Chapter 1

`md2tex` - a CLI to convert Markdown to TeX

`md2tex` is a simple but customizable markdown to TeX converter inspired by XSLT and functionnal programming. Rather than using fancy object oriented libraries, it locates elements of Markdown syntax using regular expressions and translates them into TeX syntax. It has only been tested with latin script so far.

1.1 Features

- Translation from Markdown to TeX
- Creation of a complete and valid TeX document, either from a generic template or using your own template
- Reading Markdown from custom locations and writing a TeX file to a custom location
- Several options for extra customization: type of quotes inline quotes, numbered or unnumbered headers, document classes...

1.2 Commands / How to

The advantage of having a script focused on Markdown to TeX conversion is that you can get a lot of fine tuning and extra customization.

1.2.1 Requirements

The script relies on the `minted`[1] package, so this needs to be installed on your machine. This script has been tested on a Linux machine and should work on UNIX and affiliated systems. Theoritically, it should also run on Windows, although that has not been tested. You will need to tailor the below scripts if you use Windows.

1.2.2 Automated installation (MacOS/GNU-Linux)

An installation shell script has been written to make things easier. Simply clone the repo and launch the install !

```
git clone https://github.com/paulhectork/md2tex.git # clone the repo
cd md2tex # move to the directory
bash install.sh # can also be used with other shells: zsh, ash...
```

1.2.3 Manual install

If the above method doesn't work, a manual installation is just as easy:

```
git clone https://github.com/paulhectork/md2tex.git # clone the repo
cd md2tex # move to the directory
python3 -m venv env # create virtual env
source env/bin/activate # source it
pip install -r requirements.txt # install requirements
pip install --editable . # build the package
```

1.2.4 Base functionning

This CLI should be used from the `md2tex` directory.

```
source env/bin/activate # get the proper env
md2tex path/to/markdown/file.md --options
```

[1] Website visited on 02.08.2022.

1.2.5 Arguments and options

Argument

The only **compulsory argument** is the path to the markdown file that needs to be processed.

- The file must finish with `.md` so that we're sure that a markdown file is being processed.

Optional parameters

As you can see below, there quite a few possibilities for fine-tuning. All of the below parameters are optional.

- **`o, -output-path`**: give a specific path and filename to write the `.tex` file to?
  - if none is provided, the output path defaults to `md2tex/output/input_filename.tex`
  - if the output path contain directories that do not exist, the output will not be written. Directories must be created manually beforehand.
    - if the extension of the output file is not `.tex`, a `.tex` extension will be added to the filename.
- **`-c, -complete-tex-file`**: if provided, a complete TeX file will be created, with preamble and table of contents.
  - the default template used is `utils/template.tex`. For a custom template, see `-t` below.
  - if not provided, only the body of the Markdown file will be converted. It will need to be included in another file with a premable.
- **`-t, -custom-tex-template`**: the path to a custom TeX template to build a complete document from, instead of the default template `utils/template.tex`.
  - this argument must be used in conjunction with `-c` for an actual template to be used.
  - a path to an existing template must be provided.
  - this template must contain the string `@@BODYTOKEN@@`: this token will be replaced by the contents of the converted Markdown file.
  - the document class can also be defined in the document by the CLI: in the template's preamble, write `documentclass{book}`.

- **-d, -document-class**: this argument indicates the class of the final TeX document. it indicates wether to process the Markdown file as a LaTeX `book` or as an `article`.
  - defaults to `article`.
  - if used with a custom TeX template (see `-c` and `-t`), the template must contain an `book` in the preamble where the document class is specified so that the TeX document class can actually change.
  - in fact, this argument only impacts the headers used in the TeX template.
- **-u, -unnumbered-headers**: if this argument is provided, the TeX headers (`\chatper`, `section`...) will be unnumbered.
  - by default, the headers are numbered.
- **-f, -french-quote**: if this argument is provided, anglo-saxon inline quotes will be replaced by french quotes using the `\enquote` command.
  - defaults to False: anglo-saxon quotes are used.

1.2.6 Command line help

```
1 md2tex --help
```

1.3 Examples

Output TeX files and the PDFs (using `XeLaTeX`) can be seen in the `readme/` and `examples/` directory.

- the `readme/` directory contains the canonical TeX and PDF documentation, produced by processing this README file.
- the `examples/` directory contains additional examples, including different parameters used to process this file.
- the script `readme_conversion.sh` allows to run a series of conversions and compilation of this file into different TeX and PDF files.

Example 1 - the canonical `README.tex` file in the `README` dir:

- a complete document is created (`-c`)

- it uses the custom template (`-t`) `./readme/readme_article_template.tex`
- the output is written (`-o`) to `./readme/README.tex`
- the document created is an article (`-d article`)
- the document uses unnumbered headers (`-u`) and french quotes (`-f`)

```
md2tex README.md -c -t ./readme/readme_article_template.tex -o
↪  ./readme/README.tex -d article -u -f
```

Example 2 - a partial TeX file to a custom output (`-o ./examples/README_partial_base.tex`) with default params

```
md2tex README.md -o ./examples/README_partial_base.tex
```

Example 3 - a complete document of class `book` using a custom template with unnumbered headers and french quotes

```
md2tex README.md -c -t ./readme/readme_book_template.tex -o
↪  ./examples/README_book.tex -d book -u -f
```

Example 4 - a complete document of class `article` with a custom template and numbered (default) headers

```
md2tex README.md -c -t ./readme/readme_article_template.tex -o
↪  ./examples/README_article_numbered.tex
```

Example 5 - a complete document of class `book` with default template and numbered headers

```
md2tex README.md -c -o
↪  ./examples/README_article_default_template_numbered.tex
```

Example 6 - a complete document of class `article` with default template and default params

---

```
1 md2tex README.md -c -o ./examples/README_book_default_template.tex -d
  ↪ book
```

1.4 Markdown syntax

1.4.1 Currently supported

- Line breaks: paragraph changes using `\n\n`, `<br>` and `<br/>`
- Different styles of text: **bold**, *italic* and `inline code`. **Warning**:
  - only bold and italics made using asterisks will be translated.
- Block quotes (lines beginning with `>`). **Warning**: only non-nested block quotes – or the outer level of a nested block quote – will be rendered.
- Multiline code; if possible, the code is colored using `minted`. Indentation levels are **always** respected within multiline code.
- Ordrered and unordered lists, including nested lists. **Warning**:
  - To be processed, all indentation levels must be a multiplier of the indentation of the first indented item. See below for details on how nested lists are handled.
  - Unordered lists will only be converted if they begin with `-`. any other list token will be ignored.
  - The only list token replaced in ordered lists is `\d+.` (digits followed by a point: `1.`). If an other list token is used (a letter...), the numbered list won't be processed. If additional tokens are added (for nested lists: `1. a.`), the list will be processed, but the second token (`a.`) will not be deleted. This is because the use of additional tokens is specific to the Markdown interpreter used.
- Titles and different title levels
- Images
- URLs, both in text and as Markdown hyperlinks.
- Inline quotes : french and english style, nested quotes.
- Footnotes. **Warning**: loose footnotes (references in the body that point to nothing or footnotes that point to nothing in the body) will be deleted.

1.4.2 Currently unsupported

- Markdown tables
- Strikethrough
- Emojis
- Tasks lists
- Definition lists
- Text blocks inside lists will be processed as the continuation of a list item

---

1.5 Things that will mess with the output

The things in the list below will either cause the script to exit or produce a messy output.

- Poorly formed Markdown. If a list is not properly formed, the script will exit.
- Using things that are currently not supported.
- Incoherent use of header levels (`#`, `##`...) will be processed and render a valid document, but it will mess with the appearence of the TeX file and of the table of contents, especially if the headers are numbered.
- Unnumbered lists made using non-standard markdown syntax (lists not beginning with `-`...). These will simply not be matched.
- Numbered lists using additional numbering keys, such as `1. a`. These will be matched, but in this case, `1.` will be deleted and replaced by TeX syntax; `a` will be kept, so the TeX file will contain double numbering (the TeX numbering and this second numbering key)
- Highly nested lists are not supported by default in LaTeX, but this can be changed in your preamble.
- Multiple footnotes with the same key; for the footnote processing to work, there must be unique footnote numbers and only one note per footnote number.
- Loose footnotes that point to nothing will be deleted.

Clean your file first ! You can also make the bulk of the translation using this cli and fine tune your `.tex` file later on.

1.6 About nested lists and visual indentation

List nesting and visual indentation is a bit of a pickle, so here's an explanation of how the indentation is processed and what constitutes a valid list for this tool.

1.6.1 What is a valid list?

All items must be **as indented than the first list item** or more. the indentation level of the first item will be removed before processing that list. This means that *the first example* will result in the same result as *the second*.

```
- list item with no leading space
- same here
```

```
  - list item with two leading spaces
  - same here, still a valid list
```

All indentation levels must be a multiple of the leading whitespaces of the first nested item. If the first nested list item is indented with 4 whitespaces, all nested items must be indented with a multiple of 4 spaces. For example, a list item can't be indented with 3 spaces while another one is indented with 4 spaces.

The above two rules cause the script to exit with an error message pointing to the faulty list. On top of that, in keeping with LaTeX and markdown logic, a list item can't skip a nesting level (you cannot go from an item in a list to an item in a list in a list). This will **not** stop the script, the indentation will just be reset automatically.

```
- this is an item at level 0
  - this is an item at level 1; the indentation is set as 2 whitespaces
  - same here; level 1.
   - visually, this one seems indented as level 3 as it starts with 6
      leading whitespaces.
   however, its nesting level will be reset from level 3 to level 2 to
      avoid skipping
   a nesting level
  - this item is at level 1
```

1.6.2 Technically, how does it work ?

- The complete markdown list is matched.
- Each list item is mapped to its indentation level, in number of leading spaces.
- The list is validated. If it is not valid, the script stops.
- An indentation multiplier is defined: it is the number of spaces in the first nested list item.

- This multiplier is used to map each list item to its nesting level.
- The markdown is syntax is replaced by TeX syntax.

1.6.3 Examples of valid and invalid lists

Valid lists

```
- item at level 0
- item at level 0
  - item at level 1. here, a 2-whitespace indentation is defined for the
      list.
   - item at level 2, with 4 leading spaces.
- item at level 0
```

```
   - item at level 0
   - item at level 0
```

```
- item at level 0
   - item at level 1. for this list, the indentation is set to 4 spaces.
   - visually, this item is at level 3 (12 leading spaces), but will be
      reindented to 2 spaces.
   - item at level 1.
```

Invalid lists

```
  - item at level 0
  - item at level 0
- this will cause the script to exit: the item is less indented than the
    first item.
```

```
- item at level 0
       - indentation is set at 4 spaces
        - here, there are 6 spaces. the indentation is incoherent and
           the script stops.
```

1.7 License and credits

Developped by Paul Kervegan in August 2022 and released under GNU GPL v3.

Contents