Dorgygen: Source code documentation in org-mode

Stefano Ghirlanda

March 9, 2024

WARNING: This is work in progress. It only supports C and there is no customization.

1 Introduction

Dorgygen is an Emacs package to document source code in Org documents with minimal setup. You may find it useful if you don't want/need full-blown literate programming and if you like source code without distracting markup. With dorgygen, source code is documented with short comments, which are then integrated with longer explanations in Org documents. Take this C file as an example:

```
// the main function
int                 // exit value
main( int argc,     // number of arguments
      char *argv[] // argument values
  );
```

We don't need special markup because dorgygen understands that, for example, "`// number of arguments`" is a comment about "`int argc`," which in turn is a parameter of the function `main` (courtesy of tree-sitter). When it runs, dorgygen scans source files for comments and creates corresponding Org headings and subheadings. The example file above gives:

```
* main.c

** main

- The main function.
- In: ~int argc~. Number of arguments.
- In: ~char *argv[]~. Argument values.
- Out: ~int~. Exit value.
```

Once this documentation is generated, you can add more. For example:

```
* main.c
```

```
Here is documentation about the file.

** main

- The main function.
- In: ~int argc~. Number of arguments.
- In: ~char *argv[]~. Argument values.
- Out: ~int~. Exit value.

Here is documentation about the ~main~ function.
```

When run again, dorgygen updates documentation from source files and preserves your text.

2 Installation

Install manually from here. Hopefully from MELPA soon.

3 Using dorgygen

3.1 Preparing source code

For a C project, one would typically add documentation comments to header files. The template for commenting a C function is

```
// brief function documentation
int // return type documentation
function(
  int arg1, // argument documentation
  ...
  );
```

Any of the documentation comments can be omitted, and additional comments are ignored. You can format code to your liking, provided comments are placed appropriately, and you can use the `/* ... */` style.

A `typedef` is commented through a preceding comment:

```
// a new type
typedef new_float_t float;
```

It will appear as a list item under the file-level heading.

Comments introduced by `//` can span multiple lines, provided they are consecutive:

```
// this function documentation
// is longer than one line
int // return type documentation
function(
  int arg1, // this argument documentation
            // is also longer than one line
  ...
  );
```

3.2 Preparing org documents

There is only one mandatory setting: one heading in the document should have the property `DORG_REX` set to a regular expression that matches the source files to be documented. For example, the following instructs dorgygen to document all C headers in the `src` directory:

```
* Documentation
:PROPERTIES:
:DORG_REX: src/\.h$
:END:
```

Characters that are special in Emacs regular expressions must be escaped with a backslash.

Dorgygen looks for the first header with the `DORG_REX` property around the point of invocation. Thus you can have multiple documentation sections in the same document, each with its own `DORG_REX` property.

The optional property `DORG_LAN` specifies a programming language. If it is not set, each file's extension is used to guess the language.

3.3 Generating documentation

After you have commented source files and prepared the Org document, you just run `M-x dorgygen`.

4 Customization

Running `M-x customize-group RET dorgygen` gives access to some customization:

- `doxygen-attr-list`: The content of this variable is added before each list generated by dorgygen. For example, you can set it to `#attr_latex: ...` to customize LaTeX export.

5 TODO Bugs and limitations

Please submit bugs and feature requests as issues on Github.