doxygen使用

安装

从官方网站下载二进制文件，双击安装。推荐同时安装GraphViz(v2.20版本以上），因为Doxygen可以用GraphViz生成图表。
在windows下：
- 若想生成压缩的HTML文档，则需要安装微软的HTML help workshop
- 若想生成Qt帮助文档，则需要安装qhelpgenerator
- 若想生成PDF文档，则需要安装LaTex以及Ghostscript

配置

Doxygen使用一个配置文件来管理其设置。对于每一个工程，都需要有它自己的配置文件。Doxygen通过下面的命令生成一个配置文件的模板：

```
doxygen -g <config-file>
```

<config-file>是配置文件的文件名。如果忽略该参数，则默认给出的文件名为Doxyfile。如果指定的文件名已存在，则doxygen会将已存在的文件名重命名为<config-file>.bak，然后再创建配置文件。如果使用 - 作为文件名，则doxygen会从标准输入stdin中读取文件名

配置属性

所有的配置属性都是按照格式：

```
属性名 = 属性值
属性名 = 属性值1  属性值2
```

如果不想手动编辑配置文件，则可以使用doxywizard这个GUI工具来编辑配置文件。

常用属性：
HAVE_DOT = YES ，此时会使用Graphviz，默认为NO
USE_PDFLATEX = YES（默认为YES），此时会从LATEX生成PDF文档，该属性为YES要求GENERATE_LATEX属性为YES（默认为YES）

源文件

对于小型的C/C++工程，只需要将INPU属性置空即可，doxygen会在当前目录中搜索源文件
对于大型的工程，则可以设置以下属性：
- INPUT属性：设置为工程的根目录
- FILE_PATTERNS属性：设置文件类型过滤器（如*.cpp *.h等一个或多个值)。当源文件匹配其中某一项时即可匹配。若为空，则默认过滤器为所有doxygen支持的类型
- RECURSIVE属性：设为YES则表示递归查找源代码所在的目录树
- EXCLUDE属性：排除某些文件或目录
- EXCLUDE_PATTERNS属性：通过正则表达式形式排除某些文件

运行

为了生成文档，运行命令：

```
doxygen <config-file>
```

- 具体生成的是html,latex,xml，由配置文件决定
- 默认的输出位置，由配置文件的OUTPUT_DIRECTORY决定。若为空则默认输出到doxygen启动时的目录

注释编写

C/C++

注释类型

对于代码中的每个实体都有两种注释类型：brief description以及detailed description，每一个都是可选的。对于函数还存在第三种注释类型：body description，它由函数体中发现的所有注释块组成。

detailed description

• C风格的detailed description

```
/**
 * ... detailed description...
 */
```

• Qt风格的detailed description

```
/*!
 * ... detailed description...
 */
```

• 第三种风格的detailed description

```
///
/// ... detailed description...
///
```

brief description

• 基于\brief指令的brief description 。其中detailed description必须与brief description隔一个空行

```
/*! \brief Brief description.
 *         Brief description continued.
 *
 *  Detailed description starts here.
 */
```

• 基于JAVADOC_AUTOBRIEF为YES（在配置文件中设置）的brief description。此时brief description自动在第一个'.'号结束，后面紧跟着detailed description

```
/** Brief description which ends at this dot. Details follow
 *  here.
 */
```

- 第三种C++风格的brief description。

```
/// Brief description.
/** Detailed description. */
```

在成员之后添加注释

可以在成员之后添加注释。
- Qt风格：

```
int var; /*!< Detailed description after the member */
```

- C++风格：

```
int var; /**< Detailed description after the member */
```

如果只需要添加brief description，则：

```
int var; //!< Brief description after the member
```

示例

```
//!  A test class.
/*!
  A more elaborate class description.
*/
class Test
{
  public:
    //! An enum.
    /*! More detailed enum description. */
    enum TEnum {
                 TVal1, /*!< Enum value TVal1. */
                 TVal2, /*!< Enum value TVal2. */
                 TVal3  /*!< Enum value TVal3. */
               }
         //! Enum pointer.
         /*! Details. */
         *enumPtr,
         //! Enum variable.
         /*! Details. */
         enumVar;

    //! A constructor.
    /*!
      A more elaborate description of the constructor.
    */
    Test();
    //! A destructor.
    /*!
      A more elaborate description of the destructor.
    */
   ~Test();

    //! A normal member taking two arguments and returning an integer value.
    /*!
      \param a an integer argument.
      \param s a constant character pointer.
      \return The test results
      \sa Test(), ~Test(), testMeToo() and publicVar()
    */
    int testMe(int a,const char *s);

    //! A pure virtual member.
    /*!
      \sa testMe()
      \param c1 the first argument.
      \param c2 the second argument.
    */
    virtual void testMeToo(char c1,char c2) = 0;

    //! A public variable.
    /*!
      Details.
    */
    int publicVar;

    //! A function variable.
    /*!
      Details.
    */
    int (*handler)(int a,int b);
};
```

注释位置

注释块通常位于被注释内容的前面。也可以将注释放在任何地方（除了函数体内的注释），方式为：

```
/*! \class Test
    \brief A test class.

    A more detailed class description.
*/
```

此时你必须在注释开始处放置一个指令，指示该注释所对应的实体。常用的为：
- \struct to document a C-struct.
- \union to document a union.
- \enum to document an enumeration type.
- \fn to document a function.
- \var to document a variable or typedef or enum value.
- \def to document a #define.
- \typedef to document a type definition.
- \file to document a file.
- \namespace to document a namespace.
- \package to document a Java package.
- \interface to document an IDL interface

为了document一个C++ class的member，你必须要首先document这个class。同理，若想document一个namespace中的member，你必须首先document这个namespace。最后，若想document全局的C函数，typedef,enum或者预编译指令，则你必须首先document包含它们的那个file：

```
/*! \file  xxx.h*/
```

示例

```
/*! \file structcmd.h
    \brief A Documented file.

    Details.
*/
/*! \def MAX(a,b)
    \brief A macro that returns the maximum of \a a and \a b.

    Details.
*/
/*! \var typedef unsigned int UINT32
    \brief A type definition for a .

    Details.
*/
/*! \var int errno
    \brief Contains the last error code.
    \warning Not thread safe!
*/
/*! \fn int open(const char *pathname,int flags)
    \brief Opens a file descriptor.
    \param pathname The name of the descriptor.
    \param flags Opening flags.
*/
/*! \fn int close(int fd)
    \brief Closes the file descriptor \a fd.
    \param fd The descriptor to close.
*/
/*! \fn size_t write(int fd,const char *buf, size_t count)
    \brief Writes \a count bytes from \a buf to the filedescriptor \a fd.
    \param fd The descriptor to write to.
    \param buf The data buffer to write.
    \param count The number of bytes to write.
*/
/*! \fn int read(int fd,char *buf,size_t count)
    \brief Read bytes from a file descriptor.
    \param fd The descriptor to read from.
    \param buf The buffer to read into.
    \param count The number of bytes to read.
*/
#define MAX(a,b) (((a)>(b))?(a):(b))
typedef unsigned int UINT32;
int errno;
int open(const char *,int);
int close(int);
size_t write(int,const char *, size_t);
int read(int,char *,size_t);
```

特殊指令

所有的特殊指令均是以\或者@开启。

Python

Python标准的documentation string存放在doc属性中，可以在运行时获取。doxygen可以获取这些注释来生成文档。但是此时无法使用任何doxygen的特殊指令。

```
"""@package docstring
Documentation for this module.

More details.
"""

def func():
     """Documentation for a function.

    More details.
    """
    pass

class PyClass:
    """Documentation for a class.

    More details.
    """

    def __init__(self):
        """The constructor."""
        self._memVar = 0;

    def PyMethod(self):
        """Documentation for a method."""
        pass
```

doxygen支持另外一种python注释：通过以##开启。这种注释类似于C/C++中的注释，支持特殊指令的使用

```
## @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;

    ## @var _memVar
    #  a member variable
```