#### 安装evo工具参考的博文

<https://github.com/MichaelGrupp/evo>

<HTTP://blog.csdn.net/langshibing/article/details/90257659?utm_medium=distribute.pc_relevant_t0.none-task-blog-BlogCommendFromMachineLearnPai2-1.edu_weight&depth_1-utm_source=distribute.pc_relevant_t0.none-task-blog-BlogCommendFromMachineLearnPai2-1.edu_weight>

<https://blog.csdn.net/weixin_44386661/article/details/103080551?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-2.edu_weight&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-2.edu_weight>

<https://blog.csdn.net/anchuanxu/article/details/103385483>

**Evo\_traj tum your\_pose --ref=groundtruth -p -asv**

**Evo\_traj euroc pose -p**

**evo\_traj euroc posefile --save\_as\_tum**

**Evo\_ape groundtruth yourpose -p -asv --save\_results resultfilename**

#### Diff命令

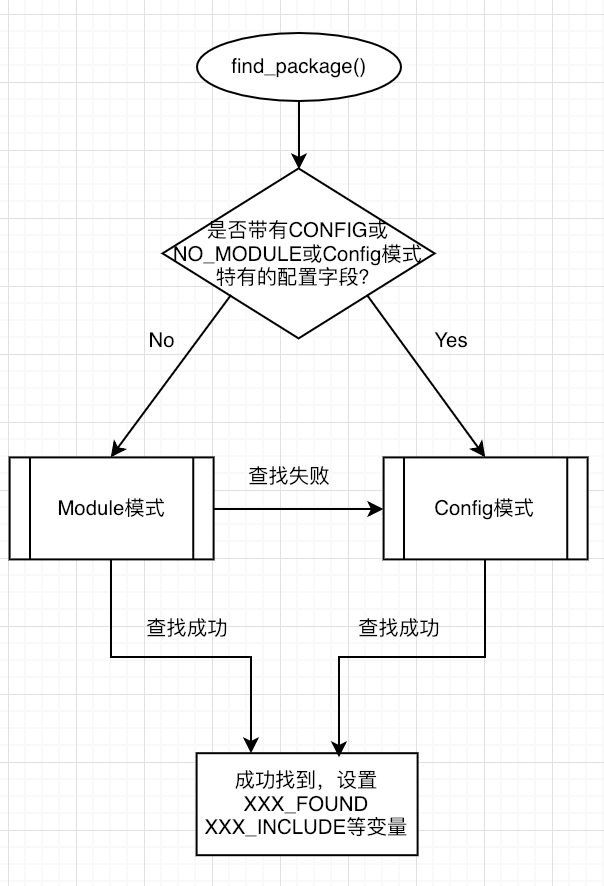
**<http://www.ruanyifeng.com/blog/2012/08/how_to_read_diff.html>**

#### Pangolin学习

<https://www.cnblogs.com/shhu1993/p/6814714.html>

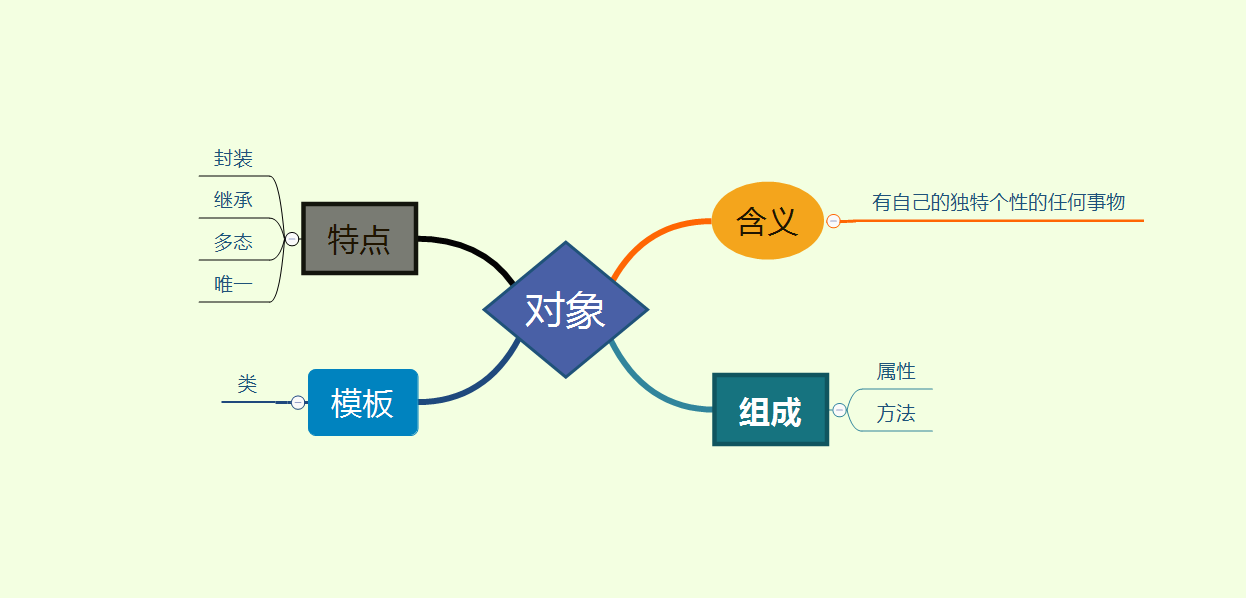
<https://mp.weixin.qq.com/s?__biz=MzI5MTM1MTQwMw==&mid=2247485156&idx=1&sn=7109ee61d663d21832b94534b6863db4&chksm=ec10b8e0db6731f62490664d3ef02491e93371d99bda926b116dc216b1411789ed78aba2a166&mpshare=1&scene=1&srcid=0505TTqkFeTOzqAV63lggaE9&pass_ticket=b%2FXHS6vjWLb%2BKGKqEBCjh7chAnhbsnjE7MJpLEZNBJmpKfemcHXmJL%2BIZ5pRwYeD#rd>

#### Find\_package



#### 面向对象与面向过程的本质的区别

<https://blog.csdn.net/jerry11112/article/details/79027834>



#### c++中default、explicit、override、final、volatile关键字

<https://blog.csdn.net/u011947630/article/details/103062773>

#### c++11多**线程**---线程锁（mutex）

<https://www.cnblogs.com/lovebay/p/11582682.html>

#### ubuntu16.**04** 检测不到扩展屏幕（解决方案）

<https://blog.csdn.net/weixin_42907473/article/details/102922294>

#### Json解析生成库

41种json库的测评

<https://github.com/miloyip/nativejson-benchmark>

Rapidjson性能最好，但似乎比较难用

试试CjsonOBject，简单易用

<https://github.com/Bwar/CJsonObject>

Json、xml、yaml文件

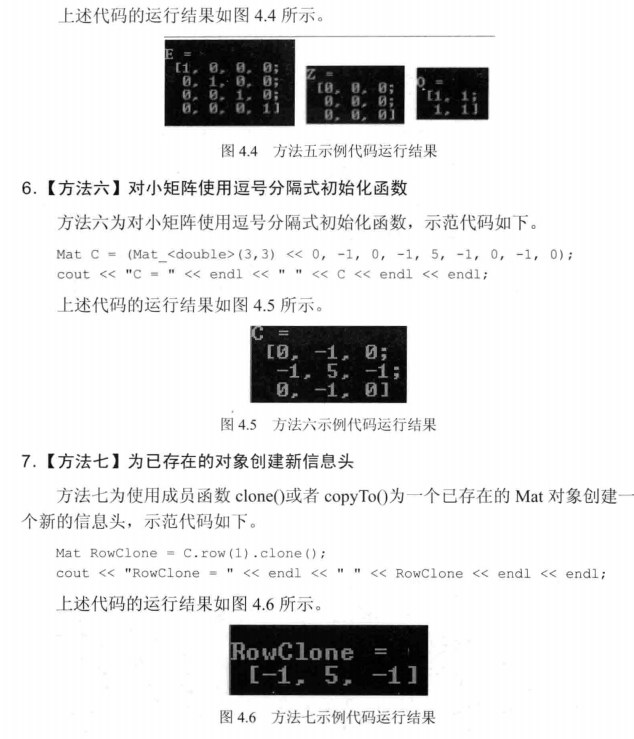
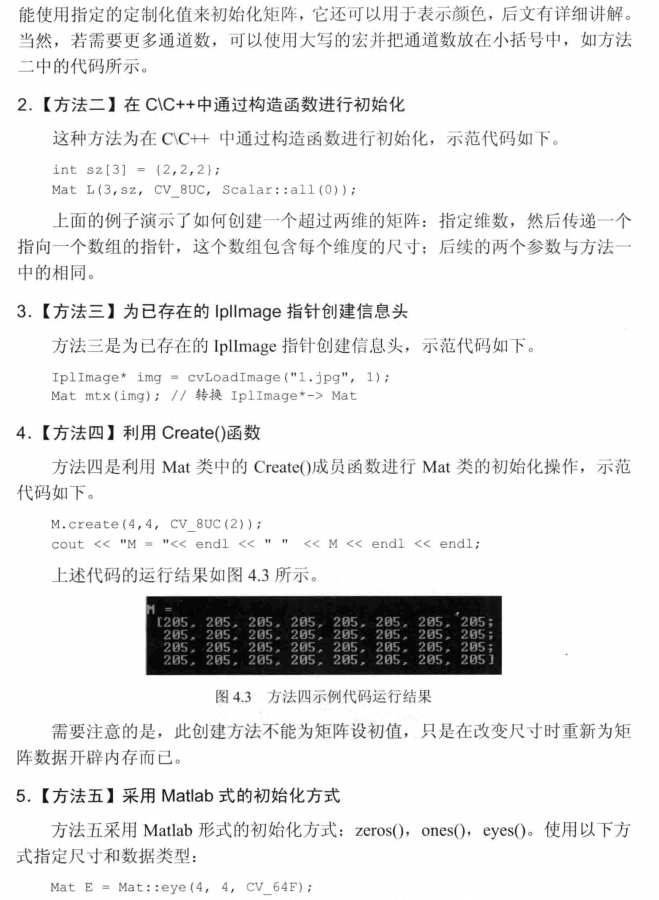
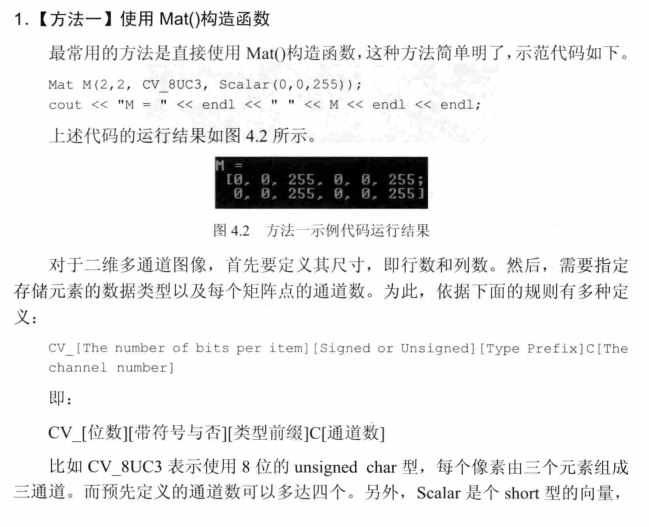
YAML是一个可读性高，用来表达数据序列化的格式。

<https://www.runoob.com/w3cnote/yaml-intro.html>



#### **Opencv**

<http://www.woshicver.com/>



Mat的访问方法主要有，利用m.at<uchar/Vecsb3b son on>(i,j)

M.ptr<>(i)[j]

指针\*pData pData+i

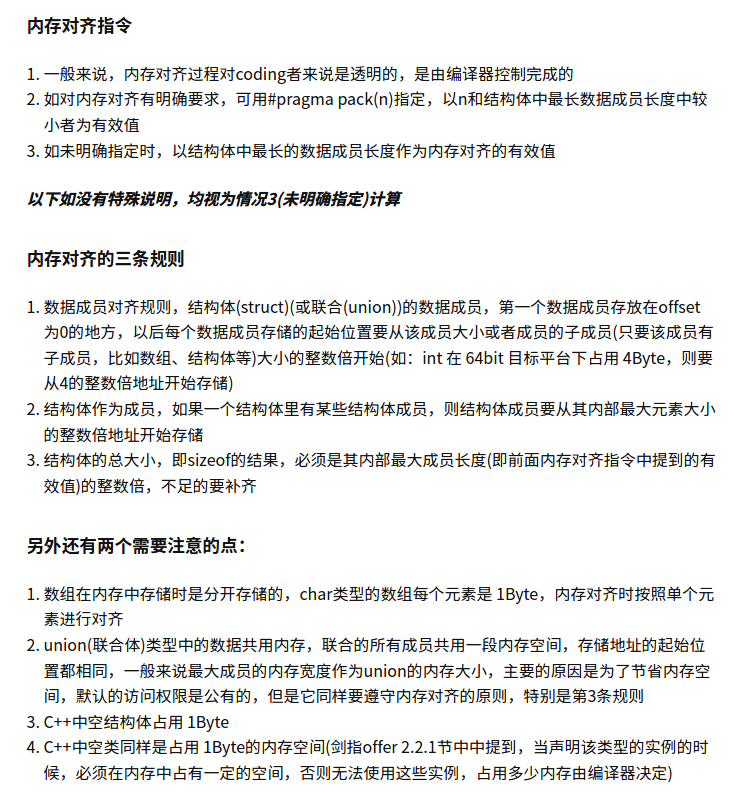
#### C++序列与反序列化

<https://www.cnblogs.com/Malphite/p/10818954.html>

Boost serialization

<https://www.cnblogs.com/cmranger/p/4772149.html>

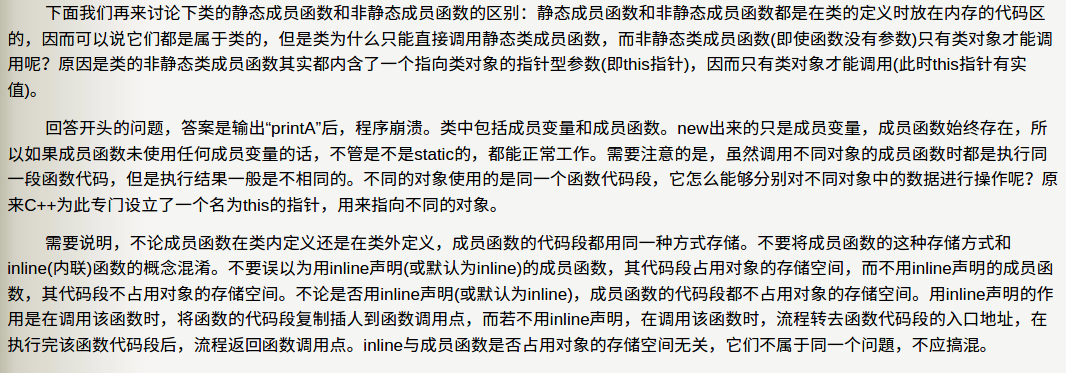
#### 内存对齐

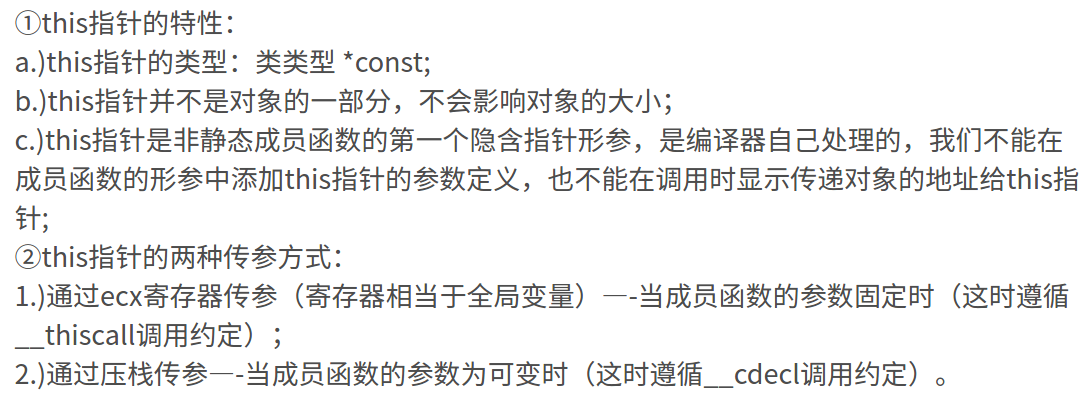


类的内存存储（成员变量和成员函数代码）虚函数指针虚基类指针

<https://zhuanlan.zhihu.com/p/103384358>

<https://www.cnblogs.com/rednodel/p/9300729.html>





ram内存布局

<https://zhuanlan.zhihu.com/p/94507500>

几个库在cmakelist中的查找名称

1.find\_package(Pangolin REQUIRED)${Pangolin\_INCLUDE\_DIRS} ${Pangolin\_LIBRARIES}

2.find\_package(OpenCV REQUIRED) ${OpenCV\_INCLUDE\_DIRS} ${OpenCV\_LIBS}

3.Boost\_FOUND - 如果找到了所需的库就设为true

Boost\_INCLUDE\_DIRS - Boost头文件搜索路径

Boost\_LIBRARY\_DIRS - Boost库的链接路径

Boost\_LIBRARIES - Boost库名，用于链接到目标程序

Boost\_VERSION - 从boost/version.hpp文件获取的版本号

Boost\_LIB\_VERSION - 某个库的版本

4.

#### Memset（）函数



--

#### C语言的htons和htonl函数、大尾端、小尾端

<https://www.cnblogs.com/sddai/p/5790479.html>



Socket编程

Sys/socket.h

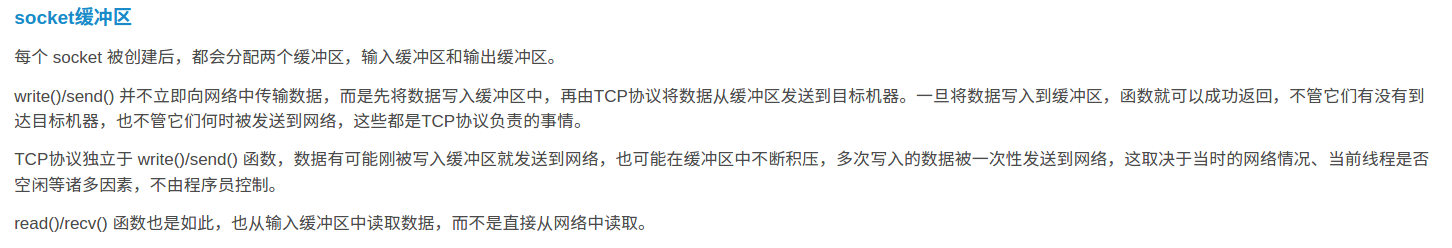
Netinet/in.h

<https://blog.csdn.net/godop/article/details/79894079>

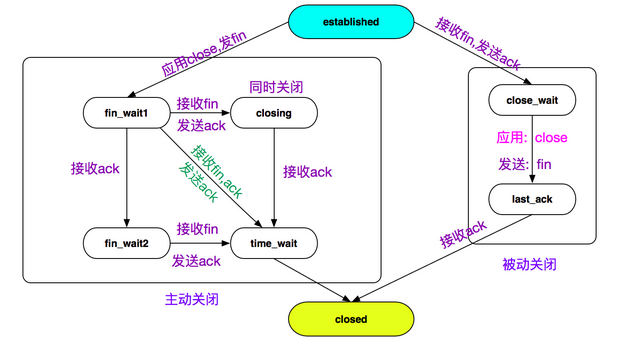
<https://www.cnblogs.com/felixzh/p/8359066.html>

socket函数send和recv函数

<https://blog.csdn.net/gaoshou_zgt/article/details/78611448?utm_medium=distribute.pc_aggpage_search_result.none-task-blog-2~all~sobaiduend~default-3-78611448.nonecase&utm_term=send%E5%87%BD%E6%95%B0%E6%96%AD%E5%BC%80%20socket>



<https://www.cnblogs.com/liedElxa/p/10795398.html>



<https://my.oschina.net/alchemystar/blog/1821680>

<https://blog.csdn.net/pashanhu6402/article/details/96428887>

Linux网卡命名enp3s0说明

<https://blog.csdn.net/weixin_34123613/article/details/89390993?utm_medium=distribute.pc_relevant_t0.none-task-blog-BlogCommendFromMachineLearnPai2-1.edu_weight&depth_1-utm_source=distribute.pc_relevant_t0.none-task-blog-BlogCommendFromMachineLearnPai2-1.edu_weight>



算法时间复杂度分析

<https://www.cnblogs.com/jack1995/p/12152833.html>

<https://ethsonliu.com/2018/04/algorithm-complexity.html>

#### Github 简明教程

<https://www.runoob.com/w3cnote/git-guide.html>

#### Terminator的安装和配置(颜色和字体)

<https://blog.csdn.net/zhangkzz/article/details/90524066>

| **键盘命令** | **说明** |
| --- | --- |
| Ctrl+Shift+T | 新建一个terminator窗口 |
| Ctrl+Shift+E | 对terminator窗口垂直分割 |
| Ctrl+Shift+O | 对terminator窗口水平分割 |
| Ctrl+Shift+W | 关闭当前的terminator窗口 |
| Ctrl+Shift+I | 另外创建一个新的terminator窗口 |
| Ctrl+Shift+N | 在分割的不同窗口之间切换（向后） |
| Ctrl+Shift+P | 在分割的不同窗口之间切换（向前） |
| Ctrl+Shift+T | 新建一个Tab窗口 |
| Ctrl+Tab | 切换下一个窗口 |
| Ctrl+Shift+T | 新建一个terminator窗口 |
| Ctrl+Shift+F | 当前terminator窗口进行搜索 |
| Ctrl+Shift+X | 当前分割的窗口最大化 |
| Alt+A | 将所有分割terminator同步 |
| Alt+O | 关闭分割terminator同步 |

#### 关于友元函数在模板类中使用出现的问题的解决办法及模板的机制

<https://blog.csdn.net/huang1600301017/article/details/90177970>



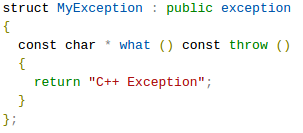
#### g++ 常用命令选项

#### Std::exception、try、catch、throw，自定义exception

<https://www.runoob.com/cplusplus/cpp-exceptions-handling.html>

自定义继承std::exception有两种方法：

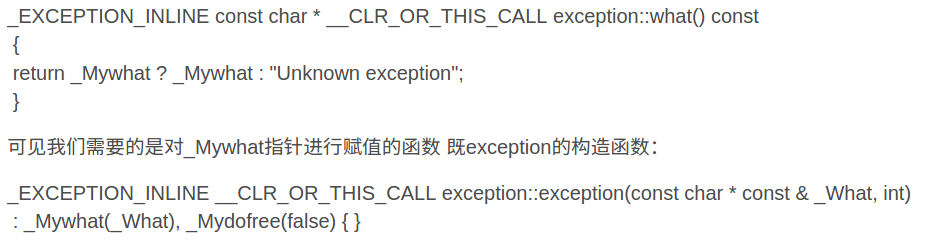
1.重载what（）函数



1. 传参父类构造函数



<https://blog.csdn.net/diffjd/article/details/78555019>



#### Rviz学习

1. 解析ros::init

<https://www.dazhuanlan.com/2019/12/14/5df3c1ef11183/?__cf_chl_jschl_tk__=4d36894f78b5cc0f9438cfb6beef58bf4638b2dd-1601261195-0-AfsjFnCLN7NhgDhVT6CChazwnYzgE4ab4ooWgmS3r_IC11hm8Dorlg_fL5pCF3cu93zq9tN4j77Iwb7Piy8dOfws2W4yPLDXS59o4iZ7cnO00mhG28RPharBU5TibUM-4LjQkVTKgwSGVlxHbn94Of1GfTOyOjDElzjqMWUg-OQJJ1xGuycYVEm_WmUWtHBdEM_UwmJTwbOJs8Tm6LEVWN3dwjSGPKdMazseEoqYYSEPztDiuRY6vDfAgsCNU2WU5-f8jw_yP6iJAHg8u6oD5G2Cat64oP0z2rdOvd3WKbOfBr17H7RMdO2kOhkylo1nxw>

<https://blog.csdn.net/wanghuiquan0712/article/details/78014232>

1. ROS Nodehandle句柄

<https://blog.csdn.net/jack_20/article/details/70746736>

ros::init(argc, argv, "node\_name"); // node name

ros::NodeHandle n; //n 命名空间为/node\_namespace

ros::NodeHandle n1("sub"); // n1命名空间为/node\_namespace/sub

ros::NodeHandle n2(n1,"sub2");// n2命名空间为/node\_namespace/sub/sub2

ros::NodeHandle pn1("~"); //pn1 命名空间为/node\_namespace/node\_name

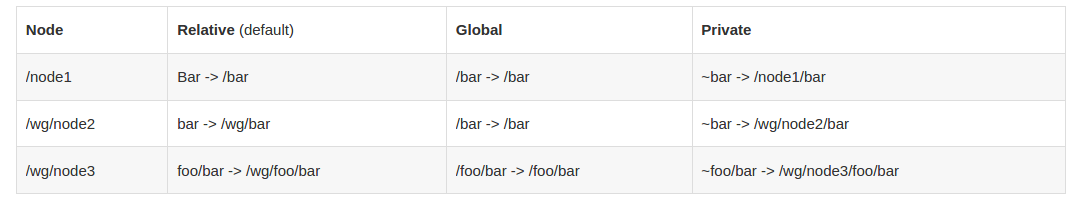
ros::NodeHandle pn2("~sub"); //pn2 命名空间为/node\_namespace/node\_name/sub

ros::NodeHandle pn3("~/sub"); //pn3 命名空间为/node\_namespace/node\_name/sub

ros::NodeHandle gn("/global"); // gn 命名空间为/global



1. 命名空间



Rviz：display 、panel、tool

#### Pluginlib

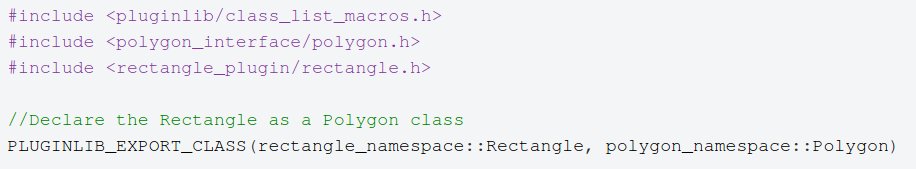
Pluginlib是一个用于ROS包来实现对插件的动态装载和卸载的c++库。这里插件的定义就是一种可以动态加载的可装载类库，能够即插即用，不用事先像传统的c++程序那样要include和声明之类的。

插件是基于一个基类的，继承于一个基类，这个基类就是一个多态接口，用于调用插件。

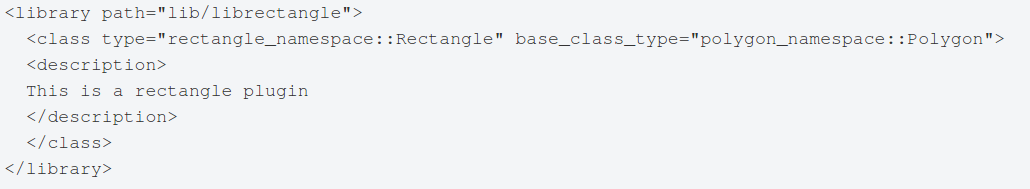
该接口基类以及其继承类不允许有带参数的构造函数。

实现方法：

1.将准备作为插件的类标记成为export类：在所有构成该plugin库的源文件中添加宏PLUGINLIB\_EXPORT\_CLASS;如下



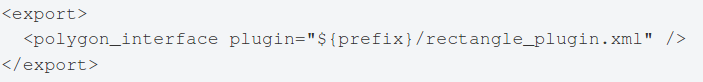
2.建立一个描述插件的xml格式文件（名字自拟），里面包含插件库的信息，插件的名字，插件的类型等（注意这里lib/librectangle的lib是devel下的lib，librectangle后面不用加.so）



<http://wiki.ros.org/pluginlib/PluginDescriptionFile>

这个额外的文件的以及PLUGINLIB\_EXPORT\_CLASS宏使得ROS系统能够加载、发现和推断该插件。该文件还提供了CLASS宏功能以外的一些信息。

3.将plugin注册到ROS系统，以使得pluginlib能够在ros包里找到.在package.xml文件里加入export标签



如果插件借口基类在别的ros包里，加入下面两行，其实ros依赖其它包都是这么写的。



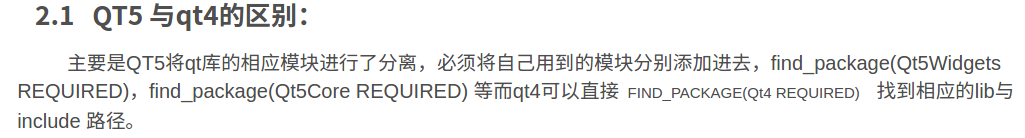
上面三步就完成了注册，接下来看怎么使用：

1.查询某一接口的plugin，这里是nav\_core



2.在使用插件的源文件中包含pluginlib/class\_loader.h，使用class\_loader类。

#### cmake编译QT5

<https://blog.csdn.net/zyh821351004/article/details/46632281>

#### C++RAII机制

RAII就是像shared\_ptr、lock\_guard这样的资源管理方式，在资源的生命周期内使用，生命周期结束后自动销毁。

<https://blog.csdn.net/quinta_2018_01_09/article/details/93638251>

#### 正则表达式

正则表达式由表达式、量词和断言组成；

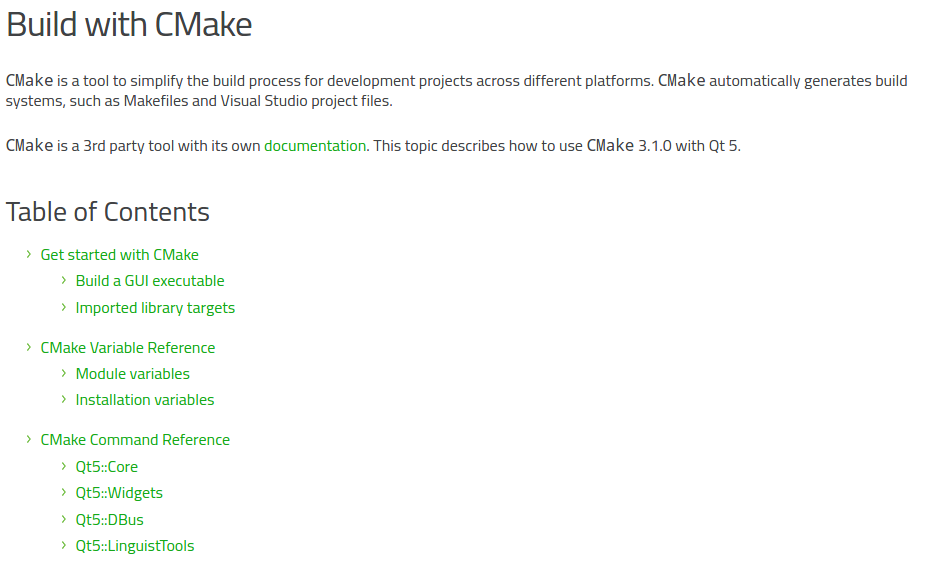
表达式：单个字符、字符集[],如a、[avf\_],连续的用-表示[a-z]也代表匹配单个字符

量词：

<https://www.runoob.com/regexp/regexp-syntax.html>

#### Cmake编译qt

<https://doc.qt.io/qt-5/cmake-get-started.html>



主要是添加对QT工程文件的编译支持：因为qt工程文件与cmake不同，编译器也不同，因而需要添加支持。

1. 对.h文件调用预编译器moc QT5\_WRAP\_CPP( )
2. 对.ui文件调用uic处理 QT5\_WRAP\_UI()
3. 对资源文件.qrc调用rcc处理QT5\_ADD\_RESOURCE()
4. #这些生成的中间文件都会在build目录下，这样的话，编译器则不能定位由uic程序产生的诸如\_ui\_mainwindow.h等文件。所以，我们需要把build目录添加到包含目录中

INCLUDE\_DIRECTORIES(${CMAKE\_CURRENT\_BINARY\_DIR})

这个好像在cmake 3.7以上就不需要

1. 最后和源文件添加到目标对象中

Moc可以设置SET (CMAKE\_AUTOMOC ON)来自动开启

同样

set(CMAKE\_AUTORCC ON)

set(CMAKE\_AUTOUIC ON)

#### 为什么C/C++要分为头文件和源文件？

<https://www.zhihu.com/question/280665935?sort=created>

#### cmake用法及常用命令总结（全）

<https://www.cnblogs.com/ZY-Dream/p/11232779.html>

#### cmake使用教程（一）-起步

<https://juejin.im/post/6844903557183832078>\

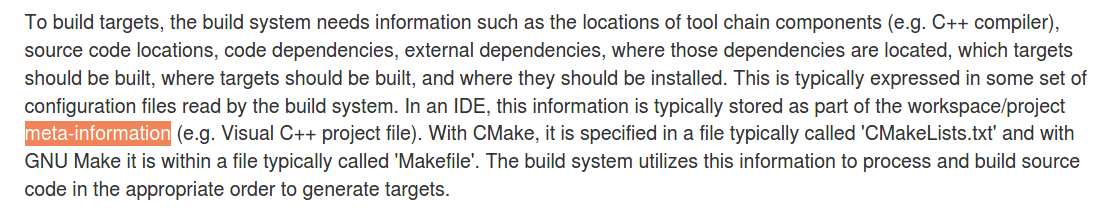
cmake学习之-configure\_file

<https://www.cnblogs.com/gaox97329498/p/10952732.html>

#### cmake的add\_custom\_command和add\_custom\_target指令

<https://blog.csdn.net/qq_38410730/article/details/102797448?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.edu_weight&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.edu_weight>

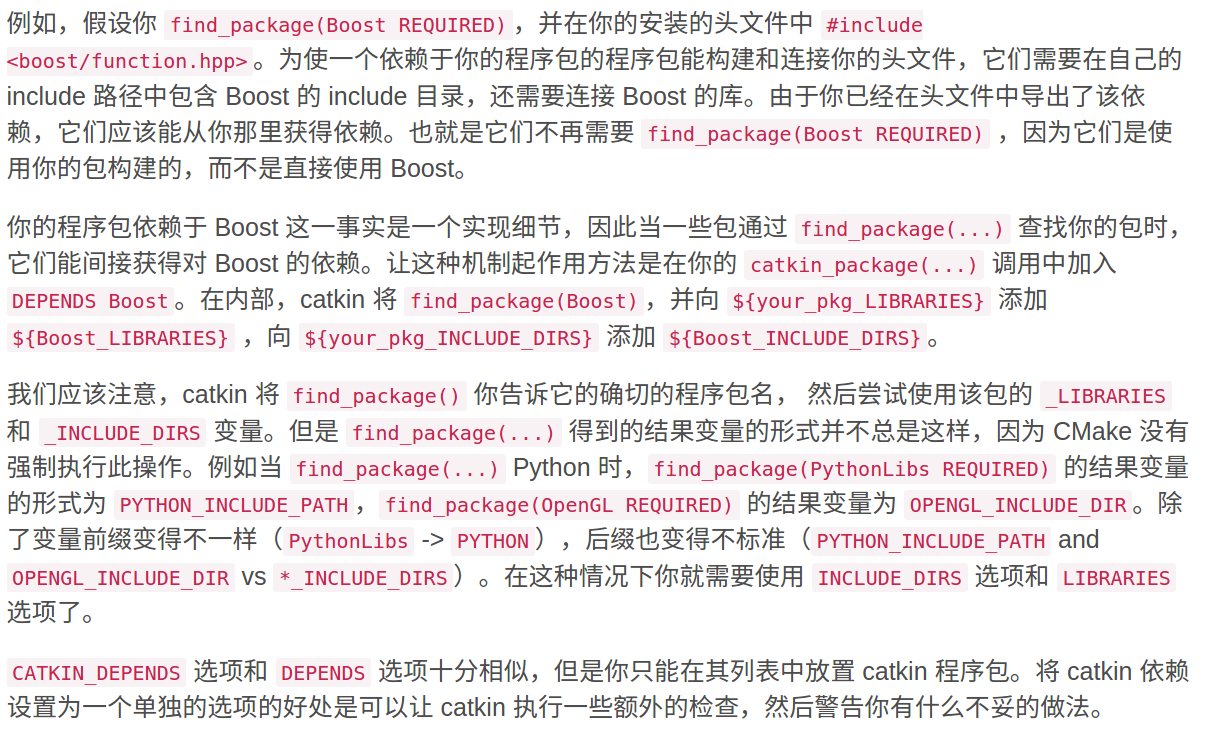
#### Build catkin



<http://wiki.ros.org/catkin/conceptual_overview>

#### ROS Catkin 教程之 catkin\_package(...) 到底在做什么？

<https://blog.csdn.net/lcc816/article/details/82962561>



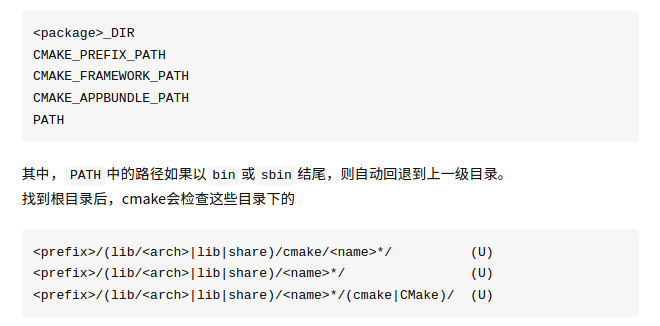
#### setup.py应用场合

<https://www.cnblogs.com/hellcat/p/9763110.html>



#### cmake find\_package路径详解

<https://zhuanlan.zhihu.com/p/50829542>





详见<https://www.cnblogs.com/liuzhenbo/p/11186613.html>

#### Cmake package的概念

<https://cmake.org/cmake/help/v3.0/manual/cmake%2Dpackages.7.html#user-package-registry>

#### [pkg-config原理及用法](https://www.cnblogs.com/sddai/p/10266624.html)

https://www.cnblogs.com/sddai/p/10266624.html