micROS 开发者指南

起草单位	起草人
军科创新院	

参与单位	参与人
军科战争院	
军科评估中心	
中电科 20 所	
中电科 28 所	
国防科大	

2019 年 5 月 19 日

micROS开发者指南

版本号	作者	备注
V1.0	姜载乐，杨懿	创建文档
V1.1	姜载乐，杨懿，王彦臻	增加封面、目录，更改字体和其他格式，更正了一些翻译错误
V1.2	姜载乐，杨懿，王彦臻	改进版权与声明部分的示例；修改原文档说明部分结构和内容；将注释合并到编码规范部分
V1.3	林彬，易伟，王彦臻	改进版权与声明部分的示例；增加函数、变量命名规范

micROS工程浩大，参与人员众多。为了保证软件代码的规范化、标准化和质量，制定本开发指南，望众开发者遵守。 指南以“可操作性”为第一原则，力求简单明了，对规范的原因一般不做详细解释。

目录

一、版权与许可

（一）版权（copyright）

所有源文件/头文件以版权声明开始(一般情况下,版权归属为micROS项目组），格式为（各单位根据代码实际开发者标注单位名称）：

Copyright (C) 2016, micROS Group, NIIDT, TAIIC, HPCL.

All rights reserved.

（二）许可（licence）

许可声明后为作者信息，格式为：

Author(s): <Author1’s Name>, <Author2’s Name>...

（三）示例

1.C++版本示例

```
/*=====================================================================
* Copyright (c) 2016, micROS Group, NIIDT, TAIIC, HPCL.
* All rights reserved.
*
* Redistribution and use in source and binary forms, with or without modification, are permitted
* provided that the following conditions are met:
*
* 1. Redistributions of source code must retain the above copyright notice, this list of conditions and
*     the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions
*     and the following disclaimer in the documentation and/or other materials provided with the
*     distribution.
* 3. All advertising materials mentioning features or use of this software must display the following
*     acknowledgement: This product includes software developed by the micROS Group and its
*     contributors.
* 4. Neither the name of the Group nor the names of contributors may be used to endorse or promote
*     products derived from this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY MICROS GROUP AND CONTRIBUTORS "AS IS"AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
* PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE MICROS, GROUP OR
```

```
* CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
* EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
* PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
* PROFITS; OR BUSINESS INTERRUPTION). HOWEVER CAUSED AND ON ANY THEORY
* OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
* NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
================================================================
* Author: Zaile Jiang, Yi Yang.
*/
```

2.Python版本示例

```
## Copyright (c) 2016, micROS Group, NIIDT, TAIIC, HPCL.
#    All rights reserved.
#
#    Redistribution and use in source and binary forms, with or without    modification, are permitted
#    provided that the following conditions are met:
#    1. Redistributions of source code must retain the above copyright notice, this list of conditions
#       and the following disclaimer.
#    2. Redistributions in binary form must reproduce the above copyright notice, this list of
#       conditions and the following disclaimer in the documentation and/or other materials provided
#       with the distribution.
#    3. All advertising materials mentioning features or use of this software must display the
#       following acknowledgement: This product includes software developed by the micROS Group.
#       and its contributors.
#    4. Neither the name of the Group nor the names of its contributors may be used to endorse or
#       promote products derived from this software without specific prior written permission.
#
#    THIS SOFTWARE IS PROVIDED BY MICROS,GROUP AND CONTRIBUTORS "AS IS"
#    AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
#    THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
#    PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
#    MICROS,GROUP OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
#    INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
#    (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
#    SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
#    HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
#    CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
#    OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
#    ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
#    Author: Zaile Jiang, Yi Yang.
```

二、代码组织结构

（一）整体结构

代码以package作为组织单位，package基于CMake进行编译。除了代码文件，每个package还必须包括以下三个文件：

- Package.xml
- CMakeLists.txt
- README.md

（二）package.xml

package.xml文件是构建的重要配置文件，在package.xml中主要包含以下信息：

- 描述信息（比如 package 的功能描述、维护者等）
- 依赖信息
- 元信息（比如作者、网页）
- package 信息（比如版本）

每个 package.xml 文件必须包含的标签包括：

- <package>：最高级 tag，属性：format，用于指定格式
- <name> ：package 名称
- <version> ：当前版本
- <description>：package 的基本描述
- <maintainer>(至少一个)：维护者
- <license>(至少一个)：协议
- <buildtool_depend>(至少一个)：一般情况下只需要指定 catkin 作为编译工具，在需要交叉编译的情况下需要增加目标机器的编译工具。

典型的 package.xml 文件如下：

```
<?xml version="1.0"?>
<package format="2">
  <name>storage</name>
  <version>0.0.0</version>
  <description>The storage package</description>
  <maintainer email="xxx@xxx.xx">xxx</maintainer>
  <license>BSD</license>

  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_export_depend>roscpp</build_export_depend>
  <exec_depend>roscpp</exec_depend>
</package>
```

（三）CMakeLists.txt

CMakeLists.txt 用于 CMake 生成构建所需要的配置文件。CMakeLists.txt需满足CMake的基本语法规范。

（四）README.md

README.md用于提供package的功能、使用说明、第三方依赖和安装步骤。基本格式为：

package名

==========

基本组成和功能介绍

==========

安装方法（optional）

==========

运行示例或结果展示（optional）

==========

备注信息（optional）

典型示例如下：

geometric_shapes

==========

This package contains generic definitions of geometric shapes and bodies, as well as tools for operating on shape messages.

Shapes represent only the form of an object.

Bodies are shapes at a particular pose. Routines such as point containment and ray intersections are provided.

Supported shapes:

- sphere
- box
- cone
- cylinder
- mesh

Note: Bodies for meshes compute the convex hull of those meshes in order to provide the point containment / ray intersection routines.

Note: [shape_tools](https://github.com/ros-planning/shape_tools) package was recently merged into this package

三、C++编码规范

（一）命名

在micROS中，对特定的资源遵循以下命名规则：

- .msg，.srv文件采用CamelCase
- 类名用CamelCase
- 成员函数和其它函数用camelCase
- topic, service名称用camel_case
- 包名字用camel_case，以micros开头，如micros_xxx
- 类成员变量用_camelCase
- 一般变量用camelCase
- 函数参数用：aCamelCase或anCamelCase，pCamelCase(指针）
- library名称用camel_case
- 全局变量用gCamelCase
- 旧版本函数前加//deprecated
- 未完成功能加TODO或者todo
- 宏定义用CAMEL_CASE
- 头文件中用#define __ACTOR_SCHEDULER__ 定义变量或功能（前后双下划线）

（二）格式

- 缩进必须是两个空格（绝不允许使用tab）；
- 大括号应独占一行；
- 双目运算符与操作的两个变量中间加空格。

示例如下：

```
Point::compareX(const Point& other) const
{
    if (x_ < other.x_)
    {
      return -1;
    }
    else if (x_ > other.x_)
    {
      return 1;
    }
    else
    {
      return 0;
   }
}
```

（三）头文件 guard

头文件必须添加头文件guard，格式为:

```
#ifndef PACKAGE_PATH_FILE_H
#define PACKAGE_PATH_FILE_H
...
#endif
```

（四）打印

使用rosconsole进行屏幕打印，不推荐printf和cout。

（五）宏

尽量使用inline函数和const变量代替宏。

（六）预编译

尽量使用#if代替#ifdef。

示例：

```
#if   DEBUG
   temporary_debugger_break();
#endif
```

不推荐使用：

```
#ifdef    DEBUG
   temporary_debugger_break();
#endif
```

（七）命名空间

推荐在自己的代码包中使用独立的命名空间；不推荐使用using other_namespace，而应该使用using other_namespace::thing。因为前者会污染整个命名空间。示例：

```
using std::list;    // I want to refer to std::list as list
using std::vector;    // I want to refer to std::vector as vector
```

不推荐使用：

```
using namespace std; // Bad, because it imports all names from std::
```

（八）继承

- 推荐用继承和虚函数来统一接口，不推荐继承基类的具体代码实现，因为会造成混乱；
- 重写的虚函数仍然添加 virtual 声明；
- 不推荐 private 继承，此情况推荐将基类作为子类的一个成员变量；
- 不推荐多重继承。

（九）异常

- 推荐使用异常代替返回错误码；
- 为异常建立相关说明文档；
- 不在析构函数和回调函数中使用异常；
- 在进入异常前，保证做好内存释放、信号量释放等工作。

（十）枚举

为枚举添加命名空间。示例：

```
namespace Choices
{
   enum Choice
   {
       Choice1,
       Choice2,
       Choice3
   };
}
typedef Choices::Choice Choice;
```

（十一）全局变量

不推荐使用全局变量和函数。

（十二）类的静态成员变量

不推荐，其让类的多个对象混乱，并且让多线程编程更加困难。

（十三）调用 exit()

不在自己编写的库里调用exit()。

（十四）断言

- 使用断言代替 if 语句检查变量的值；
- 使用 ros/assert.h 里的函数进行断言。

（十五）注释

1.注释原则

注释统一采用JavaDoc风格，格式为：

```
/**      首行有两个“*”
 *       ... text ...
*/
```

为后续文档生成方便，建议采用doxygen模式添加注释：

- 安装说明http://www.doxygen.nl/download.html
- 使用说明http://www.doxygen.nl/manual/starting.html

2.命名空间

命名空间前要添加功能描述，格式为：

```
/**
* @brief  命名空间的简单概述  \n(换行)
*  命名空间的详细概述
*/
namespace micros_xxx
{
}
```

3.类

类前要添加类的功能描述，格式为：

```
/**
* @brief  类的简单概述  \n(换行)
 *  类的详细概述
*/
class Example
{
};
```

4.函数

函数前要添加函数的详细描述，包括功能、参数、返回值和注意事项等，格式为：

```
/**
   * @brief  简要描述  \n
   *  详细描述
   * @param[in] fileName       参数描述，包括参数功能和输入/输出
   *         -r读取，详细描述
   *         -w  可写，详细描述
   * @return  返回值描述
   *       --1  返回值详细描述
   * @note  其它说明
*/
```

5.变量

关键（成员）变量前需要添加注释，格式为：

```
/**
 * @brief 关键变量的简单概述 \n(换行)
 * 关键变量的详细概述
*/
```

6.其它注释

其它编程过程中注释格式不做强制要求，可根据习惯添加。为提高代码可读性，推荐尽量多添加注释。

四、Python 编码规范

该规范适用于所有使用Python编写的micROS代码。

（一）命名

- 命名规则总结如下：
- 代码包：package_name
- 类名：ClassName
- 方法名：method_name
- 域名：field_name
- 私有变量：_private_something
- 私有程度更高的域：self.__really_private_field
- 全局变量：_global

（二）风格

- 使用 4 个空格作为缩进
- 所有的 Python 代码必须被放置在一个模块的命名空间中。用户自己开发的 Python 源代码目录将会被导出到所有依赖的路径上，因此必须避免无意中破坏他人的导入（import）。强烈建议使用模块和代码包使用相同的名字。

（三）源码结构

采用以下两种源代码目录结构：

1.没有 msg/srvs的规模较小的模块：

```
packagename
  |- src/
     |- packagename.py
  |- scripts/
     |- non-exported python files
```

2.包含 msg/srvs的模块：

```
packagename
 |- src/
    |- packagename/
       |- __init__.py
       |- yourfiles.py
 |- scripts/
    |- non-exported python files
```

（四）结点文件

在micROS中，一个结点名与可执行文件名是相同。通常情况下，在主脚本的顶部，Python文件包括#!/usr/bin/env python，其中主脚本名与结点名相同。如果你的结点很简单，这个脚本可能包含它的全部代码。否则，该结点文件很可能调用那里面的代码。

注：尽量将micROS的特定代码与需要重复使用的通用代码分开。

（五）Python 的功能/版本

当前开发使用Python 2.5版本，为了方便的过渡到Python 2.6，2.7，Python3k等更新的Python版本，建议采用以下措施：

- 采用新形式的类；
- 不使用 reduce()。sum()适用于大多数情况，而 for 循环的效率也不差；
- 尽量避免使用 map()或 filter()；
- 使用 raise Exception("msg")而不是 raise Exception, msg；
- 不使用反引号作为 repr 的快捷引用；
- 不使用"<>"，而使用"!="（注意：这并不意味不能使用"<"和">"。只有"< >"将被删除）；
- 使用 from __future__ import division；
- 使用 subprocess，而不是 popen2，os.popen；
- 避免使用 dict.has_key()，而是用 key in dict 代替；

- 尽量避免使用 zip()，range()，map()或 filter()的结果；
- 不使用 string.{atoi|atof|...}()., 而使用 int()，float()代替；
- 避免使用 print >> f, "Message"。使用 f.write("Message\n")代替（例如，sys.stderr.write("My error msg\n")）

这是由于上述特性在 Python 2.5 或 3k 中将被淘汰。

（六）注释

注释可以采用Python原生风格和doxygen风格。推荐采用doxygen风格，格式为：

```
## @package pyexample
#    Documentation for this module.
#
#    More details.
```

典型的Python代码注释示例如下：

```
## @package pyexample
#    Documentation for this module.
#
#    More details.
## Documentation for a function.
#
#    More details.
def func():
    pass
## Documentation for a class.
#
#    More details.
class PyClass:
## The constructor.
    def __init__(self):
        self._memVar = 0;

    ## Documentation for a method.
    #    @param self The object pointer.
    def PyMethod(self):
        pass

    ## A class variable.
classVar = 0;
```

doxygen注释相关文档如下：

- 安装说明http://www.doxygen.nl/download.html
- 使用说明http://www.doxygen.nl/manual/starting.html

五、测试工具/测试代码规范

（一）C++测试

C++测试用gtest。

（二）Python 测试

Python测试用unittest。

（三）ROS API 测试

ROS API的测试用rostest。