`toc2md` User Guide

Version 0.9.1

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

Carlo Alessandro Verre

carlo.alessandro.verre@gmail.com

june 17, 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.

To read this document you only need a minimum knowledge of MD and HTML.

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.

1.2. Installation

Be sure you got last pip version typing at terminal (without sudo):

```
$ python3 -m pip install --user --upgrade pip
```

Now you can install `toc2md` typing (without sudo):

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

1.3. Documentation

After installation, if you type at terminal:

```
$ cd ~/.local/lib/python3.*/site-packages/toc2md/docs
```

you can find all versions of the `toc2md` User Guide discussed in 3. After Usage:

- `toc2md.md` - original file edited by ReText and toc2md
- `toc2md.html` - exported by ReText, in input to FireFox
- `toc2md.sed.html` - toc2md.html modified by sed, in input to Writer
- `toc2md.retext.pdf` - directly exported by ReText (low quality)
- `toc2md.firefox.pdf` - exported by FireFox from toc2md.html (medium quality)
- `toc2md.writer.pdf` - exported by Writer from toc2md.sed.html (high quality)

README.md has been obtained from toc2md.md dropping page 1.

Text written on `sysout` typing at terminal:

```
$ toc2md -h
```

has been obtained extracting the chapter 2.2. Logic from toc2md.md.

Anyway the "official" version of the `toc2md` User Guide is `toc2md.writer.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 beWriter)
- 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 heading: 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:

- one 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 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:

```
<style>@media print{h2{page-break-before:always}}</style>
<br><br><br><br><br><br><br><br><br><br><br><br><center>
# Animal House
</center>
## What's Inside?

## Rains

Drops drops.

### Cats

Meow meow.

### Dogs

Bark bark.
```

Line 1 automatically inserts a page break before each level 2 heading (level 1 chapter). This has effect only when you export in HTML format towards Firefox, see beWriter.

Lines 2 and 4 center (vertically and horizontally) the document title in line 3.

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:

```
<style>@media print{h2{page-break-before:always}}</style>
<br><br><br><br><br><br><br><br><br><br><br><br><center>
# Animal House
</center>
## 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]** (maybe twice) 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.

```
<style>@media print{h2{page-break-before:always}}</style>
<br><br><br><br><br><br><br><br><br><br><br><br><center>
# Animal House
</center>
## 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:

```
<style>@media print{h2{page-break-before:always}}</style>
<br><br><br><br><br><br><br><br><br><br><br><br><center>
# Animal House
</center>
## 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 us look at what happens to the links in our TOC by translating from MD format into HTML and PDF format.

As an example, we will see various metamorphoses of this same User Manual you are reading, born as `toc2md.md`, edited and processed concurrently by Retext and toc2md.

3.1. MD to HTML

The purpose of the MD format is to prefigure an HTML file, however we will see a case in which this passage from MD to HTML can cause us some surprise.

3.1.1. HTML From ReText

We export `toc2md.md` from Retext by **File → Export → HTML** into `toc2md.html`. If we open it by by Firefox, links in our TOC work perfectly as we expected, we click a line in TOC and we jump on the desired chapter.

We suppose this will happen with any other MD editor and any other HTML browser.

3.1.2. HTML From PyPI

But let us look at this particular case. In order to publish `toc2md` package in PyPI we need a `README.md`, so we create `README.md` as a copy of `toc2md.md` less page 1. We upload `toc2md` package to PyPI for publishing. Then we go by Firefox on our brand new project page https://pypi.org/project/toc2md and under the "Project description" label we find the HTML translation of our `README.md`. We click a line in TOC, 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 therefore is out of our control.

3.2. MD to PDF

We have several ways to generate a PDF file from our `toc2md.md` file. As you can expect, from a more complex procedure we will get a higher degree of control on result, and a better final quality. We will examine three ways:

- exporting directly from ReText
- exporting from Firefox via HTML format
- exporting from LibreOffice Writer via HTML format

3.2.1. PDF From ReText

We can directly export `toc2md.md` from ReText by **File → Export → PDF** into a file we call `toc2md.retext.pdf`, to remark its origin.

We get a PDF file with page numbers in bottom right corner and nothing else.

Procedure is instantaneous, control on final result is absent, quality is good for a quick-and-dirty draft printing:

- links in table of contents do not work
- the trick **`'<style>@media print{h2{page-break-before:always}}</style>'`** to automatically insert page breaks before level 1 headings does not work
- no manual formatting of final result is possible

By the way, if you print directly from ReText by **File → Print** you obtain on paper the same result.

3.2.2. PDF From Firefox

We can:

1. open toc2md.html HTML file by Firefox
2. optionally control page headers and footers via **[≡] button → Print...→ Print... → Options → Header and Footer**
3. optionally control paper size and other print parameters via **[≡] button → Print... → Print... → Page Setup**
4. finally export into toc2md-firefox.pdf by **[≡] button → Print... → Print... → General → Print to File → Print**

Procedure is fast, control of result and final quality are good for printing but not for web publishing:

- links in table of contents do not work, but...
- the trick **`'<style>@media print{h2{page-break-before:always}}</style>'`** to automatically insert page breaks before level 1 headings works

By the way, if you print directly from Firefox by **File → Print... → Print... → Print** you obtain on paper the same result.

3.2.3. PDF From LibreOffice Writer

You can:

1. replace in `toc2md.html` each string **`'<table>'`** with string **`'<table border=1>'`** and write result in `toc2md.sed.html` (to maintain visibility of table borders, see for instance the table in 4.1. Acronyms) by typing at terminal **'`sed 's/<table>/<table border=1>/' < toc2md.html > toc2md.sed.html`'**
2. open `toc2md.sed.html` by LibreOffice Writer
3. make changes at your taste (for instance change text font or alignment, insert page breaks, insert headers and/or footers with page number or whatever else...)
4. finally export from Writer into `toc2md.writer.pdf` by **File → Export As → Export Directly As PDF**

Procedure is cumbersome, control of final result is maximum, result is an indexed, professional-looking PDF file, good for printng and web publishing:

- if you open `toc2md.writer.pdf` by a PDF browser:
  - the toc2md-generated links in TOC jump perfectly to chapters
  - the file is said "indexed" because in the side pane on the left you can view the "Index", which is a Writer-generated TOC equal to the toc2md-generated one (plus page numbers) and remains visible when you scroll down in the file
- the trick **`'<style>@media print{h2{page-break-before:always}}</style>'`** to automatically insert page breaks before level 1 headings do not work, but now you can insert page breaks where you want by **Ctrl-Enter**

To quickly repeat step 3 several times on the same file you can record your Writer commands into a Writer macro.

Writer can not distinguish '#'-headings from the others, so the two TOCs (the Writer-generated at left and the toc2md-generated in the text) will be really equal only as long as you have avoided inserting other MD headings as:

```
Something
===
```

or other HTML headings as:

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

in your MD file.

4. Appendices

4.1. Acronyms

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

4.2. 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 toc2md.pdf file has been:

- written by ReText 7.1.0
- processed by toc2md 0.9.1
- exported from ReText in HTML format and translated into PDF format by Firefox 76.0.1

During tests we used:

- ReText 7.1.0 to write MD, HTML and PDF files
- Firefox 76.0.1 and LibreOffice Writer 6.4.3.2 to translate HTML files into PDF format
- Atril 1.24.0 to check PDF files
  `<br>`