TP-NOTE(1) Version 1.15.0 | Tp-Note documentation

Table of Contents

1. NAME

Tp-Note - save and edit your clipboard content as a note file.

2. SYNOPSIS

```
tpnote [-b] [-c <FILE>] [-d] [-e] [-p <NUM>]
       [-n] [-v] [-V] [-x <DIR>|''|'-'] [<DIR>|<FILE>]
```

3. DESCRIPTION

Tp-Note is a note taking tool and a template system, that synchronizes the note's metadata with its filename. *Tp-Note* collects various information about its environment and the clipboard and stores it 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 metadata is analysed and, if necessary, its filename is adjusted. For all other file types, *Tp-Note* creates a new note annotating 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 editor of your choice. Although the templates are written for Markdown, *Tp-Note* is not tied to any specific markup language. However, *Tp-Note* comes with an optional viewer feature, that currently renders only Markdown, ReStructuredText and HTML. Note, that there is also some limited support for Asciidoc and WikiText. The note's rendition with its hyperlinks is live updated and displayed in the user's webbrowser.

After the user finished editing, *Tp-Note* analyses eventual changes in the notes metadata and renames, if necessary, the file, so that its metadata 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.

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

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 metadata 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):

```
> tpnote "./03-Favorite Readings/"
```

or

```
> cd "./03-Favorite Readings"
> tpnote
```

creates the document:

```
"./03-Favorite Readings/20211031-Favorite Readings--Note.txt"
```

with the content:

```
---
title:      "Favorite Readings"
subtitle:   "Note"
author:     "getreu"
date:       "2021-10-31"
lang:       "en_GB.UTF-8"
---
```

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 }}`", its URL with "`{{ clipboard | linktarget }}`"and its title with "`{{ clipboard | linktitle }}`". 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 metadata 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/)" | tpnote
```

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.

```
> tpnote "./03-Favorite Readings/"
```

or

```
> cd "./03-Favorite Readings/"
> tpnote
```

This creates the document:

```
"./03-Favorite Readings/20211031-Who Moved My Cheese--Note.txt"
```

with the content:

```
---
title:      "Who Moved My Cheese"
subtitle:   "Note"
author:     "getreu"
date:       "2021-10-31"
```

```
lang:       "en_GB.UTF-8"
---

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 modify 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 hyperlink

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/)`".

```
> tpnote './doc/Lecture 1'
```

This creates the following document:

```
./doc/Lecture 1/20211031-The Rust Book--Notes.txt
```

```
---
title:      "The Rust Book"
subtitle:   "URL"
author:     "getreu"
date:       "2021-10-31"
lang:       "en_GB.UTF-8"
---

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

When analyzing the clipboard's content, *Tp-Note* searches for hyperlinks in Markdown, ReStructuredText, Asciidoc and HTML format. When successful, the content template uses the link text of the first hyperlink found 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`”.

```
> tpnote
```

This creates the note: “`20211031-Todo.mdtxt`” with the following content:

```
---
title:      "Todo"
subtitle:   ""
author:     "getreu"
date:       "2021-10-31"
lang:       "en_GB.UTF-8"
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_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" | tpnote
```

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 | tpnote
```

creates the note file “`20211031-Jens Getreu's blog.md`”with the webpage’s content converted to Markdown:

```
---
title:      "Jens Getreu's blog"
subtitle:   ""
author:     "getreu"
date:       "2021-10-31"
lang:       "en"
---


<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 | tpnote
```

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 "`.txt`", 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 metadata.

Example:

```
> tpnote "Classic Shell Scripting.pdf"
```

creates the note:

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

with the content:

```
---
title:      "Classic Shell Scripting.pdf"
subtitle:   "Note"
author:     "getreu"
date:       "2021-10-31"
lang:       "en_GB.UTF-8"
---

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

The configuration file variables "`[filename] 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

Unless invoked with "`--batch`"or "`--view`", *Tp-Note* launches an external text editor after creating a new note. This also happens when "`<path>`" points to an existing "`.txt`"-file.

Example: edit the note from the previous example:

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

```
> tpnote 20211031-Favorite Readings--Note.txt
```

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:

```
> tpnote "20200306-Favorite Readings--Note.txt"
```

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.txt"
```

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. In batch mode, error messages are dumped on the console only and no alert window pops up.

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 *FILE*, --config=*FILE*

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

-d *LEVEL*, --debug=*LEVEL*

Print additional log messages. The debug level *LEVEL* must be one out of "`trace`", "`debug`", "`info`", "`warn`", "`error`"(default) or "`off`". The level "`trace`"reports the most detailed information, while "`error`"informs only about failures. A "`warn`" level message means, that not all functionality might be available or work as expected.

Use "`-b -d trace`"for debugging templates, if the HTTP server (viewer) does not work as expected "`-n -d debug`", if your text editor does not open as expected "`-n -d info --edit`" or to observe the launch of the web browser "`-n -d info --view`". The option "`-d trace`"shows all available template variables, the templates used and the rendered result of the substitution, which is particularly useful for debugging new templates. The option "`-d off`"silences all error message reporting and suppresses also the error popup window.

All error messages are dumped in the error stream `stderr` and appear on the console from where *Tp-Note* was launched:

```
tpnote.exe --debug info my_note.txt
```

On Windows the output must be redirected into a file to see it:

```
tpnote.exe --debug info my_note.txt >debug.txt 2>&1
```

Alternatively, you can redirect all logfile entries into popup alert windows.

```
tpnote.exe --popup --debug info my_note.txt
```

The same can be achieved by setting following configuration file variables (especially useful under Windows):

```
[arg_default]
debug = 'info'
popup = true
```

The value for "`[arg_default] debug`"must be one out of "`trace`", "`debug`", "`info`", "`warn`", "`error`"(default) and "`off`". They have the same meaning as the corresponding command line options.

-e, --edit

Edit only mode: opens the external text editor, but not the file viewer. This disables *Tp-Note*'s internal file watcher and web server, unless "`-v`" is given. Another way to permanently disable the web server is to set the configuration variable "`[viewer] enable=false`". When "`--edit --view`" appear together, "`--view`" takes precedence and "`--edit`" is ignored.

-p, --port=*PORT*

Set server port the web browser connects to, to the specified value *PORT*. If not given, a random free port is chosen automatically.

-n, --no-filename-sync

Whenever *Tp-Note* opens a note file, it synchronizes its YAML-metadata with its filename. "`--no-filename-sync`" disables the synchronization. Mainly useful is this flag in scripts for testing "`.txt`"-files. See section EXIT STATUS for more details. The section METADATA FILENAME SYNCHRONIZATION shows alternative ways to disable synchronisation.

-t, --tty

Tp-Note tries different heuristics to detect weather a graphic environment is available or not. For example, under Linux, the "`DISPLAY`"environment variable is evaluated. The "`--tty`" flag disables the automatic detection and sets *Tp-Note* in "console" mode, where only the non GUI editor (see configuration variable: "`[app_args] editor_console`") and no viewer is launched.

-u, --popup

Redirect log file entries into popup alert windows. Must be used together with the **--debug** option to have an effect. Note, that debug level "`error`"conditions will always trigger popup messages, regardless of **--popup** and **--debug** (unless "`--debug off`"). Popup alert windows are queued and will never interrupt *Tp-note*. To better associate a particular action with its log events, read through all upcoming popup alert windows until they fail to appear.

-v, --view

View only mode: do not open the external text editor. This flag instructs *Tp-Note* to start an internal file watcher and web server and connect the system's default web browser to view the note file and to observe live file modifications. This flag has precedence over the configuration variable "`[viewer] enable=false`". When "`--edit --view`" appear together, "`--view`" takes precedence and "`--edit`" is ignored.

-V, --version

Print *Tp-Note*'s version, its compiled-in features and the path to the sourced configuration file. The output is YAML formatted for further automatic processing.

-x *DIRECTORY*, --export=*DIRECTORY*

Print the note as HTML-rendition into *DIRECTORY*. "`-x -`"prints to *stdout*. The empty string, e.g. "`--export=`"or "`-x ""`", defaults to the directory where the note file resides. No external text editor or viewer is launched. Can be combined with "`--batch`"to avoid popup error alert windows.

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. By convention, a valid *Tp-Note* file has at least one YAML field named "`title:`" (the name of this compulsory field is defined by the "`[tmpl] compulsory_header_field`" variable in the configuration file and can be changed there).

Note that prepended text, placed before the YAML front-matter, is ignored. There are however certain restrictions: If present, the skipped text should not be too long (cf. constant "`BEFORE_HEADER_MAX_IGNORED_CHARS`"in the source code of *Tp-Note*) and it must be followed by at least one blank line:

```
Prepended text is ignored.

---
<YAML-front-matter>
---
```

```
<document-body>
```

There is no restriction about the markup language being used in the note's text body. However, the default templates assume Markdown and the file extension "`.txt`". Both can be changed easily by adapting *Tp-Note*'s configuration file. Besides the requirements concerning its header, a valid *Tp-Note* file must have a filename extension that is listed in one of the configuration file variables: "`[filename] extension_*`". The latter also determine which internal markup language render is called for *Tp-Note*'s renderer feature.

7. METADATA FILENAME SYNCHRONIZATION

Consider the following *Tp-Note*-file:

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

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 and whitespace, a *sort tag* can be any combination of `-_.`[^sort tag] and is usually used as

- *chronological sort tag*

  ```
  20140211-Reminder.doc
  20151208-Manual.pdf
  2015-12-08-Manual.pdf
  ```

- or as a *sequence number sort tag*.

  ```
  02-Invoices
  08-Tax documents
  09_02-Notes
  09.09-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*.

[^sort tag]: The characters “ _”, “ -”, “”, “ \t”and “ .”are considered to be part of the *sort tag* even when they appear in last position.

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 }}.txt” [3]

Consider the following document with the filename:

```
20211031-My file.txt
```

and the content:

```
---
title:      "1. The Beginning"
subtitle:   "Note"
author:     "getreu"
date:       "2021-10-31"
lang:       "en_GB.UTF-8"
---
```

As “ -My file.txt”is not equal to “ -'1. The Beginning--Note.txt”, *Tp-Note* will rename the file to “ 20211031-'1. The Beginning--Note.txt”. If the filename had been “ 05_02-My file.txt”, it would rename it to “ 05_02-'1. The Beginning--Note.txt”.

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:

[3] The variables “ {{ fm_title }}” and “ {{ fm_subtitle }}” reflect the values in the note’s metadata.

```
---
title:      "1. The Beginning"
date:       "2021-10-31"
sort_tag:   "20211101-"
---
```

When *Tp-Note* synchronizes the note's metadata with its filename, it will also change the sort tag from "`20211031-`"to "`20211101-`". The resulting filename becomes "`20211101-'1. The Beginning--Note.txt`".

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 metadata, 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:   "rst"
---
```

This will change the file extension from "`.txt`"to '`.rst`. The resulting filename becomes "`20211101-'1. The Beginning--Note.rst`".

Important: "`rst`"must be one of the registered file extensions listed in the "`[filename] extensions_rst`" variables in *Tp-Note*'s configuration file. If needed you can add more extensions there. If the new filename extension is not listed in one of theses variables, *Tp-Note* will not be able to recognize the note file as such and will not open it in the external text editor and viewer.

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.

The metadata filename synchronisation feature can be disabled permanently by setting the configuration file variable "`[arg_default] no_filename_sync = true`". To disable this feature for one time only, invoke *Tp-Note* with "`--no-filename-sync`". To exclude a particular note from filename synchronisation, add the YAML header field "`filename_sync: false`".

```
---
title:      "1. The Beginning"
filename_sync: false
---
```

8. CUSTOMIZATION

Tp-Note's configuration file resides typically in "`~/.config/tpnote/tpnote.toml`" on Unix or in "`C:\Users\<LOGIN>\AppData\Roaming\tpnote\config\tpnote.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. For a detailed description of the available configuration variables, please consult the "`const`"definitions in *Tp-Note*'s source code file "`config.rs`"

The configuration file is encoded according to the TOML-standard. Variables ending with "`[tmpl] *_content`"and "`[tmpl] *_filename`"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 "`{{ dir_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 trace`"option and observe what information it captures from its environment.

8.1. Template variables

All Tera template variables and functions[4] can be used within *Tp-Note*. For example "`{{ get_env(name='LANG') }}'` gives you access to the"`LANG`' environment variable.

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

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

- "`{{ path }}`"is the canonicalized fully qualified path name corresponding to *Tp-Note*'s positional command line parameter "`<path>`". If none was given on the command line, "`{{ path }}`" contains the current working directory path.
- "`{{ dir_path }}`": is identical to "`{{ path }}`" with one exception: if "`{{ path }}`"points to a file, the last component (the file name) is omitted and only the directory path is retained. If "`{{ path }}`"points to a directory, "`{{ path_dir }}`" equals "`{{ path }}`".
- "`{{ 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, the latter 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`.

The following "`{{ fm_* }}`"variables are typically generated, *after* a content template was filled in with data: For example a field named "`title:`"in the content template "`[tmpl] new_content`" will generate the variable "`fm_title`" which can then be used in the corresponding "`[tmpl] new_filename`"filename template. "`{{ fm_* }}`"variables are generated dynamically. This means, a YAmL front-matter variable "`foo:`"in a note will generate a "`{{ fm_foo }}`"template variable. On the other hand, a missing "`foo:`"will cause "`{{ fm_foo }}`" to be undefined.

Please note that "`{{ fm_* }}`"variables are available in all filename templates and in the "`[tmpl] copy_content`" content template only.

- "`{{ fm_title }}`" is the "`title:`" as indicated in the YAML front-matter of the note.
- "`{{ fm_subtitle }}`"is the "`subtitle:`"as indicated in the YAML front matter of the note.

- “`{{ fm_author }}`”is the “`author:`”as indicated in the YAML front-matter of the note.
- “`{{ fm_lang }}`” is the “`lang:`” as indicated in the YAML front-matter of the note.
- “`{{ fm_file_ext }}`” holds the value of the optional YAML header variable “`file_ext:`” (e.g. “`file_ext: "rst"`”).
- “`{{ fm_sort_tag }}`”: The sort tag variable as defined in the YAML front matter of this note (e.g. “`sort_tag: "20200312-"`”).
- “`{{ fm_all }}`”: is a collection (map) of all defined “`{{ fm_* }}`”variables. It is used in the “`[tmpl] copy_content`” template, typically in a loop like:

```
{% for key, value in fm_all %}{{ key }}: {{ value | json_encode }}
{% endfor %}
```

Important: there is no guarantee, that any of the above “`{{ 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 “`""`”.

For a more detailed description of the available template variables, please consult the “`const`” definitions in *Tp-Note*’s source code file “`note.rs`”

8.2. Template filters

In addition to *Tera*’s built-in filters[5], *Tp-Note* comes with some additional filters, e.g.: “`tag`”, “`trim_tag`”, “`stem`”, “`cut`”, “`heading`”, “`linkname`”, “`linktarget`”, “`linktitle`” and “`ext`”.

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

- “`{{ path | filename }}`”returns the final component of “`{{ path }}`”. If “`{{ path }}`”points to a file, the filter returns the complete filename including its sort tag, stem, copy-counter, dot and extension. If the “`<path>`”points to a directory, the filter returns the final directory name.
- “`{{ path | tag }}`”is the sort tag (numerical filename prefix) of the final component of “`{{ path }}`”, e.g. “`01-23_9-`”or “`20191022-`”. It is similar to “`{{ path | filename }}`” but without returning its stem, copy-counter and extension.

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

- “`{{ path | stem }}`”is similar to “`{{ path | filename }}`”but without its sort tag, copy-counter and extension. Only the stem of “`{{ path }}`”s last component is returned.
- “`{{ path | copy_counter }}`”is similar to “`{{ path | filename }}`” but without its sort tag, stem and extension. Only the copy counter of “`{{ path }}`”s last component is returned.
- “`{{ path | ext }}`”is “`{{ path }}`”“s file extension without dot (period), e.g. ”.txt`' or '`.md`’.
- “`{{ path | ext | prepend_dot }}`”is “`{{ path }}`”“s file extension with dot (period), e.g. ”`.txt`’ or “`.md`”.
- “`{{ path | trim_tag }}`”returns the final component of “`path`”which might be a directory name or a file name. Unlike the “`filename`”filter (which also returns the final component), “`trim_tag`” trims the sort tag if there is one.
- “`{{ dir_path | trim_tag }}`”returns the final component of “`dir_path`” (which is the final directory name in “`{{ path }}`”). Unlike the “`filename`”filter (which also returns the final component), “`trim_tag`” trims the sort tag if there is one.
- “`{{ 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.
- “`{{ clipboard | linkname }}`” is the name of the first Markdown or reStructuredText formatted link in the clipboard.
- “`{{ clipboard | linktarget }}`” is the URL of the first Markdown or reStruncturedText formatted link in the clipboard.
- “`{{ clipboard | linktitle }}`” is the title of the first Markdown or reStruncturedText formatted link in the clipboard.
- “`{{ 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.
- “`{{ fm_subtitle | sanit }}`”is the note’s subtitle as defined in its front-matter, sanitized in a filesystem friendly form. Special characters are omitted or replaced by “`-`” and “`_`”. See the section *Filename-template convention* for more details about this filter.
- “`{{ fm_title | sanit(alpha=true) }}`”is the note’s title as defined in its front-matter. Same as above, but strings starting with a sort tag are prepended by an apostrophe to avoid ambiguity.
- “`{{ fm_all | remove(var='fm_title') }}`”represents a collection (map) of all “`fm_*`” variables, exclusive of the variable “`fm_title`”.

8.3. Content-template conventions

Tp-Note distinguishes two template types: content-templates are used to create the note's content (front-matter and body) and the corresponding filename-templates "`[tmpl] *_filename`" are used to calculate the note's filename. By convention, content templates appear in the configuration file in variables named "`[tmpl] *_content`".

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 "`{{ dir_path | stem }}`"becomes "`{{ dir_path | stem | json_encode() }}`" or just "`{{ dir_path | stem | json_encode }}`".

8.4. Filename-template convention

By convention, filename templates appear in the configuration file in variables named "`[tmpl] *_filename`". When a content-template creates a new note, the corresponding filename-templates is called afterwards to calculate the filename of the new notes. The filename template "`[tmpl] sync_filename`"has a special role as it is synchronizes the filename of existing note files. As we are dealing with filenames we must guarantee, that the templates produce 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 variable, but is most useful with filename-templates. For example, in the "`[tmpl] sync_filename`" template, we find the expression "`{{ subtitle | sanit }}`". Note that the filter recognizes strings that represent a so called dot file name and treats them a little differently by prepending them with an apostrophe: a dot file is a file whose name starts with "`.`"and that does not contain whitespace. It may or may not end with a file extension. The apostrophe preserves the following dot from being filtered.
- "`sanit(alpha=true)`"is similar to the above, with one exception: when a string starts with a digit "`0123456789`"or "`-_`", 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. The default sort tag separator "`'`"can be changed with the configuration variable "`[filename] sort_tag_extra_separator`".

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 "`{{ path`

`| tag }}`" and "`{{ dir_path | tag }}`". As these are guaranteed to contain only the filesystem friendly characters "`0123456789 -_`", no additional filtering is required. Please note that in this case a "`sanit`"-filter would needlessly restrict the value range of sort tags as they usually end with a "`-`", a character, which the "`sanit`"-filter screens out when it appears in leading or trailing position. For this reason no "`sanit`"-filter is not allowed with "`{{ path | tag }}`" and "`{{ dir_path | tag }}`".

8.5. Register your own text editor

The configuration file variables "`[app_args] editor`" and "`[app_args] editor_console`" define lists of external text editors to be launched for editing. The lists contain by default well-known text editor names and their command line arguments. *Tp-Note* tries to launch every text editor in "`[app_args] editor`"from the beginning of the list until it finds an installed text editor. When *Tp-Note* is started on a Linux console, the list "`[app_args] editor_console`"is used instead. Here you can register text 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 debug your changes invoke *Tp-Note* with "`tpnote --debug info --popup --edit`"

When you configure *Tp-Note* to work with your text editor, make sure, that your text editor does not fork! You can check this by launching the text editor from the command line: if the command prompt returns immediately, then the file editor forks the process. On the other hand everything is OK, when the command prompt only comes back at the moment the text editor is closed. Many text editors provide an option to restrain from forking: for example the *VScode*-editor can be launched with the "`--wait`"option or *Vim* with "`--nofork`". However, *Tp-Note* also works with forking text editors. Although this should be avoided, there is a possible workaround:

```
> FILE=$(tpnote --batch) # Create the new note.
> mytexteditor "$FILE"    # The prompt returns immediatly as the editor
 forks.
> tpnote --view "$FILE"  # Launch Tp-Note's viewer.
>                         # After the editing is done...
> tpnote --batch "$FILE" # Synchronize the note's filename.
```

Remark for the advanced console user: In a similar way, you can launch a different text editor than the one configured in *Tp-Note*'s configuration file:

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

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

Register a Flatpak Markdown editor

Flathub for Linux[6] 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 text editor available as Flatpak package[7]. Before installing, make sure that you have setup Flatpack[8] 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 `tpnote.toml` and search for the “`[app_args] editor`” variable, quoted shortened below:

```
[app_args]
editor = [
    [
    '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 text editor, here *VSCode*. When launching, *Tp-Note* searches through this list until it finds an installed text 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'],`:

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

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

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

Register a console text editor running in a terminal emulator

In this setup *Tp-Note* launches the terminal emulator which is configured to launch the text editor as child process. Both should should not fork when they start (see above).

Examples, adjust to your needs and taste:

- *Neovim* in *Xfce4-Terminal*:

  ```
  [app_args]
  editor = [
    [
      'xfce4-terminal',
      '--disable-server',
      '-x',
      'nvim',
      '+colorscheme pablo',
      '+set syntax=markdown',
    ],
  ]
  ```

- *Neovim* in *LXTerminal*:

  ```
  [app_args]
  editor = [
    [
  ```

```
    'lxterminal',
    '--no-remote',
    '-e',
    'nvim',
    '+colorscheme pablo',
    '+set syntax=markdown',
  ],
]
```

- *Neovim* in *Xterm*:

```
[app_args]
editor = [
    [
      'xterm',
      '-fa',
      'DejaVu Sans Mono',
      '-fs',
      '12',
      '-e',
      'nvim',
      '+colorscheme pablo',
      '+set syntax=markdown',
  ],
]
```

8.6. Change the default markup language

Tp-Note identifies the note's markup language by its file extension and renders the content accordingly (see "`[filename] extensions_*`"variables). This ensures interoperability between authors using different markup languages. Although *Tp-Note* is flexible in opening existing note files, new notes are always created in the same markup language, which is by default *Markdown*. How to change this is shown in the following section.

Change the way how new notes are created

Tp-Note's core function is a template system and as such it depends very little on the used markup language. The default templates are designed in a way that they contain almost no markup specific code. There is one little exception though. The following configuration variables affect the way new notes are created:

1. Change the default file extension for new notes from:

```
[filename]
extension_default=.txt'
```

to:

```
[filename]
extension_default='rst'
```

2. Replace the following line in the template "`[tmpl] clipboard_content`" that defines a hyperlink in Markdown format:

```
[{{ path | tag }}{{ path | stem }}{{ path | ext | prepend_dot }}]
(<{{ path | tag }}{{ path | stem }}{{ path | ext | prepend_dot }}>)
```

with the following line encoded in *RestructuredText*:

```
`<{{ path | tag }}{{ path | stem }}{{ path | ext | prepend_dot }}>`_
```

As a result, all future notes are created as "`*.rst`" files.

Change the markup language for one specific note only

You can change the Markup language of a specific note by adding the variable "`file_ext:`" to its YAML header. For example, for *ReStructuredText* add:

```
---
title:    "some note"
file_ext: "rst"
---
```

The above change only applies to the current note only.

8.7. Change the sort tag generation scheme

Sort tags for new notes are generated with the "`[TMPL] *_filename`" templates and updated with the "`[TMPL] sync_filename`" template. By default, the characters "`_`", "`-`", *space*, "`\t`" and "`.`" are recognized as being part of a *sort tag* when they

appear at the beginning of a filename. This set of characters can be modified with the "`[filename] sort_tag_chars`" configuration variable. In addition, one special character "`[filename] sort_tag_extra_separator`"(by default "`'`") is sometimes used as "end of sort tag marker" to avoid ambiguity. Note: the above templates and character sets must be matched carefully to prevent cyclic filename change! To debug your "`[TMPL] sync_filename`" template, create a test note file "`test.txt`"and invoke *Tp-Note* with "`--debug trace`" and "`--batch`":

```
tpnote --batch --debug trace test.txt
```

8.8. Store new note files by default in a subdirectory

When you are annotating an existing file on disk, the new note file is placed in the same directory by default. To configure *Tp-Note* to store the new note file in a subdirectory, lets say "`Notes/`", instead, you need to modify the templates "`[tmpl] annotate_filename`"and "`[tmpl] annotate_content`":

Replace in "`[tmpl] annotate_filename`" the string:

```
{{ path | tag }}
```

with:

```
Notes/{{ path | tag }}
```

and in "`[tmpl] annotate_content`":

```
[{{ path | filename }}](<{{ path | filename }}>)
```

with (Linux, MacOS):

```
[{{ path | filename }}](<ParentDir../{{ path | filename }}>)
```

or with (Windows):

```
[{{ path | filename }}](<ParentDir..\\{{ path | filename }}>)
```

Please note that webbrowsers usually ignore leading "`../`"in URL paths. To work around this limitation, *Tp-Note*'s built-in viewer interprets the string "`ParentDir..`"as an alias of "`..`". It is also worth mentioning that *Tp-Note* automatically creates the subdirectory "`Notes/`" in case it does not exist.

8.9. Customize the built-in note viewer

Delay the launch of the web browser

By default, *Tp-Note* launches two external programs: some text editor and a web browser. If wished for, the configuration variable "`[viewer] startup_delay`"allows to delay the launch of the web browser some milliseconds. This way the web browser window will always appear on top of the editor window. A negative value delays the start of the text editor instead.

Change the way how note files are rendered for viewing

Besides its core function, *Tp-Note* comes with several built-in markup renderer and viewer, allowing to work with different markup languages at the same time. The configuration file variables "`[filename] extensions_*`" determine which markup renderer is used for which note file extension. Depending on the markup language, this feature is more or less advanced and complete: *Markdown* (cf. "`[filename] extensions_md`") is best supported and feature complete: It complies with the *Common Mark* specification. The *ReStructuredText* renderer (cf. "`[filename] extensions_rst`") is quit new and still in experimental state. For all other supported markup languages *Tp-Note* provides a built-in markup source text viewer (cf. "`[filename] extensions_txt`") that shows the note as typed (without markup), but renders all hyperlinks to make them clickable. In case none of the above rendition engines suit you, it is possible to disable the viewer feature selectively for some particular note file extensions: just place these extensions in the "`[filename] extensions_no_viewer`" variable. If you wish to disable the viewer feature overall, set the variable `[arg_default] edit = true`.

Change the HTML rendition template

After the markup rendition process, *Tp-Note*'s built-in viewer generates its final HTML rendition through the customizable HTML templates "`[viewer] rendition_tmpl`" and "`[viewer] error_tmpl`". The following code example taken from "`[viewer] rendition_tmpl`" illustrates the available variables:

```
[viewer]
rendition_tmpl = '''<!DOCTYPE html>
<html lang="{{ fm_lang | default(value='en') }}">
<head>
<meta charset="utf-8">
<title>{{ fm_title }}</title>
  </head>
  <body>
  <pre class="note-header">{{ fm_all_yaml }}</pre>
  <hr>
  <div class="note-body">{{ note_body }}</div>
  <script>{{ note_js }}</script>
</body>
</html>
'''
```

Specifically:

- “`{{ fm_* }}`”are the deserialized header variables. All content template variables and filters are available. See section *Template variables* above.
- “`{{ fm_all_yaml }}`”is the raw UTF-8 copy of the header. Not to be confounded with the dictionary variable “`{{ fm_all }}`”.
- “`{{ note_body }}`” is the note’s body as HTML rendition.
- “`{{ note_js }}`” is the Java-Script browser code for live updates.

Alternatively, the header enclosed by “`<pre>...</pre>`” can also be rendered as a table:

```
  <table>
    <tr><th>title:</th><th>{{ fm_title }}</th> </tr>
    <tr><th>subtitle:</th><th>{{ fm_subtitle | default(value='') }}</th></
tr>
  {% for k, v in fm_all| remove(var='fm_title')| remove(var='fm_subtitle')
 %}
    <tr><th>{{ k }}:</th><th>{{ v }}</th></tr>
  {% endfor %}
  </table>
```

The error page template “`[viewer] error_tmpl`” (see below) does not provide “`fm_*`” variables, because of possible header syntax errors. Instead, the variable “`{{ note_error }}`” contains the error message as raw UTF-8 and the variable “`{{ note_erroneous_content }}`” the HTML rendition of the text source with clickable hyperlinks:

```
[viewer]
error_tmpl = '''<!DOCTYPE html>
<html lang=\"en\">
<head>
<meta charset=\"utf-8\">
<title>Syntax error</title>
</head>
<body>
<h3>Syntax error</h3>
<p> in note file: <pre>{{ path }}</pre><p>
<hr>
<pre class="note-error">{{ note_error }}</pre>
<hr>
{{ note_erroneous_content }}
<script>{{ note_js }}</script>
</body>
</html>
'''
```

Customize the built-in HTML exporter

Customizing *Tp-Note*'s HTML export function works the same way than customizing the built-in viewer. There are some slight differences though: The role of the "`[viewer] rendition_tmpl`" template - discussed above - is taken over by the "`[exporter] rendition_tmpl`" template. In this template the same *Tera* variables are available, except "`{{ note_js }}`" which does not make sense in this context. As the exporter prints possible rendition error messages on the console, there is no equivalent to the "`[viewer] error_tmpl`" template.

8.10. Choose your favourite web browser as note viewer

Once the note is rendered into HTML, *Tp-Note*'s internal HTTP server connects to a random port at the "`localhost`" interface where the rendition is served to be viewed with a web browser. *Tp-Note*'s configuration file contains a list "`[app_args] browser`" with common web browsers and their usual location on disk. This list is executed top down until a web browser is found and launched. If you want to view your notes with a different web browser, simply modify the "`[app_args] browser`" list and put your favourite web browser on top.

In case none of the listed browsers can be found, *Tp-Note* switches into a fall back mode with limited functionality, where it tries to open the system's default web browser. A disadvantage is, that in fall back mode *Tp-Note* is not able to detect when the user closes the web browser. This might lead to situations, where *Tp-Note*'s internal HTTP server shuts down to early. In order to

check if *Tp-Note* finds the selected web browser as intended, invoke *Tp-Note* with “`tpnote --debug info --popup --view`”.

9. SECURITY AND PRIVACY CONSIDERATIONS

As discussed above, *Tp-Note*’s built-in viewer sets up an HTTP server on the “`localhost`” interface with a random port number. This HTTP server runs as long as the as long as the launched web browser window is open. It should be remembered, that the HTTP server not only exposes the rendered note, but also some other (image) files starting from the parent directory (and all subdirectories) of the note file. For security reasons symbolic links to files outside the note’s parent directory are not followed. Furthermore, *Tp-Note*’s built-in HTTP server only serves files that are explicitly referenced in the note document and whose file extensions are registered with the “`[viewer] served_mime_type`”configuration file variable. As *Tp-Note*’s built-in viewer binds to the “`localhost`”interface, the exposed files are in principle accessible to all processes running on the computer. As long as only one user is logged into the computer at a given time, no privacy concern is raised: any potential note reader must be logged in, in order to access the `localhost` HTTP server.

This is why on systems where multiple users are logged in at the same time, it is recommended to disable *Tp-Note*’s viewer feature by setting the configuration file variable “`[arg_default] edit = true`”. Alternatively, you can also compile *Tp-Note* without the “`viewer`”feature. Note, that even with the viewer feature disabled, one can still render the note manually with the “`--export`” option.

Summary: As long as *Tp-Note*’s built-in note viewer is running, the note file and all its referenced (image) files are exposed to all users logged into the computer at that given time. This concerns only local users, *Tp-Note* never exposes any information to the network.

10. EXIT STATUS

Normally the exit status is “`0`”when the note file was processed without error or “`1`”otherwise. If *Tp-Note* can not read or write its configuration file, the exit status is “`5`”.

When “`tpnote -n -b <FILE>`”returns the code “`0`”, the note file has a valid YAML header with a “`title:`”field. In addition, when “`tpnote -n -b -x - <FILE>`”returns the code “`0`”, the note’s body was rendered without error.

11. RESOURCES

Tp-Note it hosted on:

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

12. COPYING

12.1. Contribution

13. AUTHORS

Jens Getreu `<getreu@web.de>`