Best Tips for documenting code using doxygen? [closed]

Asked 16 years, 3 months ago Modified 7 years, 7 months ago Viewed 151k times

308

votes

Closed. This question is opinion-based. It is not currently accepting answers.

Closed 8 years ago.

Locked. This question and its answers are locked because the question is off-topic but has historical significance. It is not currently accepting new answers or interactions.

My team is starting to document our C code using doxygen, paying particular attention to our public API headers. There appears to be a lot of flexibility and different special commands in doxygen, which is great, but it's not clear what's a good thing and what's a bad thing without trial and error.

What are your favourite ways to mark up your code, what are your MUST DOs and DO NOTs?
Please provide your top tips, one per answer to facilitate voting.

I am looking to define our whole approach to API documentation, including providing a template to get the rest of the team started. So far I have something like this:

```
/**
 * @file   example_action.h
 * @Author Me (me@example.com)
 * @date   September, 2008
 * @brief  Brief description of file.
 *
 * Detailed description of file.
 */

/**
 * @name    Example API Actions
 * @brief   Example actions available.
 * @ingroup example
 *
 * This API provides certain actions as an example.
 *
 * @param [in] repeat  Number of times to do nothing.
 *
 * @retval TRUE   Successfully did nothing.
 * @retval FALSE  Oops, did something.
 *
 * Example Usage:
```

```
 * @code
 *    example_nada(3); // Do nothing 3 times.
 * @endcode
 */
boolean example(int repeat);
```

c++ c documentation doxygen

Share

edited Jun 6, 2014 at 22:02

balki

27.6k ● 32 ● 109 ● 155

asked Sep 9, 2008 at 11:41

Andrew Johnson

7,289 ● 4 ● 25 ● 27

7 Personally I think that the `[in]` and `[out]` parts of the `param` shouldn't be needed. Your API should specify if something is an in variable or and out variable: `const int * const a` is an `in` and `int * const a` is and `out` – Matt Clarkson Aug 24, 2012 at 8:03

6 File autorship doesn't belong in a source file, unless the author's name is part of the copyright declaration. Figuring out who wrote what is the source control blame functionality's raison d'être – Kuba hasn't forgotten Monica Aug 4, 2014 at 14:48

@Mark, I had not thought of that (+1). Of course, it is not as instantly intuitive, but it can't get out of synch, so good point. I have occassionally `#define` d `INPUT`, `MODIFY` and `OUTPUT` as empty macros, and used them both on declarations and on actucall calls. Opinion is divided on whether that is a good thing. – Mawg Nov 22, 2016 at 12:32

Comments disabled on deleted / locked posts / reviews

20 Answers

Sorted by: Highest score (default)

92 votes

You don't need and should not write the name of the file in the `@file` directive, doxygen reads the name of the file automatically. The problem with writing the name of the file is that when you rename the file you will have to change the `@file` directive as well.

Providing `@author` and `@date` information is also useless most of the time since the source control system does it a lot better than someone editing the files manually.

You also don't have to write `@brief` if you use the following Doxygen syntax and enable JAVADOC_AUTOBRIEF in doxygen's configuration:

```
/*! Short Description on the first line

    Detailed description...
    ...
*/
void foo(void) {}
```

The `@name` directive for functions is also 100% redundant most of the time and completely useless. It only brings errors when someone modifies the name of the function and not the doxygen `@name`.

Share

edited Sep 4, 2016 at 14:40

answered Apr 3, 2014 at 11:46

Étienne

4,984 ● 2 ● 38 ● 64

4 the doxygen docs says "Let's repeat that, because it is often overlooked: to document global objects (functions, typedefs, enum, macros, etc), you must document the file in which they are defined. In other words, there must at least be a /*! \file *or a* /* @file */ line in this file." it seems to suggest that \file can be necessary. however, this may simply be misinterpreted. worth clarifying. – ivo Welch Jun 22, 2014 at 1:48

7 \file is necessary but not the name of the file, that's all I'm saying ;) – Étienne Jun 22, 2014 at 6:49

2 Please Étienne, add in your answer an example of the beginning of a file usign `\file` without the filename. I think you need to set `JAVADOC_AUTOBRIEF = YES` in order to avoid using `@brief`. Can you confirm? What do you advice to provide a brief for a file? Cheers – oHo Jul 1, 2016 at 9:07

@ivoWelch Their documentation also says "For member functions or functions that are part of a namespace you should document either the class or namespace." so \file may not seem needed, if all your free functions are in namespaces? I didn't check that though... – Ela782 Sep 1, 2016 at 0:21

You don't need \file if all your functions are in a class or a namespace. Please keep in mind that I was answering to a particular question, which uses \file in a templace (and that makes sense in my opinion). – Étienne Sep 4, 2016 at 14:39

60

votes

Write a descriptive home page using @mainpage (in a separate header file just for this purpose). Consider, as shown in my example, making it a guide to your main classes/functions and modules.

Another Sample

Whilst I was getting the above-linked main oofile doxygen content back online, here's an example from some current client work using Markdown format. Using Markdown you can refer to a mainpage in markdown (in the Doxygen settings) which is great for the typical `readme.md` file included in open-source projects.

```
Lingopal
========
Developer Documentation started when Andy Dent took over support in May 2014.

There are a number of pages in Markdown format which explain key aspects:
```

```
- @ref doc/LingopalBuilding.md
- @ref doc/LingopalSigning.md
- @ref doc/LingopalDatabases.md
- @ref doc/LingopalExternals.md

See the <a href="pages.html">Related Pages list for more.</a>

-------------

_Note_

These pages, whilst readable by themselves, are designed to be run through the
[Doxygen](http://www.doxygen.com) code documentation engine which builds an
entire local cross-referenced set of docs. It uses a minor [extension of
Markdown formatting.]
(http://www.stack.nl/~dimitri/doxygen/manual/markdown.html)

The settings to generate the documentation are `Lingopal.doxy` and
`LingopalDocOnly.doxy`. The latter is used for quick re-generation of just
these additional pages.
```

Share

edited Dec 6, 2015 at 19:31

Victor Stafusa

14.5k ● 13 ● 62 ● 75

answered Jan 14, 2009 at 22:55

Andy Dent

17.9k ● 7 ● 92 ● 119

7 The link is dead, can't see what you mean. I suggest either expanding the answer to be self contained, or fix the dead link. – Rick Deckard Jun 12, 2014 at 19:50

1 sorry, will try to get it back up ASAP, was subject to a DDOS on my blog and also had to take down my Doxygen site as a consequence as it had too many files in one dir – Andy Dent Jun 14, 2014 at 0:55

54

votes

Use Groups to organise your code into modules.

Remember that you can put almost everything into multiple groups so they can be used to provide semantic tagging like the tags in Stack Overflow. For example, you might tag things as specific to a given platform.

You can also use groups to match a folder hierarchy within an IDE, as shown in my RB2Doxy sample output.

Groups work well when nested - I have a large example for the OOFILE source.

Share

edited May 14, 2017 at 20:03

Bhargav Rao

52k ● 29 ● 126 ● 141

answered Jan 14, 2009 at 22:50

Andy Dent

17.9k ● 7 ● 92 ● 119

Yes, using nested groups was what I was talking about with one of my answers. I've been experimenting with using a separate file for the hierarchy management, which seems to work well. – Andrew Johnson Mar 17, 2009 at 15:01

2 I wish I could give this and the other answers involving groups more than one vote. Doxygen can't tell you anything that is not already in your code. It can, however, present the information in a different order or filter it. Groups are one of the most useful things you can add in comments because it lets doxygen give you what you can not get by opening the source files in an editor - a significantly reorganised view of your code. – Bowie Owens Sep 22, 2011 at 1:55

6 Broken links... – Aaron Campbell Apr 9, 2015 at 23:11

I found the RB2Doxy output in a .zip file: oofile.com.au/files/REALbasic/RB2DoxySample.zip – HotOil Jan 21, 2016 at 20:00

can you please fix the links in your answer? – xaxxon Jun 6, 2017 at 2:49

49 votes

Some commands i use in my code :

- `\todo { paragraph describing what is to be done }` Useful to keep track of todos, a page will be created in final documentation containing your todo list.
- `\c <word>` Displays the argument using a typewriter font. Use this to refer to a word of code. I would use it before "TRUE" and "FALSE" in your example.
- `\a , \warning , \see` : see http://www.stack.nl/~dimitri/doxygen/commands.html#cmdc for description

Share

answered Sep 9, 2008 at 13:39

kraymer
3,424 ● 2 ● 26 ● 33

2 Thanks, some useful stuff there. \c doesn't work for TRUE/FALSE in the question because it's using retval, not return. All your retvals make a table where the first word after the command is the return value, so using \c just gives you a table of "\c". – Andrew Johnson Sep 19, 2008 at 23:33

36 votes

A good "best practice" (though not always achievable) is to provide short, working examples for every API, and pull them into the help using \includelineno (or \include for no line numbers). These can be unit tests, if they're written so users can understand them (ie, not hooked into a larger test harness). As a nice side effect, changes to the API will break the samples, so they have to be kept up to date.

You can describe an API in words, but there's nothing like seeing the actual code to understand how to use it.

Share

answered May 22, 2009 at 0:20

Hoser

749 ● 8 ● 12

Yeah, I had that in the example template. It's especially useful in the comment that concerns a group of APIs to show how they work together. – Andrew Johnson Jun 4, 2009 at 21:28

28

votes

As I find myself editing code on higher-resolution screens I've moved from using the backslash to the @ prefix on Doxygen commands. *Not so noisy* backslash has found itself now *too damned hard to make out the Doxygen commands.*

Share

answered Jan 15, 2009 at 12:47

Andy Dent

17.9k ● 7 ● 92 ● 119

27

votes

If you are sure your team will follow such a heavyweight template, fine, use it as shown.

Otherwise, it looks like JavaDoc. One of the nice things about Doxygen is how good a job it does *without* having to use use such strong markup. You don't need to use @name and with the JAVADOC_AUTOBRIEF setting you can skip @brief - just make sure the first line of the comment is a reasonable brief description.

I prefer descriptive names over enforcing documentation and encouraging people to add comments only when they add significant value. That way, the valuable comments aren't drowned out by all the *noise.*

Share

answered Jan 14, 2009 at 22:43

Andy Dent

17.9k ● 7 ● 92 ● 119

7 Inevitably if you try to follow a template, you will get a lot of comments like "This is a brief description of the class" in actual code from people forgetting to go back and add stuff intot he comments. Better nothing than these types of comments. – Greg Rogers Jan 14, 2009 at 22:50

1 The one sad thing about making documentation optional is that, in my experience, code just doesn't get documented. If the template ends up being excessively larger that what most documentation blocks actually are, you can always change it. – axs6791 Feb 10, 2009 at 20:17

4 If you make documentation optional you have the chance to audit if it was added, particularly a quick check with Doxygen changing settings to warn about undocumented classes. I'd also suggest using XML output from Doxygen so you can parse comments for common bland strings. – Andy Dent Feb 11, 2009 at 1:20

We try not to have too large a public API and we have strong code reviews so people are following the templates and the comments are reasonable. The templates we're using aren't as heavyweight as the one above. One big block per API header and a lightweight one per function. – Andrew Johnson Mar 17, 2009 at 14:59

25 votes

If you have bugs located in the code or you find bugs you can also tag in the code like this:

```
/** @bug The text explaining the bug */
```

When you then run doxygen you get a seperate Bug List alongside lists like Todo List

Share

answered Nov 14, 2008 at 11:10

eaanon01

1,079 ● 7 ● 9

15 votes

If you have a really, really big project -- big enough that Doxygen runs take over an hour -- you can cut it up into multiple modules that Doxygen later links together using tag files.

For example, if you have a big MSVC solution with twenty projects in it, you can make directory be its own Doxygen run, and then use tag-files to glue together the output the same way a linker glues together .libs to make an executable.

You can even take the linking metaphor more literally and make each Doxy config file correspond to a .vcproj file, so that each project (eg .lib or .dll) gets its own Doxy output.

Share

answered Jan 14, 2009 at 22:52

Crashworks

41.3k ● 14 ● 107 ● 171

This is a good operational tip, albeit one that I've not had to apply in person. If you have used tag files, does that affect HOW people write their Doxygen content in any way that programmers would need to be warned to follow? – Andy Dent Jan 15, 2009 at 12:53

Not really that I've noticed; in principle I suppose you could really treat each tagged piece as its own module and give it its own front page, but we just document as usual and count on Doxygen to link everything together in the end. – Crashworks Jan 16, 2009 at 7:08

15 I use a subversion post-commit hook to pull out the directories that have changed, write them to a file and then every night I automatically re-generate the doxygen html

votes

on our webserver so we always have up-to-date docco.

Every project I want documented has a little project.doxy file that contains the per-project settings and an include to the main doxygen settings - eg:

```
PROJECT_NAME           = "AlertServer"
PROJECT_NUMBER         = 8.1.2
INPUT                  = "C:/Dev/src/8.1.2/Common/AlertServer"
HTML_OUTPUT            = "AlertServer"
@INCLUDE = "c:\dev\CommonConfig.doxy"
```

For Windows SVN server, use the hook:

```
@echo off
for /F "eol=¬ delims=¬" %%A in ('svnlook dirs-changed %1 -r %2') do echo %%A >>
c:\svn_exports\export.txt
```

and then run this nightly:

```
@echo off

rem ---------------
rem remove duplicates.
type nul> %TEMP%.\TEMP.txt

for /F "eol=¬ delims=¬" %%a in (c:\svn_exports\export.txt) do (
 findstr /L /C:"%%a" < %TEMP%.\TEMP.txt > nul
 if errorlevel=1 echo %%a>> %TEMP%.\TEMP.txt
)

copy /y %TEMP%.\TEMP.txt export_uniq.cmd >nul
if exist %TEMP%.\TEMP.txt del %TEMP%.\TEMP.txt


rem ---------------
rem fetch all the recently changed directories into the svn_exports directory

for /F "eol=¬ delims=¬" %%A in (c:\svn_exports\export_uniq.cmd) do (
  svn export "file:///d:/repos/MyRepo/%%A" "c:/svn_exports/%%A"  --force
)


rem ---------------
rem search through all dirs for any config files, if found run doxygen

for /R c:\svn_exports %%i in (*.doxy) do c:\tools\doxygen\bin\doxygen.exe "%i"


rem ---------------
rem now remove the directories to be generated.
del /F c:\svn_exports
```

this removes duplicate entries, finds all projects that have a .doxy project file, and runs doxygen on them. Voila: fully documented, always up-to-date code on a webserver.

Share

answered Feb 20, 2009 at 13:02

gbjbaanb

52.6k ● 12 ● 110 ● 154

What is the purpose of INPUT in your file? – Léo Léopold Hertz 준영 Aug 30, 2009 at 13:11

It seems to be the location of your project. – Léo Léopold Hertz 준영 Aug 30, 2009 at 13:11

Does the following include some default settings for your doxygen?
c:\dev\CommonConfig.doxy – Léo Léopold Hertz 준영 Aug 30, 2009 at 13:13

My comment was an example. You can read up on doxygen's options here:
stack.nl/~dimitri/doxygen/config.html – gbjbaanb Aug 30, 2009 at 14:20

13

votes

For complex projects it may be useful to have a separate file for module management, which controls the groups and subgroups. The whole hierarchy can be in one place and then each file can simply stuff to the child groups. e.g.:

```
/**
 * @defgroup example Top Level Example Group
 * @brief    The Example module.
 *
 * @{
 */

/**
 * @defgroup example_child1 First Child of Example
 * @brief    1st of 2 example children.
 */

/**
 * @defgroup example_child2 Second Child of Example
 * @brief    2nd of 2 example children.
 */

// @}
```

Simply including the definition of a group within the { } of another group makes it a child of that group. Then in the code and header files functions can just be tagged as part of whatever group they are in and it all just works in the finished documentation. It makes refactoring the documentation to match the refactor code much easier.

Share

edited Sep 19, 2008 at 23:36

answered Sep 9, 2008 at 19:42

Andrew Johnson

7,289 ● 4 ● 25 ● 27

1 How can one hide the separate file from actual documentation? I am taking this approach (separate file 'doxyMain.h' for groups), and mark its documentation as internal to exclude it from documentation (cannot exclude the file itself because it defines the groups but it is not really part of the API). Now it shows doxyMain.h file in the file-list but without actual source. Next I will try to see if I can make it not to include it at all (it is a file solely for the purpose of documentation). – dashesy Mar 20, 2013 at 17:21

10 votes

Uses lots and lots of links. This can be done using see also links (\see or @see if you prefer), and making sure that you use any references to other class names in documentation by their correct class name. For example if you refer to class FUZZYObject as an "object", then write immediately after it the name of the class (e.g. "frazzle the objects (FUZZYObject)").

Share

answered Oct 18, 2009 at 3:47

cdiggins

18.2k ● 7 ● 109 ● 107

9 votes

Automatically build and publish your documentation. As part of automatically building the documentation, pay attention to the warnings, its very easy to write badly structure doxygen comments.

Share

answered Dec 30, 2009 at 15:40

John Naegle

8,247 ● 3 ● 41 ● 48

8 votes

Use `\example` as much as you can. It auto-links API elements to example code.

Share

answered Sep 13, 2012 at 13:58

cdiggins

18.2k ● 7 ● 109 ● 107

8 votes

Don't bother with @author or @date (@date was mentioned in another post). These are both handled by a revision control system.

Share

answered Sep 15, 2012 at 0:21

Jim Tshr

357 ● 1 ● 3 ● 10

1 how? subversion docs don't tell me about doxygen, and doxygen docs don't tell me about subversion. an example would be helpful. – ivo Welch Jun 22, 2014 at 1:40

1 When you reintegrate a dev branch in the trunk, you become the owner of all the modifications in the files. When the reintegration is done by another people than the developper, you loose the information. – Caduchon Mar 10, 2016 at 12:39

7 votes

Always include a description with your classes. Try to say how a class is used, or why it is used, not just what it is (which usually just reflects the name anyway).

Share

answered Oct 18, 2009 at 3:51

cdiggins
18.2k ● 7 ● 109 ● 107

4 votes

Group your member functions and fields if it makes sense to do so with \defgroup. This is very helpful, even if you don't say much.

Share

answered Oct 18, 2009 at 3:49

cdiggins
18.2k ● 7 ● 109 ● 107

4 votes

If you are worried that some team members will avoid documenting or you just want a working minimal sets of documentation, you can enable these in your doxygen configuration.

```
WARNINGS               = YES
WARN_IF_UNDOCUMENTED   = YES
WARN_IF_DOC_ERROR      = YES
```

As part of your doxygen build process save the warnings to a file and try to get and keep the warning count as low as possible (0 if that is reasonable). If you do this, every public and protected class member will need at least an @brief, @param for each function argument and an @return. This is good enough to describe most APIs and not too much to encumber other living codebases.

You should, of course, encourage people to document as much as they feel is required on a case by case basis, as long as they meet the minimum project standards. Don't set the minimum too high though, then you may not get useful documentation in the end.

For example, in our project, everything another coder is likely to touch should be documented. Enabling the warnings let see how close that goal we are. We also try to use @internal to describe what/why we do what we do with some of our private members.

Share

edited Dec 3, 2015 at 16:49

answered Jan 23, 2011 at 23:08

Sqeaky

1,936 ● 3 ● 22 ● 42

3

votes

If you find that the configuration directive INLINE_SOURCES puts too much code in the documentation, you can manually quote specific portions of the code using the \snippet command.

```
/**
 * Requirment XYZ is implemented by the following code.
 *
 * \snippet file.c CODE_LABEL
 */
int D()
{
   //[CODE_LABEL]
   if( A )
   {
      B= C();
   }
   //[CODE_LABEL]
}
```

note: snippet gets its files from the EXAMPLE_PATH, not where the source path is. You will have to put the same list of files and paths from INPUT directive on the EXAMPLE_PATH directive.

Share

answered Dec 3, 2015 at 20:10

Michael

2,250 ● 1 ● 20 ● 27

2

votes

For larger projects taking 5+min to generate, I found it useful to quicly be able to generate doxygen for a single file and view it in a web browser.

While references to anything outside the file won't resolve, it can still useful to see the basic formatting is ok.

This script takes a single file and the projects doxy config and runs doxygen, I've set this up to run from my IDE.

```
#!/usr/bin/env python3
"""
This script takes 2-3 args: [--browse] <Doxyfile> <sourcefile>

  --browse will open the resulting docs in a web browser.
"""
import sys
```

```
import os
import subprocess
import tempfile

doxyfile, sourcefile = sys.argv[-2:]

tempfile = tempfile.NamedTemporaryFile(mode='w+b')
doxyfile_tmp = tempfile.name
tempfile.write(open(doxyfile, "r+b").read())
tempfile.write(b'\n\n')
tempfile.write(b'INPUT=' + os.fsencode(sourcefile) + b'\n')
tempfile.flush()

subprocess.call(("doxygen", doxyfile_tmp))
del tempfile

# Maybe handy, but also annoying as default.
if "--browse" in sys.argv:
    import webbrowser
    webbrowser.open("html/files.html")
```

Share

edited Feb 6, 2014 at 13:01

answered Feb 6, 2014 at 12:48

ideasman42

47.7k ● 47 ● 216 ● 347