

# LEBHART, 以多彩的方式排版你的文章

许锦文

ProjLib@outlook.com

2021 年 8 月, 北京

## 摘要

lebhart 是 colorist 文档类系列的成员之一, 其名称取自于德文的 lebhaft (活泼), 并取了 artikel (文章) 的前三个字母组合而成。整个 colorist 系列包含用于排版文章的 colorart、lebhart 以及用于排版书的 colorbook、beaulivre。我设计这一系列的初衷是为了撰写草稿与笔记, 使之多彩而不缭乱。

lebhart 支持英语、法语、德语、意大利语、葡萄牙语、巴西葡萄牙语、西班牙语、简体中文、繁体中文、日文、俄文, 并且同一篇文档中这些语言可以很好地协调。由于采用了自定义字体, 需要用  $\text{X}\text{\LaTeX}$  或  $\text{Lua}\text{\LaTeX}$  引擎进行编译。

这篇说明文档即是用 lebhart 排版的 (使用了参数 allowbf), 你可以把它看作一份简短的说明与演示。

## 目录

1	使用示例	2
1.1	如何加载	2
1.2	一篇完整的文档示例	2
2	关于默认字体	4
3	选项	5
4	具体说明	6
4.1	语言设置	6
4.2	定理类环境及其引用	6
4.3	定义新的定理型环境	7
4.4	未完成标记	8
4.5	文章标题、摘要与关键词	9
5	目前存在的问题	10

## 开始之前

为了使用这篇文档中提到的文档类, 你需要:

- 安装一个尽可能新版本的 TeX Live 或 MikTeX 套装, 并确保 colorist 和 projlib 被正确安装在你的  $\text{T}\text{\LaTeX}$  封装中。
- 下载并安装所需的字体, 参考“关于默认字体”这一节。
- 熟悉  $\text{L}\text{\LaTeX}$  的基本使用方式, 并且知道如何用  $\text{pdf}\text{\LaTeX}$ 、 $\text{X}\text{\LaTeX}$  或  $\text{Lua}\text{\LaTeX}$  编译你的文档。

## 1 使用示例

### 1.1 如何加载

只需要在第一行写：

---

```
\documentclass{lebhart}
```

---

即可使用 lebhart 文档类。请注意，要使用 Xe<sub>L</sub>AT<sub>E</sub>X 或 Lua<sub>L</sub>AT<sub>E</sub>X 引擎才能编译。

### 1.2 一篇完整的文档示例

首先来看一段完整的示例。

---

```
1 \documentclass{lebhart}
2 \usepackage{ProjLib}
3
4 \UseLanguage{French}
5
6 \begin{document}
7
8 \title{<title>}
9 \author{<author>}
10 \date{\PLdate{2022-04-01}}
11
12 \maketitle
13
14 \begin{abstract}
15     Ceci est un résumé. \dnf<Plus de contenu est nécessaire.>
16 \end{abstract}
17 \begin{keyword}
18     AAA, BBB, CCC, DDD, EEE
19 \end{keyword}
20
21 \section{Un théorème}
22
23 \begin{theorem}\label{thm:abc}
24     Ceci est un théorème.
25 \end{theorem}
26 Référence du théorème: \cref{thm:abc}
27
28 \end{document}
```

---

如果你觉得这个例子有些复杂，不要担心。现在我们来一点点地观察这个例子。

#### 1.2.1 初始化部分

---

```
\documentclass{lebhart}
\usepackage{ProjLib}
```

---

初始化部分很简单：第一行加载文档类 lebhart，第二行加载 ProjLib 工具箱，以便使用一些附加功能。

### 1.2.2 设定语言

---

```
\UseLanguage{French}
```

---

这一行表明文档中将使用法语（如果你的文章中只出现英语，那么可以不需要设定语言）。你也可以在文章中间用同样的方式再次切换语言。支持的语言包括简体中文、繁体中文、日文、英语、法语、德语、西班牙语、葡萄牙语、巴西葡萄牙语、俄语。

对于这一命令的详细说明以及更多相关命令，可以参考后面关于多语言支持的小节。

### 1.2.3 标题，作者信息，摘要与关键词

---

```
\title{<title>}
\author{<author>}
\date{\PLdate{2022-04-01}}
\maketitle
```

```
\begin{abstract}
  <abstract>
\end{abstract}
\begin{keyword}
  <keywords>
\end{keyword}
```

---

开头部分是标题和作者信息块。这个例子中给出的是最基本的形式，事实上你还可以这样写：

---

```
\author{<author 1>}
\address{<address 1>}
\email{<email 1>}
\author{<author 2>}
\address{<address 2>}
\email{<email 2>}
...
```

---

另外，你还可以采用  $\LaTeX$  文档类的写法：

---

```
\title{<title>}
\author{<author 1>}
\address{<address 1>}
\email{<email 1>}
\author{<author 2>}
\address{<address 2>}
\email{<email 2>}
\date{\PLdate{2022-04-01}}
\subjclass{*****}
\keywords{<keywords>}

\begin{abstract}
  <abstract>
\end{abstract}

\maketitle
```

---

### 1.2.4 未完成标记

---

```
\dnf<\langle some hint \rangle>
```

---

当你有一些地方尚未完成的时候，可以用这条指令标记出来，它在草稿阶段格外有用。

### 1.2.5 定理类环境

---

```
\begin{theorem}\label{thm:abc}
  Ceci est un théorème.
\end{theorem}
Référence du théorème: \cref{thm:abc}
```

---

常见的定理类环境可以直接使用。在引用的时候，建议采用智能引用 `\cref{\langle label \rangle}`——这样就不必每次都写上相应环境的名称了。

#### 提示

如果你之后想要切换到标准文档类，只需要把前两行换为：

---

```
\documentclass{article}
\usepackage[a4paper,margin=1in]{geometry}
\usepackage[hidelinks]{hyperref}
\usepackage[palatino,amsfashion]{ProjLib}
```

---

或者使用  $\mathcal{AMS}$  文档类：

---

```
\documentclass{amsart}
\usepackage[a4paper,margin=1in]{geometry}
\usepackage[hidelinks]{hyperref}
\usepackage[palatino]{ProjLib}
```

---

#### 提示

如果你喜欢这个文档类，但又希望使用一种更加中规中矩的样式，那么不妨使用 `classical` 选项，就像这样：

---

```
\documentclass[classical]{lebhart}
```

---

## 2 关于默认字体

本文档类中默认使用 Palatino Linotype 作为英文主字体，思源宋体、思源黑体、思源等宽作为中文主字体、无衬线字体以及等宽字体，并部分使用了 Neo Euler 作为数学字体。这些字体需要用户自行下载安装。其中，思源字体系列可在 <https://github.com/adobe-fonts> 下载（推荐下载 Super-OTC 版本，这样下载的体积较小）。Neo Euler 可以在 <https://github.com/khaledhosny/euler-otf> 下载。在没有安装相应的字体时，将采用 TeX Live 中自带的字体来代替，效果可能会有所折扣。

另外，还使用了 Source Code Pro 作为英文无衬线字体、New Computer Modern Mono 作为英文等宽字体，以及 Asana Math、Tex Gyre Pagella Math、Latin Modern Math 数学字体中的部分符号。这些字体在 TeX Live 或 MikTeX 中已经提供，无需自行下载安装。

### 3 选项

lebhart 文档类有下面几个选项：

- 语言选项 `EN / english / English`、`FR / french / French`，等等
  - 具体选项名称可参见下一节的 `<language name>`。第一个指定的语言将作为默认语言。
  - 语言选项不是必需的，其主要用途是提高编译速度。不添加语言选项时效果是一样的，只是会更慢一些。
- `draft` 或 `fast`
  - 你可以使用选项 `fast` 来启用快速但略微粗糙的样式，主要区别是：
    - \* 使用较为简单的数学字体设置；
    - \* 不启用超链接；
    - \* 启用 ProjLib 工具箱的快速模式。

#### 提示

在文章的撰写阶段，建议使用 `fast` 选项以加快编译速度，改善写作时的流畅度。使用 `fast` 模式时会有“DRAFT”字样的水印，以提示目前处于草稿阶段。

- `a4paper` 或 `b5paper`
  - 可选的纸张大小。默认的纸张大小为 8.5in × 11in。
- `palatino`、`times`、`garamond`、`noto`、`biolinum` | `useosf`
  - 字体选项。顾名思义，会加载相应名称的字体。
  - `useosf` 选项用来启用“旧式”数字。
- `allowbf`
  - 允许加粗。启用这一选项时，题目、各级标题、定理类环境名称会被加粗。
- `runin`
  - `\subsubsection` 采用“runin”风格。
- `puretext` 或 `nothms`
  - 纯文本模式，不加载定理类环境。
- `nothmnum`、`thmnum` 或 `thmnum=<counter>`
  - 定理类环境均不编号 / 按照 1、2、3 顺序编号 / 在 `<counter>` 内编号。其中 `<counter>` 应该是自带的计数器（如 `subsection`）或在导言部分自定义的计数器。在没有使用任何选项的情况下将按照 `chapter`（书）或 `section`（文章）编号。
- `regionalref`、`originalref`
  - 在智能引用时，定理类环境的名称是否随当前语言而变化。默认为 `regionalref`，即引用时采用当前语言对应的名称；例如，在中文语境中引用定理类环境时，无论原环境处在什么语境中，都将使用名称“定理、定义……”。若启用 `originalref`，则引用时会始终采用定理类环境所处语境下的名称；例如，在英文语境中书写的定理，即使稍后在中文语境下引用时，仍将显示为 `Theorem`。
  - 在 `fast` 模式下，`originalref` 将不起作用。

## 4 具体说明

### 4.1 语言设置

lebhart 提供了多语言支持, 包括英语、法语、德语、意大利语、葡萄牙语、巴西葡萄牙语、西班牙语、简体中文、繁体中文、日文、俄文。可以通过下列命令来选定语言:

- `\UseLanguage{<language name>}`, 用于指定语言, 在其后将使用对应的语言设定。
  - 既可以用于导言部分, 也可以用于正文部分。在不指定语言时, 默认选定 “English”。
- `\UseOtherLanguage{<language name>}{<content>}`, 用指定的语言的设定排版 `<content>`。
  - 相比 `\UseLanguage`, 它不会对行距进行修改, 因此中西文字混排时能保持行距稳定。

`<language name>` 有下列选择 (不区分大小写, 如 `French` 或 `french` 均可):

- 简体中文: `CN`、`Chinese`、`SChinese` 或 `SimplifiedChinese`
- 繁体中文: `TC`、`TChinese` 或 `TraditionalChinese`
- 英文: `EN` 或 `English`
- 法文: `FR` 或 `French`
- 德文: `DE`、`German` 或 `ngerman`
- 意大利语: `IT` 或 `Italian`
- 葡萄牙语: `PT` 或 `Portuguese`
- 巴西葡萄牙语: `BR` 或 `Brazilian`
- 西班牙语: `ES` 或 `Spanish`
- 日文: `JP` 或 `Japanese`
- 俄文: `RU` 或 `Russian`

另外, 还可以通过下面的方式来填加相应语言的设置:

- `\AddLanguageSetting{<settings>}`
  - 向所有支持的语言增加设置 `<settings>`。
- `\AddLanguageSetting(<language name>){<settings>}`
  - 向指定的语言 `<language name>` 增加设置 `<settings>`。

例如, `\AddLanguageSetting(German){\color{orange}}` 可以让所有德语以橙色显示 (当然, 还需要再加上 `\AddLanguageSetting{\color{black}}` 来修正其他语言的颜色)。

### 4.2 定理类环境及其引用

定义、定理等环境已经被预定义, 可以直接使用。

具体来说, 预设的定理类环境包括: `assumption`、`axiom`、`conjecture`、`convention`、`corollary`、`definition`、`definition-proposition`、`definition-theorem`、`example`、`exercise`、`fact`、`hypothesis`、`lemma`、`notation`、`observation`、`problem`、`property`、`proposition`、`question`、`remark`、`theorem`, 以及相应的带有星号 \* 的无编号版本。

在引用定理类环境时, 建议使用智能引用 `\cref{<label>}`。这样就不必每次都写上相应环境的名称了。

## 例子

```
\begin{definition}[奇异物品] \label{def: strange} ...
```

将会生成

**定义 4.1** (奇异物品) 这是奇异物品的定义。定理类环境的前后有一行左右的间距。在定义结束的时候会有一个符号来标记。

`\cref{def: strange}` 会显示为: **定义 4.1**。

使用 `\UseLanguage{English}` 后, 定理会显示为:

**THEOREM 4.2** (Useless) A theorem in English.

默认情况下, 引用时, 定理类环境的名称总是与当前语言相匹配, 例如, 上面的定义在现在的英文模式下将显示为英文: **DEFINITION 4.1** and **THEOREM 4.2**。如果在引用时想让定理的名称总是与原定理所在区域的语言匹配, 即总是显示原始名称, 可以在全局选项中加入 `originalref`。

下面是定理类环境的几种主要样式:

**定理 4.3** Theorem style: theorem, proposition, lemma, corollary, ...

证明 | Proof style



Remark style



**猜想 4.4** Conjecture style

**例** Example style: example, fact, ...

**问题 4.5** Problem style: problem, question, ...

为了美观, 相邻的定义环境会自动连在一起:

**定义 4.6** First definition.

**定义 4.7** Second definition.

## 4.3 定义新的定理型环境

若需要定义新的定理类环境, 首先要定义这个环境在所用语言下的名称:

- `\NameTheorem[⟨language name⟩]{⟨name of environment⟩}{⟨name string⟩}`

其中, `⟨language name⟩` 可参阅关于语言设置的小节。当不指定 `⟨language name⟩` 时, 则会将该名称设置为所有支持语言下的名称。另外, 带星号与不带星号的同名环境共用一个名称, 因此 `\NameTheorem{envname*}{...}` 与 `\NameTheorem{envname}{...}` 效果相同。

然后用下面五种方式之一定义这一环境：

- `\CreateTheorem*{<name of environment>}`
  - 定义不编号的环境 `<name of environment>`
- `\CreateTheorem{<name of environment>}`
  - 定义编号环境 `<name of environment>`，按顺序编号
- `\CreateTheorem{<name of environment>}[<numbered like>]`
  - 定义编号环境 `<name of environment>`，与 `<numbered like>` 计数器共用编号
- `\CreateTheorem{<name of environment>}<numbered within>`
  - 定义编号环境 `<name of environment>`，在 `<numbered within>` 计数器内编号
- `\CreateTheorem{<name of environment>}<existed environment>`  
`\CreateTheorem*{<name of environment>}<existed environment>`
  - 将 `<name of environment>` 与 `<existed environment>` 或 `<existed environment>*` 等同。
  - 这种方式通常在两种情况下比较有用：
    1. 希望定义更简洁的名称。例如，使用 `\CreateTheorem{thm}(theorem)`，便可以直接用名称 `thm` 来撰写定理。
    2. 希望去除某些环境的编号。例如，使用 `\CreateTheorem{remark}(remark*)`，便可以去除 `remark` 环境的编号。

#### 提示

其内部使用了 `amsthm`，因此传统的 `theoremstyle` 对其也是适用的，只需在相关定义前标明即可。

下面提供一个例子。这三行代码：

```
\NameTheorem[CN]{proofidea}{思路}
\CreateTheorem*{proofidea*}
\CreateTheorem{proofidea}<subsection>
```

可以分别定义不编号的环境 `proofidea*` 和编号的环境 `proofidea` (在 `subsection` 内编号)，它们支持在简体中文语境中使用，效果如下所示：

思路 | `proofidea*` 环境。



思路 4.3.1 | `proofidea` 环境。



## 4.4 未完成标记

你可以通过 `\dnf` 来标记尚未完成的部分。例如：

- `\dnf` 或 `\dnf<...>`。效果为：`这里尚未完成 #1` 或 `这里尚未完成 #2: ...`。  
其提示文字与当前语言相对应，例如，在法语模式下将会显示为 `Pas encore fini #3`。

类似的，还有 `\needgraph`：

- `\needgraph` 或 `\needgraph<...>`。效果为：

`这里需要一张图片 #1`



或

这里需要一张图片 #2: ...

其提示文字与当前语言相对应，例如，在法语模式下将会显示为

Il manque une image ici #3

#### 4.5 文章标题、摘要与关键词

lebhart 同时具有标准文档类与 $\mathcal{AMS}$  文档类的一些特性。

因此，文章的标题部分既可以按照标准文档类 `article` 的写法来写：

```
\title{\<title>}
\author{\<author>\thanks{\<text>}}
\date{\<date>}
\maketitle
\begin{abstract}
  \<abstract>
\end{abstract}
\begin{keyword}
  \<keywords>
\end{keyword}
```

也可以按照  $\mathcal{AMS}$  文档类的方式来写：

```
\title{\<title>}
\author{\<author>}
\thanks{\<text>}
\address{\<address>}
\email{\<email>}
\date{\<date>}
\keywords{\<keywords>}
\subjclass{\<subclass>}
\begin{abstract}
  \<abstract>
\end{abstract}
\maketitle
```

作者信息可以包含多组，输入方式为：

```
\author{\<author 1>}
\address{\<address 1>}
\email{\<email 1>}
\author{\<author 2>}
\address{\<address 2>}
\email{\<email 2>}
...
```

其中 `\address`、`\curraddr`、`\email` 的相互顺序是不重要的。

## 5 目前存在的问题

- 对于字体的设置仍然不够完善。
- 样式的设计还不够完整，包括目录、part 以及 chapter 等。
- 由于很多核心功能建立在 ProjLib 工具箱的基础上，因此 colorist (进而 colorart、lebhart 与 colorbook、beaulivre) 自然继承了其所有问题。详情可以参阅 ProjLib 用户文档的“目前存在的问题”这一小节。
- 错误处理功能不完善，在出现一些问题时没有相应的错误提示。
- 代码中仍有许多可优化之处。