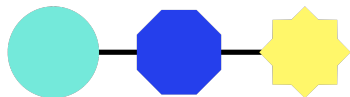


# The design concept for Ilmk—Light LaTeX Make



**Ilmk**

The Light  $\text{\LaTeX}$  Make

Takuto ASAKURA (wtsnjp)

TUG 2020

## Introduction: A demonstration

```
$ ls  
duck.tex meeting.bib meeting.tex snowman.tex
```

- ▶ Which is the main source  $\text{T}_\text{E}X$  file?
- ▶ Which  $\text{T}_\text{E}X$  engine to use?  $\text{pdfT}_\text{E}X$ ?  $\text{X}_\text{E}T_\text{E}X$ ?  $\text{LuaT}_\text{E}X$ ?
- ▶ What bib program to use?  $\text{BibT}_\text{E}X$ ? Biber?

We use  $\text{pdfT}_\text{E}X$  for this document. The main source is `meeting.tex` and the others are `\input` by it. We use  $\text{BibT}_\text{E}X$  for processing its bibliography.

1. Run `pdflatex meeting.tex`
2. Run `bibtex meeting`
3. Run `pdflatex meeting.tex` (for a few times, if necessary)

## T<sub>E</sub>X, L<sup>A</sup>T<sub>E</sub>X, and friends: The rich ecosystem

**T<sub>E</sub>X Engines** pdfT<sub>E</sub>X, X<sub>Ǝ</sub>T<sub>E</sub>X, LuaT<sub>E</sub>X, (u)pT<sub>E</sub>X, ...

**Bibliography** B<sub>I</sub>B<sub>T</sub>E<sub>X</sub>, Biber, ...

**Indexing** Makeindex, xcindy, mendex, ...

**DVIware** dvipdfm(x), dvips, ...

### Workflows by Projects

The *best* workflow is differ from one project to another

#### Example

- ▶ pdfT<sub>E</sub>X + B<sub>I</sub>B<sub>T</sub>E<sub>X</sub> + Makeindex: one of the most popular
- ▶ X<sub>Ǝ</sub>T<sub>E</sub>X and LuaT<sub>E</sub>X instead of pdfT<sub>E</sub>X: reasonable to use system fonts
- ▶ (u)pT<sub>E</sub>X + dvipdfmx: de facto standard for Japanese documents

## Telling workflows

A person may use different tools depending on purpose

### Example (In my case)

- ▶ pdfL<sup>A</sup>T<sub>E</sub>X for English documents as the first choice
- ▶ X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X if I want to use system fonts
- ▶ upT<sub>E</sub>X + dvipdfmx for Japanese documents
- ▶ LuaT<sub>E</sub>X when I want to use its Lua features

### Telling which workflow to use in a project to

- ▶ human E.g., co-authors, editors, ...
- ▶ systems E.g., text editors, IDEs, build tools, ...

It would be ideal if we can do this in an **easy** and **uniformed** way for both human and systems.

## Using generic build tools to tell the workflows?

There are numerous existing tools such as GNU Make.

- ▶ They are really useful (I have no doubt!)
- ▶ They can handle any complex workflow

### Example (simple case)

Just telling “We are using pdf $\text{\LaTeX}$  for this document” is enough.

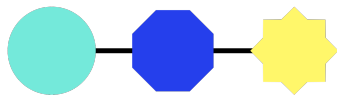
- ▶ Do we always provide Makefile for *all* documents?
- ▶ Workflows for *typical* small documents are not that complex

### Hypothesis

In many cases, just writing `%#!pdflatex` on top of the  $\text{\TeX}$  file (or similar) would be just fine.

→ I'd like to provide [an interpreter](#) for it!

## Ilmk: The motivation



# Ilmk

The Light  $\text{\LaTeX}$  Make

### Mission

Encourage people to always explicitly show the workflow for *each document* by **providing convenient ways to do it!**

- ▶ It should provide **easy ways** to specify the workflows
- ▶ It should work in **various environments**
- ▶ It should behave **exactly the same** in any environment

# The design concept

## 1. Convenience

- ▶ it supports **independent config files** (`lmk.toml`)
- ▶ *and also magic comments* in T<sub>E</sub>X file  
E.g., TOML fields, shebang-like magic comments, etc.
- ▶ a default config in *do-our-best* style, which should work fine in *typical* and *simple* L<sup>A</sup>T<sub>E</sub>X documents

## 2. Portability

- ▶ lmk is **cross-platform**; **it works solely with texlua**
- ▶ no user config (such as `~/.lmkrc`)  
∴ lmk config is a means of communicating workflows

**Note** lmk is NOT trying to replace existing tools  
→ It focuses on simple cases that people neglect using them

## Basic usage (1) lmk.toml and TOML field

### Where to write workflow

- ▶ `lmk.toml` is loaded if `lmk` is executed without arguments
- ▶ TOML field in `*.tex` files specified as arguments

### Example (TOML field)

```
1 % +++  
2 % latex = "xelatex"  
3 % +++  
4 \documentclass{article}
```

### TOML: a config format

- ▶ A small language designed for config file cf. [INI](#), [JSON](#), [YAML](#)
- ▶ It is used by several projects E.g., [Hugo](#) and [Cargo](#)
- ▶ Full spec: see <https://toml.io>



# The basics of TOML

TOML is basically **line-oriented key=value list**, kind of INI extension:

- ▶ **Comments** begin with # and continues to EOL
- ▶ **Indentation** is allowed; Defining a key multiple times is invalid
- ▶ **Basic data-types** (types in **red** are not yet supported in lmk)
  - ▶ **Strings** (basic and literal / single- and **multi-line**)
  - ▶ **Integer**, **Floats**, **Date-Time**
  - ▶ **Boolean**

## Example

```
1 # Strings
2 key = "value" # basic string (escape sequences are allowed)
3 my_favorite_primitive = '\expandafter' # literal string
4
5 # Integer
6 answer = 42
7
8 # Boolean
9 online_conference = true
```

## Data structures in TOML

- ▶ **Array**: separated by commas; values of the same data-type
- ▶ **Table**: a.k.a. hash table or dictionary; no guarantee for order
- ▶ **Inline table** and **array of tables** are not yet supported in lImk

### Example

```
1 # Array
2 tug = [ "Bachotek", "Rio de Janeiro", "Palo Alto", "Online" ]
3
4 # Table
5 [snowman] # until the next table or EOF are the key/values of this table
6 hat = "green"
7 snow = true
8
9 # Nested table
10 [duck.queen]
11 color = "pink"
12 # equivalent in JSON: { "duck": { "queen": { "color": "pink" } } }
```

## Basic usage (2) Simple keys

- ▶ `latex` (*string*):  $\text{\LaTeX}$  command to use (default: "lu<sup>l</sup>atex")  
→ `dvipdf`, `bibtex`, etc. are similar
- ▶ `max_repeat` (*integer*): to solve cross-reference (default: 5)
- ▶ `source` (*string or array of strings*): source  $\text{\TeX}$  files  
→ only valid and required in `llmk.toml`

### Example

```
1 # source TeX files
2 source = [ "test1.tex", "test2.tex" ]
3
4 # software to use
5 latex = "xelatex"
6 bibtex = "biber"
7
8 # misc
9 max_repeat = 7
```

## Flexible control (1) Array sequence and Table programs

- ▶ *sequence (string array)*: program names in the order of execution
- ▶ *programs (table of tables)*: detailed config for each program

### Example sequence

"`latex`" → "`bibtex`"

### Example programs

#### latex

```
command: "xelatex"
auxiliary: "foo.aux"
opts: "-recorder"
```

#### bibtex

```
command: "bibtex"
target: "foo.bib"
postprocess: "latex"
```

## Flexible control (2) Table programs

### Available keys in program (summary)

- ▶ `command` (*string*): command to execute
- ▶ `target` (*string*): the command is run, **only if the target file exists**
- ▶ `opts` (*string* or *array of strings*): command-line options
- ▶ `args` (*string* or *array of strings*): command-line arguments
- ▶ `auxiliary` (*string*): the file to monitor (for cross-referencing)
- ▶ `postprocess` (*string*): the program will be run after, **only if it runs**

### Special specifiers

The following specifiers are available in values for some keys:

- ▶ `%S`: source file which is processed
- ▶ `%T`: target file for each program
- ▶ `%B`: basename of `%S`

## Default config (1)

### Design concept

- ▶ Writing all config from scratch every time is meaningless  
→ Providing *do-our-best* style default config, which should work for **typical simple L<sup>A</sup>T<sub>E</sub>X** documents
- ▶ Users only need to write *diff* from the default
- ▶ No user config (such as `~/llmkrc`)  
→ A T<sub>E</sub>X file should be processed exactly as the same anywhere

### Default sequence

"`latex`" → "`bibtex`" → "`makeindex`" → "`dvipdf`"

## Default config (2)

### Default programs (summary)

#### latex

command: "lualatex"  
 auxiliary: "%B.aux"

#### bibtex

command: "bibtex"  
 target: "%B.bib"  
 postprocess: "latex"

#### dvipdf

command: "dvipdfmx"  
 target: "%B.dvi"

#### makeindex

command: "makeindex"  
 target: "%B.idx"  
 postprocess: "latex"

- ▶ There are default config also for dvips, ps2pdf, etc. cf. README
- ▶ The default programs table will be extended on demand

## Sample use cases

### Case 1: I want to use dvips instead of dvi2pdf

There is already config for dvips and ps2pdf in default programs  
 → just modifying sequence is enough:

```

1 # pLaTeX produces DVI (not PDF)
2 latex = "platex" # this is shorthand for "command" in [programs.latex]
3
4 # using dvips + ps2pdf instead of dvi2pdf
5 sequence = [ "latex", "dvips", "ps2pdf" ]

```

### Case 2: I want to use my own awesome program

You can use arbitrary command:

```

1 sequence = [ "awesome" ]
2
3 [programs.awesome]
4 command = "awesome"

```



## Cleaning actions

Cleaning actions are available (thanks @hidaruma):

- ▶ `--clean` (-c) removes temporary files such as `*.aux`, `*.log`, and `*.toc`
- ▶ `--clobber` (-C) removes all generated files including `*.pdf` and `*.synctex.gz`

Usually, the default config should work, but you can customize:

```
1 | # specifier %B represents the basename of source TeX file
2 | clean_files = ["%B.log", "%B.aux", "%B.duck"]
```

### Example

Executing the `--clean` action by using config in `foo.tex`:

```
$ llmk --clean foo.tex
```

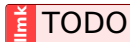
## Supports for other formats

Shebang-like format used by Emacs/YaTeX

```
| %#!uplatex
```

```
| latex = "uplatex"
```

Magic comment used by TeXShop, TeXworks, and TeXstudio



```
| %!TEX program = pdflatex  
| %!BIB program = biber
```

```
| latex = "pdflatex"  
| bibtex = "bibtex"
```

Why you want to write config in \*.tex files?

- ▶ sometimes it is annoying to open another file (especially for small and casual use cases)
- ▶ compatible with T<sub>E</sub>X-specific IDEs and Web-based editors

## Frequent Q&A (1)

How does lmk differ from latexmk?

Our goals are similar but **not exactly the same**

### Mission for lmk

Encourage people to always explicitly show the workflow for *each document* by **providing convenient ways to do it!**

Thus, there are some differences in design concept:

- ▶ lmk allows users to write config in \*.tex files
- ▶ No user config
- ▶ Less implicit decision for workflows

Does it give clear error messages?

I tried my best: lmk has **typechecker** and **own TOML perser** for this.

## Frequent Q&A (2)

### What make lmk $\LaTeX$ -specific?

Using it for general-purpose is possible in theory, but **meaningless**:

- ▶ Magic comment features are  $\TeX$ -specific **E.g.**, % is fixed
- ▶ The default config is for typical  $\LaTeX$  documents
- ▶  $\LaTeX$ -oriented rerun feature until all cross-references are solved

### How about security concerns?

Same as other build tools. But lmk requires explicit config.

#### **Warning**

Do not process unreliable  $\TeX$  documents with lmk, especially those you get from Internet, without checking their contents!

## Current status and future plan

### Current version: pre-0.1.0

- ▶ **No public release, even v0.1.0, yet**
- ▶ You have to install it manually; Don't worry, it's a single file  
→ Please visit <https://github.com/wtsnjp/lmk>
- ▶ In consideration of backward-compatibility: `lmk_version`  
→ If the compatibility is broken in the future, you'll get warning

```
| lmk_version = "0.1.0"
```

### Future plan

- ▶ It needs reference manual; at this moment we have only **README**  
→ I will make it ASAP and upload to CTAN
- ▶ Supporting other magic comment formats

## Conclusion

### Mission for llmk

Encourage people to always explicitly show the workflow for *each document* by **providing convenient ways to do it!**

For the above mission, llmk is designed to:

- ▶ provide several **easy ways** to describe the workflows
- ▶ work in **various environments**; it only requires LuaTeX in principle
- ▶ behave **exactly the same** in any environment

No more documents that no one but authors knows how to process!

Please visit <https://github.com/wtsnjp/llmk>

*Thank you! Questions and comments?*