ROS机器人开发实践

--by liuhaofeng

ROS安装与设置

- 初始化环境变量

```
echo "source /opt/ros/melodic/setup.bash" >>  ~/.bashrc
source  ~/.bashrc

ROS默认安装到/opt目录,以上为初始化ros环境(可以装多个ros版本)
```

- 安装ROS下载工具

```
sudo apt-get install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

话题通信方式

- 基本原理

```
发布者和订阅者都可以注册新的话题名，(服务中只允许客户端定义)
多对多的方式,发布者和订阅者不知道彼此的存在,都是发送或接收于某种话题,启动顺序没要求
异步方式,无反馈
```

```
# include <iostream>
# include "ros/ros.h"
# include "std_msgs/String.h"

void chatterCallback(const std_msgs::String::ConstPtr & msg)         //订阅者接收到订阅的话题消息后,会进入回调函数
{
   ROS_INFO("%s", msg.data);
}

int main(int argc, char ** argv)
{
   ros::init(argc, argv, "talker");         //ros节点初始化,名字为talker
   ros::NodeHandle n;                       //声明节点句柄对象,用来创建发布者...
   ros::Publisher chatter_pub = n.advertise<std_msgs::String>("chatter", 1000);   //发布者chatter_pub发布chatter话题的string消息,消息队列长度为1000
   ros::Subscriber chatter_sub = n.subscriber("chatter", 1000, chatterCallback);   //订阅者chatter_sub订阅chatter话题消息,队列长度为1000,注册回调函数

   ros ::Rate loop_rate(10);              //设置10hz循环频率

   while(ros::ok())
   {
      std_msgs::String msg;
      ROS_INFO("%s", msg.data.c_str());  //打印输出   String消息类型只有一个data成员存放数据

      chatter_pub.publish(msg);             //发布者发布
```

```
    ros::spinOnce();                //处理该节点的订阅话题的所有回调函数.等待进入回调函数,执行一次即退出回调,继续执行下面代码
    ros::spin();                    //循环等待回调函数,进入回调函数后一直执行回调函数,不再执行后面代码

    loop_rate.sleep();              //按频率延时
  }

  return 0;
}
```

```
# 自定义话题消息

string name
uint8  age
uint8  ok = 1
uint8  no = 0
```

- 编译功能包

```
1. add_executable
用于设置 生成的可执行文件 和 设置需要编译的代码
add_executable(talker  src/talkr.cpp)

2.  target_link_libraries
用于 可执行文件  和 要链接的库
target_link_libraries(talker  ${catkin_LIBRARIES})

3.  add_dependencies
用于设置可执行文件依赖动态生成的头文件等代码
add_dependencies(talker  ${PROJECT_NAME}_generate_messages_cpp)

4.  include_directories
用于设置头文件的相对路径
include_directories(include ${catkin_INCLUDE_DIRS})

---

# 编译消息功能包
// 编译消息时,不用指定生成文件,使用消息时,以功能包名为目录,消息文件名.h即可
1. 在package.xml添加功能包依赖项
<build_depend>message_generation</build_depend>
<run_depend>message_runtime</run_depend>

2. 在CMakeLists.txt中添加编译选项

添加依赖项
find_package(catkin REQUIRED COMPONENTS
  geometry_msgs
  roscpp
  rospy
  std_msgs
  messages_generation
)
```

```
catkin_package(
  CATKIN_DEPENDS geometry_msgs roscpp rospy std_msgs messages_runtime
)

设置需要编译的文件
add_message_files(
  FILES
  Person.msg
)

generate_messages(
  DEPNDENCIES
  std_msgs
)
```

服务通信方式

- 基本原理

```
一对多的方式,客户端和服务器知道彼此的存在,启动顺序没要求
同步方式,有反馈.客户端发送给服务器请求,等待服务器回应
```

```
 #include "ros/ros.h"
 #include "learning_comm/Add.h"       //learning_comm功能包下的编译好的头文件,在devel/include中

 bool add(learning_comm::Add::Request &req, learning_comm::Add::Response  &res)
 {
    res.sum  =  req.a + req.b;
    return true;
 }

 int main(int argc, char ** argv)
 {
    ros::init(argc, argv, "add_server");
    ros::NodeHandle  n;

    ros::ServiceClient  client = n.serviceClient<learning_comm::Add>("add_two_ints");  //不用设置消息队列
    ros::ServiceServer  service = n.advertiseService("add_two_ints", add);                    //服务端service创建服务器add_two_ints,回调函数为add

    learning_comm::Add  srv;
    srv.request.a = atoll(argv[1]);
    src.request.b = atool(argv[2]);

    if(client.call(srv))
    {
       ROS_INFO("ok");
    }
    else
    {
       ROS_ERROR("No");
```

```
    return 1;
  }

  ros::spin();        //一直等待服务请求, 若有请求跳入回调函数,执行完后还在此处等待

  return 0;
}
```

```
# 自定义服务数据  learning_comm/srv/Add.srv:

int 64  a
int64  b
---
int64  sum
```

多机通信

- 操作步骤

```
1、设置ip地址
ifconfig 查看ip地址
分别在两台计算机/etc/hosts文件中加入别名：
# pc1
192.168.1.1   pc2
#pc2
192.168.1.2   pc1
可以用ping测试

2、设置ROS_MASTER_URI
export ROS_MASTER_URI = http://pc1:11311
最好两台机器都添加到 ～/.bashrc中
```

launch启动文件

- 基本元素，launch文件可自启动roscore

```
<launch>
  <node pkg = "功能包" type="可执行文件" name="重命名" output="screen"/>
  <arg name = "arg1"  default = "value1"/>                                    <!-- arg属于launch内部变量，仅限该文件使用-->
  <param name = "param1" value = "$(arg  arg1)" />                            <!-- param存储在参数服务器中，全局变量-->
  <param file = "$(find  learn)/config/earn.yaml"  command = "load" ns = "ws"/> <!-- 加载参数文件，ns为设置命名空间-->
  <node pkg="pkg1"  type = "type1" args = "0 0 0 1"/>
  <node pkg = "pkg2" type = "type2" >
    <param name = "para2" value = "true" />
    <remap from = "/turtlebot/cmd_vel" to = "/cmd_vel" />                     <!--名称重映射，不用改话题名就可以-->
  </node>
  <include file = "$(dirname)/other.launch" />                                <!--包含其他launch文件-->
</launch>
```