toc2md User Manual

Version 0.0.7

https://pypi.org/project/toc2md/

Carlo Alessandro Verre

carlo.alessandro.verre@gmail.com

june 9, 2020

Contents

1. Before Usage

1.1. Introduction

toc2md is a small CLI utility written in Python 3, aimed to insert or update a nested chapter numbering and a linked table of contents (TOC) into a Markdown (MD) file.

"Nested chapter numbering" means level 1 chapters gets 1-item numbering (1. 2. ...), level 2 chapters gets 2-item numbering (1.1. 1.2. ...) and so on. See "Contents" above.

"Linked table of contents" means that if you click on a chapter title in the table of contents then you jump to chapter body.

In 2. Usage we show by rules and examples how to use toc2md, alone or as a coroutine together with ReText MD editor.

In 3. After Usage we discuss a couple of postprocessing scenarios.

In 4. Conclusion we dare to suggest a comparison with LibreOffice Writer...

1.2. Installation

To install toc2md type at terminal (without sudo!):

```
$ python3 -m pip toc2md
```

After installation, if typing:

```
$ python3 -V
```

result is, for instance:

```
Python 3.8.2
```

then you can find a PDF version of this document in:

```
~/.local/lib/python3.8/site-packages/toc2md/docs/toc2md.pdf
```

2. Usage

2.1. Format

Usage: toc2md [-h] [-V] file_md

Positional argument:

- file_md - MD file to be backed up, read, processed and rewritten

Optional arguments:

- -h, --help - show a help message and exit
- -V, --version - show program's version number and exit

2.2. Logic

If we type at terminal:

```
$ toc2md example.md
```

toc2md program:

- copies example.md into a timestamped backup file example-YYYY.mm.dd-HH:MM:SS.md
- writes on standard output something like:

```
file '/home/xxxx/example.md' has been saved
into '/home/xxxx/example-2020.06.05-18.19.07.md'
```

- reads example.md
- checks correct heading sequence (see below)
- deletes old chapter numbering (if any) and old TOC (if any)
- inserts new nested chapter numbering and new linked TOC
- rewrites the updated version into example.md
- writes on standard output something like:

```
file '/home/xxxx/example.md' has been updated
```

There is neither configuration file nor options nor arguments nor parameters nor commands nor statements. Logic has been kept as simple as possible, but the file to be processed must obey some rule, as follows.

toc2md processing is driven by headings in input file. A heading for us is a line which starts with a '#' character. The heading level is the number of leading '#' characters, between 1 and 6. So a heading is made by:

- a string of 1 to 6 '#' characters
- a blank
- a title

'#' characters after the sixth are silently discarded. The blank after '#' characters if not present is silently inserted. Any line not starting with '#' is treated as normal text, including other MD headings like:

```
Something
===
```

or other HTML headings like:

```
<h3>Something Else</h3>
```

which are anyway deprecated and **must** be avoided, you will see why later.

Titles in headings are:

- level 1 headings: title of the whole document
- first level 2 heading: title of the TOC
- following level 2 headings: titles of level 1 chapters (as 1. 2. ...)
- level 3 headings: titles of level 2 chapters (as 1.1. 1.2. ...)
- level 4 headings: titles of level 3 chapters (as 1.1.1. 1.1.2. ...)
- level 5 headings: titles of level 4 chapters (as 1.1.1.1. 1.1.1.2. ...)
- level 6 headings: titles of level 5 chapters (as 1.1.1.1.1. 1.1.1.1.2. ...)

So the input file **must** contain the following headings, freely intermixed with normal text, but in this

order:

- zero or one optional level 1 heading with title of the whole document
- one level 2 heading with title of the TOC
- one level 2 heading with title of the first level 1 chapter (1.)
- zero or more level 2..6 headings with titles of the following level 1..5 chapters

Beware: no heading can **ever** have a level higher than that of the previous heading plus one.

Each non-heading line in input is transcribed as it is into output, except for those between the first and second level 2 headings. These lines are supposed to be the old TOC and hence are deleted and replaced in output by the new TOC.

Chapter numbering in input headings is deleted and replaced in output by the new chapter numbering.

2.3. Example

An example will make everything clearer. As MD editor we will use ReText.

Beware: from another MD editor you could get a different behavior.

We open a terminal window and type:

```
$ retext example.md &
```

The '&' forks ReText as a separate process allowing us to use concurrently the terminal window and the ReText window.

In ReText window we write:

```
# Animal House

## What's Inside?

## Rains

Drops drops.

### Cats

Meow meow.

### Dogs

Bark bark.
```

We save (by clicking File → Save, or typing Ctrl+S, or clicking the [↓] button), then on terminal window we run:

```
$ toc2md example.md
```

and the text in ReText window magically becomes:

```
# Animal House

## What's Inside?

- [1. Rains](#1.)
    - [1.1. Cats](#1.1.)
    - [1.2. Dogs](#1.2.)
```

```
## <a name="1.">1. Rains</a>

Drops drops.

### <a name="1.1.">1.1. Cats</a>

Meow meow.

### <a name="1.2.">1.2. Dogs</a>

Bark bark.
```

Note in the newly created TOC the links pointing to chapter headings.

Beware: if after running 'toc2md example.md' ReText opens a dialog sub-window like this:

```
                              ReText

This document has been modified by other application.
Do you want to reload the file (this will discard all your changes)?

                              [X Cancel] [Reload]
```

don't click [Reload]! The sub-window means you forgot to save! Click [X Cancel] on sub-window, save ReText window, and run 'toc2md example.md' in the terminal window again.

Now we realize Cats eat Mice, Frogs like Rains, and after Rains the Sun comes, so we insert Mice in Cats, Frogs between Cats and Dogs, and Sun after Rains.

```
# Animal House

## What's Inside?

- [1. Rains](#1.)
    - [1.1. Cats](#1.1.)
    - [1.2. Dogs](#1.2.)

## <a name="1.">1. Rains</a>

Drops drops.

### <a name="1.1.">1.1. Cats</a>

Meow meow.

#### Mice

Squeak squeak.

### Frogs

Croak croak.

### <a name="1.2.">1.2. Dogs</a>

Bark bark.

## Sun

Rays rays.
```

As before, we save ReText window and we run 'toc2md example.md' in terminal window, so the text in ReText window becomes:

```
# Animal House

## What's Inside?

- [1. Rains](#1.)
    - [1.1. Cats](#1.1.)
        - [1.1.1. Mice](#1.1.1.)
    - [1.2. Frogs](#1.2.)
    - [1.3. Dogs](#1.3.)
- [2. Sun](#2.)

## <a name="1.">1. Rains</a>

Drops drops.

### <a name="1.1.">1.1. Cats</a>

Meow meow.

#### <a name="1.1.1.">1.1.1. Mice</a>

Squeak squeak.

### <a name="1.2.">1.2. Frogs</a>

Croak croak.

### <a name="1.3.">1.3. Dogs</a>

Bark bark.

## <a name="2.">2. Sun</a>

Rays rays.
```

Done!

2.4. Errors

toc2md can fail with one of the following error messages:

```
ERROR: file '...' has no '.md' extension
ERROR: file '...md' not found
ERROR: heading sequence error in line ... '#...'
```

3. After Usage

You can write a MD file for many good reasons, let's look at what happens to links in our TOC in two particular situations:

- you need to upload your MD file as is
- you want generate as final result a PDF file

3.1. Uploading README.md to PyPI

This is a special case where we encounter some trouble. Imagine we wrote by ReText and toc2md a README.md file as part of a package, say xxxx. We upload xxxx to PyPI for publishing. Then we go on our brand new project page https://pypi.org/project/xxxx and under the "Project description" label we find the HTML translation of our README.md. We click a line in table of content, and nothing happens, the link to the desired chapter does not work. Why?

The problem arises from a PyPI policy, which for some (probably good) reason does not allow internal

links in README.md. Therefore the MD-to-HTML translator PyPI uses to build our project page transforms for example the tag

```
<a name="1.">1. Before Usage</a>
```

into a bare

```
<a>1. Before Usage</a>
```

Sorry, there is nothing we can do about it, the translator is managed by PyPI maintainers, and so is completely out of our control.

3.2. Exporting in PDF format

You have several ways to generate a PDF file from your MD file. You could for example directly export your MD file from ReText in PDF format. You would get a well formatted PDF file with page numbers in bottom right corner, is ready for quick-and-dirty paper printing but not suitable for web publishing because if you open it by a PDF browser then the links in TOC do not work.

You can do better, namely:

1. write example.md using concurrently ReText and toc2md, as mentioned above
2. export your example.md from ReText into example.html (by File → Export As → HTML)
3. replace in example.html any tag '<table>' with '<table border=1>' to maintain visibility of table borders (see for instance the table in 5. Acronyms)
4. open example.html file by LibreOffice Writer
5. make changes at your taste (for instance change text font or alignment, insert page numbers, insert page breaks...)
6. finally export example.html from LibreOffice Writer into example.pdf (by File → Export As → Export Directly As PDF)

Step 3. can be accomplished by any text editor, or typing at terminal:

```
$ cp example.html temp.html
$ sed 's/<table>/<table border=1>/' < temp.html > example.html
```

In steps 4..6 you go directly from HTML to PDF format without going through ODT format.

Doing so:

- you gain a high degree of control over the final result
- if you open the final PDF file by a PDF browser:
    - the toc2md-generated links in TOC jump perfectly to chapters
    - in the side pane on the left you can view the "Index" which is a TOC generated by LibreOffice Writer equal to the toc2md-generated one (plus page numbers) and remains visible when you scroll down in the file

More exactly: LibreOffice Writer can not distinguish '#'-headings from the others, so the two TOCs will be equal only as long as you have avoided inserting other MD headings as:

```
Something
===
```

or other HTML headings as:

```
<h3>Something Else</h3>
```

4. Conclusion

With steps 1..6 in previous chapter we have defined a procedure to obtain a PDF document equipped with nested chapter numbering and linked table of contents.

As an alternative procedure we could write our document directly in LibreOffice Writer, using its Outline Numbering and Table of Contents features.

We let the reader decide which of the two solutions is the simplest and most practical.

5. Acronyms

INITIALS	MEANING
CLI	Command Language Interface
HTML	HyperText Markup Language
MD	MarkDown
ODT	Open Document Text
PDF	Page Description Format
TOC	Table Of Contents
toc2md	Table Of Contents to MarkDown

6. Credits

The toc2md project has been:

- developed in Python 3.8.2
- by Idle 3.8.2
- built and published on PyPI thanks to flit 2.3.0
- on Linux Xubuntu 20.04 LTS

This README.md file has been:

- written by ReText 7.1.0
- together with toc2md 0.9.1

During tests we used:

- ReText 7.1.0 to write MD, HTML and PDF files
- Idle 3.8.2 to edit HTML files
- Firefox 76.0.1 to check HTML files
- LibreOffice Writer 6.4.3.2 to translate HTML files into PDF format
- Atril 1.24.0 to check PDF files