1 Overview

Light \LaTeX\ Make (llmk) is yet another build tool specific for \LaTeX\ documents. Its aim is to provide a simple way to specify a workflow of processing \LaTeX\ documents and encourage people to always explicitly show the right workflow for each document.

The main features of llmk are all about the above purpose. First, you can write the workflows either in an external file `llmk.toml` or in a \LaTeX\ document source in a form of magic comments. Further, multiple magic comment formats can be used. Second, it is fully cross-platform. The only requirement of the program is the `texlua` command; llmk makes a uniform way to describe the workflows available for nearly all \TeX\ environments. Third, it behaves exactly the same in any environment. At this point, llmk intentionally does not provide any method for user configuration. Therefore, one can guarantee that a \LaTeX\ document with an llmk setup, the process of typesetting the document must be reproduced in any \TeX\ environment with the program.

1.1 Installation

This software is included in \TeX\ Live and MiK\TeX\ as Package `light-latex-make`. If you have one of the latest distributions, you normally don’t need to install it by yourself. If you want to install the development version, please refer to our `README.md`.

1.2 Learning llmk

The bundled `README.md` has a general introduction for the program. If you are new to llmk and looking for a quick guidance, you are recommended to read it first. Conversely, this document can be regarded as a reference manual: it contains detailed descriptions for every feature of llmk as much as possible, but unsuitable for getting general ideas of its basic usage.
All official resources are available from the repository on GitHub:

https://github.com/wtsnjp/llmk

Notably, you can find some example \LaTeX\ documents with llmk setups in the examples directory. The design concept of llmk is described in a separate TUGboat article [1]. It will not give you practical tips, but if you are interested in the underlying ideas of the program, it should be worth discussing. The differences from other similar tools, e.g., latexmk [3] and arara [2], are also discussed in the article.

1.3 Reporting bugs and requesting features

If you get trouble with llmk or think you have found a bug, please report it by creating either an issue or a pull request on the GitHub repository:

https://github.com/wtsnjp/llmk/issues

If you do not want to use GitHub for some reasons, it is also fine to directly send an email to the author (tkt.asakura@gmail.com).

Requests for new features are also welcome; the author cannot promise to implement the requested features, but will happy to take them into account. Before making a request, it is strongly recommended to read the article about the design concept [1].

2 Command-line interface

2.1 Command usage

The full usage of llmk can be summarized as follows:

llmk ⟨options⟩ ⟨files⟩

Herein, ⟨options⟩ are the command-line options, that start with the hyphen character -, and ⟨files⟩ are the arguments for the llmk command.

Arguments ⟨files⟩

You can specify the filenames of the source \TeX\ files, normally the files with .tex or .ltx extensions, as the arguments for the llmk command. When one or more ⟨files⟩ are specified, llmk will read either the TOML field (Section 3) or another supported magic comment (Section 4) in each of the source files and process it with the specified workflow in the given order.

As well as the tex command, you can omit the .tex extensions and just give the basenames of the files for the argument; when a ⟨basename⟩, which must not contain any dot character, is
given and the file that exactly matches to the name does not exist, llmk will automatically add the .tex extension and process it like any other if the file ⟨basename⟩.tex exists.

Note that llmk naively pass the given filenames to invoked commands. Filenames that contains special characters of TeX, e.g., # and %, are very likely to be causes of troubles. Moreover, at this point llmk does not any specific features to take care of multi-byte characters: filenames including multi-byte characters may work in some cases but can be cause of problems\footnote{The author admits that llmk needs to be enhanced in this aspect: it should have better features to treat various filenames with multi-byte characters, though the author is negative to support special characters of TeX. Suggestions and patches in this direction are especially welcome.}. Using filenames that contain only the characters in the range of the ASCII code, except special characters of TeX, is the safest way to go at any rate.

Alternatively, if the argument is not specified, llmk will read the special TOML file llmk.toml in the working directory and execute the workflow specified in the file (see Section 3). In case the argument is not specified and the llmk.toml does not exist, it will result in an error. When llmk executes the workflow specified in the file llmk.toml, all magic comments, including TOML fields and other formats, in each source \LaTeX files specified in the source array will be ignored.

**Command-line options (options)**

We have tried to implement a GNU-compatible option parser. Short options, each of which consists of a single letter, must start with a single hyphen -. Multiple short options can be specified with a single hyphen, e.g., -vs is equivalent to -v -s. Long options have to be following double hyphens --. All options must be specified before the first argument. A string beginning with a hyphen after the first argument will be treated as an argument.

When two or more options are specified, llmk applies them in the given order. If contradicting options are specified, e.g., -q v.s. -v, the option in the latter position wins over the former one.

-\texttt{-c}, \texttt{--clean}

Removes temporary files such as aux and log files. The files removed with this action can be customized with keys clean_files and extra_clean_files.

-\texttt{-C}, \texttt{--clobber}

Removes all generated files including final PDFs. The files removed with this action can be customized with the key clobber_files.

-\texttt{-d} ⟨category⟩, \texttt{--debug=⟨category⟩}, \texttt{-D}, \texttt{--debug}

Activates the specified debug category; debugging messages related to the activated category will be shown. Herein, available debug categories are: config, parser, run, fdb, programs, and all to activate all of these. You can repeat this option more than once to activate multiple categories. If you specify -D or --debug without the argument (category), it activates all available debug categories.
-h, --help

Shows a quick help message (namely a list of command-line options) and exit successfully. When this is specified, all other options and arguments are ignored.

-n, --dry-run

Show what would have been executed without actually invoking the commands. This flag is useful if you want to make sure whether your configuration will work as expected before the actual building. With option --verbose, you can get further detailed information.

-q, --quiet

This suppress most of the messages from the program.

-s, --silent

Silence messages from invoked programs. To be more specific, this redirects both standard output and standard error streams to the null device.

-v, --verbose

Make llmk to print additional information such as invoked commands with options and arguments by the program.

-V, --version

Shows the current version of the program and exit successfully. When this is specified, all other options and arguments are ignored.

2.2 Exit codes

You can grasp whether llmk successfully executed or not by seeing its status code. Note that the exit codes of invoked programs are not directly transferred as the exit code of llmk; instead, the statuses of external programs that failed, if any, are reported in the error messages.

<table>
<thead>
<tr>
<th>Code</th>
<th>Error type</th>
<th>Code</th>
<th>Error Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
<td>3</td>
<td>Parser Error</td>
</tr>
<tr>
<td>1</td>
<td>General Error</td>
<td>4</td>
<td>Type Error</td>
</tr>
<tr>
<td>2</td>
<td>Invoked program failed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3 Writing workflows in TOML format

The primary configuration format for llmk is TOML — Tom’s Obvious Minimal Language [8]. You can specify the workflows to process your \LaTeX{} documents in the format either in the special configuration file (llmk.toml) or in the TOML field (Section 3.2). You have full access to the llmk configuration with this primary format, while other supported magic comment formats (Section 4) have only partial access. You can read the entire TOML specification at its website.

3.1 TOML in llmk

The configuration for llmk written in the TOML format is read by our built-in parser. At this point, the built-in parser supports a subset of the TOML specification; only the data-types that necessary for the configuration keys (Section 3.3 & 3.4) are supported.

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bare keys</td>
<td>key</td>
<td>✓</td>
</tr>
<tr>
<td>Quoted keys</td>
<td>&quot;key&quot;</td>
<td></td>
</tr>
<tr>
<td>Dotted keys</td>
<td>tex.latex</td>
<td>✓</td>
</tr>
<tr>
<td>Basic strings</td>
<td>&quot;str&quot;</td>
<td>✓</td>
</tr>
<tr>
<td>Multi-line basic strings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Literal strings</td>
<td>'str'</td>
<td>✓</td>
</tr>
<tr>
<td>Multi-line literal strings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integer</td>
<td>123</td>
<td>✓</td>
</tr>
<tr>
<td>Boolean</td>
<td>true</td>
<td>✓</td>
</tr>
<tr>
<td>Float</td>
<td>3.14</td>
<td></td>
</tr>
<tr>
<td>Date &amp; Time</td>
<td>1979-05-27</td>
<td></td>
</tr>
<tr>
<td>Array</td>
<td>[1, 2, 3]</td>
<td>✓</td>
</tr>
<tr>
<td>Table</td>
<td>[table]</td>
<td>✓</td>
</tr>
<tr>
<td>Inline table</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Array of tables</td>
<td>[[fruit]]</td>
<td></td>
</tr>
</tbody>
</table>

3.2 Where to write

You have two options to write the configuration for llmk in TOML format: (1) creating a special file llmk.toml and (2) writing a TOML field in your \LaTeX{} source file. Either way, you have full access to the llmk configuration and specify the same workflows in (almost) the same manner.
Special file: llmk.toml

When the `llmk` command is executed without any argument, the special file `llmk.toml` is loaded automatically (Section 2.1). This filename is fixed and cannot be customized at this point. The entire content of the file must be a valid TOML; you can include supplemental information in the form of TOML comment that starts with the `#` character. The file must be encoded in UTF-8 because it is required by the TOML specification [8]. The `source` key is required in this file.

**TOML field**

The other way to pass the configuration in TOML format to `llmk` is using TOML fields — special comment areas in \LaTeX source files that are given by comment lines containing only three or more consecutive `+` characters. The following is a simple example.

```latex
% +++
% # This is a sample TOML field!
% latex = "xelatex"
% +++
\documentclass{article}
```

The formal syntax of opening and closing for TOML fields is:

```
⟨optional spaces⟩%⟨optional spaces⟩+++⟨optional pluses⟩⟨optional spaces⟩
```

where the definitions of ⟨optional spaces⟩ and ⟨optional pluses⟩ are given as follows (hereafter, whitespace \texttt{␣} denotes tab 0x09 or space 0x20).

```
⟨optional spaces⟩ → ⟨empty⟩ | \texttt{␣}⟨optional spaces⟩
⟨optional pluses⟩ → ⟨empty⟩ | +⟨optional pluses⟩
```

The line of opening and closing for TOML fields must include only the characters specified in the above. In a TOML filed, you can write TOML code for `llmk` configuration in the form of \LaTeX comment lines.

```
⟨optional spaces⟩%⟨optional spaces⟩⟨TOML line⟩⟨optional spaces⟩
```

If one or more arguments are given for the `llmk` command, it first looks for a TOML field from the beginning of each file. Only the topmost field in a file is the valid TOML field, i.e., you cannot have multiple TOML fields in a file. TOML fields have the highest priority for `llmk` configuration: if a TOML field is found in a file, other supported magic comments described in Section 4 are ignored.

Though the author recommend you to always encode your \LaTeX source file in UTF-8, you can use other encodings. In any case, the TOML lines in the fields must be consist of valid UTF-8 encoded strings. Therefore, it is recommended to use only the characters in the range of ASCII code in your TOML field if you chose the other encodings for some reasons.
3.3 Available top-level keys

Now we are going to look over all available TOML keys for llmk configuration. This section includes the full list of available top-level keys, that are effective for an entire workflow. The detailed specification for each program in your workflow can be given by the keys in the programs table, which will be described in the next section. Only the keys shown in these two sections are effective for llmk; if other key names are specified, they will be ignored. Each key requests a value of specified type; if a value of which is not the expected type, it will result in a type error.

In the string values for some specific keys, a few format specifiers are available. These specifiers will be replaced to appropriate strings before executing actions by llmk:

- \%S is replaced by the filename given to the llmk as a command-line argument or as an element of the source array.
- \%T is replaced by the target for each program.
- \%B is replaced by the basename of \%S.

Some keys have default values; the default configuration of llmk should work well for typical and simple \LaTeX documents. Only the keys you explicitly specify in the TOML format override the default configuration. If you do not write a key in your configuration, the default value will be used. In other words, you only need to write the differences from the default configuration.

bibtex [type: string] (default: "bibtex")

The command to use for the bibtex program. Reference processing tools that are compatible with \LaTeXX can be specified for this key, e.g., "biber". Internally, this key is an alias for the command key in the bibtex entry. If the command key is specified in the programs table, this alias is ineffective.

clean_files [type: array of strings] (default: see bellow)

The files to be removed with the cleaning action (-c clean). The format specifiers are available for this key. The default value is:

```
  "%B.run.xml", "%B.snm", "%B.toc", "%B.vrb"
]
```

clobber_files [type: array of strings] (default: see bellow)

The files to be removed with the clobbering action (-c clobber). The format specifiers are available for this key. The default value is:

```
[ "%B.dvi", "%B.pdf", "%B.ps", "%B.synctex.gz" ]
```
dvipdf [type: string] (default: "dvipdfmx")

The command to use for the dvipdf program. Internally, this key is an alias for the command key in the dvipdf entry. If the command key is specified in the programs table, this alias is ineffective.

dvips [type: string] (default: "dvips")

The command to use for the dvips program. Internally, this key is an alias for the command key in the dvips entry. If the command key is specified in the programs table, this alias is ineffective.

extra_clean_files [type: array of strings] (default: [])

Extra files to be removed with the cleaning action (--clean). By using this key, you can easily add files to be removed on top of the default clean_files.

latex [type: string] (default: "lualatex")

The command to use for the latex program. Internally, this key is an alias for the command key in the latex entry. If the command key is specified in the programs table, this alias is ineffective.

llmk_version [type: string]

You can declare the llmk version to use with this key. This is especially useful in consideration of compatibility. In case breaking changes are made in the future updates and an incompatible version is declared with the key, llmk will fallback to the previous behavior or at least show you a warning message. The versioning of llmk will try to follow the semantic versioning [7] and you can specify one of the versions in the following syntax:

⟨major⟩.⟨minor⟩.⟨patch⟩

Optionally, you can omit the tailing .⟨patch⟩ part.

makeindex [type: string] (default: "makeindex")

The command to use for the makeindex program. Internally, this key is an alias for the command key in the makeindex entry. If the command key is specified in the programs table, this alias is ineffective.
max_repeat [type: integer] (default: 5)

You can specify the maximum number of execution repetitions for each command in your sequence. When processing your sequence, llmk repeats a command until aux_file becomes unchanged from the former execution if the key is specified and the corresponding auxiliary file exists. This key is to prevent the potential infinite loop of repetition.

programs [type: table] (default: see Section 3.5.1)

The table that contains the detailed configuration for each program. See Section 3.4 for the details.

ps2pdf [type: string] (default: "ps2pdf")

The command to use for the ps2pdf program. Internally, this key is an alias for the command key in the ps2pdf entry. If the command key is specified in the programs table, this alias is ineffective.

sequence [type: array of strings] (default: see Section 3.5.2)

The array that contains the names of programs that will be executed by llmk. The programs specified in this array are processed in the given order with the setups specified in the programs table. Further information about the default behavior can be found in Section 3.5.2.

source [type: string or array of strings]

You can specify a \LaTeX source file as a string value or one or more files as an array of strings. This key is only effective and required in llmk.toml. Conversely, ineffective in TOML fields. The given filenames will be treated as well as those specified as command-line arguments.

3.4 Available keys in programs table

As it is described, llmk invokes each program whose name specified in the sequence array in the given order. You can control each program execution by specifying the detailed configuration in the programs table.

The programs table is a table of tables; each of the entries (a.k.a. elements) is a set of configuration consists of the keys in the following list. The programs table must have corresponding entries for all names in the sequence array, otherwise, it will result in an error.

As well as the top-level keys, some available keys in the programs table have default values, and only the values for the keys you explicitly specified in your configuration override. In string
values for some specific keys, the format specifiers described in Section 3.3 can be used in the same way.

**args** [type: string or array of strings] (default: ["%T"])

You can specify an argument to give the command as a string value or one or more arguments as an array of strings. Each argument will be surrounded by a pair of double quotations, e.g., "arg". The format specifiers are available for this key.

**aux_file** [type: string]

The auxiliary file to monitor so that to check whether rerunning for the program is necessary or not. If this key is set and the specified file that is non-empty (see aux_empty_size) exists, llmk will repeat the program execution until no change is made to the auxiliary file. This key is originally for the auxiliary file of LATEX, but the monitoring feature should be applicable to other programs. The format specifiers are available for this key.

**aux_empty_size** [type: integer]

The empty size in bytes with this key. The auxiliary files specified with the aux_file which is smaller than the value of this key are recognized as empty and will be ignored. For instance, the auxiliary files of LATEX, which normally have .aux extensions, contain \relax ⟨line break⟩ (9 bytes); thus the default value of this key for the latex program is 9.

**command** [type: string]

The name of the command to invoke. This is the only required key for each entry in the programs table.

**generated_target** [type: boolean]

This flag is to denote whether or not the target is a file that should be generated in the sequence execution. For instance, DVI files, PS files, and the input files for makeindex, which have .idx extensions are applicable. Conversely, files that you directly edit are not generated files. When the value is true, llmk will consider only the target files newer than the time that the sequence processing starts. In case the target file is older, it will be ignored and the program will not be called.

**opts** [type: string or array of strings]

You can specify a command-line option to give the command as a string value or one or more options as an array of strings. The format specifiers are available for this key.
postprocess [type: string]

The name of a program (in the programs table) to be executed as postprocess. The specified program will be called only when the current main program is executed.

target [type: string] (default: "%S")

The target filename of the program. The existence of the target file is checked by lmk right before trying to execute each program, and the programs are invoked only when it exists. This filename is also used for replacing the value for the format specifiers. When this key is not specified, the input filename that is given as a command-line argument, or the filename in the value for the source key is used as a target.

3.5 Default programs and sequence

In this section, the default values of the two important data structures for advanced control for lmk, i.e., the default configuration for the programs table and the sequence array. As it is already described in Section 3.3, programs is the table to contain detailed configuration for each program to execute by lmk, and sequence is an array to contain the names of entries in the programs table in the order of execution for a workflow. The default values of these are designed to work well for typical \LaTeX documents. Therefore, you normally do not need to change the values, but you can override any of the values by writing new values in your TOML configuration.

3.5.1 The default programs

The followings are the default values in the programs table for each entry, i.e., program, expressed in the TOML format. The default programs table contains useful settings for popular tools in the ecosystem of \LaTeX\footnote{Requests for new settings for other programs will be considered.}. Only some of them are used in the default sequence (see Section 3.5.2), but other entries can be easily used just by overriding the sequence array.

Program bibtex The entry for the Bi\LaTeX program and friends, e.g., Biber. The \texttt{latex} program is set as postprocess so that to make sure running \LaTeX command after this execution.

```
[programs.bibtex]
command = "bibtex"

[programs.bibtex]
target = "%B.bib"

[programs.bibtex]
args = ["%B"]

[programs.bibtex]
postprocess = "latex"
```
Program dvipdf  The entry for the dvipdf program and friends.

1  [[programs.dvipdf]
2  command = "dvipdfmx"
3  target = "%B.dvi"
4  generated_target = true

Program dvips  The entry for the dvips program and friends.

1  [[programs.dvips]
2  command = "dvips"
3  target = "%B.dvi"
4  generated_target = true

Program latex  The entry for the main \LaTeX programs. The default value for the command is "lualatex"; since \llmk runs on texlua, the installation of Lua\TeX is guaranteed. That is why the command is chosen for the default.

1  [[programs.latex]
2  command = "lualatex"
3  opts = ["-interaction=nonstopmode", "-file-line-error", "-synctex=1"]
4  aux_file = "%B.aux"
5  aux_empty_size = 9

Program makeindex  The entry for the Makeindex program and friends. The latex program is set as postprocess so that to make sure rerunning \LaTeX command after this execution.

1  [[programs.makeindex]
2  command = "makeindex"
3  target = "%B.idx"
4  generated_target = true
5  postprocess = "latex"

Program ps2pdf  The entry for the ps2pdf program and friends.

1  [[programs.ps2pdf]
2  command = "ps2pdf"
3  target = "%B.pdf"
4  generated_target = true
3.5.2 The default sequence

The following is the default value for the sequence array:

\[
["latex", "bibtex", "makeindex", "dvipdf"]
\]

With these default settings in the programs table the sequence array, the default behavior of \texttt{llmk} can be summarized as follows.

1. It runs the \LaTeX command, by default \texttt{luaLaTeX}, against your \LaTeX source file. This execution may be repeated when the auxiliary file exists until no change made for the file to resolve all cross-references and so on.
2. If the corresponding \BIBTeX database file, whose name matches to \texttt{%B.bib}, exists, the \BIBTeX program is executed. When the execution occurs, the \texttt{latex} program is executed again right after because the program is set as \texttt{postprocess}.
3. Identically, if the corresponding input file for Makeindex exists, the program is executed. When the execution occurs, the \texttt{latex} program is executed again right after because the program is set as \texttt{postprocess}.
4. In case the corresponding DVI file is generated in the previous steps, though this will not happen with the default value of \texttt{latex}, i.e., \texttt{lualatex}, the \texttt{dvipdf} program, by default \texttt{dvipdfmx}, is executed to produce the final PDF file.

The entries that are not used within the default sequence are to make it easier to use other tools from the above. For instance, if you want to use \texttt{dvips + ps2pdf} combination instead of \texttt{dvipdf}, you can just modify the value of sequence a bit:

\[
\texttt{sequence} = ["latex", "bibtex", "makeindex", "dvips", "ps2pdf"]
\]

4 Other supported formats

In addition to the TOML format, \texttt{llmk} also supports a few other magic comment formats that are supported in some existing tools. These features are for user convenience, but note that the aim of \texttt{llmk} is not to behave perfectly compatible with other tools.

For \texttt{llmk}, in your \LaTeX file, configuration in TOML format has the highest priority: when a TOML field is found, all the other magic comments will be ignored. The precedence of magic comment formats is as follows. Remember that all of these will be ignored if \texttt{llmk} uses the configuration in a special file \texttt{llmk.toml}.

1. TOML field (Section 3)
2. \LaTeXShop directives (Section 4.1)
3. Shebang-like directive (Section 4.2)
4.1 \TeXShop directives

\TeXShop [6], a \TeX-oriented IDE, understands special directives, which typically start with \%_!TEX, to specify a workflow for processing \LaTeX document, e.g., which \TeX engine to use. Similar directives are also supported in a few other tools such as \TeXworks [5] and \TeXstudio [9].

Among several variation of the directives, \llmk supports two of them. One is the directive to specify a variant of \TeX engine. The formal syntax of the directive is:

\begin{verbatim}
\langle TS prefix \rangle \TeX \langle optional spaces \rangle program \langle equals \rangle \langle command \rangle \langle optional spaces \rangle
\langle TS prefix \rangle \TeX \langle optional spaces \rangle TS-program \langle equals \rangle \langle command \rangle \langle optional spaces \rangle
\end{verbatim}

where the definitions of \langle TS prefix \rangle and \langle equals \rangle are given as follows.

\begin{verbatim}
\langle TS prefix \rangle \rightarrow \langle optional spaces \rangle % \langle optional spaces \rangle ! \langle optional spaces \rangle
\langle equals \rangle \rightarrow \langle optional spaces \rangle = \langle optional spaces \rangle
\end{verbatim}

The \langle command \rangle part will be passed to the latex key of \llmk. The other supported \TeXShop directive is that for the \BibTeX program. The syntax is very similar to the first one:

\begin{verbatim}
\langle TS prefix \rangle \BIB \langle optional spaces \rangle program \langle equals \rangle \langle command \rangle \langle optional spaces \rangle
\langle TS prefix \rangle \BIB \langle optional spaces \rangle TS-program \langle equals \rangle \langle command \rangle \langle optional spaces \rangle
\end{verbatim}

The \langle command \rangle part will be passed to the \bibtex key of \llmk. For both of the two directives, only the topmost ones are effective; the others will be ignored. For example, the following two configuration are equivalent:

\begin{verbatim}
1 % !TEX TS-program = xelatex
2 % !BIB TS-program = biber
3 \documentclass{article}
4 % +++
5 % latex = "xelatex"
6 % bibtex = "biber"
7 % +++
8 \documentclass{article}
\end{verbatim}

4.2 Shebang-like directive

A few existing tools, notably the YaTeX mode for Emacs [4], support the so-called shebang-like directive. This is also supported by \llmk. The syntax is:

\begin{verbatim}
\langle optional spaces \rangle %\# ! \langle optional spaces \rangle \langle command \rangle \langle optional spaces \rangle
\end{verbatim}

The \langle command \rangle part will be passed to the latex key of \llmk. This directive is only effective strictly in the first line in your \LaTeX source file.
For example, the following two configurations are equivalent:

```latex
\documentclass{article}
\begin{document}
% latex = "pdflatex"
\end{document}
```

5 Acknowledgements

This project has been supported by the \TeX\ Development Fund created by the \TeX\ Users Group (No. 29). I would like to thank all contributors and the people who gave me advice and suggestions for new features for the llmk project.

6 License

This software is released under the MIT license:

The MIT License (MIT)

Copyright 2018-2021 Takuto ASAKURA (wtsnjp)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Third-party software

toml.lua Copyright 2017 Jonathan Stoler. Released under the MIT license:

https://github.com/jonstoler/lua-toml/blob/master/LICENSE
References


