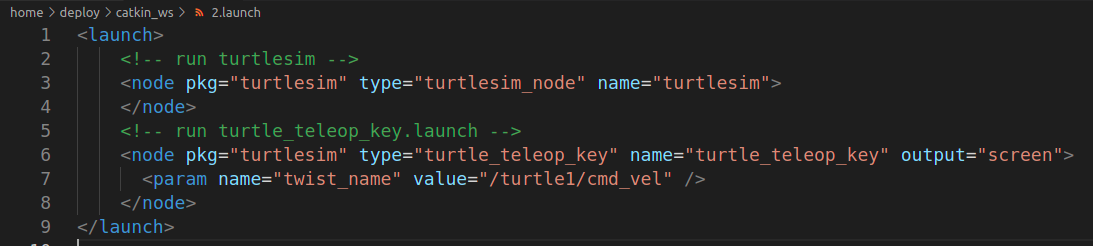
-s<!--- LeGO-LOAM -->

<node pkg="fusion" type="driver" name="driver"output="screen"/>

<node pkg="fusion" type="localization" name="localization" output="screen"/>





此处我们创建了两个分组，并以命名空间（namespace）标签来区分，其中一个名为turtulesim1，另一个名为turtlesim2，两个分组中都有相同的名为sim的turtlesim节点。这样可以让我们同时启动两个turtlesim模拟器，而不会产生命名冲突。

1. 前面我们使用roslaunch启动节点时，并没有先执行roscore。使用roslaunch启动节点时，会自动检测有没有启动roscore，如果启动了，就不再启动，如果没有启动就启动roscore

roslaunch批量创建ros参数;roslaunch 是启动ros节点的第二种方式，其使用xml文件格式，对节点进行一些配置，然后启动节点。

存放launch文件的目录不一定非要命名为launch，事实上都不用非得放在目录中，roslaunch命令会自动查找经过的包并检测可用的启动文件。然而，这种推荐的标准做法被认为是“最佳实践”。

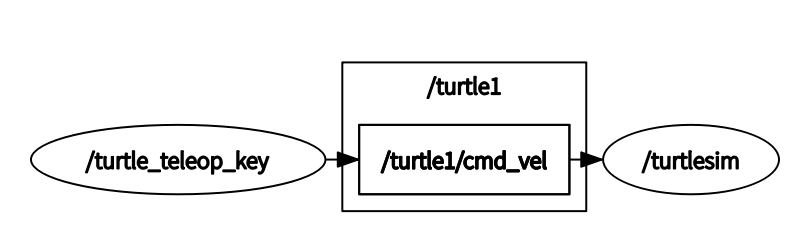
要去运行roslaunch launch turtlemimic.launch 首先



1. //执行roslaunch的终端窗口不断打印log，如果想不在终端打印，可以将output项改为log，或者不加output项 output="screen"就是将对应的log打印到屏幕上。
2. node\_pkg是包名 name代表节点名字

rosrun turtlesim turtlesim\_node

上下一一对应

1. name项是定义节点运行的名字，可以随便修改，然后可以通过rqt\_graph指令，查看修改效果，一般保持和type一致。

<node>标签中有三项必不可少，pkg项指定节点所在的包名，type项指定可执行程序名字，其与CMakeLists.txt中add\_executable()项对应，name项对应节点启动之后的名字，可使用rosnode list查看。respawn是可选，表示节点退出后，是否重新启动，默认为false。output是可选，表示log输出位置，如果是screen表示log输出到屏幕，如果是log表示输出到log文件，默认为log。

6 <param>标签用于创建节点参数，其创建的参数位于节点命名空间，name表示key值，与程序保持一致，注意这里与使用rosrun不同，rosrun中的参数key值是有下划线的，value表示值，type表示参数类型。参数类型可为int，double，bool，str以及yaml，yaml用法后续介绍。

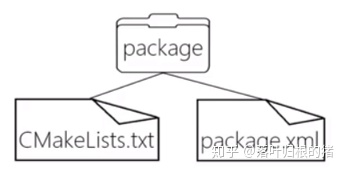
rospack  find  package\_name

判断是否为Package ：

ROS本质上就是有一个个package组成的，package可以说是ROS的细胞。

在catkin\_make的时候它会一个一个的去找package然后生成目标文件。

一个文件夹要被ROS认为是package的话，必须包含以下两个文件：



7 节点属性之节点元素的形式： <node pkg=”package-name” type=”executable-name” name=”node-name”/>在节点标签末尾的斜杠“/”是必须的，但很容易忘。你也可以这样显式地给出结束标签：<node pkg=”…”type=”…” name=”…”></node> 如果该节点有子节点，例如 remap 或者 param 元素，那么该显式结束标签是必不缺少的。还有就是name属性给节点指派了名称，它将覆盖任何通过调用 ros::int来赋予节点的名称。

roslaunch mapping fusion.launch包在哪里？

### 8 tf

如果你的odom计算没有错误，那么map-->odom的tf就是0.

### base\_link坐标系

base\_link是机器人本体坐标系，以底座中心为原点（一般为机器人的旋转中心）,跟随robot一起运动，robot身上的任何零件都是以该坐标系为母参考系。

### odom坐标系

里程计坐标系下的机器人位姿可以作为一个短期的参考，但漂移的存在使其不能作为全局的参考,在odom 坐标系中的机器人的姿态能够保证是连续的，这意味着在odom 坐标系中的移动平台的姿态总是平滑变化，没有跳变。在机器人刚开始运动的时候，odom和map坐标系是重合的。但随着时间的推移是不重合的，而出现的偏差就是里程计的累积误差.

### map坐标系

map坐标系是一个长期的全局定位参考，但在位置估计存在离散跳跃，使其不能作为局部的短期位置参考。map坐标是不连续的，这意味着在map坐标系中移动平台的姿态可以随时发生离散的跳变。

定位模块先接收odom到base\_link的 transform，并使用这个信息发布map到odom的transform。

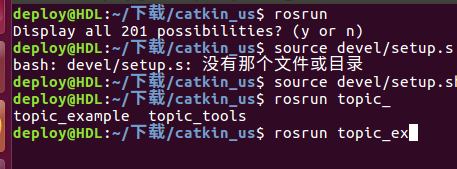
To setup your environment to use the /opt/ros/kinetic installation space, source the setup file for your shell:

source /opt/ros/kinetic/setup.bash

-------------------------------------------------------------------------

必须要首先进行source才能进行人rosrun命令，否则的话，这些命令默认就是source到了source /opt/ros/kinetic/setup.bash这个路径里面

source devel/setup.bash # 刷新坏境

编译完成后，如果有新的目标文件产生（原来没有），那么一般紧跟着要source刷新环境，使得系统能够找到刚才编译生成的ROS可执行文件。这个细节比较容易遗漏，致使后面出现可执行文件无法打开等错误。



每一个package里面都有一个这样的目录结构便于进行编译对应的文档文件

而且这些文件目录的结构都是递归进行的的，每个需要编译的文件编译自己所在目录的cmakelists等等文件，相当于是一个市，很多的市在并交在一起，再去形成一个省，结构都是类似的！

add\_executable(talker src/talker.cpp)

1这里面的src到底是指大文件夹还是小文件夹src呢？

这个完全取决于到底是Cmakelist相对位置是在哪里。

2 msg->data.c\_str());指针指向后面的地址

3 // 初始化std\_msgs::String类型的消息

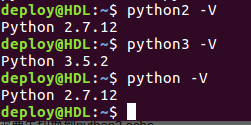
std\_msgs::String msg; 在源文件哪儿？

ROS中还有一个特殊的数据类型：Header，它含有时间戳和ROS中广泛使用的坐标帧信息。在msg文件的第一行经常可以看到Header header。

乌龟的那个算法，之所以不需要source，是因为它默认就是在ubuntu系统的文件夹里面

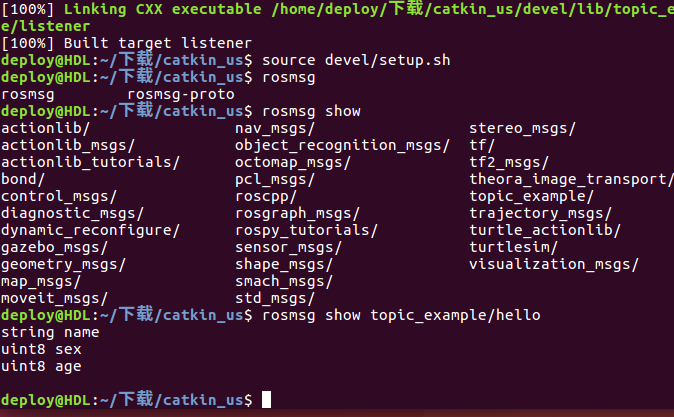
话题的通信是通过节点间发送ROS**消息**实现的。为了使发布者（turtle\_teleop\_key）和订阅者（turtulesim\_node）进行通信，发布者和订阅者必须发送和接收相同**类型**的消息。这意味着话题的**类型**是由发布在它上面消息的**类型**决定的。使用rostopic type命令可以查看发布在话题上的消息的**类型**。

|  |  |
| --- | --- |
| catkincreatepkg | 创建功能包 |
| catkin\_make | | 编译工作空间中的功能包 |
| roscd | | 功能包目录跳转 |
| rosed | | 编辑功能包中的文件 |
| rosrun | | 运行功能包中的可执行文件 |
| roslaunch | | 运行启动文件 |



针对于下面rosmsg show 出来的内容

当里面出现的topic\_example/ 右斜杠表示文件夹，子目录还有文件可选取



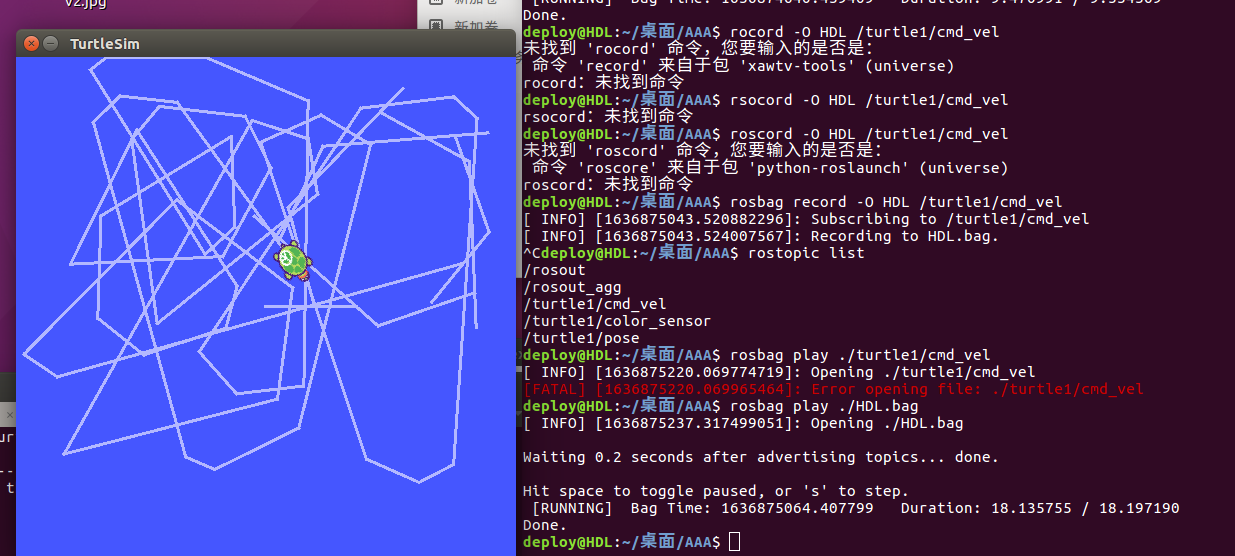
HAD里面也有对应的ROS程序，里面跑着匹配的算法其实也是可以跑得通的。

针对ROS里面的PCL库，需要重新进行安装才能打开对应的点云图片，但实际上播包时，能看整个全过程中的情况。

不过这个环境变量是基于shell的，也就是说只有在当前设置了的shell里才会有作用，因此每次打开新的shell运行程序不停地设置LD\_LIBRARY\_PATH，

是一件非常麻烦的事情，所以我们就会想有不有什么一劳永逸地方法，使得设置之后就不用再去设置了？答案是肯定的，

需要回应的是：到底什么是shell？



rosbag play 的时候，整个系统会对应的进行运动，Apollo系统那样，整个软件开发的过程中，回放数据包，整个软件会自动运行。

rosbag play all.bag --topics /chatter 播放指定的功能包

录制bag时候，topic需要手打上去

deploy@HDL:~/桌面/时间戳函数/catkin\_us$ rosbag record -O HDL /chatter

[ INFO] [1637205523.964299121]: Subscribing to /chatter

[ INFO] [1637205523.967523877]: Recording to HDL.bag.

录制bag时候，node也可以（？可以么）进行录制，也还是需要手打上去。

deploy@HDL:~/桌面/时间戳函数/catkin\_us$ rosbag record -O HDL /talker

[ INFO] [1637205581.708680259]: Subscribing to /talker

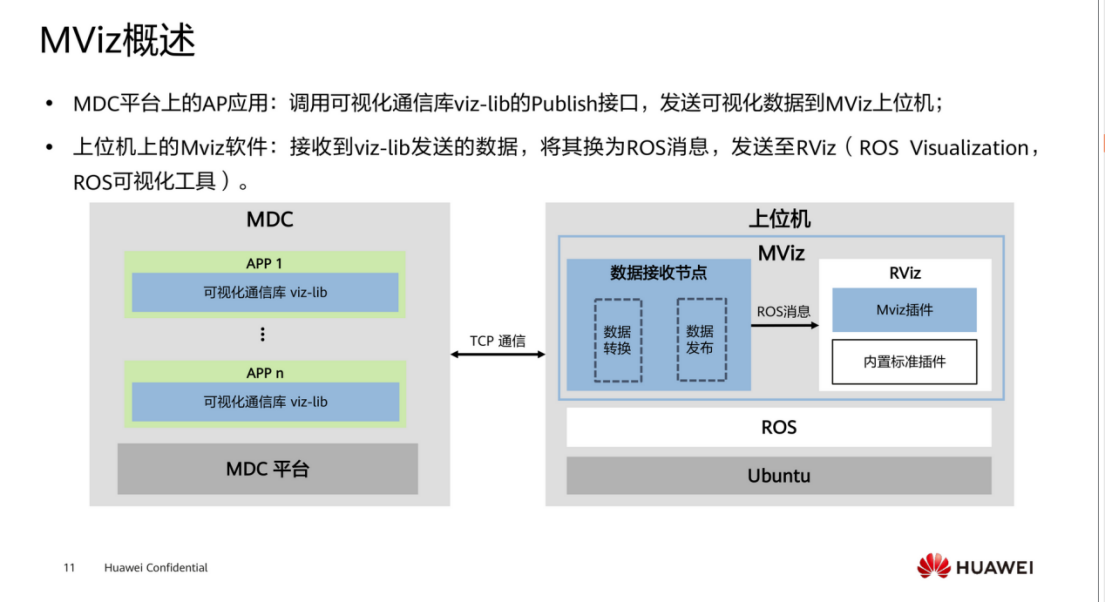
[ INFO] [1637205581.711566639]: Recording to HDL.bag.

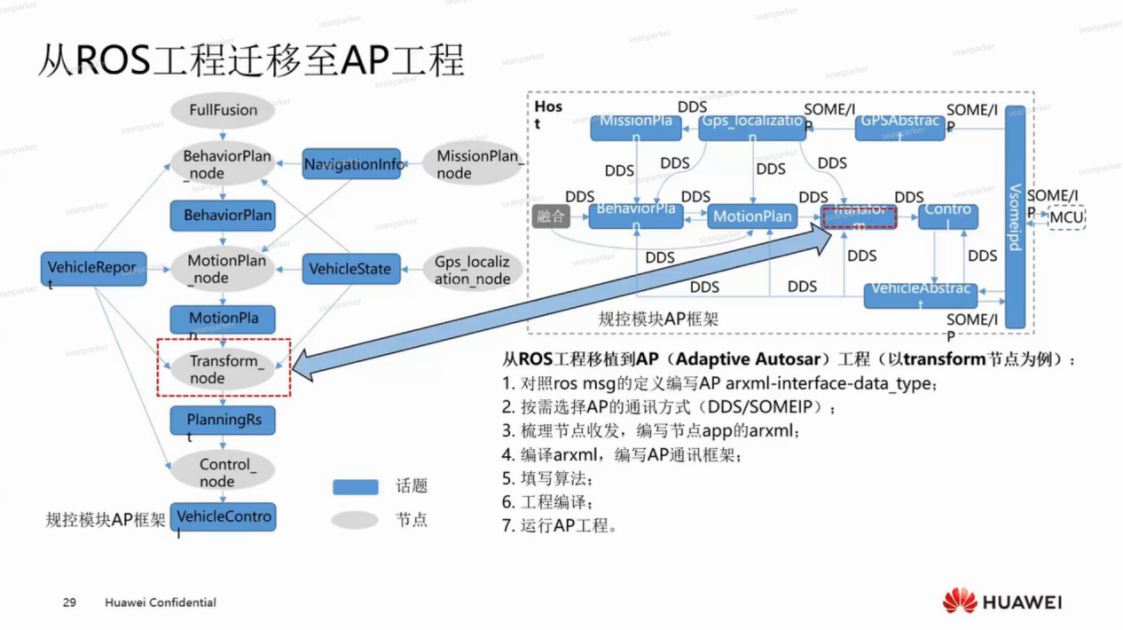
Bag Time: 1637205922.303406 是指整个包录制时候的时间，无论如何改变，整个bag time是不会变化的。

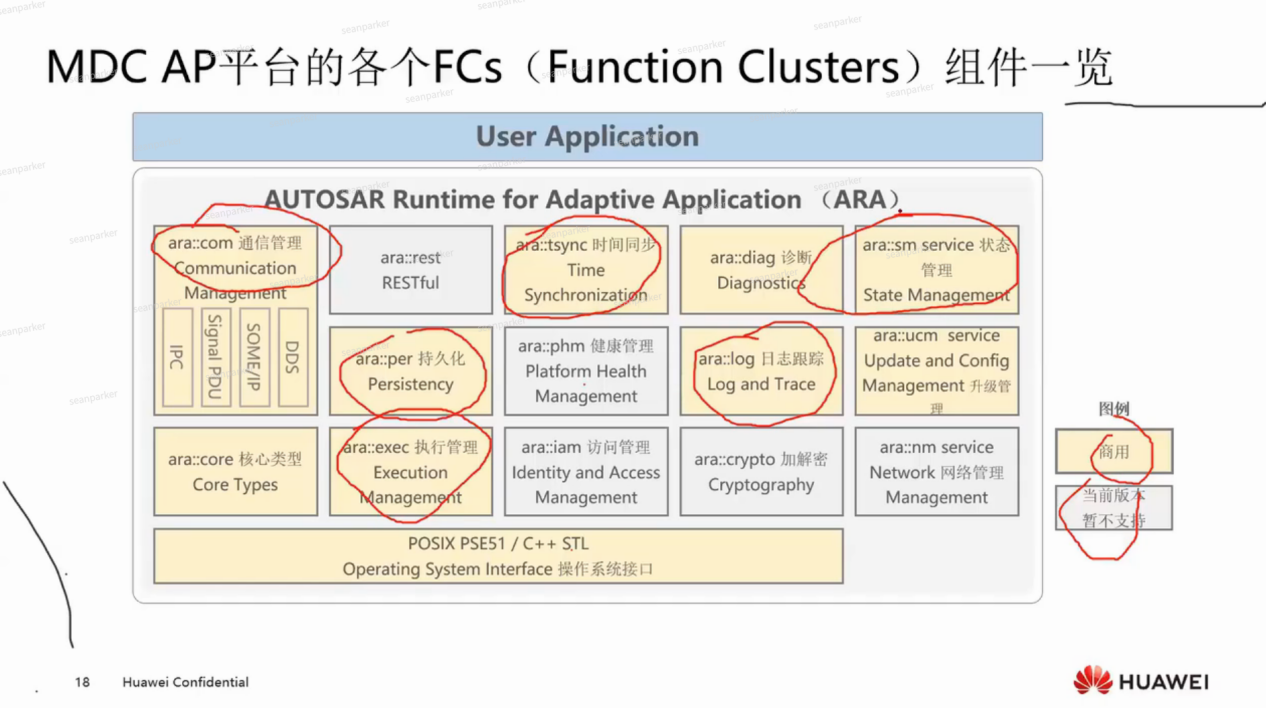
以上展示了，如果只是录制子的topic，应该进行的步骤和方法。

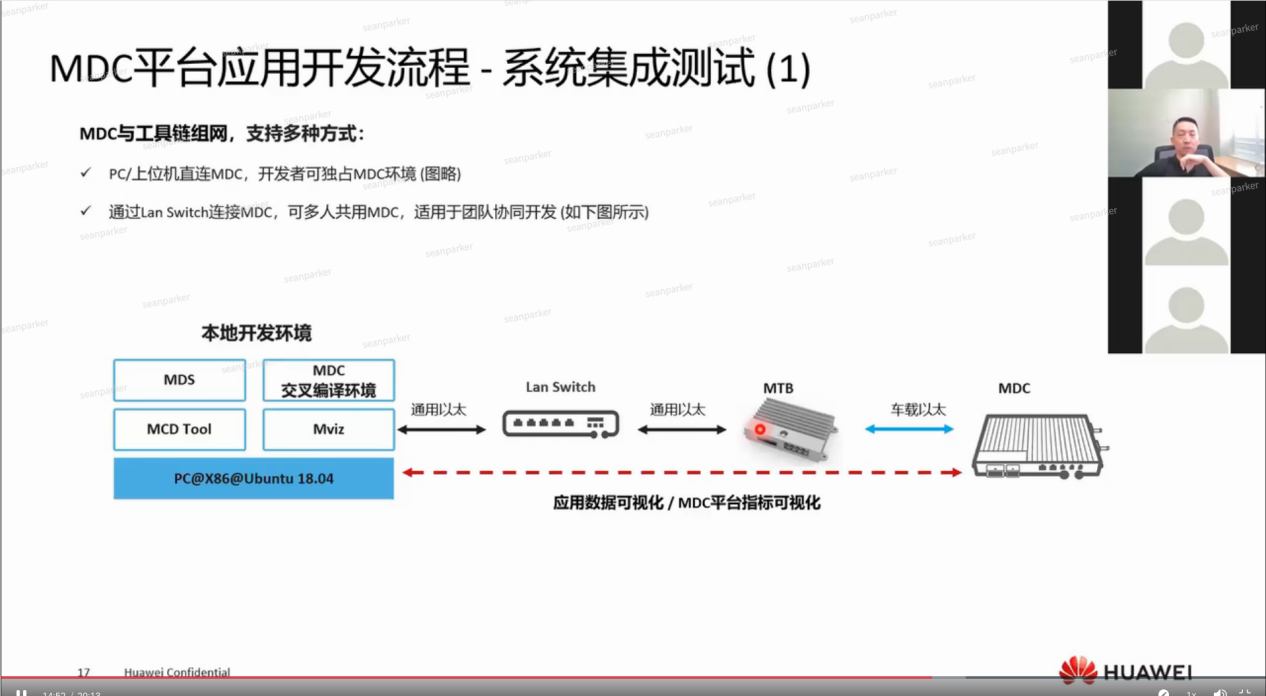
其中需要注意的是：如果进行播放子文件包必须在包名称前面加上./

其余需要强调的地方，都在上述截图中有所体现。











rm -rf mdc2had\_ws/

直接删除某个文件夹，但是不用逐一确认信息，直接删除即可。

rm -rf \*.zip 删除对应的压缩文件夹

rm -rf \*.sh

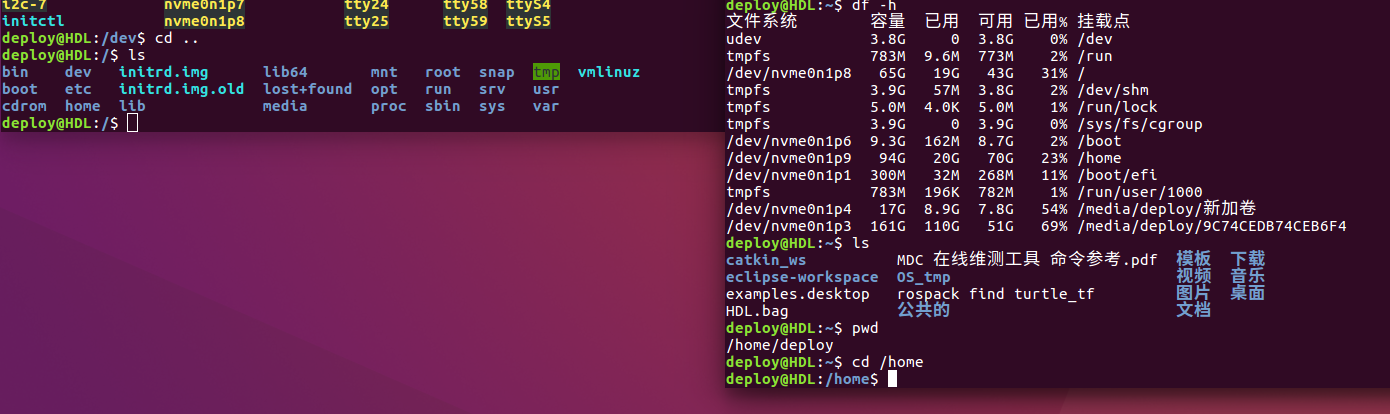
scp -r sshuser@192.168.1.6:/home/mdc/localization/ ./

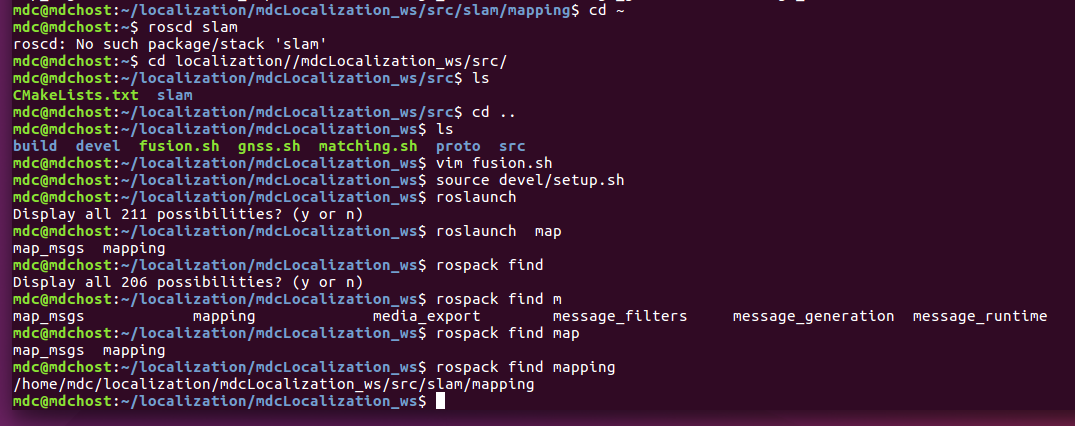
deploy@HDL:~$ scp -r code@192.168.1.192:/home/code/Pictures/ ./

将远处的文件传递到本地的文件夹下面。./表示的是传递到本地所在的文件夹下面

没有./ 的话，默认就是到了home文件夹的目录里面

linux查看根目录下所有文件夹大小的方法如下：  
1、进入根目录：cd /  
2、使用命令 ： du -sh \* 查看根目录下每个文件夹的大小  
3、进入占用空间比较大的文件夹，然后再使用2中命令查找大文件。



挂载点指挂到对应根目录位置（根目录不一定全用上），到根目录指令是cd/

关于上面的情况需要说明的是：rospack find +包名 输出这个命令后得到的是包名字所在的目录

找到对应的包名字以后，roslaunch对应的包名字 对应的launch文件

1关于在ros里，一定要source devel/setup.sh

这样就会将即将运行的代码铺在了ros里面默认的程序小乌龟上面了

2 关于程序，如果以n开头就是虚拟节点注意一下，这时候要下电，重新上电进行

3

process[rosout-1]: started with pid [3751]

started core service [/rosout]

terminate called after throwing an instance of ‘std::runtime\_error’

what(): locale::facet::\_S\_create\_c\_locale name not valid

[rosout-1] process has died [pid 3751, exit code -6, cmd /opt/ros/jade/lib/rosout/rosout \_\_name:=rosout \_\_log:=/home/odroid/.ros/log/9cdba11a-3419-11e1-a50e-36e66a0e97b1/rosout-1.log].

log file: /home/odroid/.ros/log/9cdba11a-3419-11e1-a50e-36e66a0e97b1/rosout-1\*.log

export LC\_ALL="C"

最好把它写在.bashrc文件中，这样省的每次都要输入这个命令。 vim ~/.bashrc

4

执行对应的代码

方法一：所以每一次将.sh文件打开时候，第一行出现source devel/setup.sh就是要将对应文件起始点位置进行了重新source里面的内容.也就是将对应动态库加进去

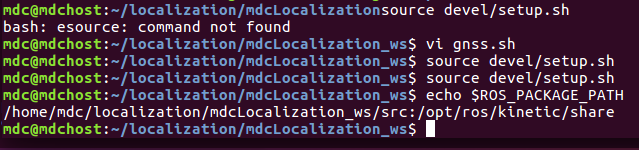
cd /home/mdc/localization/Work\_E115\_locate\_gps\_v1/Debug/bin

export LD\_LIBRARY\_PATH=/home/mdc/localization/Work\_E115\_locate\_gps\_v1/Debug/lib:$LD\_LIBRARY\_PATH

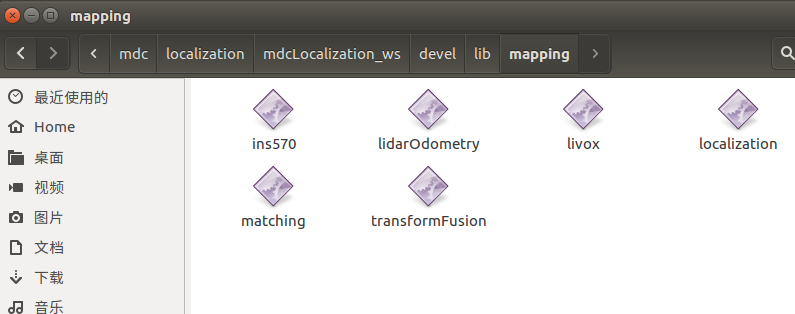
./Work\_E115\_locate\_gps\_v1

方法二：就是首先source一下，然后对应的节点就会发布出来了，但是在source之后为什么会出现那么多的，我理解是，在编译好文档内容的可执行文件的基础上进行的。

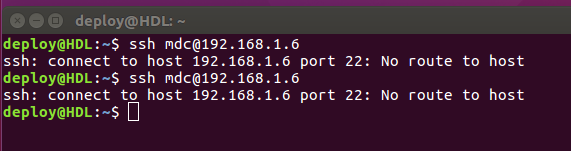
关于环境变量的设置，以及环境变量的查询



针对与到lib之前是常见的目录：



关于host



在mdc的目录下面往本地计算机方向进行传递，传递到了home的文件夹下面

scp -r 2021.bag [deploy@192.168.1.168](mailto:deploy@192.168.1.168):

[deploy@HDL:~$scp-rlivoxv2\_ws.tar.gz](mailto:deploy@HDL:~$scp-rlivoxv2_ws.tar.gz) mdc@192.168.1.6:/home/mdc

mdc@192.168.1.6's password:

livoxv2\_ws.tar.gz 100% 22MB 21.7MB/s 00:01

以上是处于deploy的目录里面往mdc里面进行传递

ps aux最初用到Unix Style中，用来查看系统进程的命令

而ps命令（Process Status）就是最基本同时也是非常强大的进程查看命令.

使用该命令 可以确定有哪些进程正在运行和运行的状态、进程是否结束、进程有没有僵尸、哪些进程占用了过多的资源等等.总之大部分信息都是可以通过执行该命令得到的.

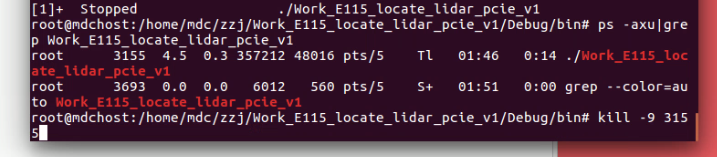
ps是显示当前状态处于running的进程，grep表示在这些里搜索，而ps aux是显示所有进程和其状态。

$ ps aux | grep amoeba

查到amoeba的进程

$ kill -s 9 pid 杀死进程

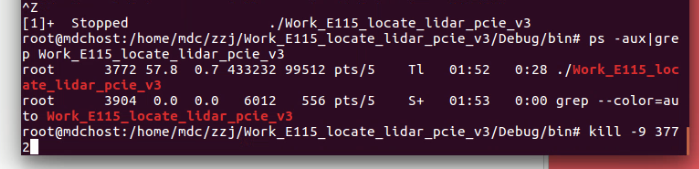
先对V1进行kill



再对V3进行kill

kill - 9 表示强制杀死该进程;而 kill 则有局限性,例如后台进程,守护进程等

韩用这个命令，将执行了的可执行文件进行了kill

每当发布一个node name的时候，会自带将对应的几个topic进行启动，比如说livox。当把它启动的时候，会自动带着其他的几个topic进行启动。那六个节点需要了解一下！

录制数据到底是录制什么，topic还是node？

答案是topic，因为一个程序有很多的topic，当我们只需要记录一个topic的时候，或者几个topic的时候。就需要在下面的程序中进行录制，-O subset是录制之后程序名字的事情。

rosbag record -O subset /turtle1/cmd\_vel /turtle1/pose

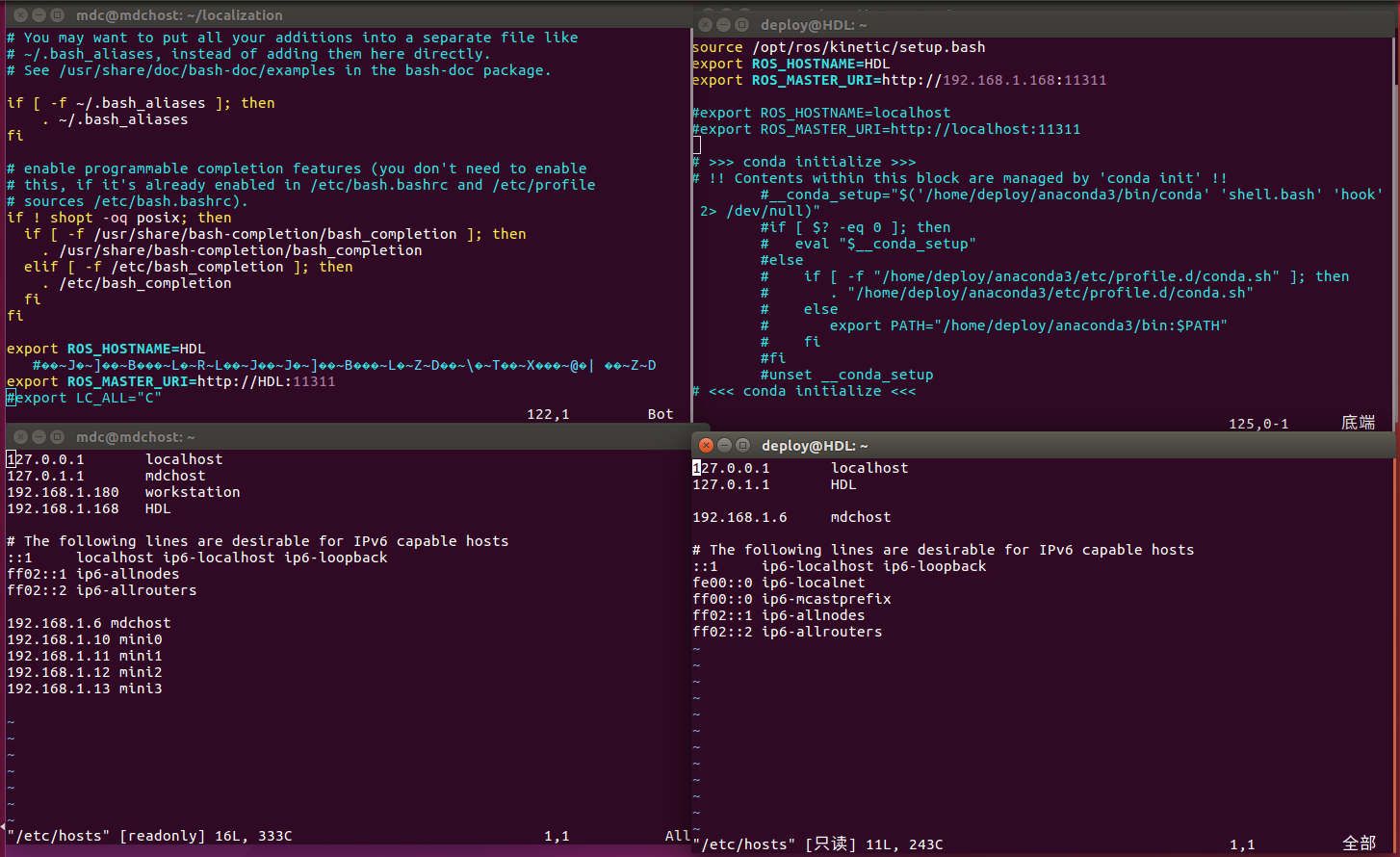
rostopic echo [topic] 查看某个话题发布的消息

rosrun pcl\_ros bag\_to\_pcd 2021-11-17-01-49-00.bag cloud ./

一个命令行就可以完成.bag 的数据包转换成为pcd的格式

关于MDC的原始主从机的配置模样,就是在下面的配置中什么都没有!

每经过更改配置以后,需要进行的是,将当前对话框关闭,然后重新打开新的配置



ROS\_HOSTNAME分别对应主从机的hostname，ROS\_MASTER\_URI全部对应为主机的ip，需替换成自己的。所以上图从机的hostname应改为mdchost

<https://blog.csdn.net/night___raid/article/details/106243017>

https://blog.csdn.net/qq\_43481884/article/details/108895044

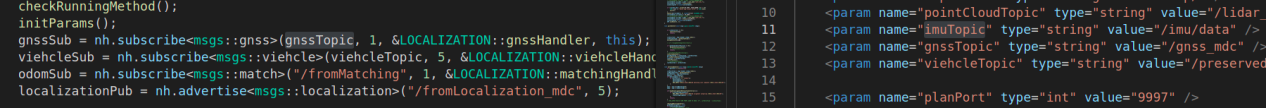
针对上图mdchost用户名里面出现的HDL,是一种对192.168.1.123的简写

针对topic的名字,首先软件文件夹先是经过了编译,然后是生成对应对应的可执行文件,很关键!gnss.sh的文件见下面所示

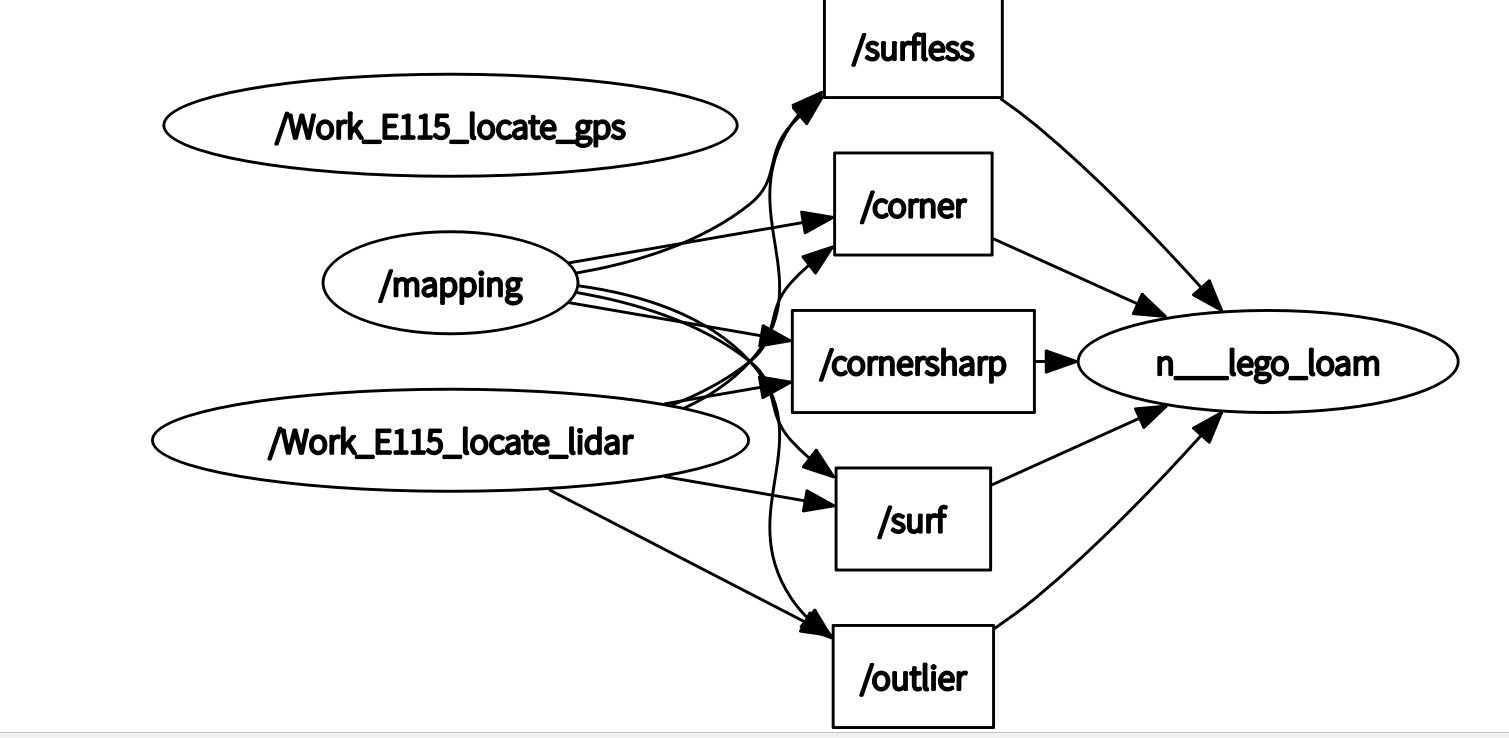
source devel/setup.sh

roslaunch mapping gnss\_mdc.launch

当执行下面的gnss\_mdc.launch文件时候,左侧的话题名称gnssTopic自动就会变成了右侧的/gnss\_mdc



下图左侧/mapping是livox节点发送的,右侧是lidarOdometry



rm -rf \* 将所在的文件夹下面的文件全部进行删除!

在源代码中修改参数每次都需要重新编译才能运行，费时费力，我们可以利用ROS的参数服务器，并且把修要修改的参数写在launch文件中，这样每次修改launch文件后Ctrl+S，就可以马上运行看看效果。

launch文件为了调试而生的存在!

cmake -DCMAKE\_BUILD\_TYPE=None -DCMAKE\_INSTALL\_PREFIX=/usr -DBUILD\_GPU=ON -DBUILD\_apps=ON -DBUILD\_examples=ON -DCMAKE\_INSTALL\_PREFIX=/usr ..

sudo apt-get install ros-kinetic-desktop-full

安装ROS kinetic桌面完整版

sudo apt-get install ros-kinetic-PACKAGE 安装ROS kinetic独立功能包

sudo apt-get install ros-(ROS version name)-cv-bridge

sudo apt-get install ros-kinetic-pcl-ros

sudo apt-get install ros-(ROS version name)-vision-opencv

pkg-config --modversion opencv

查看opencv版本查看

