1.引言

我们知道启动ROS相关应用要么是roslaunch要么是rosrun，实际项目中不可能每次手动去运行这些命令，简单的就是写到脚本里去，但是涉及复杂的，就需要使用代码去处理

2.在代码中启动roslaunch和rosrun

直接贴出代码

```
import subprocess
import rospy
import rosnode

class launch_demo:
    def __init__(self, cmd=None):
        self.cmd = cmd

    def launch(self):
        self.child = subprocess.Popen(self.cmd)
        return True

    def shutdown(self):
        self.child.terminate()
        self.child.wait()
        return True

if __name__ == "__main__":
    rospy.init_node('launch_demo',anonymous=True)

    launch_nav = launch_demo(["roslaunch", "pibot_simulator", "nav.launch"])

    launch_nav.launch()

    r = rospy.Rate(0.2)
    r.sleep()

    rospy.loginfo("switch map...")
    r = rospy.Rate(1)
    r.sleep()

    rosnode.kill_nodes(['map_server'])

    map_name =
```

```
"/home/pibot/ros_ws/src/pibot_simulator/maps/blank_map_with_obstacle.yaml"

    map_node = subprocess.Popen(["rosrun", "map_server", "map_server", map_name,
"__name:=map_server"])

    while not rospy.is_shutdown():
        r.sleep()
```

上面使用python代码启动了一个PIBOT模拟器的导航，然后5s后切换了一个地图

- 使用subprocess.Popen可以启动一个进程(roslaunch或者rosrun)
- 使用rosnode.kill_nodes可以杀死一个rosnode

3.代码中导航相关应用

定点导航

```
from launch_demo import launch_demo
import rospy

import actionlib
from actionlib_msgs.msg import *
from move_base_msgs.msg import MoveBaseAction, MoveBaseGoal
from nav_msgs.msg import Path
from geometry_msgs.msg import PoseWithCovarianceStamped
from tf_conversions import transformations
from math import pi

class navigation_demo:
    def __init__(self):
        self.set_pose_pub = rospy.Publisher('/initialpose',
PoseWithCovarianceStamped, queue_size=5)

        self.move_base = actionlib.SimpleActionClient("move_base", MoveBaseAction)
        self.move_base.wait_for_server(rospy.Duration(60))

    def set_pose(self, p):
        if self.move_base is None:
            return False

        x, y, th = p

        pose = PoseWithCovarianceStamped()
        pose.header.stamp = rospy.Time.now()
        pose.header.frame_id = 'map'
        pose.pose.pose.position.x = x
        pose.pose.pose.position.y = y
        q = transformations.quaternion_from_euler(0.0, 0.0, th/180.0*pi)
        pose.pose.pose.orientation.x = q[0]
```

```
        pose.pose.pose.orientation.y = q[1]
        pose.pose.pose.orientation.z = q[2]
        pose.pose.pose.orientation.w = q[3]

        self.set_pose_pub.publish(pose)
        return True

    def _done_cb(self, status, result):
        rospy.loginfo("navigation done! status:%d result:%s"%(status, result))

    def _active_cb(self):
        rospy.loginfo("[Navi] navigation has be actived")

    def _feedback_cb(self, feedback):
        rospy.loginfo("[Navi] navigation feedback\r\n%s"%feedback)

    def goto(self, p):
        goal = MoveBaseGoal()

        goal.target_pose.header.frame_id = 'map'
        goal.target_pose.header.stamp = rospy.Time.now()
        goal.target_pose.pose.position.x = p[0]
        goal.target_pose.pose.position.y = p[1]
        q = transformations.quaternion_from_euler(0.0, 0.0, p[2]/180.0*pi)
        goal.target_pose.pose.orientation.x = q[0]
        goal.target_pose.pose.orientation.y = q[1]
        goal.target_pose.pose.orientation.z = q[2]
        goal.target_pose.pose.orientation.w = q[3]

        self.move_base.send_goal(goal, self._done_cb, self._active_cb,
self._feedback_cb)
        return True

    def cancel(self):
        self.move_base.cancel_all_goals()
        return True

if __name__ == "__main__":
    rospy.init_node('navigation_demo',anonymous=True)

    launch_nav = launch_demo(["roslaunch", "pibot_simulator", "nav.launch"])
    launch_nav.launch()

    r = rospy.Rate(0.2)
    r.sleep()

    rospy.loginfo("set pose...")
    r = rospy.Rate(1)
    r.sleep()
    navi = navigation_demo()
    navi.set_pose([-0.7,-0.4,0])

    rospy.loginfo("goto goal...")
    r = rospy.Rate(1)
```

```
    r.sleep()
    navi.goto([0.25,4, 90])

    while not rospy.is_shutdown():
        r.sleep()
```

上面完成设置机器人位置和导航到某一位置的功能