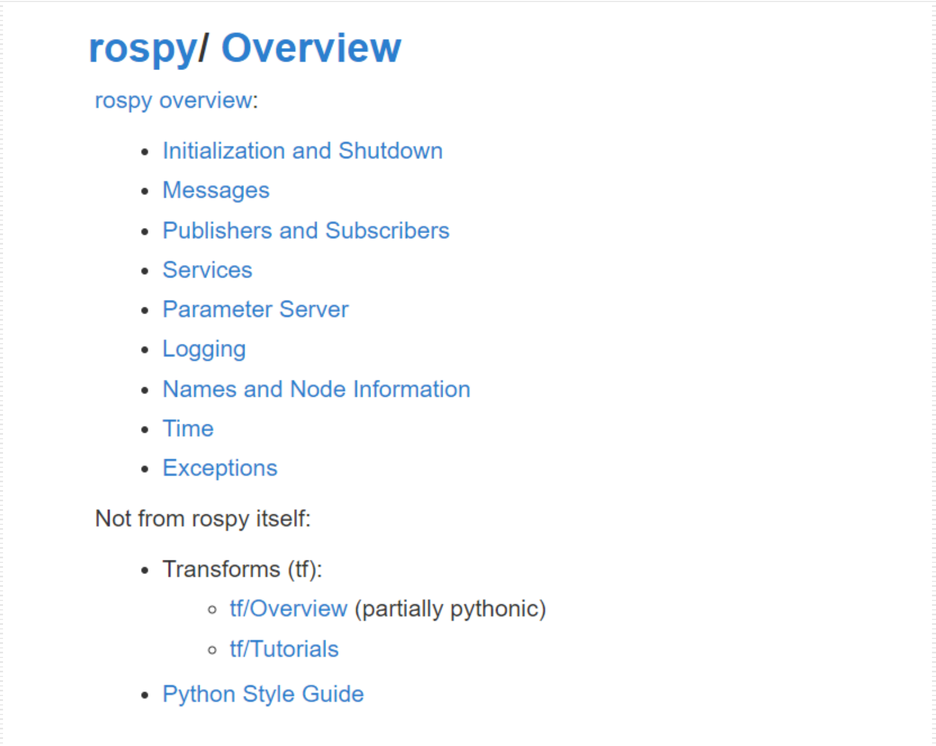
Node ----节点控制相关

Topic ----话题控制相关

Service ----服务控制相关

Param ----参数控制相关

Time ----时间控制相关



rospy/ Overview

rospy overview:

- Initialization and Shutdown
- Messages
- Publishers and Subscribers
- Services
- Parameter Server
- Logging
- Names and Node Information
- Time
- Exceptions

Not from rospy itself:

- Transforms (tf):
  - tf/Overview (partially pythonic)
  - tf/Tutorials
- Python Style Guide

9.4 rospy简介

Rospy-Node相关

```
import rospy
rospy.init_node('my_node')
```

函数	描述
init_node(name)	注册和初始化node
MasterProxy get_master()	获取master的句柄
bool is_shutdown()	返回是否关闭
on_shutdown(fn)	在node关闭时调用函数
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

9.4 rospy简介

//queue_size=None同步，其他整数异步

rospy-Topic相关

pub = rospy.Publisher(‘topic name’，std_msgs.msg.String, 10)

pub.publish(msg)

pub.unregister()

函数	描述
[[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)	等待指定topic的一个message
spin() 没有spinOnce() ！！！	触发topic或service的处理，会阻塞直到关闭
Publisher类	
__init__(self, name, data_class, queue_size=None)	构造函数
publish(self,msg)	成员函数发布消息
unregister(self)	成员函数停止发布
Subscriber类	
__init__(self, name, data_class, call_back=None, queue_size=None)	构造函数
unregister(self)	成员函数停止订阅

9.4　rospy简介

Service相关

s = rospy.Service(‘service name’, service_type, handle_function)

client = rospy.ServerProxy(‘service name’, service_type)

client.call(req) 或者 response=client(req)

函数	描述
wait_for_service(service, timeout=None)	阻塞直到服务可用，无返回值
Service类	
__init__(self, name, service_class, handler)	构造函数提供服务
shutdown(self)	成员函数关闭服务
ServiceProxy类	//实际上就是client
init(self,name, service_class)	构造函数 服务的请求方
call(self,*args, **kwds)	调用服务
__call__(self,*args, **kwds)	调用服务

9.4 rospy简介

Param相关

函数	描述
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)	参数是否存在参数服务器上
str search_param()	搜索参数

9.4　rospy简介

Time相关

Duration理解为一段时间
Time理解为一个时刻

函数	**描述**
Time类	rospy.Time(1,0)　rospy.Time.now()
__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
函数	
Time get_rostime()	当前时刻的Time对象
float get_time()	返回当前时间，返回float 单位秒
sleep(duration)	执行挂起
Rate 类	rate=rospy.Rate(5)　rate.sleep()
__init__(self, frequency)	构造函数
sleep(self)	挂起 考虑上一次的rate.sleep()时间
Time remaining(self)	成员函数 剩余sleep时间
Duration类	rospy.Duration(1,0)
__init__(self,secs=0, nsecs=0)	构造函数 秒和纳秒

提纲

- ROS编程简介
- roscpp简介
- roscpp 举例demo
- rospy 简介
- rospy 举例demo
- 机器人运动控制实例

9.5　rospy举例demo

本节对rospy编程进行举例应用，分别实现：

话题基本操作demo—topic_demo；

服务基本操作demo—service_demo；

服务参数器demo　— param_demo；

9.5　rospy举例demo

topic_demo

功能描述：两个node，一个发布模拟的GPS消息（格式为自定义，包括坐标和工作状态），另一个接收并处理该信息（计算到原点的距离）。

步骤：

①package

②msg

③talker.py

④listener.py

⑤CMakeList.txt&package.xml

⑥编译（虽然python不需要编译，但ROS生成msg需要）

9.5　rospy举例demo

gps.msg

```
float32 x
float32 y
string state
```

↓ **catkin_make**

~/catkin_ws/devel/lib/python2.7/dis-pacakges/topic_demo/msg/__init__.py

```
from topic_demo.msg import gps
```

9.5 rospy举例demo

③pylistener.py

```
#!/usr/bin/env python

import rospy
import math
from topic_demo.msg import gps

def callback(gps):
        distance = math.sqrt(math.pow(gps.x, 2) + math.pow(gps.y, 2))
        rospy.loginfo(‘Listener: GPS distance=%f, state : %s‘, distance, gps.state)

def listener():
      rospy.init_node(‘pylistener’)
      rospy.Subscriber(‘gps_info’, gps, callback)
      rospy.spin()

If __name__ == ‘__main__’:
      listener()
```

9.5　rospy举例demo

④pytalker.py

```
#!/usr/bin/env python
import rospy
from topic_demo.msg import gps

def talker():
  rospy.init_node('pytalker', anonymous=True)
  pub = rospy.Publisher('gps_info', gps, queue_size=10)
  rate = rospy.Rate(1)
   x = 1.0
   y = 2.0
   state = 'working'
   while not rospy.is_shutdown():
      rospy.loginfo('Talker: GPS: x=%f , y = %f')
      pub.publish(gps(state, x, y))
      x = 1.03 * x
      y = 1.01 * y
      rate.sleep()

If __name__ == '__main__':
   talker()
```

9.5 rospy举例demo

⑤CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(topic_demo)

find_package(catkin REQUIRED COMPONENTS message_generation roscpp rospy std_msgs)
add_message_files(FILES gps.msg)
generate_messages(DEPENDENCIES std_msgs)

catkin_package(CATKIN_DEPENDS  roscpp rospy std_msgs message_runtime)

include_directories(include ${catkin_INCLUDE_DIRS})

add_executable(talker src/talker.cpp )
add_dependencies(talker topic_demo_generate_messages_cpp)
target_link_libraries(talker ${catkin_LIBRARIES})

add_executable(listener src/listener.cpp )
add_dependencies(listener topic_demo_generate_messages_cpp)
target_link_libraries(listener ${catkin_LIBRARIES})
```

9.5 rospy举例demo

⑤package.xml

```
<?xml version="1.0"?>
<package>
  <name>topic_demo</name>
  <version>0.0.0</version>
  <description>The publish_subscribe_demo package</description>
  <maintainer email="hanhaomin008@126.com">davidhan</maintainer>
  <license>BSD</license>
  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>message_generation</build_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>rospy</build_depend>
  <build_depend>std_msgs</build_depend>
  <run_depend>roscpp</run_depend>
  <run_depend>rospy</run_depend>
  <run_depend>std_msgs</run_depend>
  <run_depend>message_runtime</run_depend>
  <export>
    <!-- Other tools can request additional information be placed here -->
   </export>
</package>
```

9.5 rospy举例demo

⑥编译

```
cd ~/catkin_ws
catkin_make
```

⑦运行

设置权限

```
chmod u+x pytalker.py pylistener.py
```

刷新ROS环境

```
source ~/catkin_ws/devel/setup.bash
```

启动master

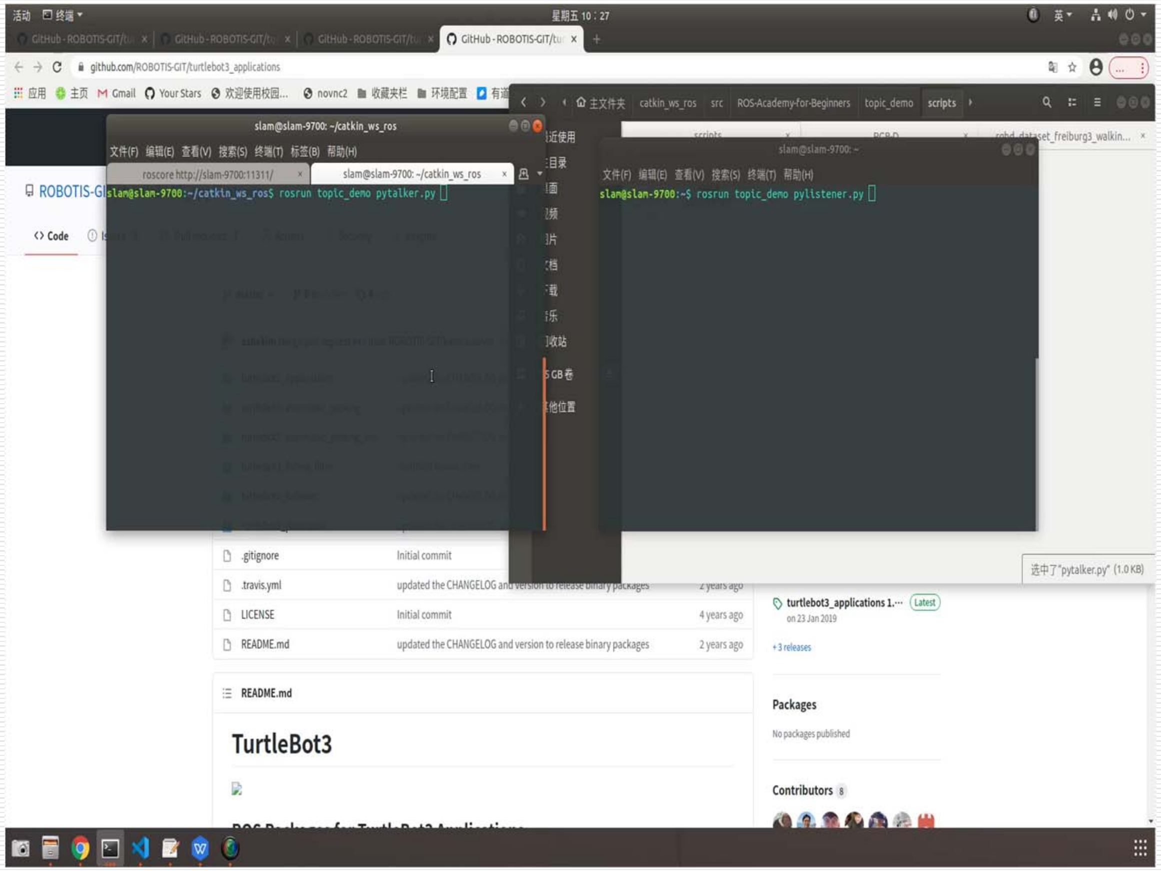
```
roscore
```

运行程序

```
rosrun 包名 pytalker.py
rosrun 包名 pylistener.py
```

9.5 rospy举例demo

topic_demo



```
slam@slam-9700:~/catkin_ws_ros$ rosrun topic_demo pytalker.py
```

```
slam@slam-9700:~$ rosrun topic_demo pylistener.py
```

9.5 rospy举例demo

service_demo

功能描述：两个node，客户端发布模拟身份信息注册请求（格式自定义，包括姓名、年龄），服务器接收处理该信息，并返回信息。

步骤：

①package

②srv

③server_demo.py

④client_demo.py

⑤CMakeList.txt&package.xml

⑥编译（虽然python不需要编译，但ROS生成msg需要）

9.5　rospy举例demo

Greeting.srv

```
string name
int32 age
---
string feedback
```

catkin_make

~/catkin_ws/devel/lib/python2.7/dis-pacakges/service_demo/srv/__init__.py

生成几种类型：

service_demo.srv.Greeting
service_demo.srv.GreetingRequest
Service_demo.srv.GreetingResponse

```
from service_demo.srv import *
```

9. 5　rospy举例demo

③server_demo.py

```
#!/usr/bin/env pypthon
import rospy
from service_demo.srv import *

def server_srv():
      rospy.init_node(  ‘greetings_server’  )
      s = rospy.Service(‘greetings’, Greeting, handle_function)   #定义程序的server端
      rospy.loginfo(‘Ready to handle the request:’)
      rospy.spin()

def handle_function(req):
       rospy.loginfo(‘Request from’, req.name, ‘with age’, req.age)
       return GreetingResponse(‘Hi %s. I’m server!’%req.name)

If __name__==‘__main__’:
       server_srv()
```

9. 5　rospy举例demo

④client_demo.py

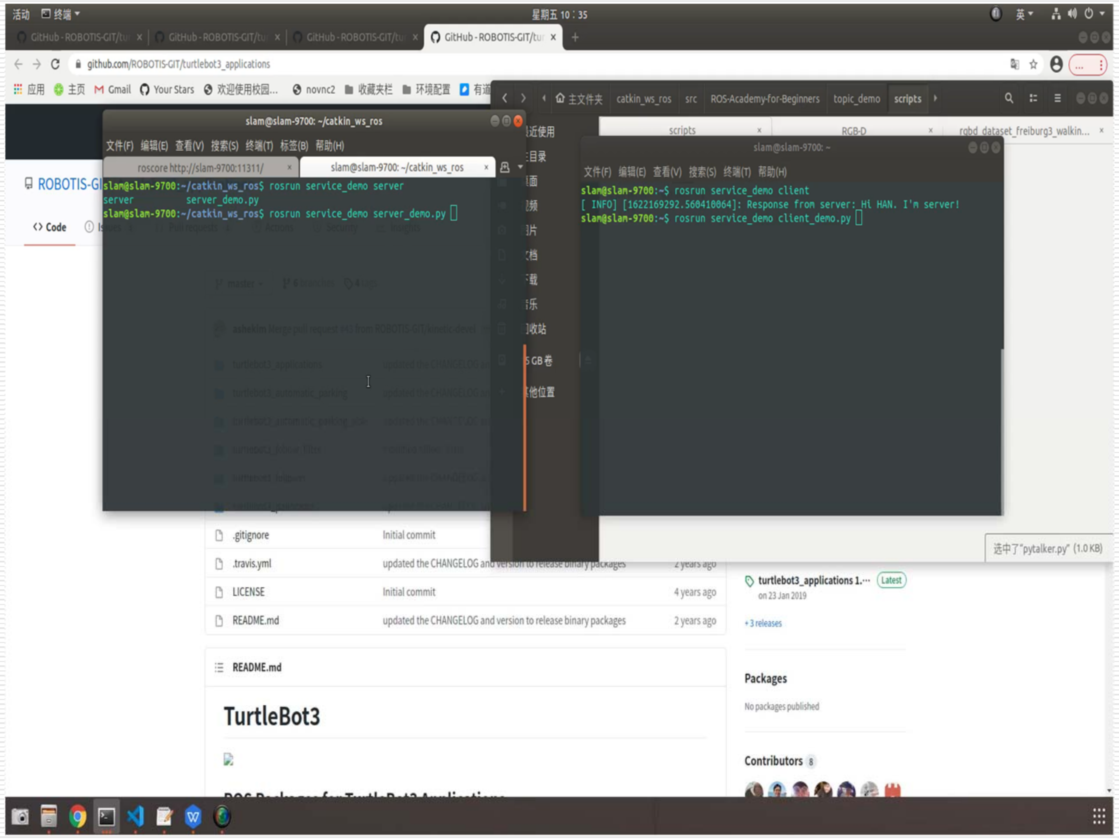
```
#!/usr/bin/env pypthon
import rospy
from service_demo.srv import *

def client_srv():
      rospy.init_node( ‘greetings_client’ )
      rospy.wait_for_service(‘greetings’)
      try:
          greetings_client = rospy.ServiceProxy(‘greetings’, Greeting)
          rosp = greetings_client(‘HAN’, 20)     #   rosp=greetings_client.call(‘HAN’, 20)
          rospy.loginfo(‘Message From Server: %s’%rosp.feedback)
      except rospy.ServiceExceptioin, e:
          rospy.logwarn(‘Service call failed:%s’%e)

If __name__==‘__main__’:
       client_srv()
```

9.5 rospy举例demo

service_demo



```
slam@slam-9700:~/catkin_ws_ros$ rosrun service_demo server
server          server_demo.py
slam@slam-9700:~/catkin_ws_ros$ rosrun service_demo server_demo.py
```

```
slam@slam-9700:~$ rosrun service_demo client
[ INFO] [1622169292.560410064]: Response from server: Hi HAN. I'm server!
slam@slam-9700:~$ rosrun service_demo client_demo.py
```

9.5　rospy举例demo

param_demo.py

```
import rospy

def param_demo():
       rospy.init_node(‘param_demo’)
       rate = rospy.Rate(1)
       while(not rospy.is_shutdown):
              parameter1 = rospy.get_param(‘/param1’)
              rospy.delete_param(‘/param1’)
              rospy.set_param(‘/param1’, 1)
              if(rospy.has_param(‘/param2’)):
                         rospy.loginfo(‘/param2 exists’)
              else:
                       ros[y.loginfo(‘/param3 does not exist’)
              params = rospy.get_param_names()
              rospy.loginfo(‘param list: %s’, params)
        rate.sleep()

If __name__=‘__main__’:
      param_demo()
```

提纲

- ROS编程简介
- roscpp简介
- roscpp 举例demo
- rospy 简介
- rospy 举例demo
- 机器人运动控制实例

9.6 机器人运动控制Demo

rospy Topic

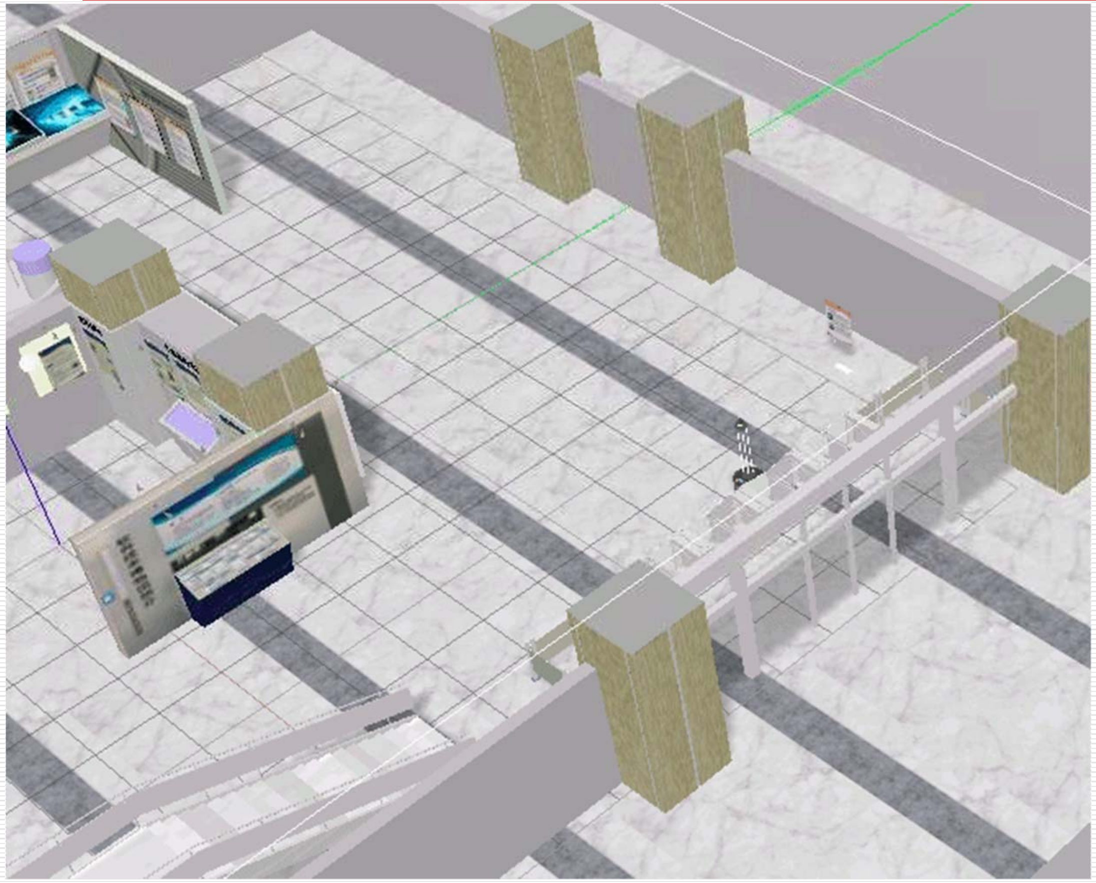
让机器人沿着x轴的方向，也就是前方以0.5m/s的速度运动，同时有一个0.5rad/s的角速度绕着z轴进行旋转。分别使用rostopic命令和rospy中的topic实现。

9.6 机器人运动控制Demo

在Ubuntu的终端输入下面这段命令：
这条命令用于向/cmd_vel话题，发类型为geometry_msgs/Twist的信息，其中linear下x方向，机器人向前方向的速度设置为每秒0.5米，angular下z方向，机器人顺时针旋转，速度为每秒0.5米。此信号值发布一次。

```
rostopic pub /cmd_vel geometry_msgs/Twist  "
linear:
  x: 0.5
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.5"
```

9.6 机器人运动控制Demo



在这一例子中，我们让机器人沿着x轴的方向，也就是前方以0.5m/s的速度运动，同时有一个0.5rad/s的角速度绕着z轴进行旋转。一旦发布了这一消息，机器人就会按照消息上的命令一直执行，要想使机器人停下来需要重新发布话题消息，将机器人的线速度和角速度都设置为0，即：

```
rostopic pub /cmd_vel geometry_msgs/Twist "linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0"
```

9.6 机器人运动控制Demo

在仿真环境下我们用ROSPY实现以上功能

1、编写程序topic_demo1.py

首先`rospy.init_node('topic_demo')`初始化节点`topic_demo`，`pub = rospy.Publisher('/cmd_vel', Twist, queue_size=1)`，创建话题发布程序，话题是`/cmd_vel`，消息的数据类型是`Twist`，设置数据处理时间是每秒1次。机器人速度赋值`move.linear.x = 0.5, move.angular.z = 0.5`。

```
for i in xrange(5):
    pub.publish(move)
    rate.sleep()
```

9.6 机器人运动控制Demo

执行5次，共5s，移动命令。之后将机器人速度设置为0，`move.linear.x = 0, move.angular.z = 0`，发布消息是机器人停止运动，`pub.publish(move)`。

```
#!/usr/bin/env python

import rospy
from std_srvs.srv import Empty, EmptyResponse # import the service message python
classes generated from Empty.srv.
from geometry_msgs.msg import Twist

rospy.init_node('topic_demo')
pub = rospy.Publisher('/cmd_vel', Twist, queue_size=1)
rate = rospy.Rate(1)
move = Twist()
move.linear.x = 0.5
move.angular.z = 0.5
for i in xrange(5):
    pub.publish(move)
    rate.sleep()

move.linear.x = 0
move.angular.z = 0
pub.publish(move)
```

9.6 机器人运动控制Demo

2、添加ROS主从配置

```
vim ~/.bashrc
```

```
#export
ROS_MASTER_URI=http://192.168.8.101:11311
export ROS_MASTER_URI=http://127.0.0.1:11311
#export ROS_HOSTNAME=192.168.8.xxx
export ROS_HOSTNAME=127.0.0.1
```

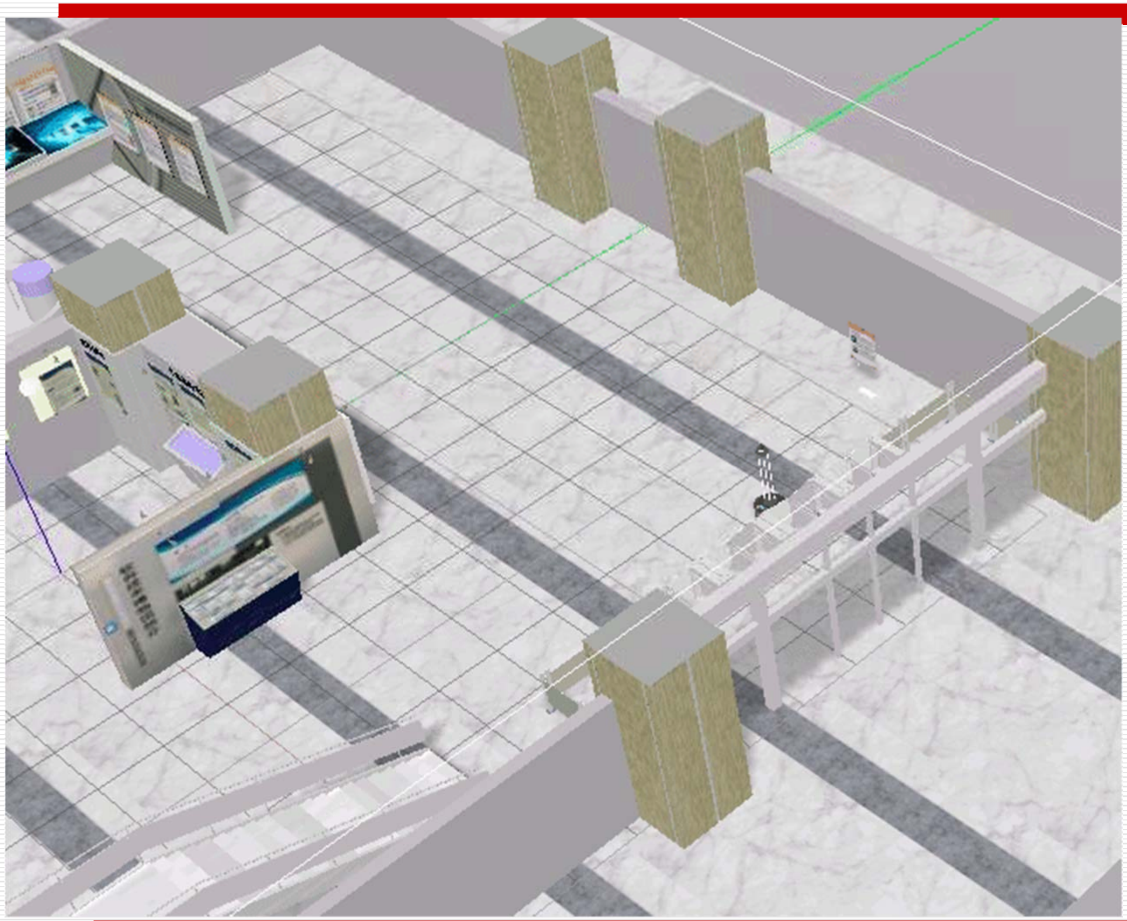
3、启动XBot Gazebo仿真

```
roslaunch robot_sim_demo robot_spawn.launch
```

4、运行程序

```
python topic_demo1.py
```

9.6 机器人运动控制Demo



练习1： rospy Service

练习2： rospy Action

让机器人沿着x轴的方向，也就是前方以0.5m/s的速度运动，同时有一个0.5rad/s的角速度绕着z轴进行旋转。要求分别使用rospy中的service实现。