Doxygen

zakaria.meliani

March 2019

Abstract

All files can be download from courses and exercice

1 Introduction

Doxygen is a documentation system for C, C++, Fortran, Python, and others languages. It allows to generate documentation for you developments:

- from comments inserted in the code source.
- from code structure it self.

The generated documentation can be used with browser (in HTML) and/or an off-line reference manual (in LaTeX), RTF, DOC files.

1.1 Comments in code source

There are two types of comments in code sources. The first kind are comments at the beginning of each file. It describes the file and lists the authors and known bugs. The second kind of comments are more local within the file that describe your procedures/function associated arguments and parameters and class/type/structure in use within the file.

1.2 Linux

sudo apt-get install Graphviz
sudo apt-get install doxygen doxygen-gui

1.3 mac

brew install Graphviz
brew install doxygen

1.4 Windows

from this link: http://doxygen.nl/files/doxygen-1.8.15-setup.exe

1.5 GIT repository

If you the following : g++, python, cmake are installed. You can install with git repository git clone https://github.com/doxygen/doxygen.git

cd doxygen
mkdir build
cd build
cmake -G "Unix Makefiles" ..
make
make install

2 Generate documentation

Doxygen documentation of any "object" (file, function, class, ...) consists of brief and detailed description using specific Doxygen comments which are different form others comments

2.1 C code

```
/**
 * This is multilignes comments
 * for C and C++ language
**/
```

Doxygen C code comment 1

Or

```
/*!
 * You can use this forms
 * for C and C++ language
*/
```

Doxygen C code comment 2

Or

```
//!
//!  You can use this forms
//!  for C and C++ language
//!
```

Doxygen C code comment 3

Or

```
///
///  You can use this forms
///  for C and C++ language
///
```

Doxygen C code comment 4

Doxygen supports also in-line comments, both brief and detailed kind; these blocks can only be used to document members and parameters!

```
int i_param = 5; /** on the same line**/
```

Doxygen C code line comment

2.2 File description

Each file should be describe as follow

```
/*
_____________________________________________________________________
          FILE HEADER
_____________________________________________________________________
 TITLE          : project name
 PROJECT        : sub-project name
 MODULE         : name of the module or program
 URL            : ...
 AFFILIATION    : Observatoire
 DATE           : ...
 REVISION       : ... V0.8
*/
/*!
 * @file [filename]
 *
 * @author [your name]
 * @date
 *
 @brief [brief description]
 @details [detailed description but not need]
 @section
 @f[
   \vec{v}\cdot\vec{\nabla}\vec{v}
 @f]
 @see [https://URL]
 @bug no bug
 @return [Hello word]
 */
int main(int argc, char** argv)
{
  ...
}
```

codes/C_codes/Simple_files/header_file_C.c

1. First you should use @file to attribute name to file documentation file

   @file [file name]

2. The authors names/institutions is set using

   /** * @author [authors name] * ## Institution * Observatoire de Paris * ## adress * Meudon */

   Here we use Markdown to institution and address in bold

3. At the main program file header it is possible to impose it as the main page by adding

   !> **@mainpage** [add title]

2.3 Class description

It important when use a class to declare it with

1. **@class** [class name]

   and

2. **@enum** to document an enumeration type.

```
/**
* @file       class.C
* @author     MELIANI
* @version    1.0
* @date       01 Mars 2019
* @brief      TP du cours Doxygen
*
* @details    Ce programme permet de mettre en pratique
*             les notions vues pendant le cours
*             Calcul de la documentation de class en C
*
* @todo
* plus generale
*
* @mainpage Cours Doxygen
*
* @section intro Introduction
*     This code it use to learn Doxygen with fortran
* @section optims Optimisations
*    \subsection loops Les boucles
*       blablabla
*    \subsection cstes Les constantes
*       blablabla
*/

class TheClass
{
  public:

    /**
     * An enum.
     * More detailed enum description.
     */

    enum TEnum {
          TVal1, /**< enum value TVal1. */
          TVal2, /**< enum value TVal2. */
          TVal3  /**< enum value TVal3. */
         }
       *enumPtr, /**< enum pointer. Details. */
       enumVar;  /**< enum variable. Details. */
```

```
1042        /**
            * A constructor.
1044        * A more elaborate description of the constructor.
            */
1046        TheClass();

1048        /**
            * A destructor.
1050        * A more elaborate description of the destructor.
            */
1052       ~TheClass();

1054        /**
            * a normal member taking two arguments and returning an
        integer value.
1056        * @param a an integer argument.
            * @param s a constant character pointer.
1058        * @see TheClass()
            * @see ~TheClass()
1060        * @see testMeToo()
            * @see publicVar()
1062        * @return The test results
            */
1064        int AME_MEMBER(int a,const char *s);

1066        /**
            * A pure virtual member.
1068        * @see testMe()
            * @param c1 the first argument.
1070        * @param c2 the second argument.
            */
1072        virtual void testMeToo(char c1,char c2) = 0;

1074        /**
            * a public variable.
1076        * Details.
            */
1078        int publicVar;

1080        /**
            * a function variable.
1082        * Details.
            */
1084        int (*handler)(int a,int b);
    };
```

codes/C_codes/Simple_files/class.C