Записки программиста

Блог о программировании, а также электронике, радио, и всяком таком

Генерируем документацию к проектам на C/C++ с Doxygen

12 февраля 2018

Программисты, как правило, не очень любят писать тесты. Но куда сильнее они не любят писать документацию. Тесты хотя бы представляют собой программы, а документация — это что? Просто текст. Вот пусть кто-нибудь другой его и пишет, технические писатели например! Впрочем, если речь идет не о пользовательской документации, а об описании классов и методов, предназначенном для других программистов, тут откосить вряд ли удастся. К счастью, есть Doxygen, способный существенно помочь со столь неприятной для многих работой.

Fun fact! Несмотря на то, что в рамках этого поста мы будем говорить о Doxygen исключительно в контексте языков C и C++, он также поддерживает языки Java, Python, PHP, и другие.

Установка Doxygen осуществляется как-то так:

```
# если у вас Ubuntu:
sudo apt-get install doxygen

# если у вас Arch Linux:
sudo pacman -S doxygen
```

Далее переходим в каталог с нашим проектом и создаем шаблонный файл Doxyfile командой:

```
doxygen -g
```

В Doxyfile содержится краткое описание проекта, его версия и подобные вещи. Следующие значения стоит сразу изменить.

Имя проекта:

```
PROJECT_NAME           = "Project Name"
```

Версия проекта:

```
PROJECT_NUMBER         = 0.1
```

Краткое описание проекта:

```
PROJECT_BRIEF           = "Yet another NoSQL DBMS"
```

Куда писать сгенеренные доки:

```
OUTPUT_DIRECTORY        = doxygen
```

Отключаем LaTeX, так как HTML обычно достаточно:

```
GENERATE_LATEX          = NO
```

Где искать файлы, из которых генерировать документацию — список файлов и директорий через пробел:

```
INPUT                   = src include
```

Включаем рекурсивный обход директорий:

```
RECURSIVE               = YES
```

Всего в файле около 2500 строк с подробным описанием всех параметров. Однако помимо приведенных выше параметров вам вряд ли придется что-то менять.

Правила хорошего тона гласят, что для каждой процедуры или метода класса (независимо от того, публичные они или приватные) должно быть как минимум краткое описание того, что делает процедура или метод, описание каждого из аргументов, а также описание возвращаемого значения. Если метод делает еще что-то, например, как-то меняет состояние класса, это также обязательно нужно задокументировать.

Аналогично, краткое описание того, что они из себя представляют, должно быть у самих классов и структур. Не лишним будет рассказать, является ли класс копируемым или thread-safe. Краткое описание должно быть у каждого атрибута класса.

Рассмотрим пример:

```
/**
 * @brief Entry point
 *
 * Execution of the program
 * starts here.
 *
 * @param argc Number of arguments
 * @param argv List of arguments
 *
 * @return Program exit status
 */
int main(int argc, char** argv) {
```

Doxygen генерирует документацию из комментариев, начинающихся с /** или /*!. Поддерживаются и другие метки, но, по моим наблюдениям, на практике они используются

реже.

Для обозначения того, к чему именно относится часть комментария, используются специальные тэги. В терминологии Doxygen они зовутся командами. Команды начинаются со знака @ или \. К наиболее частым командам я бы лично отнес следующие:

- @brief — краткое описание метода или класса;
- @param foo — описание аргумента foo процедуры или метода;
- @return — описание возвращаемого значения;
- @class Foo — описание конкретного класса Foo;
- @file fname — описание конкретного файла;
- @mainpage Title — комментарий содержит текст для титульного листа документации;
- @see Ref — ссылка на связанный класс или метод;
- @throws Foo или @exception Foo — метод бросает исключение Foo;
- @warning — предупреждение. Очень ярко выделяется в документации, трудно не заметить;
- @private — пометить класс или метод, как приватный. Удобно, когда не хочется палить в документации какие-то детали реализации;
- @todo — что-то нужно доделать. Doxygen генерирует отдельную страницу со списком всех @todo;
- @deprecated — помечает класс или метод устаревшим. Как и с @todo, Doxygen генерирует отдельную страницу со списком всех устаревших классов и методов;

В действительности, Doxygen поддерживает куда больше команд. Например, он позволяет писать многостраничную (@page) документацию с разделами (@section) и подразделами (@subsection), указывать версии методов и классов (@version), и не только. Ознакомиться с полным списком команд можно здесь.

Fun fact! Doxygen понимает Markdown в комментариях. Краткую шпаргалку по Markdown вы найдете в заметке про создание статического блога на Pelican.

Для генерации и просмотра документации просто говорим:

```
doxygen
firefox doxygen/html/index.html
```

Если вас интересуют конкретные примеры, рекомендую посмотреть документацию по libevent, wxWidgets или Assimp. Любой проект куда проще поддерживать, если у него есть такая же классная документация.

Дополнение: Вас также может заинтересовать пост Построение UML-диаграмм с помощью PlantUML. В Doxygen имеется поддержка PlantUML.

Метки: C/C++.

Вы можете прислать свой комментарий мне на почту, или воспользоваться комментариями в Telegram-группе.

- ## Коротко о себе

  Меня зовут Александр, позывной любительского радио R2AUK. Здесь я пишу об интересующих меня вещах и временами — просто о жизни.

  Вы можете следить за обновлениями блога с помощью RSS и Telegram. Также я являюсь одним из ведущих подкаста DevZen и выкладываю видео на YouTube.

  Мой e-mail — *afiskon@gmail.com*. Если вы хотите мне написать, прошу предварительно ознакомиться с FAQ.

- Поиск

- ## Основные рубрики

  - Антенны
  - Беспроводная связь
  - C/C++
  - Go
  - Linux
  - PostgreSQL
  - Python
  - STM32
  - СУБД
  - Электроника