TP-NOTE(1) Version 1.7.5 | Tp-Note documentation

Table of Contents

1. NAME

Tp-Note - fast note taking with templates and filename synchronization.

2. SYNOPSIS

```
tp-note [-V] [-b] [-d] [-v] [-c <config-file>] [<path>]
```

3. DESCRIPTION

Tp-Note is a note-taking-tool and a template system, that consistently synchronizes the note's meta-data with its filename. *Tp-Note* collects various information about its environment and the clipboard and stores them in variables. New notes are created by filling these variables in predefined and customizable *Tera*-templates. In case "`<path>`"points to an existing "*Tp-Note*"-file, the note's meta-data is analysed and, if necessary, its filename is modified. For all other file types, *Tp-Note* creates a new note that annotates the file "`<path>`"points to. If "`<path>`"is a directory (or, when omitted the current working directory), a new note is created in that directory. After creation, *Tp-Note* launches an external text editor of your choice. Although the note's structure follows "`pandoc`"-conventions, it is not tied to any specific Markup language.

After the user finished editing, *Tp-Note* analyses eventual changes in the notes meta-data and renames, if necessary, the file, so that its meta-data and filename are in sync again. Finally, the resulting path is printed to "`stdout`", log and error messages are dumped to "`stderr`".

This document is Tp-Note's technical reference. More information can be found in Tp-Note's user manual[1] and at Tp-Note's project page[2].

4. OPERATION MODES

Tp-Note operates in 4 different modes, depending on its commend-line-arguments and the clipboard state. Each mode is usually associated with one content-template and one filename-template.

4.1. New note without clipboard

In case the clipboard is empty while starting, the new note is created with the templates: "`tmpl_new_content`"and "`tmpl_new_filename`". By default, the new note's title is the parent's directory name. The newly created file is then opened with an external text editor, allowing to change the proposed title and to add other content. When the text editor closes, *Tp-Note* synchronizes the note's meta-data and its filename. This operation is performed with the "`tmpl_sync_filename`" template.

Example: the clipboard is empty and `<path>` is a directory (or empty):

[1] https://blog.getreu.net/projects/tp-note/tp-note--manual.html
[2] https://blog.getreu.net/projects/tp-note/

```
> tp-note "./03-Favorite Readings/"
```

or

```
> cd "./03-Favorite Readings"
> tp-note
```

creates the document:

```
"./03-Favorite Readings/20200306-Favorite Readings--Note.md"
```

with the content:

```
---
title:      "Favorite Readings"
subtitle:   "Note"
author:     "getreu"
date:       "March  6, 2020"
lang:       "en_GB.UTF-8"
revision:   "1.0"
---
```

4.2. New note based on clipboard data

When "`<path>`" is a directory and the clipboard is not empty, the clipboard's content is stored in the variable "`{{ clipboard }}`". In addition, if the content contains an hyperlink in Markdown format, the hyperlink's name can be accessed with "`{{ clipboard | linkname }}`" and its URL with "`{{ clipboard | linkurl }}`". The new note is then created with the "`tmpl_clipboard_content`" and the "`tmpl_clipboard_filename`"templates. Finally, the newly created note file is opened again with some external text editor. When the user closes the text editor, *Tp-Note* synchronizes the note's meta-data and its filename with the template "`tmpl_sync_filename`".

Note: this operation mode also empties the clipboard (configurable feature).

Clipboard simulation

When no mouse and clipboard is available, the clipboard feature can be simulated by feeding the clipboard data into `stdin`:

```
> echo "[The Rust Book](https://doc.rust-lang.org/book/)" | tp-note
```

Tp-Note behaves here as if the clipboard contained the string: “`[The Rust Book](https://doc.rust-lang.org/book/)`”.

The clipboard contains a string

Example: While launching *Tp-Note* the clipboard contains the string: “`Who Moved My Cheese?\n\nChapter 2`” and `<path>` is a directory.

```
> tp-note "./03-Favorite Readings/"
```

or

```
> cd "./03-Favorite Readings/"
> tp-note
```

This creates the document:

```
"./03-Favorite Readings/20200306-Who Moved My Cheese--Note.md"
```

with the content:

```
---
title:      "Who Moved My Cheese"
subtitle:   "Note"
author:     "getreu"
date:       "2020-09-11"
lang:       "en_GB.UTF-8"
revision:   "1.0"
---

Who Moved My Cheese?

Chapter 2
```

We see from the above example, how the “`tmpl_clipboard_content`”content template extracts the first line of the clipboards content and inserts it into the header’s “`title:`” field. Then, it copies the entire clipboard content into the body of the document. However,

if desired or necessary, it is possible to modifiy all templates in *Tp-Note*'s configuration file. Note, that not only the note's content is created with a template, but also its filename: The "`tmpl_clipboard_filename`"filename template concatenates the current date, the note's title and subtitle.

The clipboard contains a Markdown link

Example: `<path>` is a directory, the clipboard is not empty and it contains the string: "`I recommend:\n[The Rust Book](https://doc.rust-lang.org/book/)`".

```
> tp-note './doc/Lecture 1'
```

This creates the following document:

```
./doc/Lecture 1/20200911-The Rust Book--Notes.md
```

```
---
title:      "The Rust Book"
subtitle:   "URL"
author:     "getreu"
date:       "2020-09-11"
lang:       "en_GB.UTF-8"
revision:   "1.0"
---

I recommend:
[The Rust Book](https://doc.rust-lang.org/book/))
```

When the clipboard content contains an hyperlink in Markdown format, the template will use the name of the first hyperlink as document title.

The clipboard contains a string with a YAML header

Example: `<path>` is a directory, the clipboard is not empty and it contains the string: "`---\ntitle: Todo\nfile_ext: mdtxt\n---\n\nnothing`".

```
> tp-note
```

This creates the note: "`20200911-Todo.mdtxt`" with the following content:

```
---
title:      "Todo"
subtitle:   ""
author:     "getreu"
date:       "2020-09-11"
lang:       "en_GB.UTF-8"
revision:   "1.0"
file_ext:   "mdtxt"
---

nothing
```

Technically, the creation of the new note is performed using the YAML header variables: "`{{ fm_title }}`", "`{{ fm_subtitle }}`", "`{{ fm_author }}`", "`{{ fm_date }}`", "`{{ fm_lang }}`", "`{{ fm_revision }}`", "`{{ fm_sort_tag }}`"and "`{{ fm_file_ext }}`"which are evaluated with the "`tmpl_copy_content`" and the "`tmpl_copy_filename`" templates.

Note, that the same result can also be achieved without any clipboard by tying in a terminal:

```
> echo -e "---\ntitle: Todo\nfile_ext: mdtxt\n---\n\nnothing" | tp-note
```

Furthermore, this operation mode is very handy with pipes in general, as shows the following example: it downloads some webpage, converts it to Markdown and copies the result into a *Tp-Note* file. The procedure preserves the webpage's title in the note's title:

```
curl 'https://blog.getreu.net' | pandoc --standalone -f html -t
 markdown_strict+yaml_metadata_block | tp-note
```

creates the note file "`20200910-Jens Getreu's blog.md`"with the webpage's content converted to Markdown:

```
---
title:      "Jens Getreu's blog"
subtitle:   ""
author:     "getreu"
date:       "2020-09-11"
lang:       "en"
revision:   "1.0"
---
```

```
<a href="/" class="logo">Jens Getreu's blog</a>

-   [Home](https://blog.getreu.net)
-   [Categories](https://blog.getreu.net/categories)
```

Use Tp-Note in shell scripts

To save some typing while using the above pattern, you can create a script with:

```
> sudo nano /usr/local/bin/download
```

Insert the following content:

```
#!/bin/sh
curl "$1" | pandoc --standalone -f html -t markdown_strict
+yaml_metadata_block | tp-note
```

and make it executable:

```
> sudo chmod a+x /usr/local/bin/download
```

To execute the script type:

```
> download 'https://blog.getreu.net'
```

4.3. New note annotating some non-Tp-Note file

When "`<path>`" points to an existing file, whose file-extension is other than "`.md`", a new note is created with a similar filename and a reference to the original file is copied into the new note's body. If the clipboard contains some text, it is appended there also. The logic of this is implemented in the templates: "`tmpl_annotate_content`" and "`tmpl_annotate_filename`". Once the file is created, it is opened with an external text editor. After editing the file, it will be - if necessary - renamed to be in sync with the note's meta-data.

Example:

```
> tp-note "Classic Shell Scripting.pdf"
```

creates the note:

```
"Classic Shell Scripting.pdf--Note.md"
```

with the content:

```
---
title:      "Classic Shell Scripting.pdf"
subtitle:   "Note"
author:     "getreu"
date:       "March  6, 2020"
lang:       "en_GB.UTF-8"
revision:   "1.1"
---

[Classic Shell Scripting.pdf](Classic Shell Scripting.pdf)
```

The configuration file variable "`note_file_extensions`"lists all file extensions that *Tp-Note* recognizes and opens as own file types. Others are treated as described above.

This so called *annotation* mode can also be used with the clipboard: when it is not empty, its data is appended to the note's body.

4.4. Editing notes

If not invoked with "`--batch`", *Tp-Note* launches an external text editor after creating a new note. This also happens when "`<path>`" points to an existing "`.md`"-file.

Example: edit the note from the previous example:

```
> cd "./03-Favorite Readings"
> tp-note 20200306-Favorite Readings--Note.md
```

4.5. Automatic filename synchronization before and after editing

Before launching the text editor and after closing it, *Tp-Note* synchronizes the filename with the note's metadata. When the user changes the metadata of a note, *Tp-Note* will replicate that

change in the note's filename. As a result, *all your note's filenames always correspond to their metadata*, which allows you to find your notes back quickly.

Example:

```
> tp-note "20200306-Favorite Readings--Note.md"
```

The way how *Tp-Note* synchronizes the note's metadata and filename is defined in the template "`tmpl_sync_filename`".

Once *Tp-Note* opens the file in an text editor, the note-taker may decide updating the title in the note's YAML metadata section from "`title: "Favorite Readings"`"to "`title: "Introduction to bookkeeping"`". After closing the text editor the filename is automatically updated too and looks like:

```
"20200306-Introduction to bookkeeping--Note.md"
```

Note: the sort-tag "`20200306`"has not changed. The filename synchronization mechanism by default never does. (See below for more details about filename synchronization).

5. OPTIONS

-b, --batch

Do not launch the external text editor or viewer. All other operations are available and are executed in the same way.

Tp-Note ignores the clipboard when run in batch mode with "`--batch`". Instead, if available, it reads the `stdin` stream as if the data came from the clipboard.

-c *CF*, --config=*CF*

Load the alternative config file *CF* instead of the default one.

-d, --debug

Print additional log-messages on console. It shows the available template variables, the templates used and the rendered result of the substitution. This option particularly useful for debugging new templates. On Windows, the output must be redirected into a file to see it. To do so open the command-prompt and type:

```
tp-note.exe -d >debug.txt 2>&1
```

-v, --view
Launch the external text editor, if possible, in read-only-mode.

-V, --version
Print *Tp-Note*'s version and exit. When combined with "`--debug`", additional technical details are printed.

6. THE NOTE'S DOCUMENT STRUCTURE

A *Tp-Note*-note file is always UTF-8 encoded. As newline, either the Unix standard "`\n`"or the Windows standard "`\r\n`"is accepted. *Tp-Note* writes out newlines according the operating system it runs on.

Tp-Note is designed to be compatible with "`Pandoc`'s and'`RMarkdown`s document structure as shown in the figure below.

```
---
<YAML-front matter>
---
<document-body>
```

The YAML front matter starts at the beginning of the document with "`---`"and ends with "`...`"or "`---`". Note that according to the YAML standard, string-literals are always encoded as JSON strings.

There is no restriction about the markup language used in the note's text body. However, the default templates assume that Markdown and the file extension "`.md`"is used. Both can be changed easily by adapting *Tp-Note*'s configuration file.

7. META-DATA FILENAME SYNCHRONIZATION

Consider the following *Tp-Note*-file:

```
20151208-Make this world a better place--Suggestions.md
```

The filename has 4 parts:

```
{{ fm_sort_tag }}-{{ fm_title }}--{{ fm_subtitle }}.{{ fm_file_ext }}
```

A so called *sort-tag* is a numerical prefix at the beginning of the filename. It is used to order files and notes in the file system. Besides numerical digits, a *sort-tag* can be any combination of `0123456789-_`[3] and is usually used as

- *chronological sort-tag*

  ```
  20140211-Reminder.doc
  20151208-Manual.pdf
  ```

- or as a *sequence number sort-tag*.

  ```
  02-Invoices
  08-Tax documents
  09_02-Notes
  ```

When *Tp-Note* creates a new note, it prepends automatically a *chronological sort-tag* of today. The "`{{ fm_title }}`"part is usually derived from the parent directory name omitting its own *sort-tag*.

A note's filename is in sync with its meta-data, when the following is true (slightly simplified, see the configuration file for the complete definition):

> filename on disk without *sort-tag* == "`-{{ fm_title }}--{{ fm_subtitle }}.md`" [4]

Consider the following document with the filename:

```
20200306-My file.md
```

and the content:

```
---
title:      "1. The Beginning"
subtitle:   "Note"
```

[3] The characters "`_`" and "`-`" are considered to be part of the *sort-tag* when they appear in last position.

[4] The variables "`{{ fm_title }}`" and "`{{ fm_subtitle }}`" reflect the values in the note's metadata.

```
author:     "getreu"
date:       "March  6, 2020"
lang:       "en_GB.UTF-8"
revision:   "1.1"
---
```

As “`-My file.md`”is not equal to “`-'1. The Beginning--Note.md`”, *Tp-Note* will rename the file to “`20200306-'1. The Beginning--Note.md`”. If the filename had been “`05_02-My file.md`”, it would rename it to “`05_02-'1. The Beginning--Note.md`”.

Note: When the YAML front matter does not contain the optional “`sort_tag`”variable, *Tp-Note* will never change a sort-tag. Nevertheless, it might change the rest of the filename!

The reason why by default *Tp-Note* does not change sort-tags is, that they define their order in the file listing. In general this order is independent of the notes content. The simplest way to organize the sort-tags of your files is by renaming them directly in your file-system. Nevertheless, in some cases you might want to have full control over the whole filename through the note’s YAML front matter. For example, if — for some reason — you have changed the document’s date in the front matter and you want to change the chronological sort tag in one go. In order to overwrite the note’s sort-tag on disk, you can add a “`sort_tag`” variable to its front matter:

```
---
title:      "1. The Beginning"
...
date:       "March  7, 2020"
sort_tag:   "20200307-"
...
---
```

When *Tp-Note* synchronizes the note’s metadata with its filename, it will also change the sort-tag from “`20200306-`” to “`20200307-`”. The resulting filename becomes “`20200307-'1. The Beginning--Note.md`”.

The “`sort_tag`”variable also becomes handy, when you want to create one single note without any sort-tag:

```
---
title:      "1. The Beginning"
...
sort_tag:   ""
```

```
...
---
```

In the same way, how it is possible to pin the sort-tag of the note from within the note's meta-data, you can also change the file extension by adding the optional "`file_ext`"variable into the note's front matter:

```
---
title:      "1. The Beginning"
...
file_ext:   "mdtxt"
...
---
```

This will change the file extension from "`.md`"to '`.mdtxt`. The resulting filename becomes "`20200307-'1. The Beginning--Note.mdtxt`".

Important: "`mdtxt`" must be one of the registered file extensions listed in the "`note_file_extensions`"variable in Tp-Note's configuration file. If needed you can add more extensions there.

Note: When a "`sort_tag`"variable is defined in the note's YAML header, you should not change the sort-tag string in the note's file name manually by renaming the file, as your change will be overwritten next time you open the note with *Tp-Note*. However, you can switch back to *Tp-Note*'s default behaviour any time by deleting the "`sort_tag`"line in the note's metadata. The same applies to the "`file_ext`" variable.

8. CUSTOMIZATION

Tp-Note's configuration file resides typically in "`~/.config/tp-note/tp-note.toml`" on Unix or in "`C:\Users\<LOGIN>\AppData\Roaming\tp-note\config\tp-note.toml>`" on Windows. When *Tp-Note* starts, it tries to find its configuration file. If it fails, it writes a default configuration file. *Tp-Note* is best customized by starting it once, and then modifying its default configuration.

The configuration file is encoded according to the TOML-standard. Variables starting with "`tmpl_*`" are *Tera-Template*-strings (see: https://tera.netlify.com/docs/#templates).

Tp-Note captures and stores its environment in *Tera-variables*. For example, the variable "`{{ path }}`" is initialized with the note's target directory. The variable "`{{ clipboard }}`"contains the content of the clipboard. To learn more about variables,

launch *Tp-Note* with the "`--debug`"option and observe what information it captures from its environment.

8.1. Template variables

All Tera template variables and functions[5] can be used within *Tp-Note*. For example "`{{ get_env(name='LANG') }}'` gives you access to the"`LANG`' environment variable.

In addition *Tp-Note* defines the following variables:

- "`{{ file }}`"is the canonicalized fully qualified file name corresponding to *Tp-Note*'s positional parameter "`<path>`". If "`<path>`"points to a directory the content of this variable is identical to "`{{ path }}`".
- "`{{ path }}`" is same as above but without filename and extension.
- "`{{ clipboard }}`"is the complete clipboard text. In case the clipboard's content starts with a YAML header, the latter does not appear in this variable.
- "`{{ clipboard_header }}`"is the YAML section of the clipboard data, if one exists. Otherwise: empty string.
- "`{{ stdin }}`"is the complete text content originating form the input stream "`stdin`". This stream can replace the clipboard when it is not available. In case the input stream's content starts with a YAML header, it does not appear in this variable.
- "`{{ stdin_header }}`" is the YAML section of the input stream, if one exists. Otherwise: empty string.
- "`{{ extension_default }}`"is the default extension for new notes (can be changed in the configuration file),
- "`{{ username }}`" is the content of the first non-empty environment variable: `LOGNAME`, `USER` or `USERNAME`.
- "`{{ fm_title }}`"is the "`title:`"as indicated in the YAML front matter of the note (only available in filename-templates and in "`tmpl_copy_content`").
- "`{{ fm_subtitle }}`"is the "`subtitle:`"as indicated in the YAML front matter of the note (only available in filename-templates and in "`tmpl_copy_content`").
- "`{{ fm_author }}`"is the "`author:`"as indicated in the YAML front matter of the note (only available in filename-templates and in "`tmpl_copy_content`")
- "`{{ fm_lang }}`"is the "`lang:`"as indicated in the YAML front matter of the note (only available in filename-templates and in "`tmpl_copy_content`")

[5] https://tera.netlify.com/docs/#templates

- “`{{ fm_revision}}`”is the “`revision:`”as indicated in the YAML front matter of the note (only available in filename-templates and in “`tmpl_copy_content`”)
- “`{{ fm_file_ext }}`” holds the value of the optional YAML header variable “`file_ext:`” (e.g. “`file_ext: "rst"`”). This variable is only available with the “`tmpl_sync_filename`”, “`tmpl_copy_content`” and the “`tmpl_copy_filename`” templates! Note, that “`{{ fm_file_ext }}`” is undefined, when the corresponding YAML header variable is not present in the note’s header.
- “`{{ fm_sort_tag }}`”: The sort variable as defined in the YAML front matter of this note (e.g. “`sort_tag: "20200312-"`”). This variable is only available in the “`tmpl_sync_filename`”, “`tmpl_copy_content`” and the “`tmpl_copy_filename`” templates! Note, that “`{{ fm_sort_tag }}`” is undefined, when the corresponding YAML header variable is not present in the note’s header.

Except for “`{{ fm_title }}`”, there is no guarantee, that any of the “`{{ fm_* }}`” variables is defined! Depending on the last content template result, certain variables might be undefined. Please take into consideration, that a defined variable might contain the empty string “`""`”.

8.2. Template filters

In addtion to *Tera*’s built-in filters[6], *Tp-Note* comes with some additional filters, e.g.: “`tag`”, “`stem`”, “`cut`”, “`heading`”, “`linkname`”, “`linkurl`” and “`ext`”.

A filter is always used together with a variable. Here some examples:

- “`{{ file | tag }}`”is the sort-tag (numerical filename prefix) of the current note on disk, e.g. “`01-23_9-`”or “`20191022-`”. Useful in content templates, for example to create new notes based on a path with a filename (e.g. “`tmpl_annotate_content`”).
- “`{{ path | stem }}`” is the parent directory’s name of the note on disk.
- “`{{ file | stem }}`” is the note’s filename without sort-tag and extension.
- “`{{ file | ext | prepend_dot }}`”is the note’s filename extension without the dot (period), e.g. “`md`” od “`mdtxt`”.
- “`{{ clipboard | cut }}`” is the first 200 bytes from the clipboard.
- “`{{ clipboard | heading }}`”is the clipboard’s content until end of the first sentence ending, or the first newline.

[6] https://tera.netlify.app/docs/#built-in-filters

- "`{{ clipboard | linkname }}`"is the name of the first Markdown formatted link in the clipboard.
- "`{{ clipboard | linkurl }}`"is the URL of the first Markdown formatted link in the clipboard.
- "`{{ file | ext }}`" is the filename extension of the current note on disk.
- "`{{ username | json_encode }}`"is the username Json encoded. All YAML front matter must be Json encoded, so this filter should be the last in all lines of the front matter section.
- "`{{ subtitle | sanit }}`"the note's subtitle as defined in its front-matter, sanitized in a filesystem friendly form. Special characters are omitted or replaced by "`-`" and "`_`".
- "`{{ title | sanit(alpha=true) }}`"the note's title as defined in its front-matter. Same as above, but strings starting with a number are prepended by an apostrophe.

8.3. Content-template conventions

Tp-Note distinguishes two template types: content-templates "`tmpl_*_content`" used to create the note's content (front-matter and body) and filename-templates "`tmpl_*_filename`" used to calculate the note's filename.

Strings in the YAML front matter of content-templates are JSON encoded. Therefore, all variables used in the front matter must pass an additional "`json_encode()`"-filter. For example, the variable "`{{ path | stem }}`" becomes "`{{ path | stem | json_encode() }}`" or just "`{{ path | stem | json_encode }}`".

8.4. Filename-template convention

The same applies to filename-template-variables: in this context we must guarantee, that the variable contains only file system friendly characters. For this purpose *Tp-Note* provides the additional Tera filters "`sanit`" and "`sanit(alpha=true)`".

- The "`sanit()`" filter transforms a string in a file system friendly from. This is done by replacing forbidden characters like "`?`"and "`\\`"with "`_`"or space. This filter can be used with any variables, but is most useful with filename-templates. For example, in the "`tmpl_sync_filename`"template, we find the expression "`{{ subtitle | sanit }}`".
- "`sanit(alpha=true)`" is similar to the above, with one exception: when a string starts with a digit "`0`-`9`,"the whole string is prepended with `'`. For example: "`1 The Show Begins`" becomes "`'1 The Show Begins`". This filter should always

be applied to the first variable assembling the new filename, e.g. "`{{ title | sanit(alpha=true )}}`". This way, it is always possible to distinguish the sort-tag from the actual filename.

In filename-templates most variables must pass either the "`sanit`" or the "`sanit(alpha=true)`"filter. Exception to this rule are the sort-tag variables "`{{ file | tag }}`" and "`{{ path | tag }}`". As these are guaranteed to contain only the filesystem-friendly characters: "`0..9-_`", no additional filtering is required. In addition, a "`sanit()`"-filter would needlessly restrict the value range of "`{{ file | tag }}`"and "`{{ path | tag }}`": a sort tag usually ends with a "`-`", a character that the "`sanit`"-filter screens out when it appears in leading or trailing position. For this reason no "`sanit`"-filter is allowed with "`{{ file | tag }}`" and "`{{ path | tag }}`".

8.5. Register your own external text editor

The configuration file variables "`editor_args`" and "`viewer_args`" define a list of external text editors to be launched for editing. "`viewer_args`"is used when *Tp-Note* is invoked with "`--view`"in viewer mode. The list contains well-known text editor names and its command-line arguments. *Tp-Note* tries to launch every text editor from the beginning of the list until it finds an installed text editor. When *Tp-Note* is started on a Linux console, an alternative file editor list used: "`editor_console_args`"and "`viewer_console_args`". Here you can register file editors that do not require a graphical environment, e.g. "`vim`" or "`nano`".

In order to use your own text editor, just place it at the top of the list. To make this work properly, make sure, that your text editor does not fork! You can check this when you launch the text editor from the command-line: if the prompt returns immediately, then it forks the process. In contrast, it is Ok when the prompt only comes back at the moment when the text editor is closed. Many text editors provide an option not to fork: for example the *VScode*-editor can be launched with the "`--wait`"option and `vim` with `vim --nofork`. However, *Tp-Note* also works with forking text editors. Then, the only drawback is, that *Tp-Note* can not synchronize the filename with the note's metadata when the user has finished editing. It will still happen, but only when the user opens the note again with *Tp-Note*.

Remark for the advanced console user: It is also possible to launch a different editor without changing the configuration file:

```
> FILE=$(tp-note --batch); vi "$FILE"; tp-note --batch "$FILE"
```

Whereby "`FILE=$(tp-note --batch)`"creates the note file, "`vi "$FILE"`"opens the "`vi`"-file editor and "`tp-note --batch "$FILE"`" synchronizes the filename.

8.6. Register a Flatpak Markdown editor

Flathub for Linux[7] is a cross-platform application repository that works well with *Tp-Note*. To showcase an example, we will add a *Tp-Note* launcher for the *Mark Text* Markdown file editor available as Flatpak package[8]. Before installing, make sure that you have setup Flatpack[9] correctly. Then install the application with:

```
> sudo flatpak install flathub com.github.marktext.marktext
```

To test, run *Mark Text* from the command-line:

```
> flatpak run com.github.marktext.marktext
```

Then open *Tp-Note*'s configuration file `tp-note.toml` and search for the "`editor_args`" variable, quoted shortened below:

```
editor_args = [
    ['typora'],
    [
    'code',
    '-w',
    '-n',
],
#...
]
```

The structure of this variable is a list of lists. Every item in the outer list corresponds to one entire command line launching a different file editor, here *Typora* and *VSCode*. When launching, *Tp-Note* searches through this list until it finds an installed file editor on the system.

In this example, we register the *Mark Text* editor at the first place in this list, by inserting '`['flatpak', 'run', 'com.github.marktext.marktext'],`:

```
editor_args = [
    [
```

[7] https://www.flathub.org/home
[8] https://www.flathub.org/apps/details/com.github.marktext.marktext
[9] https://flatpak.org/setup/

```
    'flatpak',
    'run',
    'com.github.marktext.marktext',
],
    ['typora'],
    [
    'code',
    '-w',
    '-n',
],,
#...
]
```

Save the modified configuration file. Next time you launch *Tp-Note*, the *Mark Text*-editor will open with your note.

8.7. Change the markup language

Tp-Note is markup language agnostic, however the default templates define *Markdown* as default markup language. To change this, just edit the following 3 variables:

1. Change the variable "`extension_default`". Example: "`extension_default='rst'`".
2. Change the variable "`note_file_extension`". Example: "`note_file_extensions = [ 'rst', 'rest', 'restructuredtext' ]`".
3. The last line in the template "`tmpl_clipboard_content`" defines a hyperlink in Markdown format. Change the link format according to your markup language convention.

9. RESOURCES

Tp-Note it hosted on:

- Github: https://github.com/getreu/tp-note and on
- Gitlab (mirror): https://gitlab.com/getreu/tp-note.

10. COPYING

Licensed under either of

- Apache Licence, Version 2.0 (LICENSE-APACHE or http://www.apache.org/licenses/LICENSE-2.0)
- MIT licence (LICENSE-MIT or http://opensource.org/licenses/MIT)

at your option.

10.1. Contribution

Unless you explicitly state otherwise, any contribution intentionally submitted for inclusion in the work by you, as defined in the Apache-2.0 licence, shall be dual licensed as above, without any additional terms or conditions. Licensed under the Apache Licence, Version 2.0 (the "Licence"); you may not use this file except in compliance with the Licence.

11. AUTHORS

Jens Getreu `<getreu@web.de>`