

lint安装和使用说明

0.说明

cpplint是一款c++代码风格和质量保证工具，用于帮助团队不同开发者使用相同的代码规范，保证工程代码质量。但是cpplint并不能保证发现代码中的所有问题，也不会尝试修复这些问题，所以，在使用cpplint之前请仔细阅读《C++代码风格指南》。在此，使用git的hook机制在git commit之前检查代码风格是否合规，如果不合规，则可能提交失败。

1.安装

1.下载cpplint.py脚本

```
1 | wget https://git.io/JYRCf -O cpplint.py
```

2.将cpplint.py脚本配置到系统内

```
1 | sudo mkdir /opt/cpplint
2 | sudo mv cpplint.py /opt/cpplint/
3 | sudo chmod +x /opt/cpplint/cpplint.py
4 | sudo ln -sf /opt/cpplint/cpplint.py /usr/bin/cpplint
```

3.执行完上述命令后，终端输入cpplint --help查看是否有帮助信息打印，以验证安装成功

2.对工程启用lint

1.系统已经安装好cpplint

2.下载pre-commit脚本

```
1 | wget https://git.io/JYR8f -O pre-commit
```

3.将pre-commit拷贝到项目的.git/hooks/目录下

```
1 | mv pre-commit $(project_root)/.git/hooks/
2 | chmod +x ../.git/hooks/pre-commit
```

3.对团队工程启用lint

1.系统已经安装好cpplint

版本维护者:

2.将pre-commit拷贝到项目的.githooks/目录，赋予可执行权限，上传到远程仓库

模块开发者:

2.更新本地仓库到最新版本

3.开启lint

```

1  对于2.9.0及以上版本的git:
2  git config core.hooksPath .githubhooks
3
4  对于2.6.0以下版本的git:
5  find .git/hooks -type l -exec rm {} \;
6  find .githubhooks -type f -exec ln -sf ../../{} .git/hooks/ \;

```

4.使用lint

1.cpplint支持手动配置规则，将CPPLINT.cfg文件放在项目根目录之下，该配置文件将对整个工程起作用，配置文件包含一系列键值对，目前支持以下配置

```

1  set noparent
2  filter+=filter1,-filter2,...
3  exclude_files=regex
4  linelength=120
5  headers=hpp,hxx

```

- set noparent指的是将不再向上层查找CPPLINT.cfg文件，必须有
- filters:控制要检查的错误类型。如果没有这一行，将对所有错误类型进行检查，以+或者-开头控制加入和去掉哪些错误类型，如:

```
filters=-build/header_guard,-legal/copyright
```

表示不对头文件保护符和版权声明进行检查，所有错误类型请参阅《C++代码风格指南》。

- exclude_files(**最常用**):控制工程中不进行lint的文件，正则表达式或者字符串形式(相对于工程根目录)。例如：
 - 若要对某一个文件进行检查，比如rs_perception_3.0的demo中的cnn_detection_demo.cpp


```
exclude_files=release_demo/ros_demo/cnn_detection_demo.cpp
```
 - 若要对某个文件夹下所有文件进行检查，比如rs_perception_3.0的ros_demo


```
exclude_files=release_demo/ros_demo
```
 - 若要对某个文件夹下的所有头文件进行检查，比如rs_perception_3.0的common模块下的所有.h头文件


```
exclude_files=perception/common/*.h
```
 - 若要对rs_pcl文件夹下的所有.cpp文件进行检查


```
exclude_files=.*rs_pcl/*.cpp
```
 - 若同时不对多个文件或者目录进行检查，这些正则表达式或者字符串应以逗号,分隔开（不加空格）

.....

总之，匹配同一个文件或者文件夹有多种表达方式，正则匹配或者直接写出路径，原则就是保证不要匹配到预期之外的文件或目录

- linelength:控制每行的最长字符长度，推荐120
- headers:控制将哪些文件当做.h头文件对待

对于团队工程，由工程总维护者制定CPPLINT.cfg，用git进行版本控制。模块开发者请不要修改CPPLINT.cfg的内容

2.每个检测出来的问题会给出所在的行，不合规类型，更改建议，和一个不合规的置信度分数1-5，5代表问题很确定，1代表问题不太确定。

3.如果确定某行报告的某类不合规是假阳性，可以临时通过在这一行行尾添加注释 `//`

`NOLINT(category)` 对该行禁用该类别lint, `// no lint` 或者 `// NOLINT(*)` 代表对该行禁用所有类别的lint，并及时反馈。

4.对项目开启lint并配置好CPPLINT.cfg之后，即可在执行git commit时对要提交的代码进行合规检查,推荐使用命令：

```
git commit -m "your message" > commit_log.txt 2>&1
```

将报告log到文件中，方便修改，修改之后重新commit，直到成功。