ProjLib Toolkit

User Manual

Jinwen Xu

June 2021, Beijing

Abstract

The ProjLib toolkit is designed to simplify the preparation before writing LaTeX documents. With ProjLib loaded, you no longer need to set up the theorem-like environments nor configure the appropriate multilingual settings. Additionally, a series of auxiliary functionalities are introduced.

1 The main package

1.1 How to load it

Just add the following line to your preamble:

```
\usepackage{ProjLib}
```

Attention

Since cleveref is used internally, ProjLib needs to be placed after varioref and hyperref.

1.2 Options

ProjLib offers the following options:

- `draft` or `fast`
  - Fast mode. The functionality will be appropriately reduced to get faster compilation speed, recommended to use during the writing phase.
- `palatino`, `times`, `garamond`, `biolinum` | `useosf`
  - Font options. As the name suggest, font with corresponding name will be loaded.
  - The `useosf` option is used to enable the old-style figures.
- `nothms`, `nothmnum`, `regionalref`
  - Options from PJLthm, please refer to the section on this package for details.
- `author`
  - Load PJLauthor. For more information about its functionality, see the section on this package.
- `amssim`
  - Load PJLamssim. For more information about its functionality, see the section on this package.

In addition, there are also some options of the components that should be passed as global options of your document class, such as the language options `EN` / `english` / `English`, `FR` / `french` / `French` etc. of PJLlang, and `paperstyle`, `preview` of PJLpaper. For more information, please refer to the corresponding sections.

Corresponding to: ProjLib 2021/06/28

2 The components

2.1 PJLamssim

PJLamssim is used to simulate some features of the amsart class in a standard class, including:

- \address, \curraddr, \email and \dedicatory macro(the first three are provided by PJLauthor)
- \keywords macro
- \subjclass macro
- \thanks can be written outside \author
- The abstract environment can be placed before \maketitle

These modifications would only take place in standard classes. In the $\mathcal{AMS}$ classes, PJLamssim does not have any effect.

2.2 PJLauthor

PJLauthor offers \address, \curraddr and \email, and allows you to enter multiple groups of author information. The standard usage is like this:

```
\author{⟨author 1⟩}
\address{⟨address 1⟩}
\email{⟨email 1⟩}
\author{⟨author 2⟩}
\address{⟨address 2⟩}
\email{⟨email 2⟩}
...
```

The mutual order of \address, \curraddr and \email is not important.

2.3 PJLdate

PJLdate offers the \PLdate⟨*yyyy-mm-dd*⟩ (or \PJLdate⟨*yyyy-mm-dd*⟩) macro to convert ⟨*yyyy-mm-dd*⟩ into the date format of the currently selected language. For example, in current English context, \PLdate{2022-04-01} would become "April 1, 2022", while in French context as "1er avril 2022".

For details on how to select a language, please refer to the section on PJLlang.

2.4 PJLdraft

PJLdraft offers the following macros:

- \dnf or \dnf<...>. The effect is: `To be finished #1` or `To be finished #2: ...`.
  The prompt text changes according to the current language. For example, it will be displayed as `Pas encore fini #3` in French mode.
- \needgraph or \needgraph<...>. The effect is:

  `A graph is needed here #1`

  or

  `A graph is needed here #2: ...`

  The prompt text changes according to the current language. For example, in French mode, it will be displayed as

  `Il manque une image ici #3`

For details on how to select a language, please refer to the section on PJLlang.

2.5 PJLlang

PJLlang offers multi-language support, including simplified Chinese, traditional Chinese, English, French, German, Japanese, and Russian (among them, Chinese, Japanese, and Russian require appropriate TEX engines and fonts to support).

PJLlang provides language options. The names of these options have three types, which are abbreviations (such as `EN`), lowercase (such as `english`), and capital letters (such as `English`). For the option names of a specific language, please refer to ⟨*language name*⟩ below. Among them, the first specified language ⟨*first language*⟩ will be used as the default language, which is equivalent to specifying `\UseLanguage{`⟨*first language*⟩`}` at the beginning of your document.

> **Tip**
>
> It is recommended to use these language options and pass them as global options. In this way, only the specified language is set, thus saving the TEX memory and significantly improving the compilation speed.

The language can be selected by the following macros:

- `\UseLanguage{`⟨*language name*⟩`}` is used to specify the language. The corresponding setting of the language will be applied after it. It can be used either in the preamble or in the main body. When no language is specified, "English" is selected by default.
- `\UseOtherLanguage{`⟨*language name*⟩`}{`⟨*content*⟩`}`, which uses the specified language settings to typeset ⟨*content*⟩. Compared with `\UseLanguage`, it will not modify the line spacing, so line spacing would remain stable when CJK and Western texts are mixed.

⟨*language name*⟩ can be (it is not case sensitive, for example, `French` and `french` have the same effect):

- Simplified Chinese: `CN`, `Chinese`, `SChinese` or `SimplifiedChinese`
- Traditional Chinese: `TC`, `TChinese` or `TraditionalChinese`
- English: `EN` or `English`
- French: `FR` or `French`
- German: `DE`, `German` or `ngerman`
- Italian: `IT` or `Italian`
- Portuguese: `PT` or `Portuguese`
- Portuguese (Brazilian): `BR` or `Brazilian`
- Spanish: `ES` or `Spanish`
- Japanese: `JP` or `Japanese`
- Russian: `RU` or `Russian`

In addition, you can also add new settings to selected language:

- `\AddLanguageSetting{`⟨*settings*⟩`}`
  - — Add ⟨*settings*⟩ to all supported languages.
- `\AddLanguageSetting(`⟨*language name*⟩`){`⟨*settings*⟩`}`
  - — Add ⟨*settings*⟩ to the selected language ⟨*language name*⟩.

For example, `\AddLanguageSetting(German){\color{orange}}` can make all German text displayed in orange (of course, one then need to add `\AddLanguageSetting{\color{black}}` in order to correct the color of the text in other languages).

2.6 PJLlogo

PJLlogo offers the \ProjLib macro to draw the logo, which looks like ProjLib. It is similar to ordinary text macros and can be used with different font size macros:

\tiny: ProjLib
\scriptsize: ProjLib
\footnotesize: ProjLib
\normalsize: ProjLib
\large: ProjLib
\Large: ProjLib
\LARGE: ProjLib
\huge: ProjLib
\Huge: ProjLib

2.7 PJLmath

PJLmath offers the following shortcuts:

i) \mathfrak{·} ⟶ \mf· or \frak· . For example, \mfA (or \mf{A}) has the same effect as \mathfrak{A}. This works for both upper and lower case, producing:

$$\mathfrak{abcdefghijklmnopqrstuvwxyz}$$
$$\mathfrak{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$$

ii) \mathbb{·} ⟶ \bb· . This only works for uppercase alphabet and the number 1.

$$\mathbb{ABCDEFGHIJKLMNOPQRSTUVWXYZ1}$$

There are also special command for well-known algebraic structures: \N, \Z, \Q, \R, \C, \F, \A.

$$\mathbb{NZQRCFA}$$

iii) \mathcal{·} ⟶ \mc· or \cal· . This only works for uppercase alphabet.

$$\mathcal{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$$

iv) \mathscr{·} ⟶ \ms· or \scr· . This only works for uppercase alphabet.

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

2.8 PJLpaper

PJLpaper is mainly used to adjust the paper color. It has the following options:

- `paperstyle` = ⟨*paper style name*⟩
  - — Set the paper color style. The options available for ⟨*paper style name*⟩ are: `yellow`, `dark` and `nord`.
- `yellowpaper`, `darkpaper`, `nordpaper`
  - — Same as `paperstyle` with the corresponding ⟨*paper style name*⟩ specified.
- `preview`
  - — Preview mode. Crop the white edges of pdf file for the convenience of reading.

It is recommended to use them as global options of the document class. In this way, the paper settings would be clear at a glance.

2.9 PJLthm

PJLthm offers the configuration of theorem-like environments. It has the following option:

- `nothms`
  - — Theorem-like environments will not be defined. You may use this option if you wish to apply your own theorem styles.
- `nothmnum`
  - — Theorem-like environments will not be numbered.
- `regionalref`
  - — When referencing, name of the theorem-like environment will change with the current language (by default, the name will always remain the same; for example, when referencing a theorem written in the French context, even if one is currently in the English context, it will still be displayed as "Théorème"). In `fast` mode, this option is automatically enabled.

Preset environments include: `assumption`, `axiom`, `conjecture`, `convention`, `corollary`, `definition`, `definition-proposition`, `definition-theorem`, `example`, `exercise`, `fact`, `hypothesis`, `lemma`, `notation`, `observation`, `problem`, `property`, `proposition`, `question`, `remark`, `theorem`, and the corresponding unnumbered version with an asterisk `*` in the name. The titles will change with the current language. For example, `theorem` will be displayed as "Theorem" in English mode and "Théorème" in French mode. For details on how to select a language, please refer to the section on PJLlang.

> **Tip**
>
> When referencing a theorem-like environment, it is recommended to use `\cref{`⟨*label*⟩`}`. In this way, there is no need to explicitly write down the name of the corresponding environment every time.

If you need to define a new theorem-like environment, you must first define the name of the environment in the language to use:

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

For ⟨*language name*⟩, please refer to the section on PJLlang. When ⟨*language name*⟩ is not specified, the name will be set for all supported languages. In addition, environments with or without asterisk share the same name, therefore, `\NameTheorem{envname*}{...}` has the same effect as `\NameTheorem{envname}{...}`.

And then define this environment in one of following five ways:

- `\CreateTheorem*{`⟨*name of environment*⟩`}`
  - — Define an unnumbered environment ⟨*name of environment*⟩
- `\CreateTheorem{`⟨*name of environment*⟩`}`
  - — Define a numbered environment ⟨*name of environment*⟩, numbered in order 1,2,3,…
- `\CreateTheorem{`⟨*name of environment*⟩`}[`⟨*numbered like*⟩`]`
  - — Define a numbered environment ⟨*name of environment*⟩, which shares the counter ⟨*numbered like*⟩
- `\CreateTheorem{`⟨*name of environment*⟩`}<`⟨*numbered within*⟩`>`
  - — Define a numbered environment ⟨*name of environment*⟩, numbered within the counter ⟨*numbered within*⟩

- \CreateTheorem{⟨*name of environment*⟩}(⟨*existed environment*⟩)
  \CreateTheorem*{⟨*name of environment*⟩}(⟨*existed environment*⟩)
  - Identify ⟨*name of environment*⟩ with ⟨*existed environment*⟩ or ⟨*existed environment*⟩*.
  - This method is usually useful in the following two situations:
    1. To use a more concise name. For example, with \CreateTheorem{thm}(theorem), one can then use the name thm to write theorem.
    2. To remove the numbering of some environments. For example, one can remove the numbering of the remark environment with \CreateTheorem{remark}(remark*).

Tip

This macro utilizes the feature of amsthm internally, so the traditional theoremstyle is also applicable to it. One only needs declare the style before the relevant definitions.

Here is an example. The following code:

```
\NameTheorem[EN]{proofidea}{Idea}
\CreateTheorem*{proofidea*}
\CreateTheorem{proofidea}<subsection>
```

defines an unnumbered environment proofidea* and a numbered environment proofidea (numbered within subsection) respectively. They can be used in English context. The effect is as follows (the actual style is related to the document class):

Idea | The proofidea* environment.

Idea 2.9.1 | The proofidea environment.

3 Known issues

- PJLauthor is still in its preliminary stage, its effect is not as good as the relatively mature authblk.
- PJLlang: It is still quite problematic with the configuration of polyglossia, so main features are implemented through babel for now.
- PJLlang: There are some problems with the language options. For example, chinese will cause errors with babel. Also, conflicts among multiple options may occur.
- PJLpaper: the preview option is mainly implemented with the help of package geometry, so it does not work quite as well in the KOMA document classes.
- PJLthm: The numbering and theorem-style settings of the theorem-like environments cannot be accessed by the user at present.
- PJLthm: The localization of cleveref is not yet complete for all supported languages of PJLlang, especially for Chinese, Japanese and Russian.
- Error handling mechanism is incomplete: no corresponding error prompt when some problems occur.
- There are still many things that can be optimized in the code. Some takes too long to run, especially the setup of theorem-like environments in PJLthm.

4 Usage example

4.1 Standard classes

In standard classes, one usually only need to configure the page size, hyperlinks and load ProjLib before actually start writing the document. Below is a complete example.

```
\documentclass{article}
\usepackage[a4paper,margin=.75in]{geometry}
\usepackage[hidelinks]{hyperref}
\usepackage[palatino]{ProjLib} % Load the toolkit and use font Palatino

\UseLanguage{French} % Use French from here

\begin{document}

\title{Le Titre}
\author{Auteur}
\date{\PJLdate{2022-04-01}}

\maketitle

\begin{abstract}
    Ceci est un résumé. \dnf<Plus de contenu est nécessaire.>
\end{abstract}

\section{Un théorème}

%% Theorem-like environments can be used directly
\begin{theorem}\label{thm:abc}
    Ceci est un théorème.
\end{theorem}

Référence du théorème: \cref{thm:abc} % It is recommended to use clever reference

\end{document}
```

If PJLamssim is loaded, then one can adopt the $\mathcal{AMS}$ writing style in the document (of course, the original way is also valid, so always adding the option `amssim` usually does not cause problems). This way, the line that introduces ProjLib should be written as:

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

4.2 The $\mathcal{AMS}$ classes

In $\mathcal{AMS}$ classes, one usually only need to configure the page size, hyperlinks and load ProjLib before actually start writing the document. Below is a complete example.

```
\documentclass{amsart}
\usepackage[a4paper,margin=.75in]{geometry}
\usepackage[hidelinks]{hyperref}
\usepackage[palatino]{ProjLib} % Load the toolkit and use font Palatino

\UseLanguage{French} % Use French from here

\begin{document}

\title{Le Titre}
\author{Auteur 1}
\address{Adresse 1}
\email{\href{Courriel 1}{Courriel 1}}
\author{Auteur 1}
\address{Adresse 1}
\email{\href{Courriel 2}{Courriel 2}}
\date{\PJLdate{2022-04-01}}
\subjclass{*****}
\keywords{...}

\begin{abstract}
    Ceci est un résumé. \dnf<Plus de contenu est nécessaire.>
\end{abstract}

\maketitle

\section{Première section}

%% Theorem-like environments can be used directly
\begin{theorem}\label{thm:abc}
    Ceci est un théorème.
\end{theorem}

Référence du théorème: \cref{thm:abc} % It is recommended to use clever reference

\end{document}
```