L05 Documentation

Markus Raab

Institute of Information Systems Engineering, TU Wien

Introduction

Learning Outcomes

After successful completion of L05
students will be able to use techniques to

- remember basics
- generate documentation
- verify documentation

Correctness

Documentation tends to be:

- outdated
- incorrect
- not helpful

Make Sure That

- documentation gets reviewed
- documentation is in source code management
- small distance to code
- avoid redundant information

Different Types of Documentation

tutorials	for learning
how-to	solving a problem
reference	searching for details e.g. man pages, API docu
explanations	in `doc/dev` "How?"
decisions	for background information "Why?"
examples	for copy&paste

Different People

beginners never forget everybody starts as beginner

advanced understanding how to improve

expert learn how to teach others,
improve upon what the software is doing

L05 Documentation

Markus Raab

Institute of Information Systems Engineering, TU Wien

Generate

Markdown

Minimal formatting abilities but implemented by many tools

- `[Link](/linktarget)` or `[Link](relative/link)`
- `*italics*` and `**bold**`
- `` `inline code` `` or ```` ``` ```` code fences
- ```
  - item
  - item
  - item
  ```
  ```

Views

The same (markdown) file can be viewed via:

- directly viewing the source `doc/help/kdb.md`
- website `https://www.libelektra.org/manpages/kdb` (rendered by marked)
- API docu
  `https://doc.libelektra.org/api/master/html/doc_help_kdb_md.html`
  (rendered by doxygen)
- `man kdb` (rendered by ronn)
- `kdb -help` or `kdb help <command>`
- GitHub `https://master.libelektra.org/doc/help/kdb.md`
- From qt-gui (rendered by discount)

→ 5 different markdown renderer involved

Plugins

For plugins, documentation even changes the build process:

```
1 - infos = Information about the toml plugin is in keys below
2 - infos/author = Jakob Fischer <jakobfischer93@gmail.com>
3 - infos/licence = BSD
4 - infos/provides = storage/toml
5 - infos/needs = base64
6 - infos/recommends = type
7 - infos/placements = getstorage setstorage
8 - infos/status = experimental unfinished
9 - infos/metadata = order comment/# comment/#/start comment/#/s
10 - infos/description = This storage plugin reads and writes TOM
```

Conclusion

- reuse of documentation by generation
- avoids duplication and errors
- avoids CI checks for inconsistencies
- generation by CI

L05 Documentation

Markus Raab

Institute of Information Systems Engineering, TU Wien

Verify

Goals

Not every documentation-related task can be generated:

- in the text we want to refer to the behavior
- we want to verify if given examples are correct

Problems with Unit Tests

1. difficult to read
2. code cannot be directly copied (asserts)
3. cannot be easily integrated in tutorials

$\rightarrow$ specialized verification language for documentation

Verification of Tutorials

```
1 ␣␣␣␣```sh
2 ␣␣␣␣kdb␣set␣user:/tests/something
3 ␣␣␣␣#␣RET:␣0
4
5 ␣␣␣␣kdb␣get␣user:/tests/something
6 ␣␣␣␣```
```

Syntax

1. starts with ```` ```sh ````
2. comments introduce checks
3. otherwise is shell code to be executed
4. `#>` verifies stdout output
5. `# RET:` verifies return code (if not 0)

Conventions

1. Test data in `/tests`.
2. Generate temporary files if needed or use HERE.

Conclusions

1. If possible, generate.
2. Otherwise, verify.
3. Keep user and purpose in mind.

L05 Documentation

Markus Raab

Institute of Information Systems Engineering, TU Wien

Meeting

Learning Outcomes

After successful completion of L05
students will be able to use techniques to

- remember basics
- generate documentation
- verify documentation

Make Sure That

Task

What should you make sure as FLOSS maintainer in respect to documentation?

Make Sure That

- documentation gets reviewed
- documentation is in source code management
- small distance to code
- avoid redundant information

Make Sure That

Task

Which types of documentation do you need to treat differently?

Different Types of Documentation

tutorials	for learning
how-to	solving a problem
reference	searching for details e.g. man pages, API docu
explanations	in `doc/dev` "How?"
decisions	for background information "Why?"
examples	for copy&paste

Task

Break.

Views

Task

Which different views can be provided by generation of documentation?

Views

The same (markdown) file can be viewed via:

- directly viewing the source `doc/help/kdb.md`
- website `https://www.libelektra.org/manpages/kdb` (rendered by marked)
- API docu
  `https://doc.libelektra.org/api/master/html/doc_help_kdb_md.html`
  (rendered by doxygen)
- `man kdb` (rendered by ronn)
- `kdb -help` or `kdb help <command>`
- GitHub `https://master.libelektra.org/doc/help/kdb.md`
- From qt-gui (rendered by discount)

→ 5 different markdown renderer involved

Conclusions

Task

When do we generate, when do we verify, and for which users?

Conclusions

1. If possible, generate.
2. Otherwise, verify.
3. Keep user and purpose in mind.

H2: Corrections

Task

How did you correct the review notes?

T1: Continuous Integration

Task

Do you have enough tasks?

Task

Fix CI pipeline.

Feedback

- Feedback Talk
- ECTS breakdown realistic?

L06 Entry Barriers