**Doxygen: A Quick Setup and Usage Manual**

**(Introduction to Doxygen, Installation, and Usage manual)**

**Contains: Setup and usage manual, and includes SE Lab Assignment – 1 question.**

**Arun Avinash Chauhan**

Doxygen is a popular tool used to generate documentation from your code. It is what is called a “documentation system”. Doxygen can be used to generate code for a variety of languages but for simplicity and consistency, given it is only one lab session, we will be sticking to using Doxygen to document C++ code.

**How does Doxygen work?**

The two steps to using Doxygen are as follows:

1. To use Doxygen, you need to write comments in your code in a format that Doxygen understands. Comments are to be included in your .h files, primarily, and in your .cpp files. Though “Doxygen won’t use them extensively” [Using Doxygen: Quick Guide; University of Wisconsin Madison] from .cpp files. Doxygen is good at generating documentation for class definitions (variables, methods, etc.) but not so good with documenting the algorithm (which is in your .cpp file)
2. Once you have your .h and .cpp files ready, simply run Doxygen, which generates an ‘html’ folder. Go to that folder and click on ‘index.html’. The documentation of your code is in an easy-to-read html format.

But before we go to using Doxygen, it needs to be installed (duh!).

**Installation of Doxygen:**

Before I get into the installation of Doxygen, I would like you to know that in the Software Industry, a lot of things, almost everything in fact, does not work the first time (unless of course it is preinstalled by your organization). There are also a lot of times when the code you are working on, its dependencies, etc. do not work. This is very normal in an engineer’s life. So here’s a mantra - Google! Everyone does. And if there is something I fail at clarifying / clearing / resolving, you (and I) can both rely on Google. ‘Googling’ is something every software engineer should be well versed with. That note aside, let’s get to installation of Doxygen.

**Installing Doxygen on Ubuntu**

Open terminal and run the following steps:

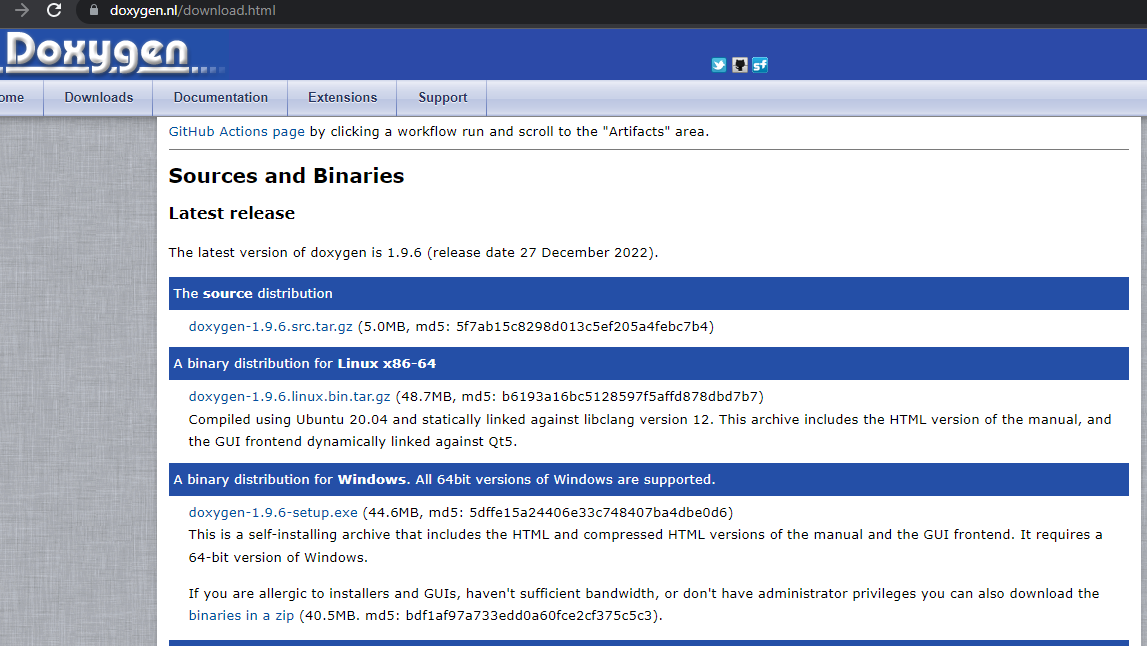
1. > sudo apt-add-repository universe
2. > sudo apt-get update
3. > sudo apt-get install doxygen

<https://askubuntu.com/questions/1291874/how-to-install-doxygen-on-ubuntu-20-04>

**Installing Doxygen on Linux**

Head over to the Doxygen official website given in the link below and download the latest release of Doxygen.

[Doxygen Download Page](https://www.doxygen.nl/download.html)



There, scroll down and download the executable which is in .tar.gz format. The section to download the .tar.gz format from will be labeled ‘A binary distribution for Linux x86-64'. Once downloaded extract the contents, navigate into the folder through your file browser, and right-click there and click open in terminal.

Or alternatively, you can open the terminal and cd into the extracted folder containing the make file with contents shown in the below screenshot.

From there, type the command

> sudo make install

Enter the password when prompted and Doxygen should get installed into /usr/local as shown in the screenshot below.

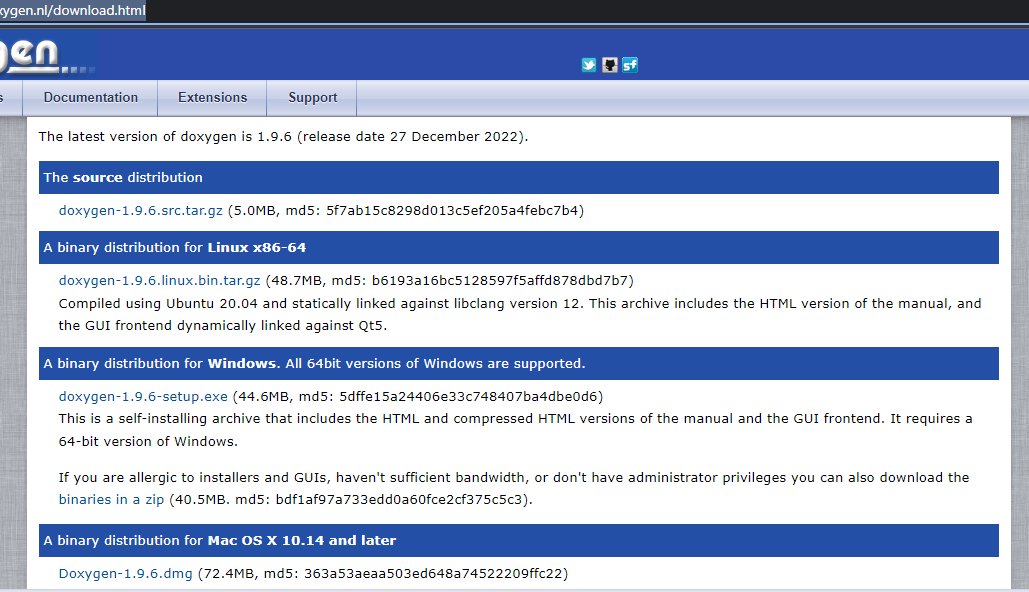
Then type the command below to make the config file

> doxygen -g

**Installing Doxygen on Windows**

Goto:

<https://www.doxygen.nl/download.html>



Select the installation .exe file, and run it as you would any .exe file, and the required components will be installed on your PC

**Installing Doxygen on Macintosh**

Download the installer from the link mentioned in Windows installation. The installer will be of type .dmg

At the point, I am hoping you have installed Doxygen on your systems. Having Doxygen on your systems is a prerequisite to being able to perform Assignment – 1.

**Assignment - 1**

1. **A small example to get Doxygen up and running:**

The following is a sample .h file with comments in a format understood by Doxygen (/\*\* and /// - more on comments later). This file has been adapted from Wikipedia [<http://en.wikipedia.org/wiki/Doxygen>].

**Sample code:**

/\*\*

\* @file

\* @author Jan Doe

\* @version 1.0

\*

\* @section LICENSE

\*

\* This program is free software; you can redistribute it and/or

\* modify it under the terms of the GNU General Public License as

\* published by the Free Software Foundation; either version 2 of

\* the License, or (at your option) any later version.

\*

\* @section DESCRIPTION

\*

\* The time class represents a moment of time.

\*/

/\*\*

\* @brief A sample Time class

\* @author Future DB Guru

\*

\* This is a simple class to demonstrate how Doxygen is used.

\* It implements a dummy Time class.

\*/

class Time {

public:

/\*\*

\* Constructor that sets the time to a given value.

\*

\* @param timemillis Number of milliseconds

\* passed since Jan 1, 1970.

\*/

Time (int timemillis) {

// the code

}

/\*\*

\* Get the current time.

\*

\* @return A time object set to the current time.

\*/

static Time now () {

// the code

}

};

**Using Doxygen to generate documentation (Worked on Ubuntu / Linux but also applicable to Windows and Mac)**

Copy the code above into a file called ‘firstexample.h’. Then run the following command to generate a Doxygen configuration file:

> doxygen –g

This command creates a Doxygen configuration file, which you can edit. The name of this configuration file is ‘Doxyfile’. Use your favorite editor (I used gedit – not my favorite but one I remember) to edit your Doxyfile

> gedit Doxyfile

The main things to edit here are PROJECT\_NAME and INPUT. INPUT mentions the directory in which the C++ files / projects are located. Currently, you can set it to ‘.’ (dot) which means current directory.

PROJECT\_NAME = “SE Lab: Assignment – 1 (Try Doxygen)”

INPUT = .

For clarity, I suggest you create a directory called ‘**SELab-Assignment1**’ in your ‘Home’ directory and then set the path in INPUT to that directory. In case you made a directory SELab-Assignment1 and have your code ‘firstexample.h’ in that directory, the INPUT variable should be set as follows (Note, bunny is my username, you have your own path to that directory)

PROJECT\_NAME = “SE Lab: Assignment – 1 (Try Doxygen)”

INPUT = /home/bunny/SELab-Assignment1

This done, finally set the HAVE\_DOT variable in the Doxyfile to NO. By default it is set to YES, but most systems do not have the dot tool preinstalled. Therefore, just set the variable to NO.

HAVE\_DOT = NO

The final step is to generate the documentation, by running

> doxygen Doxyfile

The above command will create an html directory, with a file called index.html. Open that file in a browser, and you can read the documentation for your code. There are other parameters (with supported help comments) in the Doxyfile. Go through those parameters, tinker and change, see what happens!

**Quick Tutorial:**

Now that the software is installed and basic documentation generated (I am assuming that by this point, it is installed), let's get right into some of the basics:

Doxygen works by taking the comments which are specifically formatted according to Doxygen’s pre-agreed syntax. Doxygen calls these special syntaxes as “Tags”. Let us see some of those before we actually get into using them!

**Doxygen tags and symbols to use in your comments**

Doxygen supports several methods for incorporating documentation inside the comments. The 2 simplest methods to include in C++ source code are

/\*\* /\*\*<

… comments… (or in case the comments are after an entity) … comments…

\*/ \*/

And

/// Comments.

Note the extra asterisk (\*) in line 1 of the first option and extra slash symbol (/), (i.e. in C++ we just need 2 slashes for a comment line) These syntaxes are used to tell the Doxygen parser that it is documentation and it needs to be extracted. The above 2 syntaxes must be placed just above an entity so that the Doxygen parser can associate the comments to that particular entity.

**What is an entity in Doxygen? An entity can be a global variable, a structure declaration, enum declaration, or a function.**

Next, let’s look at the documentation of parameters. Here another special Doxygen syntax is used which is the @ syntax. Doxygen calls these structural commands. There are several of them and the most useful ones are the following:

@file, The file name must be present in the file header for inclusion into the documentation generation process

@param, Parameter documentation for functions

@page, Markdown page name

@mainpage, Main markdown page for the project

@tableofcontents, Generates “table of contents” for the markdown page

Here is a list of the most frequently used doxygen tags:

Here is a list of some of the most frequently used doxygen tags:

@file @return  
@author @param  
@date @pre  
@version @post  
@mainpage @throw

The tags in the first column above are used for documenting an individual file, or a project, while those in the second column are used for documenting a function. Coming back to the assignment:

1. **A second, more elaborate program on doxygen**

Next, we take up another short example of doxygen documentation. Here a sample file, [doxygen\_example.cpp](https://cs.smu.ca/~porter/csc/common_341_342/doxygen/doxygen_example.cpp2html) is provided, which contains a short, but more elaborate example of doxygen documentation. At the same time, it gives a good idea about doxygen documentation.

Download doxygen\_example.cpp code from the link above, save it to the previously created directory SELab-Assignment1 and run the steps you did while generating the documentation for firstexample.h.

This done, go back again to the html directory which was previously created inside the directory ‘SELab-Assignment1’, and open the file ‘index.html’. *What do you see? Check the documentation generated for doxygen\_example.cpp. How is it different from the documentation generated earlier?*

In the short example given above, look for the above-mentioned doxygen comments and tags, as well as some HTML tags, and then study the corresponding HTML output to see the effect of each tag. (or in other words), Open doxygen\_example.cpp and understand how the various comments and comment tags are given, and map them (or see how they reflect) at the output documentation generated by Doxygen.

**A short formal note on the working of Doxygen (from** [**Getting started on Doxygen**](https://cs.smu.ca/~porter/csc/common_341_342/doxygen/doxygen_info.html)**)**

When the doxygen tool processes one or more input source code files, it produces, by default, in the current working directory, a subdirectory named html, which contains all the HTML files that document that source code, according to the doxygen comments contained in the source code. The starting point for browsing this documentation, as you might expect, is the file index.html.

For the above example, to produce the html directory and all the files in it, you just have to put the two files doxygen\_example.cpp and the doxygen configuration file Doxyconfig in the directory where you want the html directory to appear, and then enter this command:

prompt> doxygen Doxyconfig

The doxygen configuration file for any source code file or project can have any name you like. However, it will generally be convenient to give a configuration file for a given source code file or a given project a name that shows its connection with that file or project. If you look at the configuration file for the example you can easily see that for a different project you will want to change the values of PROJECT\_NAME, PROJECT\_NUMBER and INPUT, but you can probably leave the other parameter values set as they are. Note that INPUT contains a whitespace separated list of all files that you want doxygen to process.

1. **In the final part of the assignment, you can take any of your existing .cpp and .h files, preferably multiple files, and comment them in such a manner that when you run doxygen, the documentation is generated**

**Lab Assignment – 1, therefore consists of the following and is worth 10 M**

1. Installation of Doxygen (2M)

2. Generating documentation (8 M)

* 1. Generating documentation for first sample (2M)
  2. Generating documentation for second sample (2M)
  3. Generating documentation for .cpp and .h files from your existing code (4M)