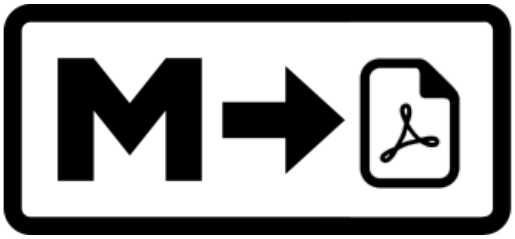

MarkDown Multi Tool *(mdmt)*

Concept

This tool deals with markdown files and it's used to convert them to other formats such as **PDF** and **HTML** files.

- I usually write manuals as MarkDown files, there's no need for a GUI editor even if it helps a lot. Printing or rendering MD files outside a proper tool might be a pain. Fortunately HTML or PDF files doesn't suffer this problem.
- Want a full blown formatting language with a lot of pro features ? Markdown is not the right format for you, and this utility is not needed at all. Pick up TeX (LaTeX, ConTeXt, ...) for that, Knuth's work is still awesome even after 50 years. Please stay away from proprietary formats.

Purpose

It's still a small utility but it already has achieved its basic tasks such as:

- No human interaction, used in massive batch processes, just a small task in an automation process.
- Lightweight and highly portable. Even if pandoc is a massive tool and has plenty of features (and I like it) it also has a lot of dependencies, having a full blown Haskell environment installed (~400Mb) might not be your best choice for a small/minimal machine or container.
- Highly portable means: python3 required, webkit *wkhtmltopdf* required, nothing more. Script is just **one** file, there's no need (yet) to define it as a module or require massive python deps. Linux and MacOS users might be happy with it.
- Do not reinvent the wheel. Building a full blown HTML renderer is an interesting task but not something I need to achieve when webkit is already in place. In fact it uses webkit and its rendering features to properly deal with HTML pages and scripts.
- Layout friendly and customizable. CSS is a thing for a while and a quite nice style sheet language, reinventing some sort of custom language for rendering pages might not be a good idea when webkit+css might do the same for you.
- No proprietary extensions. Rendering with plain CSS, display pages with webkit, automation with batch scripting. As easy as it gets.
- Custom themes available. Default (and now only) CSS template file resembles github.com style HTML pages layout, I've barely taken formatting from there because that's what I use most and what I personally prefer. It's totally up to you to create your own CSS layout and define page rendering according to your needs, dark layouts and so on.

Requirements

- ***python3***. This is a python utility, basic libs supplied are fine, there's no need to install additional things
- ***wkhtmltopdf***. Webkit command line renderer. It uses the webkit engine from the command line, if you already have any major linux desktop environment you should already have it. The package is usually named `wkhtmltox`. MacOS users might download it directly or use *brew* for that (`brew install wkhtmltopdf`)

NOTE: This *PDF* document is a practical example of what `mdmt` could do.
This user manual has been generated by joining all markdown files in this git project repository

Usage

```
$ ./mdmt --help
Usage: mdmt [options]
       Markdown Management Tool utility
       Convert markdown files to your favorite format

Options:
  -h, --help            show this help message and exit
  -v, --verbose         Verbose output information
  -i INPUTFILE, --input=INPUTFILE
                        Input file to convert (*.md)
  -l INPUTLIST, --input-list=INPUTLIST
                        Concat *.MD list from stdin as one sigle file
  -o OUTPUTFILE, --output=OUTPUTFILE
                        Destination file name (md,htm,html,cnfdoc,pdf)
  -s INPUTTYPE, --typein=INPUTTYPE
                        Force input to one of these values:
                        (md,htm,html,cnfdoc,pdf) autodetected by default
  -t OUTPUTTYPE, --typeout=OUTPUTTYPE
                        Force output to one of these values:
                        (md,htm,html,cnfdoc,pdf) autodetected by default
  -p PAGE, --page=PAGE  Page size (when applicable) (A4,Letter)
  -T TEMPLATE, --template=TEMPLATE
                        CSS template to apply, default: mdmt.css
  -P PLUGINS, --plugins=PLUGINS
                        Specify plugins to use on markdown formatting
  -I, --plugins-help    Complete help on plugins and their usage
  -H HEADER, --header-split=HEADER
                        Split a markdown by headers
```

Examples:

MarkDown -> HTML

```
# Simple conversion, Default "mdmt.css" template file is used when found in path
mdmt --input=tests/README.md --output=tests/README.html

# Same as above but explicitly specify output file type, autodetection is skipped
mdmt --input=tests/README.md --output=tests/README --type=html
```

```
# Use specific template file during conversion
#     Filename might be absolute or relative to current working path
mdmt --input=tests/README.md --output=tests/README.html --template=templates/mdmt.css
```

MarkDown -> PDF

```
# Simple conversion, Default "mdmt.css" template file is used when found in path
mdmt --input=tests/README.md --output=tests/README.pdf

# Same as above but explicitly specify output file type, autodetection is skipped
mdmt --input=tests/README.md --output=tests/README --type=pdf

# Use specific template file during conversion
#     Filename might be absolute or relative to current working path
mdmt --input=tests/README.md --output=tests/README.pdf --template=templates/mdmt.css
```

HTML -> PDF

```
# Convert HTML file to a PDF file, templates are not used
./mdmt --verbose --input=tests/index.html --output=tests/output/index.pdf
```

Concatenate MarkDown -> PDF

```
# - Concatenate a bunch of separate markdown files into one single stream
# - Convert markdown files stream to a single PDF file (join)
# - Use specific template file during conversion
#       template filename might be absolute or relative to current working path
cat << EOT |mdmt --input-list - --output=tests/README.pdf --template=templates/mdmt.css
tests/README.md
tests/SEC-RES-REQ-1.md
EOT

# Same command with input stream generated from another command
ls -1 tests/*.md | \
    mdmt --input-list - --output=tests/README.pdf --template=templates/mdmt.css
```

Split by headers

Split on different files based on selected headings, few notes are necessary to understand how it works:

- `--output` is **mandatory** and it must be an existing directory. Few examples: - `--output=tests`. Valid if "*tests*" is an existing (and writable) directory - `--output=tests/a.md`. Invalid because it is a file, **but** output files **will be saved** to `tests` if directory exists and it is writable.
- Each single generated file will be saved when selected header is found in source, allowed values are `1..6`. This structure will be generated: - `--header-split 1` Files will be saved as "`[nn] [title].md`" in the target directory.
  `[nn]` is calculated on the number of sections found in the source file (starting from "0").
  `[title]` is the header title name, allowed characters: [a-zA-Z0-9] - `--header-split 2..6` Directories with `H1` headers will be created inside `--output` dir. Naming conventions will follow what already described above. Inside each directory a `README.md` file with contents from upper sections will be created and files for selected section will be stored separately with "`[nn] [title].md`" convention previously described. This schema aims to reproduce some sort of "README.md"+Tree structure just like common github repos.

Examples:

```
# H1 Split
mdmt --input=tests/README.md --output=tests/ --template=templates/mdmt.css --header-split 1

# H2..n Split
```

```
mdmt --input=tests/README.md --output=tests/ --template=templates/mdmt.css --header-split 2
mdmt --input=tests/README.md --output=tests/ --template=templates/mdmt.css --header-split 3
#...
mdmt --input=tests/README.md --output=tests/ --template=templates/mdmt.css --header-split 6
```

Project TODOs

This is not a never finished project, something has to work and something else will be added on upcoming releases (maybe). Here are planned features, wishlists, requirements and improvements. Some of them might see the light in upcoming releases, few will remain here for some time. Feel free to fund the project if you want to add your favorite feature or contribute if possible.
Here's current wishlist:

- code block color rendering. It's already working but a better formatted (and colored) output is preferred.
- code block CSS support. As a programmer I usually add a lot of code blocks, basic CSS is not enough.

Please add your pull requests or file an issue for your favorite features, each single feature has to be fully MarkDown compatible, no extra custom features will be added. Use TeX instead.

Changelog

Changelog documentation on various mdmt versions, known issues, inner working and fixes along the way

---

> **v1.0.2 (22/05)**
> *Bugfix and new features tests*

- **New features:**
  - Testing new features for confluence docs, not yet ready for the prime time
- **Bugfix:**
  - Template path might be absolute or a relative path.

> **v1.0.1 (22/05)**
> *Feature release, parameters for forcing input/output extension files*

- **New features:**
  - `--typein`, `--typeout` options to avoid in/out file types detections
- **Internals:**
  - Test unit sample commands for `--typein`, `--typeout`.

> **v1.0.0 (22/05)**
> *Stable release, runtime custom formatting plugins available*

- **New features:**
  - Markdown formatting plugins activation available at runtime, new parameter `--plugins` - `--plugins-help` option for extensive help on available plugins.
- **Internals:**
  - Testing on big files, it might take out a lot of memory but does a rather decent job. - Feature freeze, stability and patches releases

only for a while.

---

v0.9.2 (22/04)
minor layout cleanup and document formatting

- **New Features:**
  - `text-align:justify` is now the default text alignment on standard (and still only) CSS theme. - plugin **nl2br** is not pleasant on HTML page rendering, CSS `text-align:justify` has been elected as default page justification on all text. Layout is more appealing even if rendered text is not exactly like the markdown (but markdown does not have text justification at all). - Added a new script to automatically generate a complete user manual for this project, **HTML** and **PDF** files are now created by `mdmt` itself ! - User manuals available as HTML or PDF in the `doc/` directory. - `README.md` gets automatically updated from the document generation script. - ***md*** and ***html*** files are now both supported as `--input` parameter but **html -> md** conversion is still not yet implemented.
- **Internals:**
  - Minor error checking added to improve script reliability. - Input and output files are checked before dealing with them

---

v0.9.1 (22/04)
Usable, all planned features are in place

- **New features:**
  - Slightly optimized code generation and layout cleanup - Better exceptions trapping - CSS minor fixes, default layout is mainly oriented to GitHub look and feel - Providing more CSS templates for different layouts is not yet a priority but it's possible on future releases
- **Internals:**
  - HTML javascript code cleanup. Avoiding to generate static HTML pages with JS code in it, this really helps during PDF generation due to the fact the webkit plain rendering is good doesn't handle runtime code properly. Emoji is still there as the only exception.

---

v0.9 (22/04)
Code cleanup

- **New features:**
  - Added `--verbose` flag for additional output when needed. - mermaid code block support now added, now it's fully supported on HTML and PDF files generation. - Major rewrite and some code optimization. - Internet access is required during Mermaid blocks generation (if any), there's no need later on.
- **Internals:**
  - Code optimizations are related to a stable and usable mermaid support for PDF and HTML generation. - External mermaid service from their CDN is now used when images are generated. Raw HTML pages might be rendered on the fly with some JS code but newly added methods *totally avoid* to inject JavaScript code in the build process. This approach fairly and properly render all Mermaid blocks equally on PDF and HTML. There are a lot of known issues with webkit libs and their Javascript interpreter due to the fact it is way less powerful than Google V8 or equivalent engines. Statically generating graphs directly from MarkDown code seems to be the best option, GitHub does exactly the same.

---

v0.3 (22/04)
PDF proper fixes when used with webkit rendering engine

- **New features:**
  - Major cleanup for PDF generation, reliable and based on tmp files to avoid webkit known issues
- **Internals:**
  - Fonts and formatting are left on the CSS side, need to fix it properly. - webkit rendering is really good but workaround must be placed to fix its errantic behavior when local resources are loaded from main page. - Basically this is a webkit fix for known issues release.

---

v0.2 (22/04)
HTML generation seems to be finally acceptable

- **New features:**
  - md join from multiple files from stdin
- **Internals:**
  - Syntax fixes, python linter is now happy - Focusing on single pages generation multiple file joins are not yet ready for prime time - Working on speed and efficiency, no refactoring yet - Getting emoji as UTF-8 chars. Somewhat important when importing chat conversations

---

> **v0.1 (22/04)**
> *Nothing special but it works, a rather good RC1*

- **New features:**
  - md -> html - md -> pdf - md multiple split - md join from multiple files - using python builtin markdown module and external `wkhtmltopdf` utility
- **Internals:**
  - barebone test unit in place for multiple tests - software is still for personal usage, it just works, no heavy input checks or sanity improvements - tested with small markdown files *(<=100 Kb)*
