

ProjLib 工具集

使用指南

许锦文

ProjLib@outlook.com

2021 年 10 月, 巴黎

摘要

ProjLib 工具集之设计目的为简化 \LaTeX 文档撰写前的准备工作。只需要加载 ProjLib, 多语言设置就已准备就绪, 定理类环境已被设置好可供直接使用, 并且引入了一系列辅助功能。

目录

开始之前	1	4 具体组件	6
1 简介	1	4.1 主要功能	6
2 使用示例	2	4.1.1 PJLlang: 多语言支持	6
2.1 如何加载	2	4.1.2 PJLthm: 带有智能引用与多语言支持的定理类环境	7
2.2 一篇完整的文档示例	2	4.2 次要功能	10
2.2.1 初始化部分	3	4.2.1 PJLauthor: 增强的作者信息块	10
2.2.2 设定语言	3	4.2.2 PJLdate: 智能日期处理	10
2.2.3 标题与作者信息	3	4.2.3 PJLdraft: 未完成标记	10
2.2.4 未完成标记	4	4.2.4 PJLlogo: ProjLib 图标	11
2.2.5 定理类环境	4	4.2.5 PJLmath: 数学符号与捷径	11
3 主宏包的选项	5	4.2.6 PJLpaper: 纸张设置	12
		5 目前存在的问题	13

开始之前

为了使用这套工具集, 你需要:

- 安装一个尽可能新版本的 TeX Live 或 MikTeX 套装, 并确保 projlib 被正确安装在你的 TeX 封装中。
- 熟悉 \LaTeX 的基本使用方式, 并且知道如何用 pdf \LaTeX 、X \LaTeX 或 Lua \LaTeX 编译你的文档。

1 简介

ProjLib 这一名称可以看成是英文 Project Library (项目库) 或法文 Projet Libre (自由项目) 的缩写 (作者更喜欢法文的全称)。其主要目的是提供多语言支持和带有智能引用的定理类环境。除此之外, 还附加了一些额外功能, 如支持作者附加信息、未完成标记、数学符号与捷径等。

ProjLib 工具箱由主宏包 ProjLib 以及一系列由 “PjL” 缩写开头的内部组件构成。你可以通过下一节的使用实例来了解它的使用方式。

2 使用示例

2.1 如何加载

加载 ProjLib 工具箱十分容易，只需要在导言部分加入这一行即可：

```
\usepackage{ProjLib}
```

注意事项

由于其内部使用了 cleveref，ProjLib 需要放在 varioref、hyperref 的后面。

2.2 一篇完整的文档示例

首先来看一段完整的示例。

```
1 \documentclass{article}
2 \usepackage[a4paper,margin=.75in]{geometry}
3 \usepackage[hidelinks]{hyperref}
4 \usepackage[palatino]{ProjLib} % Load the toolkit and use font Palatino
5
6 \UseLanguage{French} % Use French from here
7
8 \begin{document}
9
10 \title{\<title>}
11 \author{\<author>}
12 \date{\PLdate{2022-04-01}}
13
14 \maketitle
15
16 \begin{abstract}
17   \<abstract text> \dnf\<some hint>
18 \end{abstract}
19
20 \section{Un théorème}
21
22 \begin{theorem}\label{thm:abc}
23   Ceci est un théorème.
24 \end{theorem}
25
26 Référence du théorème: \cref{thm:abc} % It is recommended to use clever reference
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。

2.2.1 初始化部分

```
\documentclass{article}
\usepackage[a4paper,margin=.75in]{geometry}
\usepackage[hidelinks]{hyperref}
\usepackage[palatino]{ProjLib}
```

在标准文档类中，通常只需简要设置页面尺寸、超链接，再载入 ProjLib，即可直接开始写作。这里使用了 ProjLib 的 palatino 字体选项。关于 ProjLib 的所有可用选项，可以参阅下一节。

当然，你也可以使用 amsart 文档类，设置是相同的。

2.2.2 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¹。

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。

2.2.3 标题与作者信息

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

这一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，如果开启 $\mathcal{A}\mathcal{M}\mathcal{S}$ 风格，那么文章中还可以采用 $\mathcal{A}\mathcal{M}\mathcal{S}$ 文档类的写法（此时原始的写法也是成立的）。为此，引入 ProjLib 时应该加入 amsfashion 选项²：

```
\usepackage[amsfashion,palatino]{ProjLib}
```

而相应地，你也就可以使用这些命令：

```
\dedicatory{\<dedicatory>}
\subjclass{*****}
\keywords{\<keywords>}
```

¹不过，你需要自行引入相应语言的支持与字体。例如，对于中文，你可能需要载入 ctex 宏包并设置字体。作为补充，你可以尝试作者的 einfant 或 lebhart 文档类，其中相应的设置都已经完成了，详细资料可以通过 texdoc minimalist 或 texdoc colorist 获知。

²由于这一选项会修改 \LaTeX 的一些内部指令，有可能与一些宏包或文档类发生冲突，因此没有默认启用。

另外, 这种情况下, 也可以允许摘要出现在 `\maketitle` 的前面, 如同在 $\mathcal{A}\mathcal{M}\mathcal{S}$ 文档类中所要求的那样:

```
\begin{abstract}
  <abstract text>
\end{abstract}
\maketitle
```

2.2.4 未完成标记

```
\dnf<<some hint>>
```

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。

2.2.5 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{<label>}`——这样就不必每次都写上相应环境的名称了。

3 主宏包的选项

ProjLib 提供下列选项以供选择:

- `draft` 或 `fast`
 - 快速模式。功能会适当缩减, 但能够提高编译速度, 建议在撰写阶段使用。
- `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。
- `nothms`、`delaythms`、`nothmnum`、`thmnum` 或 `thmnum=<counter>`、`complexname`、`simplename`、`regionalref`、`originalref`
 - 来自用于设置定理类环境的组件 PJLthm 的选项, 详细信息请参阅有关这一宏包的小节。
- `author`
 - 加载用于增强作者信息块的组件 PJLauthor。关于其详细功能, 请参阅有关该宏包的小节。
- `amsfashion`
 - 允许 $\mathcal{M}\mathcal{S}$ 文档类的写法。此时 `author` 选项会被自动开启。

另外, 还有一些组件的参数可以作为文档类的全局参数传递, 例如 `EN/english/English`、`CN/chinese/Chinese` 等语言选项 (来自 PJLlang), `paperstyle` 和 `preview` 等纸张选项 (来自 PJLpaper)。详细信息可以参阅对应的小节。

4 具体组件

4.1 主要功能

4.1.1 PJLlang: 多语言支持

PJLlang 提供了多语言支持, 包括简体中文、繁体中文、英文、法文、德文、日文、俄文 (其中中文、日文、俄文需要相应的 $\text{T}_{\text{E}}\text{X}$ 引擎与字体支持)。

PJLlang 提供语言选项, 这些选项的名称有三种, 分别是缩写 (如 `EN`)、小写 (如 `english`)、首字母大写 (如 `English`)。具体的选项名称可以参见下文的 $\langle\text{language name}\rangle$ 。其中, 第一个指定的语言 $\langle\text{first language}\rangle$ 会被作为默认语言, 相当于在文档开头指定 `\UseLanguage{\langle\text{first language}\rangle}`。

提示

为了提高编译速度, 建议使用语言选项, 并将其作为全局参数传递。这样, 只会对指定语言进行设置, 节省了 $\text{T}_{\text{E}}\text{X}$ 内存, 从而能显著提高编译速度。

在文档中, 可以通过下列命令来选定语言:

- `\UseLanguage{\langle\text{language name}\rangle}`, 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{\langle\text{language name}\rangle}\langle\text{content}\rangle`, 用指定的语言的设定排版 $\langle\text{content}\rangle$ 。
 - 相比较 `\UseLanguage`, 它不会对行距进行修改, 因此中西文字混排时能够保持行距稳定。

$\langle\text{language name}\rangle$ 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`
- 英语: `EN` 或 `English`
- 法语: `FR` 或 `French`
- 德语: `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`
- 日语: `JP` 或 `Japanese`
- 俄语: `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{\langle\text{settings}\rangle}`
 - 向所有支持的语言增加设置 $\langle\text{settings}\rangle$ 。
- `\AddLanguageSetting(\langle\text{language name}\rangle)\langle\text{settings}\rangle`
 - 向指定的语言 $\langle\text{language name}\rangle$ 增加设置 $\langle\text{settings}\rangle$ 。

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)。

4.1.2 PJLthm: 带有智能引用与多语言支持的定理类环境

PJLthm 提供定理类环境的设置。它支持下列选项:

- `nothms`
 - 不设定定理类环境。如果你希望使用自己的定理样式, 可以使用这一选项。
- `delaythms`
 - 将定理类环境设定推迟到导言结尾。如果你希望定理类环境跟随自定义计数器编号, 则应考虑这一选项。
- `nothmnum`、`thmnum` 或 `thmnum=<counter>`
 - 使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 (在启用 `delaythms` 选项的情况下)。在没有使用任何选项的情况下将按照 `chapter` (书) 或 `section` (文章) 编号。
- `complexname`、`simplename`
 - 对于预设的定理名称, 使用详细版本 (默认) 或简易版本。详细版本包含单复数的差别, 并且在可能的情况下加入了定冠词; 简易版本则统一采用单个单词的名称, 不区分单复数。
- `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。

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`, 以及相应带有星号 * 的无编号版本。它们在显示时会依据当前语言而变化, 例如在中文语境下 `theorem` 会显示为 “定理”, 而在英文语境下则会显示为 “Theorem”。关于如何选定语言, 请参阅关于 PJLlang 的小节。

提示

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。

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。有两种方式:

- 简易设置: `\NameTheorem[<language name>]{<name of environment>}{<name string>}`
 - 这种方式只设置单独的显示名称, 智能引用等名称与之取为相同 (特别地, 以这种方式设置时智能引用名称将不区分单复数)。当不指定 `<language name>` 时, 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。
- 详细设置 (推荐):

```
\NameTheorem{<name of environment>}{  
  <language name 1>={  
    name=<Name>,  
    crefname={<name>}{<names>},
```

```

    Crefname={\langle Name \rangle}{\langle Names \rangle},
    autorefname=\langle name \rangle,
    theoremheading=\langle Name \rangle,
  },
  \langle language name 2 \rangle={...},
}

```

或

```

\NameTheorem[\langle language name \rangle]{\langle name of environment \rangle}{
  name=\langle Name \rangle,
  crefname={\langle name \rangle}{\langle names \rangle},
  Crefname={\langle Name \rangle}{\langle Names \rangle},
  autorefname=\langle name \rangle,
  theoremheading=\langle Name \rangle,
}

```

- 这种方式可以具体设置各个名称。当不指定 $\langle language name \rangle$ 时，将允许使用完整界面；在指定语言时则只设定相应语言。同样，带星号与不带星号的同名环境共用一个名称，因此 $\backslash\text{NameTheorem}\{\text{envname}*\}$ 与 $\backslash\text{NameTheorem}\{\text{envname}\}$ 效果相同。

提示

除此以外，你也可以在定义相应的定理类环境时为之命名，可以参见后文对 $\backslash\text{CreateTheorem}$ 的说明。

然后，用下面五种方式之一定义这一环境：

- $\backslash\text{CreateTheorem}*\{\langle name of environment \rangle\}$
 - 定义不编号的环境 $\langle name of environment \rangle$
- $\backslash\text{CreateTheorem}\{\langle name of environment \rangle\}$
 - 定义编号环境 $\langle name of environment \rangle$ ，按顺序编号
- $\backslash\text{CreateTheorem}\{\langle name of environment \rangle\}[\langle numbered like \rangle]$
 - 定义编号环境 $\langle name of environment \rangle$ ，与 $\langle numbered like \rangle$ 计数器共用编号
- $\backslash\text{CreateTheorem}\{\langle name of environment \rangle\}<\langle numbered within \rangle>$
 - 定义编号环境 $\langle name of environment \rangle$ ，在 $\langle numbered within \rangle$ 计数器内编号
- $\backslash\text{CreateTheorem}\{\langle name of environment \rangle\}(\langle existed environment \rangle)$
 $\backslash\text{CreateTheorem}*\{\langle name of environment \rangle\}(\langle existed environment \rangle)$
 - 将 $\langle name of environment \rangle$ 与 $\langle existed environment \rangle$ 或 $\langle existed environment \rangle*$ 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 $\backslash\text{CreateTheorem}\{\text{thm}\}(\text{theorem})$ ，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 $\backslash\text{CreateTheorem}\{\text{remark}\}(\text{remark}*)$ ，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。

你也可以在定义定理类环境的同时为之命名，只需要在之后再加入一组括号进行设置：

```
\CreateTheorem{<name of environment>}{  
  <language name 1>={  
    name=<Name>,  
    crefname={<name>}{<names>},  
    Crefname={<Name>}{<Names>},  
    autorefname=<name>,  
    theoremheading=<Name>,  
  },  
  <language name 2>={...},  
}
```

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 (具体样式与所在的文档类有关)：

思路 | `proofidea*` 环境。



思路 4.1.1 | `proofidea` 环境。



当然，你也可以使用更加精细的名称：

```
\NameTheorem{proofidea}{  
  CN = {  
    name = 思路,  
    crefname = {思路}{思路},  
    Crefname = {思路}{思路},  
  }  
}  
\CreateTheorem*{proofidea*}  
\CreateTheorem{proofidea}<subsection>
```

或者在定义时设置名称 (对于 `proofidea*` 与 `proofidea` 只需要设置一次即可)：

```
\CreateTheorem*{proofidea*}  
\CreateTheorem{proofidea}<subsection>{  
  CN = {  
    name = 思路,  

```

```

        crefname = {思路}{思路},
        Crefname = {思路}{思路},
    }
}

```

4.2 次要功能

4.2.1 PJLauthor: 增强的作者信息块

PJLauthor 提供了 `\address`、`\curraddr`、`\email` 命令，并且允许输入多组用户信息。标准的输入方式是这样的：

```

\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。

另外，你可以通过选项 `amsfashion` 以使用 $\mathcal{M}\mathcal{S}$ 方式写作。具体来说，效果为：

- 提供 `\dedicatory`、`\keywords` 及 `\subjclass` 命令
- `\thanks` 可以写在 `\author` 之外
- `abstract` 环境可以放在 `\maketitle` 的前面

注意

这些功能只在标准文档类中启用。在 $\mathcal{M}\mathcal{S}$ 文档类中，PJLauthor 不会起到任何效果。

4.2.2 PJLdate: 智能日期处理

PJLdate 提供了 `\PLdate<yyyy-mm-dd>` (或 `\PJLdate<yyyy-mm-dd>`) 命令，以将 `<yyyy-mm-dd>` 转换为当前语言的日期格式显示。例如，在当前的中文语境下，`\PLdate{2022-04-01}` 会被显示为“2022年4月1日”，而在英文语境下则会被显示为“April 1, 2022”。

关于如何选定语言，请参阅关于 PJLlang 的小节。

4.2.3 PJLdraft: 未完成标记

PJLdraft 提供了下列命令：

- `\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。
- `\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

关于如何选定语言，请参阅关于 PJLlang 的小节。

4.2.4 PJLlogo: ProjLib 图标

PJLlogo 提供了 `\ProjLib` 命令用于绘制 Logo，效果为：ProjLib。它与普通的文字指令效果类似，可以用于不同的字号：

<code>\tiny:</code>	ProjLib
<code>\scriptsize:</code>	ProjLib
<code>\footnotesize:</code>	ProjLib
<code>\normalsize:</code>	ProjLib
<code>\large:</code>	ProjLib
<code>\Large:</code>	ProjLib
<code>\LARGE:</code>	ProjLib
<code>\huge:</code>	ProjLib
<code>\Huge:</code>	ProjLib

4.2.5 PJLmath: 数学符号与捷径

PJLmath 提供下列捷径：

- i) `\mathfrak{.}` \rightarrow `\mf.` 或 `\frak.`。例如，`\mfA` (或 `\mf{A}`) 与 `\mathfrak{A}` 效果相同。这对大写、小写字母都有效：

a b c d e f g h i j k l m n o p q r s t u v w x y z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z 1

- ii) `\mathbb{.}` \rightarrow `\bb.`。这只针对大写字母或数字 1。

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z 1

对于常见的代数结构有这些特殊命令：`\N`, `\Z`, `\Q`, `\R`, `\C`, `\F`, `\A`。

N Z Q R C F A

- iii) `\mathcal{.}` \rightarrow `\mc.` 或 `\cal.`。这只针对大写字母。

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- iv) `\mathscr{.}` \rightarrow `\ms.` 或 `\scr.`。这只针对大写字母。

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

另外，PJLmath 还提供了一些 \LaTeX 中未自带的符号。

<code>\abs</code>	<code>\abs{a} \rightarrow a </code>	绝对值符号
<code>\norm</code>	<code>\norm{a} \rightarrow \ a\ </code>	范数符号
<code>\injection</code>	<code>\injection \rightarrow \hookrightarrow</code>	表示单射的箭头符号
<code>\surjection</code>	<code>\surjection \rightarrow \twoheadrightarrow</code>	表示满射的箭头符号
<code>\bijection</code>	<code>\bijection \rightarrow \xrightarrow{\sim}</code>	表示双射的箭头符号

这些捷径和符号是以一种安全的方式定义的，它们不会与已有的命令或你自己定义的命令相冲突。因此即使你用不到这些捷径或符号，也不用担心它们的存在会带来错误。

4.2.6 PJLpaper：纸张设置

PJLpaper 主要用于调节纸张颜色。它支持下列选项：

- `paperstyle = <paper style name>`
 - 设定纸张色彩样式。<paper style name> 可供选择的选项有：yellow、dark 与 nord。
- `yellowpaper`、`darkpaper`、`nordpaper`
 - 设定纸张色彩样式。效果与相应名称的 `paperstyle` 相同。
- `preview`
 - 预览模式，将会把 pdf 文件的白边去掉以方便阅读。

为了使用的方便，建议把这些选项作为文档类的全局参数，这样对于文档的纸张设定一目了然。

5 目前存在的问题

- PJLauthor 仍然处于初步阶段，在很多方面还远远比不上相对成熟的 authblk。
- PJLlang：针对 polyglossia 的设置仍然存在许多问题，因此现在主要功能都是通过 babel 实现的。
- PJLpaper 的 preview 功能主要是通过 geometry 宏包实现的，因此在 KOMA 文档类中效果不好。
- PJLthm 对于定理类环境的编号与样式设定目前还无法由用户更改。
- PJLthm：智能引用针对所有 PJLlang 已支持语言的本地化尚不完整，主要是中文、日文与俄文。
- 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，有些部分耗时过长，特别是 PJLthm 对定理类环境的定义。