doxygen使用详解

为代码写注释一直是大多数程序员有些困扰的事情。当前程序员都能接受为了程序的可维护性、可读性编码的同时写注释的说法，但对哪些地方应该写注释， 注释如何写，写多少等这些问题，很多程序员仍然没有答案。更头痛的是写文档，以及维护文档的问题，开发人员通常可以忍受编写或者改动代码时编写或者修改对 应的注释，但之后需要修正相应的文档却比较困难。如果能从注释直接转化成文档，对开发人员无疑是一种福音。而doxygen就能把遵守某种格式的注释自动 转化为对应的文档。

Doxygen是基于GPL的开源项目，是一个非常优秀的文档系统，当前支持在大多数unix（包括linux），windows家族，Mac系统 上运行，完全支持C++, C, Java, IDL（Corba和Microsoft家族）语言，部分支持PHP和C#语言，输出格式包括HTML、latex、RTF、ps、PDF、压缩的HTML和unix manpage。有很多开源项目（包括前两篇文章介绍的log4cpp和CppUnit）都使用了doxygen文档系统。而国内的开发人员却使用的不 多，这里从开发人员使用的角度介绍这个工具，使开发人员用最少的代价尽快掌握这种技术，并结合这个工具探讨如何撰写注释的问题。以下以linux下的 C++语言为例进行介绍，以下讨论基于doxygen1.3.3。

1. doxygen使用步骤
由于只是工具的使用，这里不介绍它的原理，直接从使用步骤开始。Doxygen的使用步骤非常简单。主要可以分为：
1）第一次使用需要安装doxygen的程序
2）生成doxygen配置文件
3）编码时，按照某种格式编写注释
4）生成对应文档
doxygen的安装非常简单， linux下可以直接下载安装包运行即可，下载源代码编译安装也是比较通用的编译安装命令。请参考其安装文档完成安装。

Doxygen在生成文档时可以定义项目属性以及文档生成过程中的很多选项，使用下面命令能够产生一个缺省的配置文件：
doxygen -g [配置文件名]
可以根据项目的具体需求修改配置文件中对应的项，具体的修改过程在下面介绍。修改过的配置文件可以作为以后项目的模板。

让doxygen自动产生文档，平常的注释风格可不行，需要遵循doxygen自己的格式。具体如何写doxygen认识的注释在第3节详细介绍。

OK，代码编完了，注释也按照格式写好了，最后的文档是如何的哪？非常简单，运行下面的命令，相应的文档就会产生在指定的目录中。
doxygen [配置文件名]

需要注意的是doxygen并不处理所有的注释，doxygen重点关注与程序结构有关的注释，比如：文件、类、结构、函数、变量、宏等注释，而忽略函数内变量、代码等的注释。

2. doxygen配置文件
doxygen配置文件的格式是也是通常的unix下配置文件的格式：注释'#'开始；tag = value [,value2…]；对于多值的情况可以使用 tag += value [,value2…]。

对doxygen的配置文件的修改分为两类：一种就是输出选项，控制如何解释源代码、如何输出；一种就是项目相关的信息，比如项目名称、源代码目 录、输出文档目录等。对于第一种设置好后，通常所有项目可以共用一份配置，而后一种是每个项目必须设置的。下面选择重要的，有可能需要修改的选项进行解释 说明，其他选项在配置文件都有详细解释。

TAG 缺省值 含义
PROJECT_NAME 项目名称
PROJECT_NUMBER 可以理解为版本信息
OUTPUT_DIRECTORY 输出文件到的目录，相对目录（doxygen运行目录）或者绝对目录
INPUT 代码文件或者代码所在目录，使用空格分割
FILE_PATTERNS *.c *.cc *.cxx *.cpp *.c++ *.java *.ii *.ixx *.ipp *.i++ *.inl *.h *.hh *.hxx *.hpp *.h++ *.idl *.odl 指定INPUT的目录中特定文件，如：*.cpp *.c *.h
RECURSIVE NO 是否递归INPUT中目录的子目录
EXCLUDE 在INPUT目录中需要忽略的子目录
EXCLUDE_PATTERNS 明确指定的在INPUT目录中需要忽略的文件，如：FromOut*.cpp

OUTPUT_LANGUAGE English 生成文档的语言，当前支持2、30种语言，国内用户可以设置为Chinese
USE_WINDOWS_ENCODING YES（win版本）
NO（unix版本） 编码格式，默认即可。
EXTRACT_ALL NO 为NO，只解释有doxygen格式注释的代码；为YES，解析所有代码，即使没有注释。类的私有成员和所有的静态项由EXTRACT_PRIVATE和 EXTRACT_STATIC控制
EXTRACT_PRIVATE NO 是否解析类的私有成员
EXTRACT_STATIC NO 是否解析静态项
EXTRACT_LOCAL_CLASSES YES 是否解析源文件（cpp文件）中定义的类
SOURCE_BROWSER NO 如果为YES，源代码文件会被包含在文档中
INLINE_SOURCES NO 如果为YES，函数和类的实现代码被包含在文档中
ALPHABETICAL_INDEX NO 生成一个字母序的列表，有很多类、结构等项时建议设为YES
GENERATE_HTML YES 是否生成HTML格式文档
GENERATE_HTMLHELP NO 是否生成压缩HTML格式文档（.chm）
GENERATE_LATEX YES 是否乘车latex格式的文档
GENERATE_RTF NO 是否生成RTF格式的文档
GENERATE_MAN NO 是否生成man格式文档
GENERATE_XML NO 是否生成XML格式文档

3. doxygen注释
3.1 注释风格
下面是工作量最大部分，安装doxygen格式写注释。通常代码可以附上一个注释块来对 代码进行解释，一个注释块由一行或者多行组成。通常一个注释块包括一个简要说明（brief）和一个详细说明（detailed），这两部分都是可选的。 可以有多种方式标识出doxygen可识别的注释块。
1）JavaDoc类型的多行注释。

2）QT样式的多行注释。

3） /// …text….
4） //! …text….
简要说明有多种方式标识，这里推荐使用@brief命令强制说明，例如：

以上这些注释格式用来对紧跟其后的代码进行注释。doxygen也允许把注释放到代码后面，具体格式是放一个'<'到注释开始部分。例如：
```
int var1 ;
int var2; ///< ….text….
```

注释和代码完全分离，放在其他地方也是允许的，但需要使用特殊的命令加上名称或者声明进行标识，比如：class、struct、union、 enum、fn、var、def、file、namespace、package、interface（这些也就是doxygen关注的注释类型）。这里 不推荐使用，建议注释尽量放在代码前后。具体使用方式参见doxygen手册。

3.2 doxygen常用注释格式
通常的选择上面的一、两种注释风格，遇到头文件中各种类型定义，关键变量、宏的定义，在其前或者后使用 @brief 定义其简要说明，空一行后继续写其详细的注释即可。

对函数的注释，是比较常常需要注释的部分。除了定义其简要说明以及详细注释，还可以使用param命令对其各个参数进行注释，使用return命令对返回值进行注释。常见的格式如下：

```
int funcl(int a, int b);
```

进行设计时，通常有模块的概念，一个模块可能有多个类或者函数组成，完成某个特定功能的代码的集合。如何对这个概念进行注释？doxygen提供了group的概念，生成的模块的注释会单独放在一个模块的页面中。使用下面的格式定义一个group。

code

group中的代码可以有自己的注释。单纯定义一个模块，去除{ 和}命令即可。任何其他代码项（比如类、函数、甚至文件）如果要加入到某个模块，可以在其doxygen注释中使用ingroup命令即可。Group之间使用ingroup命令，可以组成树状关系。

把多个代码项一起添加到某个模块中可以使用addtogroup命令，格式和defgroup相似。

对于某几个功能类似的代码项（比如类、函数、变量）等，如果希望一起添加注释，而又不想提升到模块的概念，可以通过下面的方式：

```
//@{

code…
//@}
```

对这种组进行命名可以使用name命令。此时中间代码可以有自己的注释。如：

```
//@{
code…
//@}
```

3.3 doxygen常用注释命令
doxygen通过注释命令识别注释中需要特殊处理的注释，比如函数的参数、返回值进行突出显示。上面也提到了一些注释命令（如：brief、param、return、以及group相关的命令），下面对其他一些常用的注释命令进行解释说明。
@exception <exception-object> {exception description} 对一个异常对象进行注释。
@warning {warning message } 一些需要注意的事情
@todo { things to be done } 对将要做的事情进行注释
@see {comment with reference to other items } 一段包含其他部分引用的注释，中间包含对其他代码项的名称，自动产生对其的引用链接。
@relates <name> 通常用做把非成员函数的注释文档包含在类的说明文档中。
@since {text} 通常用来说明从什么版本、时间写此部分代码。
@deprecated
@pre { description of the precondition } 用来说明代码项的前提条件。
@post { description of the postcondition } 用来说明代码项之后的使用条件。
@code 在注释中开始说明一段代码，直到@endcode命令。

@endcode 注释中代码段的结束。

到此为止，常用的doxygen的注释格式讨论完毕，我们能够按照一定的格式撰写doxygen认识的注释，并能够使用doxygen方便快捷的生成对应的文档，不过注释中应该写些什么，如何撰写有效的注释可能是困扰开发人员的一个更深层次的问题。

4. 注释的书写
注释应该怎么写，写多还是写少。过多的注释甚至会干扰对代码的阅读。写注释的一个总的原则就是注释应该尽量用来表明作者的 意图，至少也应该是对一部分代码的总结，而不应该是对代码的重复或者解释。对代码的重复或者解释的代码，看代码可能更容易理解。反映作者意图的注释解释代 码的目的，从解决问题的层次上进行注释，而代码总结性注释则是从问题的解答的层次上进行注释。

推荐的写注释的过程是首先使用注释勾勒出代码的主要框架，然后根据注释撰写相应的代码。对各种主要的数据结构、输出的函数、多个函数公用的变量进行详细地注释。对代码中控制结构，单一目的的语句集进行注释。下面是一些写注释时需要注意的要点：
避免对单独语句进行注释；
通过注释解释为什么这么做、或者要做什么，使代码的读者可以只阅读注释理解代码；
对读者可能会有疑问的地方进行注释；
对数据定义进行注释，而不是对其使用过程进行注释；
对于难于理解的代码，进行改写，而不要试图通过注释加以说明；
对关键的控制结构进行注释；
对数据和函数的边界、使用前提等进行注释；

5. 参考资料
1. doxygen homepage
http://www.stack.nl/~dimitri/doxygen/

2. doxygen manual
http://www.stack.nl/~dimitri/doxygen/manual.html

3. Code Complete: A Practical Handbook of Software Construction. Redmond, Wa.: Microsoft Press, 880 pages, 1993. ISBN: 1-55615-484-4.

4. 简介doxygen
http://www.stack.nl/~dimitri/doxygen/doxygen_intro_cn.html

5. 10 Minutes to document your code
http://www.codeproject.com/tips/doxysetup.asp

6. 使用doxygen
http://www.csdn.net/Develop/article/16%5C16383.shtm