**BASICS OF DOXYGEN**

1.Compiling from source on Windows install bison, flex, and tar.

2.Download doxygen's source tarball and put it somewhere (e.g. use c:\tools)

**Now start a new command shell and type**

1. cd c:\tools
2. tar zxvf doxygen-x.y.z.src.tar.gz

to unpack the sources.Now your environment is setup to build doxygen. Inside the doxygen-x.y.z directory you will find a winbuild directory containing a Doxygen.sln file. Open this file in Visual Studio.

**Installing the binaries on Windows**

1. Doxygen can use the dot tool of the GraphViz package to render nicer diagrams.
2. If you set the HAVE\_DOT tag to YES then doxygen will assume the dot tool is available from the path.
3. This tool is part of Graphviz, a graph visualization toolkit from AT&T and Lucent Bell Labs
4. If you want to produce compressed HTML files (see GENERATE\_HTMLHELP) in the config file, then you need the Microsoft HTML help workshop.
5. If you want to produce Qt Compressed Help files (see QHG\_LOCATION) in the config file, then you need qhelpgenerator which is part of Qt. You can download Qt from Qt Software Downloads.
6. In order to generate PDF output or use scientific formulas you will also need to install LaTeX and Ghostscript.
7. For LaTeX a number of distributions exists. Popular ones that should work with doxygen are MikTex and XemTex.
8. Ghostscript can be downloaded from Sourceforge.
9. After installing LaTeX and Ghostscript you'll need to make sure the tools latex.exe, pdflatex.exe, and gswin32c.exe are present in the search path of a command box.
10. The executable doxygen is the main program that parses the sources and generates the documentation.
11. Check if doxygen supports your programming language.These languages are supported by default: C, C++, C#, Objective-C, IDL, Java, VHDL, PHP, Python, Tcl, Fortran, and D. It is possible to configure certain file type extensions to use certain parsers.
12. Creating a configuration file. Doxygen uses a configuration file to determine all of its settings. Each project should get its own configuration file.
13. To simplify the creation of a configuration file, doxygen can create a template configuration file for you. To do this call doxygen from the command line with the -g option:

doxygen -g <config-file>

* where <config-file> is the name of the configuration file. If you omit the file name, a file named Doxyfile will be created. If a file with the name <config-file> already exists,
* doxygen will rename it to <config-file>.bak before generating the configuration template.
* For a small project consisting of a few C and/or C++ source and header files, you can leave INPUT tag empty and doxygen will search for sources in the current directory.

**Running doxygen**

1. To generate the documentation you can now enter:

doxygen <config-file>

1. Depending on your settings doxygen will create html, rtf, latex, xml and/or man directories inside the output directory.
2. As the names suggest these directories contain the generated documentation in HTML, RTF, , XML and Unix-Man page format
3. doxygen writes a Makefile into the latex directory (on the Windows platform also a make.bat batch file is generated).
4. Conversion to PDF is also possible if you have installed the ghostscript interpreter; just type make pdf (or make pdf\_2on1).
5. To get the best results for PDF output you should set the PDF\_HYPERLINKS and USE\_PDFLATEX tags to YES.
6. In this case the Makefile will only contain a target to build refman.pdf directly.

**RTF output**

Doxygen combines the RTF output to a single file called refman.rtf. This file is optimized for importing into the Microsoft Word.

**XML output**

The XML output consists of a structured "dump" of the information gathered by doxygen. Each compound (class/namespace/file/...) has its own XML file and there is also an index file called index.xml.

A file called combine.xslt XSLT script is also generated and can be used to combine all XML files into a single file.

Doxygen also generates two XML schema files index.xsd (for the index file) and compound.xsd (for the compound files).

This schema file describes the possible elements, their attributes and how they are structured, i.e. it the describes the grammar of the XML files and can be used for validation or to steer XSLT scripts.

In the addon/doxmlparser directory you can find a parser library for reading the XML output produced by doxygen in an incremental way (see addon/doxmlparser/include/doxmlintf.h for the interface of the library)

**Man page output**

The generated man pages can be viewed using the man program. You do need to make sure the man directory is in the man path (see theMANPATH environment variable).

If the [**EXTRACT\_ALL**](http://www.stack.nl/~dimitri/doxygen/manual/config.html#cfg_extract_all) option is set to NO in the configuration file (the default), then doxygen will only generate documentation for documented entities.

If you want to document the members of a file, struct, union, class, or enum, it is sometimes desired to place the documentation block after the member instead of before. For this purpose you have to put an additional < marker in the comment block. Note that this also works for the parameters of a function.

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

For functions one can use the [**@param**](http://www.stack.nl/~dimitri/doxygen/manual/commands.html#cmdparam) command to document the parameters and then use [in], [out], [in,out] to document the direction.

void foo(int v /\*\*< [in] docs for input parameter v. \*/);

Here the special command \class is used to indicate that the comment block contains documentation for the class Test. Other structural commands are:

* \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.

The design goal for Markdown's formatting syntax is to make it as readable as possible.

Markdown: to make a paragraph you just separate consecutive lines of text by one or more blank lines.

 header is followed by a line containing only ='s or -'s.

# This is a level 1 header

### This is level 3 header #######

Block quotes can be created by starting each line with one or more >'s

Simple bullet lists can be made by starting a line with -, +, or \*.

- Item 1

More text for this item.

- Item 2

+ nested list item.

+ another nested item.

- Item 3

## Code Blocks

Preformatted verbatim blocks can be created by indenting each line in a block of text by at least 4 extra spaces

A horizontal ruler will be produced for lines containing at least three or more hyphens, asterisks, or underscores. The line may also include any amount of whitespace.

Examples:

- - -

\_\_\_\_\_\_

To emphasize a text fragment you start and end the fragment with an underscore or star. Using two stars or underscores will produce strong emphasis.

Examples:

\*single asterisks\*

\_single underscores\_

\*\*double asterisks\*\*

\_\_double underscores\_\_

To indicate a span of code, you should wrap it in backticks (`).

Use the `printf()` function.

Doxygen supports both styles of make links defined by Markdown: inline and reference.

### Inline Links

[The link text](http://example.net/)

### Reference Links

[link name]: http://www.example.com "Optional title"

## Images

Markdown syntax for images is similar to that for links. The only difference is an additional ! before the link text.

Examples:

![Caption text](/path/to/img.jpg)

## Automatic Linking

To create a link to an URL or e-mail address Markdown supports the following syntax:

<http://www.example.com>

<address@example.com>

Note that doxygen will also produce the links without the angle brackets.

## Tables

First Header | Second Header

------------- | -------------

Content Cell | Content Cell

Content Cell | Content Cell

~~~~~~~~~~~~~{.py(file extension)}

# A class

class Dummy:

pass

~~~~~~~~~~~~~

which will produce:

1 # A class

2 class Dummy:

3  pass

Markdown is quite strict in the way it processes block-level HTML:

block-level HTML elements â€” e.g. <div>, <table>, <pre>, <p>, etc. â€” must be separated from surrounding content by blank lines, and the start and end tags of the block should not be indented with tabs or spaces.

**Lists**

By putting a number of column-aligned minus (-) signs at the start of a line, a bullet list will automatically be generated. Instead of the minus sign also plus (+) or asterisk (\*) can be used.

There are three ways to include formulas in the documentation.

1. Using in-text formulas that appear in the running text. These formulas should be put between a pair of \f$ commands, so
2. The distance between \f$(x\_1,y\_1)\f$ and \f$(x\_2,y\_2)\f$ is
3. \f$\sqrt{(x\_2-x\_1)^2+(y\_2-y\_1)^2}\f$.

results in:

The distance between $(x_1,y_1)$ and $(x_2,y_2)$ is $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$.

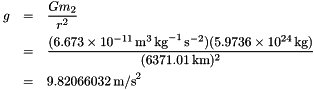
1. Unnumbered displayed formulas that are centered on a separate line. These formulas should be put between \f[ and \f] commands. An example:
2. \f[
3. |I\_2|=\left| \int\_{0}^T \psi(t)
4. \left\{
5. u(a,t)-
6. \int\_{\gamma(t)}^a
7. \frac{d\theta}{k(\theta,t)}
8. \int\_{a}^\theta c(\xi)u\_t(\xi,t)\,d\xi
9. \right\} dt
10. \right|
11. \f]

results in:

\[ |I_2|=\left| \int_{0}^T \psi(t) \left\{ u(a,t)- \int_{\gamma(t)}^a \frac{d\theta}{k(\theta,t)} \int_{a}^\theta c(\xi)u_t(\xi,t)\,d\xi \right\} dt \right| \]

1. Formulas or other latex elements that are not in a math environment can be specified using \f{environment}, where environment is the name of the $\mbox{\LaTeX}$ environment, the corresponding end command is \f}. Here is an example for an equation array
2. \f{eqnarray\*}{
3. g &=& \frac{Gm\_2}{r^2} \\
4. &=& \frac{(6.673 \times 10^{-11}\,\mbox{m}^3\,\mbox{kg}^{-1}\,
5. \mbox{s}^{-2})(5.9736 \times 10^{24}\,\mbox{kg})}{(6371.01\,\mbox{km})^2} \\
6. &=& 9.82066032\,\mbox{m/s}^2
7. \f}

which results in:



Doxygen can use the "dot" tool from graphviz to generate more advanced diagrams and graphs. Graphviz is an open-source, cross-platform graph drawing toolkit and can be found at <http://www.graphviz.org/>

If you have the "dot" tool in the path, you can set [**HAVE\_DOT**](http://www.stack.nl/~dimitri/doxygen/manual/config.html#cfg_have_dot) to YES in the configuration file to let doxygen use it.

Source files that are used as input to doxygen can be parsed by doxygen's built-in C-preprocessor.

By default doxygen does only partial preprocessing. That is, it evaluates conditional compilation statements (like #if) and evaluates macro definitions, but it does not perform macro expansion.

By setting the following in the config file:

ENABLE\_PREPROCESSING = YES

MACRO\_EXPANSION = YES

EXPAND\_ONLY\_PREDEF = YES

PREDEFINED = "DECLARE\_INTERFACE(name)=class name" \

"STDMETHOD(result,name)=virtual result name" \

"PURE= = 0" \

THIS\_= \

THIS= \

\_\_cplusplus

The following output formats are *directly* supported by doxygen:

**HTML**

Generated if GENERATE\_HTML is set to YES in the configuration file.

**$\mbox{\LaTeX}$**

Generated if GENERATE\_LATEX is set to YES in the configuration file.

**Man pages**

Generated if GENERATE\_MAN is set to YES in the configuration file.

**RTF**

Generated if GENERATE\_RTF is set to YES in the configuration file.

Note that the RTF output probably only looks nice with Microsoft's Word. If you have success with other programs, please let me know.

**XML**

Generated if GENERATE\_XML is set to YES in the configuration file.

The following output formats are *indirectly* supported by doxygen:

**Compiled HTML Help (a.k.a. Windows 98 help)**

Generated by Microsoft's HTML Help workshop from the HTML output if GENERATE\_HTMLHELP is set to YES.

**Qt Compressed Help (.qch)**

Generated by Qt's qhelpgenerator tool from the HTML output if GENERATE\_QHP is set to YES.

**Eclipse Help**

Generated from HTML with a special index file that is generated when GENERATE\_ECLIPSEHELP is set to YES.

**XCode DocSets**

Compiled from HTML with a special index file that is generated when GENERATE\_DOCSET is set to YES.

**PostScript**

Generated from the $\mbox{\LaTeX}$ output by running make ps in the output directory. For the best results PDF\_HYPERLINKS should be set to NO.

**PDF**

Generated from the $\mbox{\LaTeX}$ output by running make pdf in the output directory. To improve the PDF output, you typically would want to enable the use of pdflatex by setting [**USE\_PDFLATEX**](http://www.stack.nl/~dimitri/doxygen/manual/config.html#cfg_use_pdflatex) to YES in the configuration file. In order to get hyperlinks in the PDF file you also need to enable [**PDF\_HYPERLINKS**](http://www.stack.nl/~dimitri/doxygen/manual/config.html#cfg_pdf_hyperlinks).

## 1. Client side searching

To enable it set [**SEARCHENGINE**](http://www.stack.nl/~dimitri/doxygen/manual/config.html#cfg_searchengine) to YES in the config file and make sure [**SERVER\_BASED\_SEARCH**](http://www.stack.nl/~dimitri/doxygen/manual/config.html#cfg_server_based_search) is set to NO.

## 2. Server side searching

To enable this set both [**SEARCHENGINE**](http://www.stack.nl/~dimitri/doxygen/manual/config.html#cfg_searchengine) and [**SERVER\_BASED\_SEARCH**](http://www.stack.nl/~dimitri/doxygen/manual/config.html#cfg_server_based_search) to YES in the config file and set [**EXTERNAL\_SEARCH**](http://www.stack.nl/~dimitri/doxygen/manual/config.html#cfg_external_search) to NO.

## 3. Server side searching with external indexing

To enable this search method set [**SEARCHENGINE**](http://www.stack.nl/~dimitri/doxygen/manual/config.html#cfg_searchengine), [**SERVER\_BASED\_SEARCH**](http://www.stack.nl/~dimitri/doxygen/manual/config.html#cfg_server_based_search) and [**EXTERNAL\_SEARCH**](http://www.stack.nl/~dimitri/doxygen/manual/config.html#cfg_external_search) all to YES.

## 4. Windows Compiled HTML Help

If you are running doxygen on Windows, then you can make a compiled HTML Help file (.chm) out of the HTML files produced by doxygen. This is a single file containing all HTML files and it also includes a search index. There are viewers for this format on many platforms, and Windows even supports it natively.

To enable this set [**GENERATE\_HTMLHELP**](http://www.stack.nl/~dimitri/doxygen/manual/config.html#cfg_generate_htmlhelp) to YES in the config file. To let doxygen compile the HTML Help file for you, you also need to specify the path to the HTML compiler (hhc.exe) using the [**HHC\_LOCATION**](http://www.stack.nl/~dimitri/doxygen/manual/config.html#cfg_hhc_location) config option and the name of the resulting CHM file using [**CHM\_FILE**](http://www.stack.nl/~dimitri/doxygen/manual/config.html#cfg_chm_file).

## 5. Eclipse Help Plugin

To enable the help plugin set [**GENERATE\_ECLIPSEHELP**](http://www.stack.nl/~dimitri/doxygen/manual/config.html#cfg_generate_eclipsehelp) to YES, and define a unique identifier for your project via [**ECLIPSE\_DOC\_ID**](http://www.stack.nl/~dimitri/doxygen/manual/config.html#cfg_eclipse_doc_id), i.e.:

GENERATE\_ECLIPSEHELP = YES

ECLIPSE\_DOC\_ID = com.yourcompany.yourproject

## To avoid that everyone has to start writing their own indexer and search engine, doxygen provides an example tool for each action:doxyindexer for indexing the data and doxysearch.cgi for searching through the index.

## © copyright- Gaurav Panwar