|  |
| --- |
| ROS 开发文档 |
| 作者：Chu Chu |
| xx年xx月xx日 |

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 修订内容 | 修订人 | 修订时间 |
| 初版 | 改动了xxx | xxx | xx年xx月xx日 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[第一章 ROS基本功能 1](#_Toc9060)

[1.1新建工作空间 1](#_Toc27469)

[1.2新建package 1](#_Toc1413)

[1.3新建节点 1](#_Toc26606)

[1.4自定义msg 4](#_Toc22705)

[1.5自定义srv 5](#_Toc31970)

[运行程序 6](#_Toc12689)

[1.6一些问题 7](#_Toc25002)

[1.6 一些知识 8](#_Toc4386)

[第二章 节点设计 8](#_Toc1252)

[2.1 串口通信 8](#_Toc7884)

[2.2 共享文件夹报错 12](#_Toc15993)

[第三章 13](#_Toc28790)

[3.1 的的 13](#_Toc23526)

[第四章 13](#_Toc14347)

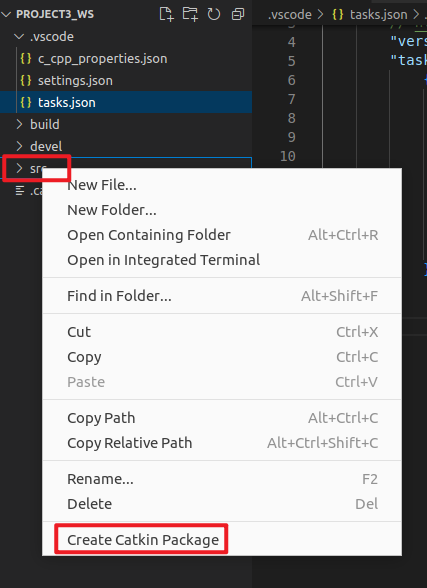
[4.1 13](#_Toc8238)

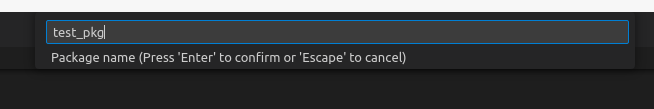
# ROS基本功能

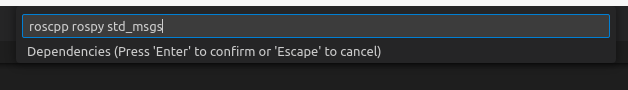
## 新建工作空间

<http://www.autolabor.com.cn/book/ROSTutorials/chapter1/14-ros-ji-cheng-kai-fa-huan-jing-da-jian/142-an-zhuang-vscode.html>

## 新建package



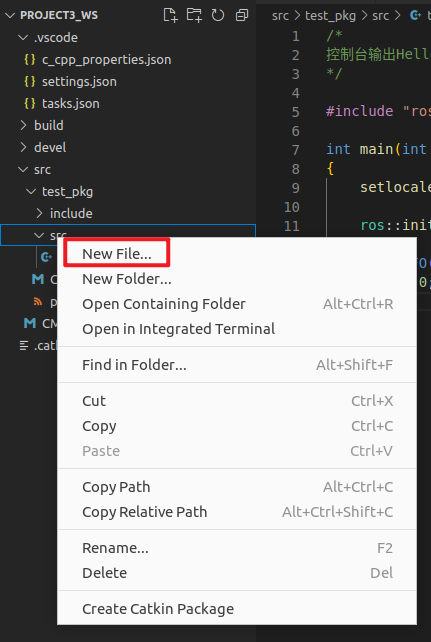




roscpp rospy std\_msgs

## 新建节点

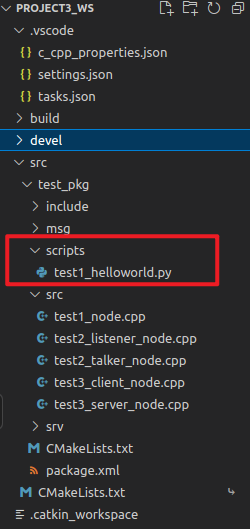
C++





执行到这一句才算新建了一个节点

Python



#!/usr/bin/env python3

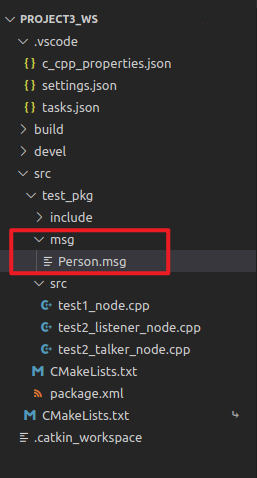
import rospy

if \_\_name\_\_=="\_\_main\_\_":

rospy.init\_node("Hello")

rospy.loginfo("Hello world!")

## 自定义msg



包目录下新建msg文件夹以及msg文件——编辑msg文件

修改配置文件

package.xml

大概在51行对应位置添加（该操作只需要执行一次）

<build\_depend>message\_generation</build\_depend>

<exec\_depend>message\_runtime</exec\_depend>

CMakeLists.txt

在第10行左右增加message\_generation

find\_package(catkin REQUIRED COMPONENTS

roscpp

rospy

std\_msgs

message\_generation

)

# 需要加入 message\_generation,必须有 std\_msgs

在第50行左右（这一步每增加一个新msg都要做一次）

## 配置 msg 源文件

add\_message\_files(

FILES

Person.msg

)

# 生成消息时依赖于 std\_msgs

generate\_messages(

DEPENDENCIES

std\_msgs

)

注意只放开第三行

#执行时依赖

catkin\_package(

# INCLUDE\_DIRS include

# LIBRARIES demo02\_talker\_listener

CATKIN\_DEPENDS roscpp rospy std\_msgs message\_runtime

# DEPENDS system\_lib

)

此时编译一下会生成消息文件的中间件（devel/include/包文件夹/msg名.h文件）

为了方便代码提示以及防止误抛异常，还需要配置c\_cpp\_properties.json

{

"configurations": [

{

"browse": {

"databaseFilename": "",

"limitSymbolsToIncludedHeaders": true

},

"includePath": [

"/opt/ros/noetic/include/\*\*",

"/usr/include/\*\*",

"/xxx/yyy工作空间/devel/include/\*\*" //配置 head 文件的路径

],

"name": "ROS",

"intelliSenseMode": "gcc-x64",

"compilerPath": "/usr/bin/gcc",

"cStandard": "c11",

"cppStandard": "c++17"

}

],

"version": 4

}

## 自定义srv

基本流程与msg一致，区别在于.srv文件格式（上面是客户端请求消息格式，下面是服务器相应回传的消息格式）例如：

实现两数相加

# 客户端请求时发送的两个数字

int32 num1

int32 num2

---

# 服务器响应发送的数据

int32 sum

add\_service\_files(

FILES

AddInts.srv

)

其他配置步骤参考新建msg章节

此时编译一下会生成消息文件的中间件

1. devel/include/包文件夹/服务名.h文件
2. devel/include/包文件夹/服务名Request.h文件
3. devel/include/包文件夹/服务名Response.h文件

为了方便代码提示以及防止误抛异常，还需要配置c\_cpp\_properties.json。与新建msg章节相同

## 运行程序

### C++

修改配置文件（package文件夹下面的CMakeLists.txt）

如果只增加了节点（注意不是源代码文件），需要在Build处（约113行）对应位置添加

add\_executable(节点名称

src/C++源文件名.cpp

)

target\_link\_libraries(节点名称

${catkin\_LIBRARIES}

)

#其中节点名称需要与ros初始化给的名字一致，源文件名是初始化节点的源文件全名，只需要对应修改中文处

如果增加了引用自定义消息的节点

add\_executable(节点名称 srcC++源文件名.cpp)

add\_dependencies(节点名称 ${PROJECT\_NAME}\_generate\_messages\_cpp)

target\_link\_libraries(节点名称

${catkin\_LIBRARIES}

)

如果增加了使用了服务的节点

add\_executable(节点名称 src/C++源文件名.cpp)

add\_dependencies(节点名称 ${PROJECT\_NAME}\_gencpp)

target\_link\_libraries(节点名称

${catkin\_LIBRARIES}

)

编译执行

Ctrl+Shift+B,确保没有报错

去到工作空间目录，bash运行source ./devel/setup.bash（每打开一个新的terminal 都需要source一次）

继续运行rosrun 包名 节点名

**一劳永逸解决source问题**

去到主目录找到.bashrc文件（被隐藏了）

去到工作空间/devel目录下找到setup.bash文件，记录下文件位置

打开.bashrc文件，在最后一行添加

source 相对文件位置/setup.bash

我这里是

source ~/Project/zjrobot\_ws/devel/setup.bash

### Python

1. 添加可执行权限

去到script目录，命令行执行chmod +x 自定义文件名.py

1. 编辑Cmake.txt 文件

catkin\_install\_python(PROGRAMS scripts/自定义文件名.py

DESTINATION ${CATKIN\_PACKAGE\_BIN\_DESTINATION}

)

1. 运行

roscore

cd 工作空间

source ./devel/setup.bash (需要去到工作空间目录)

rosrun 包名 自定义文件名.py

## ROS C++调用so文件

参考：<https://blog.csdn.net/weixin_44041199/article/details/116792591>

调用动态链接库包括两个文件：.so文件 .h文件。假设目前有个动态链接库包括两个文件：libhello.so hello.h

第一步 放置so文件与h文件

So文件放在 工作空间/lib目录下

h文件放在 包/include目录下

第二步 编辑Cmake文件

包含头文件路径

include\_directories( include ${catkin\_INCLUDE\_DIRS} )

包含so文件路径

link\_directories( lib ${catkin\_LIB\_DIRS} )

链接

add\_executable(main src/main.cpp) target\_link\_libraries(main ${catkin\_LIBRARIES} hello )

//注意：这里hello要去掉前面的lib与后面的.so

第三步 调用

在需要调用动态链接库的源文件里添加头文件，然后主函数里面就可以直接调用了

如果编译通过，但运行报错找不到so文件

ldd 命令查看so文件包含情况

解决办法：去到.bashrc文件下面把so文件路径添加进去

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/your/path

## 在ROS包里面跑cpp程序

Caktin里面包含了cpp编译器，因此在ros里面跑普通的cpp程序也没有什么问题。

1.6.1 修改配置文件

add\_executable(运行时候的名字 src/c++文件名.cpp)

target\_link\_libraries(运行时候的名字 ${catkin\_LIBRARIES})

例如：有一个helloworld.cpp文件

add\_executable(hello src/helloworld.cpp)

target\_link\_libraries(hello ${catkin\_LIBRARIES})

1.6.2运行

在ros包目录下面

source ./devel/setup.bash

rosrun zjrobot\_pkg hello

## 开机ROS自启动

思路：Ubuntu自带开机自启动脚本功能，只需要提前编写好脚本，添加进ubuntu里面的自启动列表里就行（需要提前准备好launch文件）

步骤一 编写sh脚本

在工作空间文件夹下面新建startup文件夹，文件夹里面新建start.sh文件

编辑sh文件

#! /bin/bash

source /opt/ros/noetic/setup.sh

source /home/julis/Project/hi-kvision\_-ros\_-driver/devel/setup.bash

roslaunch /home/julis/Project/hi-kvision\_-ros\_-driver/src/hk\_opencv\_mono/launch/zjrobot.launch

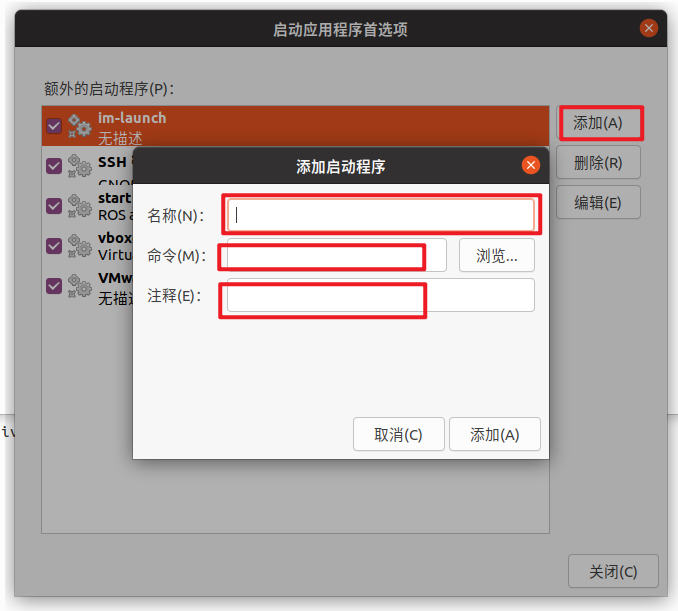
以上内容需要做对应修改

步骤二 测试sh文件

在startup文件夹下面终端运行./startup.sh

步骤三 添加进自启动列表里





名称 注释自选，命令直接浏览到sh文件即可。

步骤四 重启电脑，大功告成

## 一些问题

### 问题：.vscode文件夹里面没有tasks.json 文件

解决方案：Ctrl+Shift+B，选择:catkin\_make:build，点击小齿轮

### 执行python程序的时候有如下报错



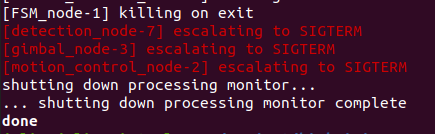
解决方法：在程序的第一行添加#!/usr/bin/env python3

### rosnode list不能显示部分节点（节点在运行，但显示不出来）

解决办法：那个节点的cpp文件是否忘了添加handler

ros::NodeHandle n;

### Ctrl C不能正常关闭，并产生如下报错



解决方法：

while(ros::ok)

改成

while(！ros::isShuttingDown())

### 开机自启动部分节点启动失败

解决方法：查看失败的节点代码，是否有用到了参数。如果在参数初始化之前使用了getparam，那该节点就会开机自启动失败。但是手动启动没问题。解决方法只要在getparam之前加个延时，等待其他节点把参数初始化完成即可（或者你自己在getparam之前初始化好参数）。

### 因为python版本问题导致catkin\_make报错

解决办法

首次用这个命令

  $ catkin\_make -DPYTHON\_EXECUTABLE=/usr/bin/python3

然后便可以自己用catkin\_make命令

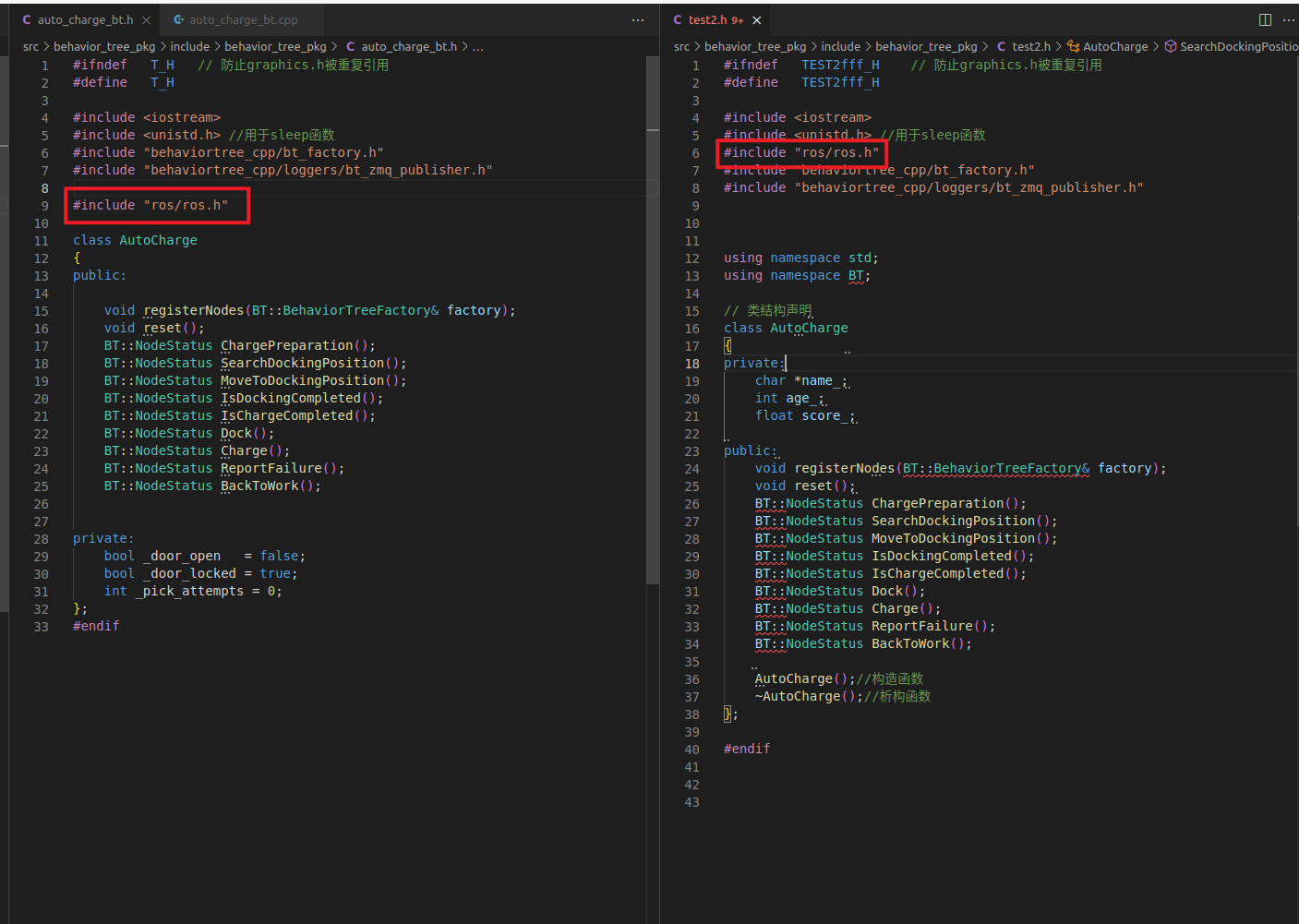
### 因为版本问题导致python节点无法运行（典型报错就是找不到yaml包 ）

环境：ros noetic

解决办法：在python文件开头指定好python版本

[#!/usr/bin/python3](https://www.jianshu.com/p/400c612381dd).8

### 在同时引用ros.h以及其他库函数时其他命名空间出现莫名其妙的错误



解决方法：把ros.h的引用放到所有引用之后

## 一些知识

1. 一个节点可能由多个源文件组成
2. 调用自定义c++头文件与源文件的方法：

<http://www.autolabor.com.cn/book/ROSTutorials/di-3-zhang-ros-tong-xin-ji-zhi-jin-jie/32-roszhong-de-tou-wen-jian-yu-yuan-wen-jian/322-zi-ding-yi-yuan-wen-jian-diao-yong.html>

1. roslaunch执行多个节点时是不会严格按照书写顺序执行的，所以节点在设计时就要确保任意顺序启动无问题

# 节点设计

## 串口通信

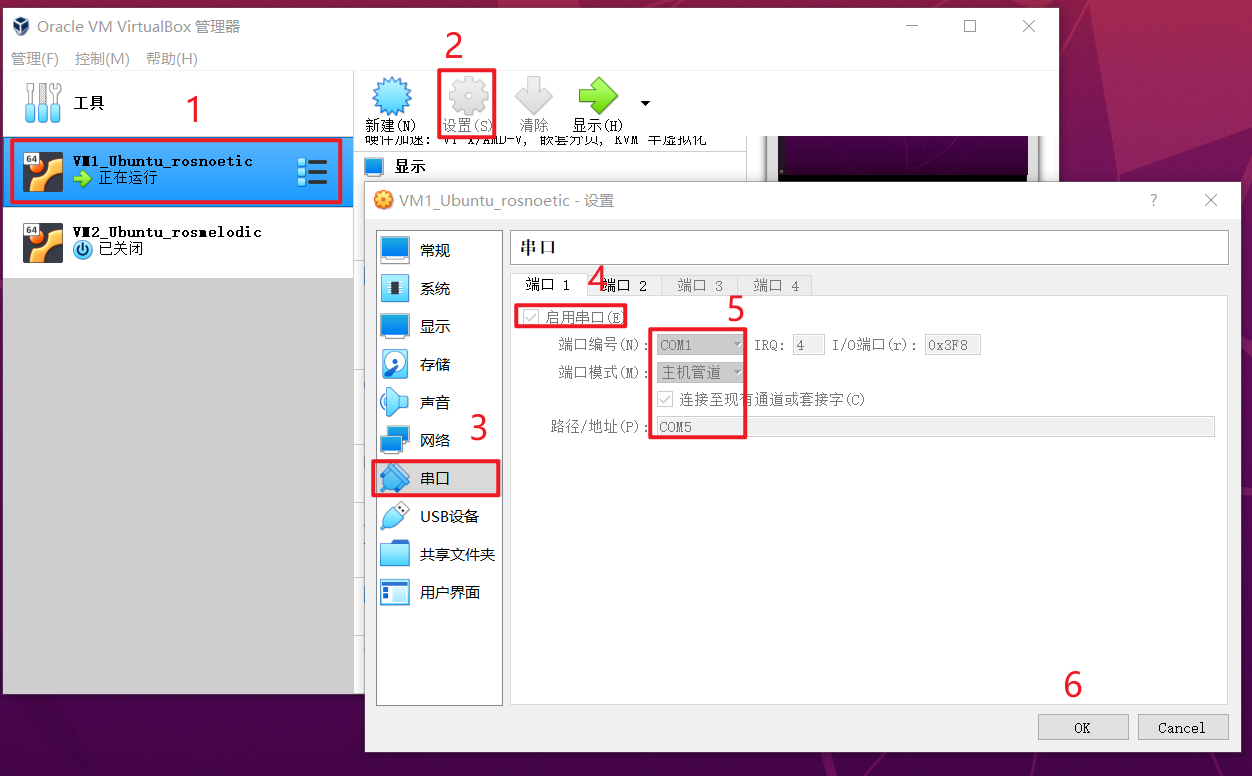
* 1. 准备工作

需要安装串口功能包：sudo apt install ros-noetic-serial

如果没有用虚拟机，则直接去到下一步

如果用了虚拟机，则需要设置下串口透传，设置方法：

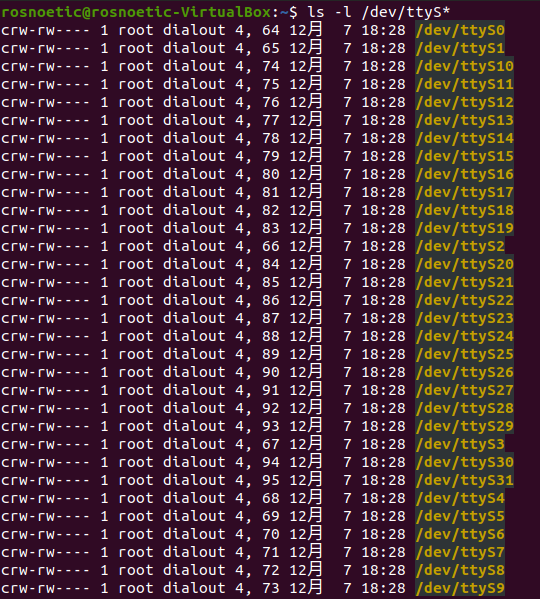
在虚拟机关闭的情况下



我们设置宿主里面的COM5作为linux里面的COM1

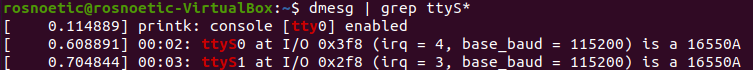
然后启动虚拟机。注意，一旦虚拟机启动，宿主机器的串口就会被占用。

设置完毕之后在linux里面命令行输入 ls -l /dev/ttyS\* 查看所有可用的串口



可以看到可用的端口

命令行输入 dmesg | grep ttyS\* 查看正在通信的端口



注意，linux里面端口名字不叫COM，而叫ttyS+数字。一般来说ttyS0对应COM1

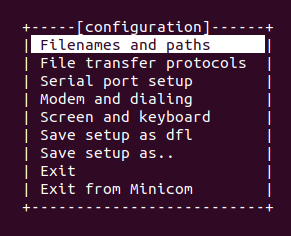
* 1. linux里

我们用到的串口调试工具叫minicom

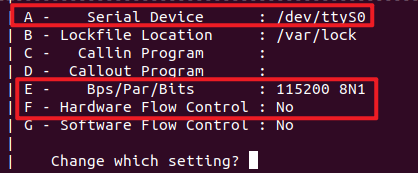
命令行输入sudo apt-get install minicom 安装串口调试工具

命令行输入 dmesg | grep ttyS\* 寻找可用串口，这里我们的可用端口是ttyS0

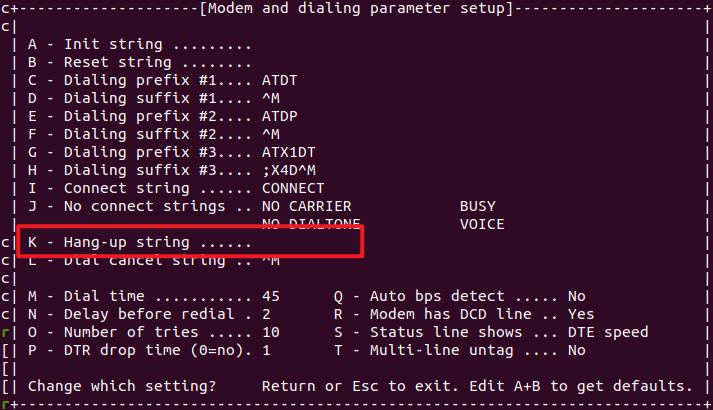
命令行输入 sudo minicom -s配置串口



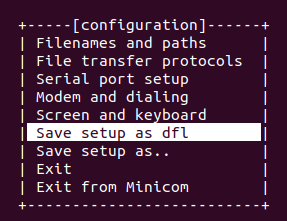
需要配置的项目有两个，首先进入选项Serial port setup里面，



然后去到Modem and dialing里面，把K改为空



最后save setup as dfl



然后选择Exit就自动打开了串口，正常情况下要打开串口直接命令行 sudo minicom。

退出先Ctrl A，然后X

* 1. 到这里只是完成了Linux系统对串口的识别，下面需要在ros里面读取到串口消息

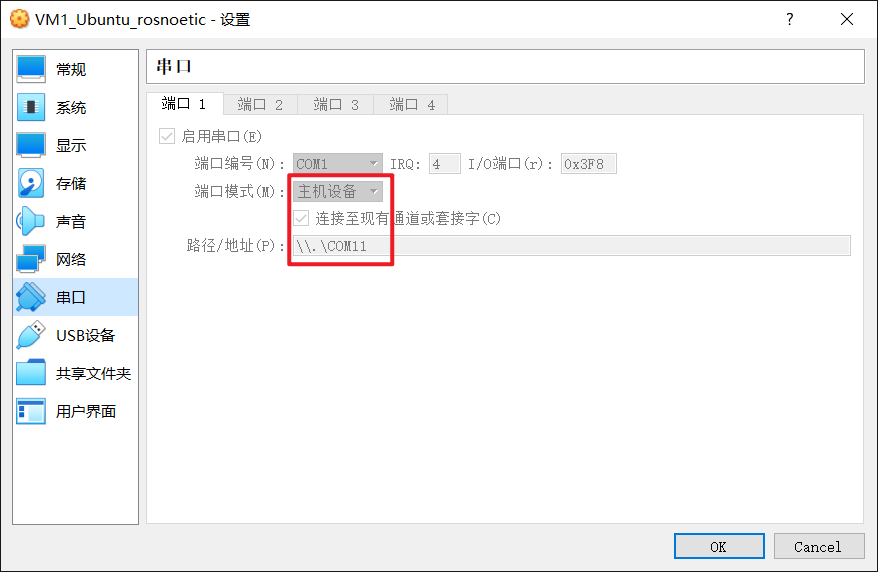
可能的报错

1 新添加一个串口之后，可能出现



解决方法

修改路径/地址名称。



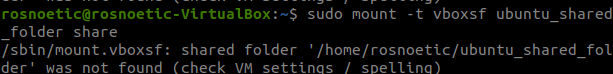
更多操作查看这里：<https://www.cnblogs.com/lcw/p/3159455.html>

## 共享文件夹报错

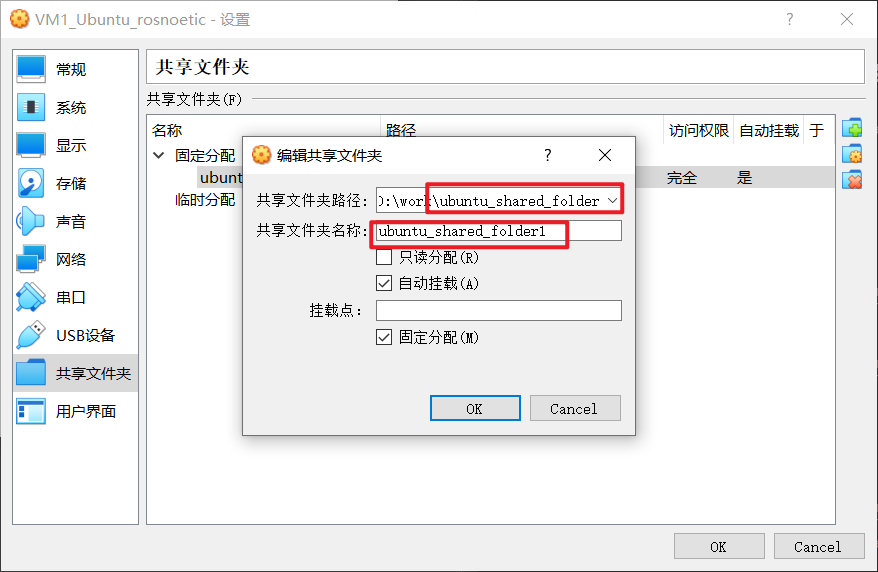
挂在共享文件夹指令

sudo mount -t vboxsf Share share\_folder

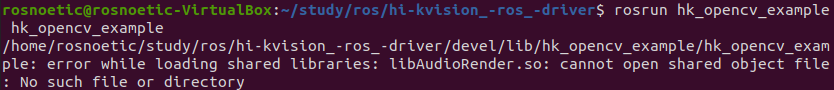
前一个是Vbox 下的共享文件夹名称，后面一个参数是Ubuntu系统里面文件夹名称



共享文件夹路径名不能等于共享文件夹名称



so文件动态链接库报错



原因：系统找不到动态链接库文件

解决办法：

1 把动态链接库位置增加到搜索目录（该方法未跑通）

2 把所有so文件移动到系统一定会访问的地址：usr/lib



知识点：ROS在编译过程中需要链接一次动态链接库，运行时候还要链接一次

编译过程中的链接一般没问题，运行过程时候的链接容易出问题

知识点：

创建pkg的时候少加了一些引用时，可以去pkg的CMakeList里面第10行左右手动添加

# 环境搭建

## Window+虚拟机

虚拟机方式：

记得把显存从16M改成128M

准备工作：

ROS安装

开发环境搭建

ROS功能包

共享文件夹设置

串口通信设置

生产环境搭建

## Ubuntu系统

需要提前安装好的包：

# Behavior Tree行为树

## 简介

机器人行为逻辑的控制一般用行为树或者状态机或者两者结合实现。本章介绍如何用现在较为流行的C++行为树库以及可视化设计工具Groot来进行行为树开发

## 环境安装

Groot依赖了BehaviorTree包，所以先安装BehaviorTree，再安装Groot

BehaviorTree安装

出到了V4，由于Groot开源版本V1只支持V3，所以这里我们安装V3.8版本的行为树

安装需要的功能包

sudo apt-get install libzmq3-dev libboost-dev

git clone https://ghproxy.com/https://github.com/BehaviorTree/BehaviorTree.CPP

cd BehaviorTree.CPP

mkdir build

cd build

编译

cmake ..

make -j8

sudo make install

这里一般不会出问题

Groot安装

git clone https://ghproxy.com/https://github.com/BehaviorTree/Groot.git

cd Groot

git submodule update --init --recursive

mkdir build

cd build

cmake ..

Make

这里会出的几个问题

1. Groot编译时候找不到BehaviorTree 包，导致编译失败

解决方法：sudo apt-get install ros-$ROS\_DISTRO-behaviortree-cpp-v3

1. 全部安装完毕之后找不到Groot启动文件

解决办法：观察make的信息，里面显示了启动文件被移动到了

/usr/local/lib/groot/Groot

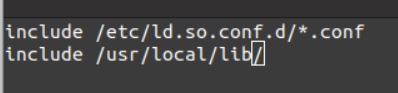
1. 运行Groot时报错找不到BehaviorTree库



解决办法：

sudo vim /etc/ld.so.conf

在里面加上库的路径



## 使用方法

CmakeLists文件修改

set(CMAKE\_CXX\_STANDARD 17)

set(CMAKE\_CXX\_STANDARD\_REQUIRED ON)

find\_package(behaviortree\_cpp)

不加这个会导致一系列报错

2.

include\_directories(

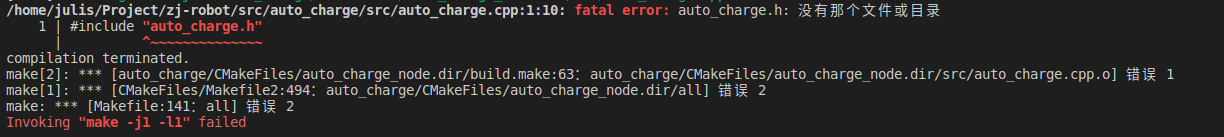
include

${catkin\_INCLUDE\_DIRS}

${PROJECT\_SOURCE\_DIR}/include/auto\_charge

)

不加这个会报错找不到头文件



target\_link\_libraries(auto\_charge\_node

${catkin\_LIBRARIES}

BT::behaviortree\_cpp

)

## 设计流程

1. 用Groot设计行为树
2. 另存得到行为树xml文件
3. 微调复制放到程序里（去掉include行，复制包括root之间的内容）

## Tips

如何实时观察行为树运行情况

#include "behaviortree\_cpp/loggers/bt\_zmq\_publisher.h"

BT::PublisherZMQ publisher\_zmq(tree);

另外需要在tickWhileRunniung函数前面加个延时，给Groot时间用来连接

sleep(5);

tree.tickWhileRunning();

增加节点状态变化输出

#include "behaviortree\_cpp/loggers/bt\_cout\_logger.h"

BT::printTreeRecursively(tree.rootNode());

## 参考

状态机与行为树对比：https://zhuanlan.zhihu.com/p/463182588

行为树&Groot官方地址https://www.behaviortree.dev/

安装参考：https://shoufei.blog.csdn.net/article/details/125473506?spm=1001.2014.3001.5502

# 仿真

## Gazebo机器人建模的几种方法

1. urdf导入
   1. 具体步骤：
      1. 获取urdf文件（来源：solidworks导出或者自己手写）
      2. roslaunch

<launch>

<!-- 设置参数 -->

<param name="robot\_description" textfile="$(find simulation)/urdf/zjrobot.urdf" />

<!-- 启动 gazebo -->

<include file="$(find gazebo\_ros)/launch/empty\_world.launch" />

<arg name="world\_name" value="$(find simulation)/gazebo/my\_world.world"/>

<!-- 在 gazebo 中显示机器人模型 -->

<node pkg="gazebo\_ros" type="spawn\_model" name="model" args="-urdf -model mycar -param robot\_description" />

</launch>

1. 直接在gazebo里面画模型
2. 直接 加载stl文件

## Gazebo建模思路

Gazebo建模分为两块：机器人本体建模，环境建模（机器人轨道也是属于环境的一种）

环境建模思路（对于自定义自己设计的环境）：

1. 也当成机器人建模，用sw导出。结果：会出现各种问题，浮空抖动。该方法不可行
2. 参考gazebo模型sdf代码，环境都是只有一个mesh文件，然后添加碰撞与视觉效果即可。针对这个思路建模过程如下:
   1. 打开gazebo任意保存一个model
   2. 用保存的model为模板修改sdf文件
   3. 在对应位置添加mesh文件

## Gazebo仿真问题整理

注意：用虚拟机跑gazebo会有各种各样的问题，建议用一台专门跑linux的机器用来跑仿真。亲测用了单独的linux系统之后，很多问题均未出现

1. SW直接另存为的stl文件导入到gazebo之后放大了1000倍

有好几种解决办法：

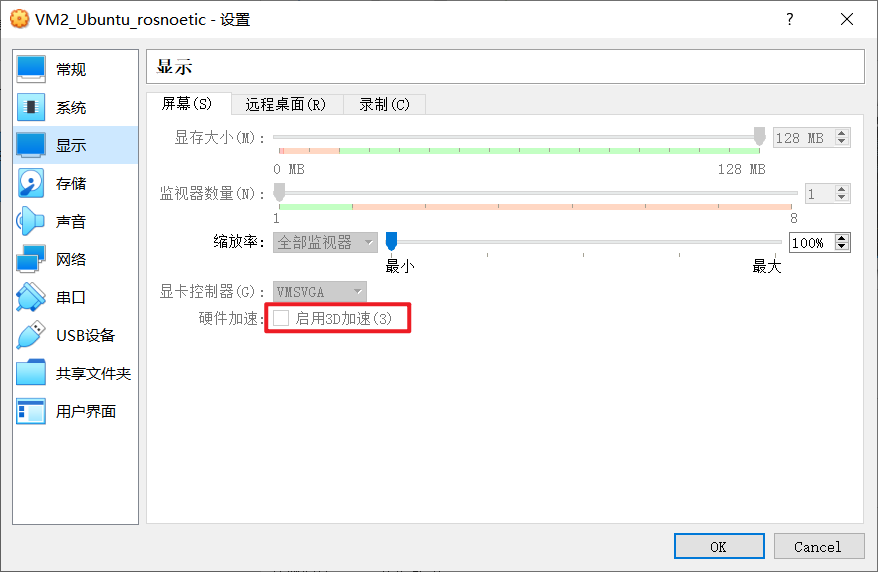
* 1. SW里面对模型进行1000倍缩放（不推荐）
  2. 用swtourdf插件导出的stl文件（推荐）
  3. 任意得到一个gazebo model文件模板，在里面引用stl文件，然后在碰撞与视觉属性里面各缩放1000倍（推荐）

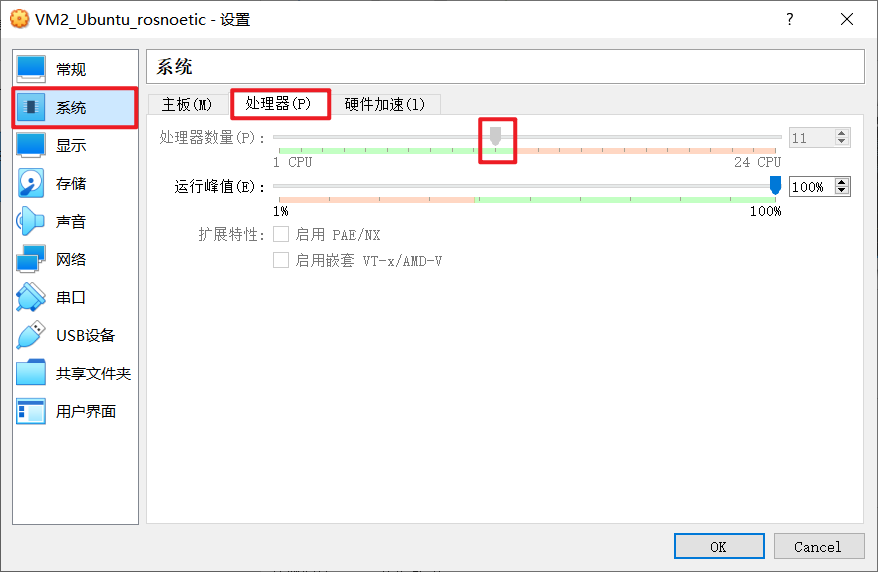
1. 自己写的urdf文件导入到gazebo中一运行无限掉落

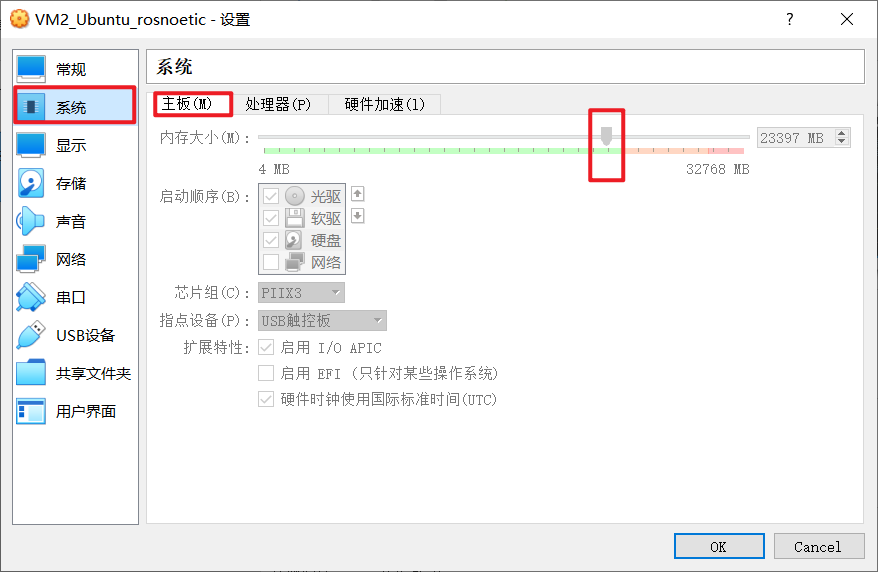
多半是因为urdf中没有定义碰撞

1. 首次运行gazebo黑屏

解决办法：如果是用虚拟机环境，关闭3D硬件加速，然后增加内存以及CPU核心数







1. 虚拟机环境和单机环境下roslaunch运行的gazebo无法保存世界,界面卡死

解决办法：如果连接了外置屏幕，拔掉外置屏幕，或者鼠标两个屏幕间来回游走，保存界面才会刷新

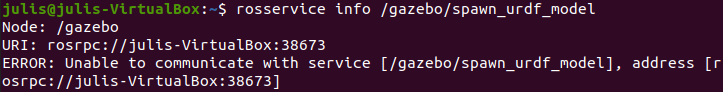
虚拟机环境下缩小放大gazebo软件可能会刷新，也有可能只刷新一次，看人品

1. 单机环境下sudo gazebo运行的世界无法spawn模型，卡在waiting for service
2. 虚拟机环境下sudo gazebo无法通过spawn 加载urdf模型

运行查看gazebo的任何一个服务info

rosservice info /gazebo/spawn\_urdf\_model

发现不能与服务通信上，怀疑是sudo 启动gazebo有些问题



运行rosservice list 发现很多gazebo的服务是启动了的

sudo的gazebo无法加载urdf模型，但是可以直接导入世界，能否把urdf模型提前放到世界文件里然后加载？

按照上面这个思路有两个方案：

1. urdf转sdf放到world文件里。结果：sudo方法卡在preparing the world打不开，roslaunch方法可以打开。神奇！
2. 在gazebo一个个导入stl文件组成机器人，然后另存为世界。结果：可行，但比较繁琐
3. 打不开gazebo

解决方法：分别运行以下两句命令，把之前打开的gazebo进程关闭干净

sudo killall gzclient

sudo killall gzserver

1. 单机模式下打开gazebo发现载入不了urdf模型

解决办法：查看urdf是否注释了inertia属性，少了必要属性模型就会加载不出来

1. 单机模式下打开gazebo有黄字报错

解决办法：查看报错内容，是否提示有重复的节点上次关闭gazebo没有关干净。kill掉即可

rosnode kill /节点名

1. 刚载入模型时模型嵌入到地面里导致物理仿真异常
2. urdf模型在gazebo里面修改过之后，fix to world 连杆与关节不见了
3. 同样的world文件（模型来自于1. 官方模型库 2. roslaunch spawn urdf模型 然后保存为世界），sudo打开能够正常显示模型，roslaunch打开啥模型都不存在。而另外导入stl以及人为sdf写入world文件的，roslaunch打开能显示
4. 如何调整导入的机器人的重置世界之后的初始位姿
   1. 直接模型位置拖动，保存成世界。该方法每次重新打开gazebo才能起作用
   2. 貌似reset world与重启gazebo所参考的初始位姿是不同的，即便在gazebo里面设置好新的位姿，reset之后还是恢复到刚刚导入模式时候的位姿。因此解决办法就是在world文件中找到reset参考的位姿进行对应修改。

解决办法：去到对应的world文件，拉到最下面有个pose（world里面有多个model属性，选择最后面的才能找到真正影响reset的pose属性）

1. 加快仿真速度
   1. 关掉Scene中的shadows（gui中关了下次打开还有，直接在world文件中关）

# 机器人导航

## 相关功能包

准备工作

请先安装相关的ROS功能包:

* 安装gmapping 包(用于构建地图):sudo apt install ros-<ROS版本>-gmapping
* 安装地图服务包(用于保存与读取地图):sudo apt install ros-<ROS版本>-map-server
* 安装 navigation 包(用于定位以及路径规划):sudo apt install ros-<ROS版本>-navigation

新建功能包，并导入依赖: gmapping map\_server amcl move\_base

# 其他

| 序号 |  |  |  |
| --- | --- | --- | --- |
| 1 |  |  |  |
| 2 |  |  |  |

表1- 1

图1- 1