# 使用 dtx 文档类文学化编程

冯振华

2019/05/08

**简介**

这份文件曾在 2018 年 3 月 25 日左右完成, 但是当时处于学习初始阶段, 所以没有深入学习理解就匆匆进入 cexam.sty 等的编写工作. 但是, 随着理解的深入以及使用 LaTeX3 可以使用更友好的语法来描述问题. 同时受 CTeX 宏集的影响, 我决定使用 LaTeX3 重写 cexam.sty, 这份文件就是为了这一目的及适应 LaTeX3 而再次深入学习理解后改写的.

## 第 1 节　查阅源码和字体设定

### 1.1　查阅宏包的源码

这个小节是为了能够快速找到相应的宏包的dtx 源文件，以方便引用相应的内容而设置的。因为我的电脑安装的是 Texlive[1] 所以采用了它的管理包命令来处理

```
tlmgr info package
```

上述命令可以列出所需宏包 package 的具体信息，可以用 vim 直接打开参考源文件而直接引用到我的说明文档中，以方便今后使用。

### 1.2　本文档对字体的设置

之所以做这个设置，因为我在写中文文档时，对于中文的粗体、斜体等设置用 XeLaTeX 编译时会出现警告，在 ctxdoc.cls 中设置的字体, 在我的电脑上没有安装, 但是我安装了完整的 Windows 字体和 Adobe 及 Funder 字体, 这是三个比较完整的字体. 所以在 ctxdoc.cls 中我作了对应的设置. 具体修改如下:

```
42 \RequirePackage[UTF8, punct = kaiming, heading, fontset = windows,
43   linespread = 1.2, sub3section]{ctex}
```

同时注释掉第 48 行和第 72—77 行的字体设置. 这样便完成了字体设定, 可以正常编写dtx 文件了.

## 第 2 节　由 doc 和 docstrip 分离 dtx 文件

一般宏包文件发布时都以dtx 文件的方式发布，同时会有一个对应的文件.ins 文件（ins 就是 install 的缩写）,dtx 文件包含了源文档及对应的说明文件, 下面的讨论就是使用这两个工具生成对应宏包和说明文档的工作过程.

[1]目前 2019 年 5 月 8 号,TexLive 版本为 TexLive2019.

## 2.1　由 package.ins 文件生成 package.sty

将 package.dtx 和 package.ins 放入同一文件夹内, 可以在终端下执行下列命令就可以生成宏包package.sty 了 [2].

```
latex package.ins
```

## 2.2　.ins 文件简介

ins 三个字母是 install 的缩写，顾名思义，这个文件是和安装相关的。.ins 文件通常用来控制 TEX 从 .dtx 文件里释放宏包文件，它结构简单，大概是这样子的：

1. 作为注释的版权信息
2. 载入 docstrip.tex :　　\input docstrip(\keepsilent,\usedir)
3. 写入 DocStrip 的控制命令
4. 编写将写入生成文件的版权信息
5. 生成文件指令
6. 写入提示信息用以完成安装
7. 结束安装文件

关于这一部分可以参考一 ctex 宏包的版权声明信息。它有以下几个特点

- 每一行都以百分号开头，因此在声明版权的同时不会影响正常的编译流程
- 版权声明部分，首先是声明版权的归属
- 接下来，声明许可协议为 LPPL。大多数 TEX 相关的宏包或软件，都在 LPPL 协议下发布

使用命令\input docstrip.tex 载入docstrip.tex。使用命令 \keepsilent 关闭 DocStrip 的日志输出功能。默认情况下，DocStrip 会详细输出它的每一步操作。对于大多数人来说，成百上千行的日志徒惹人嫌弃，因此关闭它。更多的控制命令可以在终端下执行命令

```
texdoc docstrip
```

来直接参考 DocStrip 的说明文档。

在\preamble 和 \endpreamble之间的部分，将被写入由 DocStrip 生成的所有文档（默认情况下）。比如由 ctex 宏包的这段文字将写入每一个生成的文件。

```
\generate
  {
     \usedir{tex/latex/ctex}
     \file{ctex.sty}         {\from{\jobname.dtx}{package,style}}
     \file{ctexcap.sty}        {\from{\jobname.dtx}{package,ctexcap}}
     \file{ctexsize.sty}        {\from{\jobname.dtx}{package,ctexsize}}
     \file{ctexart.cls}        {\from{\jobname.dtx}{class,article}}
     \file{ctexbook.cls}        {\from{\jobname.dtx}{class,book}}
     \file{ctexrep.cls}        {\from{\jobname.dtx}{class,report}}
     \file{ctexspa.def}
      {
        \from{\jobname.dtx}            {ctexspa}
        \from{ctexpunct.spa}    {}
      }
  }
```

\generate　\generate{⟨*Control commands for generating files*⟩}

New: 2018-03-25　这个命令里面写的有宏包安装的目录，生成文件的命令，可以有多个。

[2] 有的文件是 dtx 和 ins 文件合二为一的，这种情况在后面讨论

\usedir　\usedir{⟨*Macro Pack Installation Diretory*⟩}

New: 2018-03-25 此命令设置宏包的安装目录，比如说:tex/latex/ctex

\from　\from{⟨\jobname.*dtx*⟩}{⟨*Options*⟩}

New: 2018-03-25 此命令指名了根据相应的关键字生成对应的宏包或者文档类。

## 2.3　由 package.dtx 文件生成 package.sty

在终端下运行命令 latex docstrip 得到如下信息：

```
**************************************************************
* This program converts documented macro-files into fast *
* loadable files by strippin off (nearly) all comments ! *
**************************************************************

**************************************************************
* No Configuration file found, using default settings.   *
**************************************************************

**************************************************************
* First type the extension of your input file(s):        *
\infileext=dtx
**************************************************************

**************************************************************
* Now type the extension of you output file(s)  :        *
\outfileext=cls
**************************************************************

**************************************************************
* Now type the name(s) of option(s) to include  :        *
\Options=class
**************************************************************

**************************************************************
* Finally give the list of input file(s) without         *
* extension seperated by commas if necessary    :        *
\filelist=iuthesis
**************************************************************
```

执行 latex docstrip 命令以后，出现一系列对话框，上面二行星号间为每次出现的信息，分别回答三个问题依次为：输入文件的拓展名、生成文件的拓展名、和生成文件对应的尖括号内的关键字、dtx 文件名（不含拓展名）。

\Msg　\Msg{⟨*Console Displays Text*⟩}

New: 2018-03-25 此命令内的信息将会在处理完 .ins 文件之后显示在控制台，用来提示使用者将文件置入指定的目录。TEX 会忽略连续的空格，在第一行的\obeyspaces则会让 TEX 输出这些空格。

\endbatchfile　\endbatchfile

New: 2018-03-25 显式地结束安装文件，此后的所有内容都会被忽略。上面的介绍就是一个最简单的 .ins 文件内容了。

## 2.4　.ins 内部配置

docstrip 执行时读取配置文件 docsrip.cfg，一个重要的参数是 \BaseDirectory，指明系统 TEX 路径, 比如 \BaseDirectory{/usr/local/texmf}. 然后声明使用目录 \DeclareDir{name}tex/latex}, 那么在 .dtx 文件中使用 \usedirectory{name} 实际上就是将文件放在 /usr/local/texmf/tex/latex.

\UseTDS 则声明了 BaseDirectory 之后, 不用再繁琐的进行 DeclarDir, 而直接 useDir , 规则就是按照 TeX Directory System 的命令方法.

**注意**: docstrip 不会自己创建文件夹, 所以如果目标文件夹不存在, 则需要用户自己建立. 前面讨论了 .ins 文件中必要的命令, 下面补充完整. 下面这些配置也可以直接写在 .ins 文件里, 也可以略去不写.

\askforoverwritetrue
\askforoverwritefalse
\askonceonly
\preamble
\endpreamble
\postamble
\endpostamble
\declarepreamble
\usepreamble
\nopreamble

New: 2019-05-10

\askforoverwritetrue和 \askforoverwritefalse: 如果文件存在是否询问覆盖.
\askonceonly: 统一回答一次.
\preamble \endpreamble : 加到所有文件的头部信息, 前缀为%% (准确的是 \MetaPrefix)
\postamble \endpostamble: 加入到所有文件的尾部信息.
\declarepreamble\somename : 定义新的头部信息 (不同文件可能使用不同的头部信息)
\usepreamble\somename : 使用名为 somename 的 preamble
\nopreamble : 不使用任何 preamble
\declarepostamble\somename : 定义新的尾部信息 (不同文件可能使用不同的尾部信息)
\usepostamble\somename: 使用名为 somename 的 postamble
\nopostamble: 不使用任何 postamble

# 第 3 节 .dtx 文件简介

## 3.1 dtx 文件处理分析

.dtx 文件比 .ins 文件复杂。.ins 文件在整个过程中会被读取一次，而 .dtx 文件会被读取三次。下面对三个步骤作简略描述。

处理 .ins 文件的时候，会载入 .dtx 文件。这时候 .dtx 文件中以% 开头的行将被全部忽略，而以% 开头的行则会用于记录 <option>。程序会根据 .ins 文件中\generate命令里\from指示的 <option> 抽取 .dtx 文件中 <option> 相关内容。这个过程中，没有以% 开头的行会根据 <option> 写入文件，而以% 开头的行则被抑制并保护起来。最终生成宏包文件。

第二遍处理 .dtx 文件的时候，在读入\documentclassltxdoc 之前，所有的内容与通常意义上的 LaTeX 代码完全一致。因此需要依靠\iffalse和%\iffalse来保护相关代码，这些代码通常是 README 文件和 LICENSE 文件。在读入\DocInput{filename.dtx} 之后，.dtx 文件会被第三次载入处理。处理完成之后遇到\end{document}，后续的内容被全部忽略。最终生成说明文档。

第三遍处理 .dtx 文件是被\DocInput载入的。此时，文档内（几乎）所有的% 都被忽略。因为 LaTeX 只能有一个\documentclass和一对\begin{document} 及\end{document}，所以\end{document} 之前的所有内容都应当在这一次处理的时候被忽略。因此这部分内容应该被 \iffalse和\fi保护起来。

## 3.2 .dtx 文件组成

1. 包含在% \iffalse 和% \fi中间的版权信息
2. 包含在%\iffalse 和%\fi中间的宏包基本信息
3. 包含在% <*driver> 和% </driver> 中间的 ltxdoc 文档类及相关代码，这部分代码也包含在% \iffalse和%\fi中间
4. 在 \end{document} 之后的说明文档，这部分文档应该隐藏在行首的% 之后
5. 在说明文档最后的代码说明，以及夹杂在说明中间的代码，其中代码说明也应该隐藏在行首的% 之后，而代码本身不应该被% 保护
6. 对于注释, 因为% 已经被定义为特殊的含义, 它不能再作为 doc 部分的注释符号, docstrip 重新定义了^^A 作为内部的注释标记.2023-11-13 10:21 发现新的 docstrip 将注释符号重新定义为^^M, 所以后续再编写宏包时需要注意。

以上文档部分经过本人修改，多引用自 LaTeX 开源小屋, 它的后面还一部分叫**人肉编译器**在这里我没有打出来，大家可以直接点链接去看原始文档。已经于 2019 年 5 月 8 日加入了对二合一文件的理解, 这一部分也可以直接参考下一节关于二合一文件的处理.

### 3.3　编译生成文档

编译生成文档时可执行的命令：

1. xelatex package.ins 生成 package.sty 文件
2. xelatex package.dtx 生成说明文档
3. latex package.dtx 生成对应宏包文件
4. pdflatex package.dtx 生成说明文档（不含中文）
5. latex docstrip 填入相应选项则生成对应的.sty 和.cls 文件

## 第 4 节　.dtx 与.ins 二合一的情况

这一部分详细讨论将 .dtx 文件和 .ins 文件放在同一个文件 (.dtx 文件) 中的处理情况.[3] 这种二合一文件的结构可以用一个示例来解释, 如下

```
1\% \iffalse meta-comment
2\%<*internal>
3\iffalse
4\%</internal>
5\%<*readme>
6Readme
7\%</readme>
8\%<*internal>
9\fi
10\begingroup
11 \def\nameoflatexe{LaTeX2e}
12\expandafter\endgroup\ifx\nameoflatexe\fmtname\else
13\csname fi\endcsname
14\%</internal>
15\%<*install>
16\input docstrip.tex
17\keepsilent
18\generate{
19 \usedir{tex/latex/\jobname}
20 \file{\jobname.cls}{\from{\jobname.dtx}{class}}
21 \nopreamble\nopostamble
22 \file{README}{\from{\jobname.dtx}{readme}}}
23 \Msg{* Happy TeXing!}
24 \endbatchfile
25 \%</install>
26 \%<*internal>
27 \fi
28 \%</internal>
29 \%<*driver>
30 \ProvidesFile{skeleton.dtx}[2015/01/20 v1.0 A Skeleton File]
31 \documentclass{ltxdoc}
32 \begin{document}
33   \DocInput{\jobname.dtx}
34 \end{document}
35 \%</driver>
36 \% \fi
37 \% \CheckSum{0}
38 \% \GetFileInfo{\jobname.dtx}
39 \% \DoNotIndex{\test}
40 \% \StopEventually{}
```

[3] 2019 年 5 月 8 日下午学习完成后, 于晚上添加. 具体来源在 这几个 \iffalse 和 \fi 是什么意思中对此问题的具体解答.

```
41 \% \section{Implementation}
42 \%<*class>
43 \%    \begin{macrocode}
44 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
45 \ProvidesClass{skeleton}[2015/01/20 v1.0 A Skeleton File]
46 \%    \end{macrocode}
47 \%</class>
48 \%<class>\endinput
49 \% \Finale
50 \endinput
```

上述文件是引用知乎上用户提问时的文件, 由于其比较小巧, 所以便于说理, 同时也是从分析此文件开始真正理解二合一文件的, 所以直接放在这里了. 理解这文件的核心在于第 10 行–第 13 行的代码.

第 10 行定义了一个临时命令 \nameoflatexe 由于它是临时起作用, 所以可以使用任何你想要的名称都可以. 由于它是临时的, 所以需要将它限定于 \begingroup 和 \endgroup 之间. 但是根据不同的的功能需求在这时作了特定的设置. 命令 \fmtname 包含了当前执行此文件的程序, 如 (pdf/Xe)LaTeX 编译时, \fmtname 就是 LaTeX2e , 当使用 TeX 编译时, 它就不是 LaTeX2e .

## 4.1 获得宏包文件

使用 TeX 编译二合一的 .dtx 文件.TeX 读入文件, 其中第 1 行 \iffalse 应当于第 36 行的 \fi 配对, 第 3 行的 \iffalse 和第 9 行的 \fi配对. 第 12 行的 \ifx 和第 13 行及第 27 行的\fi 配对. 文件前面包含% 的行被忽略, 第 3 行和第 9 行之间的部分也不会被执行, 这样文件就从第 10 行开始执行, 第 11 行定义了 \nameoflatexe 第 11 行使用\expandafter使\ifx 先于 \endgroup 执行. 这样就来到了 \ifx, 由于此时编译程序为 TeX , 所以程序就到了 \else 部分, 于是 \csname␣fi\endcsname 执行, 则\ifx 执行完毕, 再执行 \endgroup 限定住了 \nameoflatexe 的作用范围, 实际上此命令也就没有用了. 紧接着调入 docstrip.tex 程序, 执行分离程序, 一直到第 24 行 \endbatchfile 则程序终止. 由于一直到程序运行终止都还没有运行到第 27 行的 \fi, 所以这不会出现任何错误.

这里有一点需要注意, 被 <*internal> 和 </internal> 所标记的部分, 是为了在生成各个宏包时避免将 \iffalse,\fi 等的输出. 因为各个分离部分都不含这二个标记, 所以它仅仅限于前述作用.

## 4.2 获得说明文档

使用 XeLaTeX 编译二合一的 .dtx 文件. XeLaTeX 读入文件, 其中第 1 行 \iffalse 应当于第 36 行的 \fi 配对, 第 3 行的 \iffalse 和第 9 行的 \fi配对. 第 12 行的 \ifx 和第 13 行及第 27 行的\fi 配对. 文件前面包含% 的行被忽略, 第 3 行和第 9 行之间的部分也不会被执行, 这样文件就从第 10 行开始执行, 第 11 行定义了 \nameoflatexe 第 11 行使用\expandafter使\ifx 先于 \endgroup 执行. 这样就来到了 \ifx, 由于此时编译程序为 XeLaTeX 所以 \ifx 判断的结果为真, 则 \else 到第 13 行 \endcsname 将失效所以 \ifx 还没有执行完成, 直到第 27 行的 \fi, 然后再执行 \endgroup. 所以第 10 行到第 27 行之间的部分将不被执行, 也就是跨过 docstrip 程序分离宏包的部分. 这样程序忽略其间被% 注释掉的部分. 运行到了第 30 行, 一直到第 33 行使用 \DocInput 重新调入.dtx 文件,% 将失去注释符的作用, 在调入文件后再恢复注释作用 (这是命令\DocInput的作用) 同时由于第 1 行的 \iffalse 和第 36 行的 \fi 配对, 所以这一部分将不被执行. 则程序运行到第 37 行, 而第 37 行一直到第 50 行是说明文档的内容, 则一直遇到第 50 行的 \endinput 则输入 .dtx 文件执行完毕, 回到第 34 行, 遇到 \end{document} 则过程结束. 我们将获得和宏包对应的说明文档.

# 第 5 节　l3doc.dtx 文件的结构

l3doc.dtx 文件的结构大体可以分为如下结构

```
1 \iffalse meta-comment
2  版权声明部分
3 \def\nameofplainTeX{plain}
4 \ifx\fmtname\nameofplainTeX\else
5   \expandafter\begingroup
6 \fi
7 \input l3docstrip.tex
8 \askforoverwritefalse
9 \preamble
10  the preamble information
11 \endpreamble
12 \% stop docstripadding \endinput

13 \postamble
14  the postamble information
15 \endpostamble

16 \generate {\file{package.sty}{\from{\jboname.dtx}{package}}}

17 \ifx\fmtname\nameofplainTeX
18    \expandafter\endbatchfile
19 \else
20   \expandafter\endgroup
21 \fi
22 </driver>
23 <driver|class>\RequirePackage{expl3,calc}
24 <*driver>
25 \PassOptionsToPackage{fontset = windows}{ctex}
26 \documentclass{ctxdoc}
27 \usepackage{framed}
28 \begin{document}
29 \DocInput{\jobname.dtx}
30 \end{document}
31 </driver>
32 \fi

33 \title{\jobname}
34 \author{<++>}
35 \date{<++>}
36 \maketitle
37 \tableofcontents

38 \begin{documentation}
39 \section{Introduction}
40 Code and documentation for this class have been written prior to the
41 \end{documentation}

42 \begin{implementation}
43 \section{\pkg{l3doc} implementation}
44 \end{implementation}
```

此文件也是二合一文件, 但是实现代码和 ctex.dtx 不同, 所以放在此处作为第二种方法. 这代码的的解释为: 第 1 行的 \iffalse 和第 32 行的 \fi 配对, 第 3-6 行和第 17-21 行配对. 在第 3 行定义了 \nameofplainTeX 为 plain , 然后对比编译程序的名字是否为 plain . 如果是, 则略去 \else 后到第 6 行的 \fi 部分, 即不执行 \begingroup 接下来就是调入 l3docstrip.tex 文件, 即开始进行宏包和类文件的分离工作. 直到第 17 行, 由于文件名 \nameofplainTeX 就是 plain , 则执行 \endbatchfile 则程序结束. 反之, 如果在第 4 行的判断为否, 则执行第 5 行的 \begingroup 同时第 17-21 行, 执行第 20 行的 \endgroup. 则介于 \begingroup和 \endgroup 的部分一直到最后所有的部分都被执行.

对比二种二合一文件，显然可以得出他们的不同：

1. ctex.dtx 执行\xetex 只生成pkg.sty 或class.cls 文件，执行\xelatex只生成说明文档ctex.pdf
2. l3doc.dtx 执行\tex 只生成 l3doc.cls 或其他宏包，执行\latex则同时生成 l3doc.cls 和 l3doc.pdf

## 第 6 节　宏包的版本信息设置

2019 年 07 月 31 日星期三 11:58:50 CST 在研究 LaTeX3 后，昨天完成cexam.sty 中的核心测行程序，今天计划将代码加入到cexam.dtx 源文件中. 但是一直不是很明白ctex.dtx 中所使用的版本控制的工作原理. 在source3 中找到答案，于是今天写此节以补充今后的开发.

一般在编写的宏包的时候我们必须声明所编写的是宏包还是类文件，但是在 LaTeX2e 中和 LaTeX3 中所使用的命令稍有不同. 同时，我们后续会对相应的宏包进行维护和升级，这时就必须对版本信息进行更改. 如果所开发的是一个宏包，则方法是并没有明显的不同. 但是一个 dtx 是同时维护了许多个宏包，则如果每次升级都逐一修改版本号就不是很方便了. 所以这里需要借助 \GetIdInfo 来完成. 在 LaTeX3 中和 LaTeX2e 中的方法开列如下

```
LaTeX2e : \ProvidesPackage{cexam}[2019/01/11 v2.0 for china middle school exam]
LaTeX3  : \ProvidesExplPackage{cexam}{2019/07/30}{v3.0}{for chinese examination}
LaTeX3  : \GetIdInfo $Id: <SVN info field >$ {<description>}
          \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
          {\ExplFileVersion}{\ExplFileDescription}
Example : \GetIdInfo$Id: ctex.dtx c498d8c 2017-04-01
                    21:33:50 +0800 Qing Lee <sobenlee@gmail.com> $
```

这里主要讨论 LaTeX3 的方法，上面所列出的两种表示方法是等效的. 但是第二种方法更适合维护多个宏包，此处命令\GetFileInfo的组成部分⟨*SVN ifo field*⟩我们借助ctex.sty 来说明. 在\GetFileInfo后紧跟着一个 $ 之后是 Id:. 再往后是宏包的名称，版本，时间. 对于时间 +0800 开始我不是很理解，查阅后得之这是东八区的符号，也就是北京时间. 在2017-04-01 一直到第二个 $ 之间的部分不参与生成名字，版本，时间. 因为这三个部分是以空格分离的，所以在其之后是起作用. 在第二个 $ 之后还应当有描述信息，但是在此文件中并没有，因为 ctex 是一个宏集，它包含了多个宏包，不同的宏包不同的用途，所以各个宏包单独写描述，当使用 docstrip 分离时分别写入对应文件. 同时，对于不同的宏包，名称也不一样，如果版本号也有不一样的地方，则需要单独写出.

## 第 7 节　vim 编写 dtx 文件的设置

这一部分介绍用 vim 编写 dtx 文件的准备工作，因为我习惯用 vim 工作，这是一款神器，非常高效，通俗点就是相当好使！

初次接触 vim 的同学，在终端下执行命令启动使用说明，一个 30 分钟的说明，可以教会你掌握基本的使用方法，以后你的能力将会飞快的增长，记住一定要静下心来耐心学完。

```
vimtutor
```

### 7.1　vim 滚屏命令

每次滚一行的命令是 CTRL-E（上滚）和 CTRL-Y (下滚)。可以把 CTRL-E 想象为是多给你一行（one line Extra）。

正向滚动一整屏的命令是 CTRL-F（送去两行）。反向的命令是 CTRL-B 。幸运地，CTRL-F 是向前（forward）滚动，CTRL-B 是向后（backward）滚动，这比较好记。

## 7.2　.vimrc 配置文件

一般常用的一些配置放在这个文件中，则每次启动 vim 则会打开相应的功能。这里主要记录几个专门为编写 dtx 文件定制的命令。由于是针对 dtx 文件编写的，所以采用了脚本的形式来写。具体设置如下

在~/.vimrc 文件中加入以下命令：

```
autocmd BufNewFile,BufRead *.dtx source ~/.vim/ftplugin/dtx.vim
```

## 7.3　dtx.vim 文件设置

为了针对 dtx 文件中输入最多的是每行加入一个百分号， 以及经常输入 macro 、macrocode 及 function 等环境，特定制以下命令，注意在具体的 dtx.bim 文件内容中 imap 后面的映射内容不应当断行，此处的断行是为了排版，而 frameverb 环境是不会自动断行的，所以我进行了手工断行处理.

```
"专为输入dtx文件设置的映射
"用来换行自动在开始加入一个百分号
imap <F2> <CR>%<Space>
"加入宏代码环境
imap <F3> <CR>%<Space><Space><Space><Space>\begin{macrocode}
      <CR><CR>%<Space><Space><Space><Space>\end{macrocode}<Esc>k0i
"加入无选项的macro环境
imap <F4> <CR>%<Space>\begin{macro}{}<Esc>2li<CR>%<Space><CR>
      %<Space><Space><Space><Space>\begin{macrocode}<CR><Esc>0i<CR>
      %<Space><Space><Space><Space>\end{macrocode}<CR>%<Space>\end{macro}
      <Esc>5k4li
"加入有选项的macro环境
imap <F6> <CR>%<Space>\begin{macro}[]<Esc>2li{}<CR>%<Space>
      <CR>%<Space><Space><Space><Space>\begin{macrocode}<CR><Esc>0i
      <CR>%<Space><Space><Space><Space>\end{macrocode}<CR>%<Space>
      \end{macro}<Esc>5k4li
imap <F8> <CR>%<Space>\begin{function}[]<Esc>2li{u}<CR>
      %<Space><Space><Space>\begin{syntax}<CR><Esc>0i%<Space><Space>
      <Space><Space><Space><CR>%<Space><Space><Space>\end{syntax}
      <CR>%<Space><Space><CR>%<Space>\end{function}<Esc>5k4li
"加入function环境，用来补充命令描述
```

上述设置具体如下

1. F2 在插入模式下换行并在第二行开头加入一个% 然后再输入一个空格
2. F3 插入 macrocode 代码环境
3. F4 插入 macro 环境
4. F6 插入带选项的 macro 环境
5. F8 插入 function 环境

在编写 cexam 的实践过程中，我发现这几个命令分开来输入比较方便，所以又做了一次修订，如下

```
"专为输入dtx文件设置的映射
"用来换行自动在开始加入一个百分号
imap <F2> <CR>%<Space>
"加入宏代码环境
imap <F3> <CR>%<Space><Space><Space><Space>\begin{macrocode}<CR><CR>
   %<Space><Space><Space><Space>\end{macrocode}<Esc>k0i
"加入无选项的macro环境
imap <F4> <CR>%<Space>\begin{macro}{}<Esc>2li<CR>%<Space><CR>%
   <Space>\end{macro}<Esc>2k4li
"加入使用示例
imap <F6> <CR>%<Space>\begin{synlax}{}<Esc>2li<CR>%<Space><++><CR>%
```

```
    <Space>\end{syntax}<Esc>2k4li
"加入有选项的macro环境
imap <F8> <CR>%<Space>\begin{function}[]<Esc>2li{<++>}<CR>%
      <Space><Space><++><CR>%<Space>\end{function}<Esc>2k4li
"加入function环境，用来补充命令描述
```

# 第 8 节　常规文本标记

本节的许多命令源自于 ltxdoc , 同时加入了一些改进.[4]

\cmd
\cs

\cmd [⟨*options*⟩] ⟨*control sequence*⟩
\cs [⟨*options*⟩] {⟨*csname*⟩}

New: 2019-05-09

这些命令用来打印控制序列. \cmd\foo 产生 "\foo" 和 \cs{foo}的效果相同. 一般来说, \cs 更加健壮, 因为这不依懒于正确的 catcode 值, 因此推荐使用.

这些命令兼容 @@ l3docstrip 语法, 在输出文件中正确替换它们. 但仅在 %<@@=⟨*module*⟩> 声明之后起作用.

此外, 这些命令可以在 \cs 的参数中使用. 例如 \cs{\meta{name}:\meta{signature}} 产生 \⟨*name*⟩:⟨*signature*⟩.

⟨*options*⟩ 是一些 "键—值" 表, 它包含以下键:

- index=⟨*name*⟩: 将⟨*csname*⟩加入索引就好像编写了 \cs{⟨*name*⟩} 一样.
- no-index: ⟨*csname*⟩ 不被索引.
- module=⟨*module*⟩: ⟨*csname*⟩ 被⟨*module*⟩中所列出的命令索引;⟨*module*⟩ 尤其可以是 "TEX and LATEX 2ε" 命令中 TeX , 或者是那些出现在主索引中的空命令. 默认情况下, 自动从命令名中推断出⟨*module*⟩.
- replace 是一个指示是否像 l3docstrip 取代 @@ 那样的布尔值 (默认为真).

这些命令允许在 (大多数) 下划线之后对控制序列进行断字. 默认情况下, 使用一个断字符来标记断字, 但这可以通过 cs-break-nohyphen 选项进行更改. 若要完全禁止控制序列的断字, 使用 cs-break-off.

\tn

\tn [⟨*options*⟩] {⟨*csname*⟩}

New: 2019-05-08

类似于 \cs 但是用于引用传统的 TEX 或 LATEX 2ε 命令. 这实际上等价于

\cs [module=TeX␣,␣replace=false,⟨*options*⟩] {⟨*csname*⟩}.

\meta

\meta {⟨*name*⟩}

New: 2019-05-08

\meta 打印带 ⟨*angle brackets*⟩ 的意大利斜体的 ⟨*name*⟩. 在 function 环境或类似环境中, 尖括号 <...> 被设置为 \meta{...}的缩写. 数学模式中的下划线也可以应用. 例如 \meta{arg_{xy}} 产生 "⟨$arg_{xy}$⟩".

\Arg
\marg
\oarg
\parg

\Arg{⟨*name*⟩}

使用 \meta 打印 ⟨*name*⟩ 并用添加大括号. 根据 LATEX 2ε 的语法, ltxdoc 中的\marg 、\oarg 和 \parg分别被用于产生花括号, 方括号和圆括号.

New: 2019-05-08

\file
\env
\pkg
\cls

\pkg {⟨*name*⟩}

它们都采用一个参数, 并分别用作表示文件, 环境, 包名称及类名称的语义命令.

New: 2019-05-09

[4] 这一部分内容根据 l3doc.dtx 文件中的内容翻译而来, 由于本人英文水平有限, 参考了百度翻译和理解, 并不是逐字对应翻译的.

\NB
\NOTE
New: 2019-05-09

```
\NB {⟨tag⟩} {⟨comments⟩}
\begin{NOTE} {⟨tag⟩}
   ⟨comments⟩
\end{NOTE}
```

在源文件中做默认没有被排版的笔记. 当 show-notes 类选项被激活时, 注释分别以逐字和逐字模式排版.

## 8.1　文档中函数的描述

function (*env.*)
syntax (*env.*)

两个重度使用的环境, 定义为用来描述 expl3 的函数和变量.

```
\begin{function}{\function_one:, \function_two:}
  \begin{syntax}
    |\foo_bar:| \Arg{meta} \meta{test_1}
  \end{syntax}
\meta{description}
\end{function}
```

效果如下:

\function_one: |\foo_bar:| {⟨*meta*⟩} ⟨$test_1$⟩
\function_two: ⟨*description*⟩

函数环境有一个选项来指示它描述的函数是可展开的, 禁止展开的还是以条件形式定义的. 这些选项为EXP, rEXP, TF, pTF 和 noTF; 注意 pTF 意味着 EXP 因为其必须是可展开的, 同时noTF的意味着除了 TF 之外, 在文档中还应当添加 TF 选项. 举一个例子:

```
\begin{function}[pTF]{\cs_if_exist:N}
  \begin{syntax}
    \cs{cs_if_exist_p:N} \meta{cs}
  \end{syntax}
\meta{description}
\end{function}
```

效果如下:

\cs_if_exist_p:N ⋆ \cs_if_exist_p:N ⟨*cs*⟩
\cs_if_exist:N*TF* ⋆ ⟨*description*⟩

variable (*env.*)

若要记录一个变量而非函数, 请使用 variable 环境; 它的行为和上面的 function 环境相同.

texnote (*env.*)

此环境用于调用 function 环境内的部分, 类似的部分仅对 TEX 经验丰富的开发人员感兴趣.

macro (*env.*)

用于标记宏/函数实现的 LATEX $2_\varepsilon$ 的良好环境仍然是 macro 环境. 在 l3doc 中有一些变化: 它现在可以接受函数的逗号分隔列表, 以避免大量连续的 \end{macro} 情况. 空格和新行将被忽略 (选项 [verb] 可以防止这种情况).

```
% \begin{macro}{\foo:N, \foo:c}
%   \begin{macrocode}
... code for \foo:N and \foo:c ...
%   \end{macrocode}
% \end{macro}
```

如果你正在记录一个辅助宏, 则通常不需要突出显示它, 也不需要检查它, 例如, 是否具有测试函数以及是否在 function 环境中具有文档模块.l3doc 将从名称中存在__ 的情况中提取, 或者你可以使用 \begin{macro}[int] 强制标记它. 在这种情况下, 边注将以灰色打印.

对于记录 expl3 类型的条件, 你可以向此函数传递一个 TF 选项 (并从函数名中省略它), 以表示函数提供了 T,F, 和 TF 后缀. 类似的 pTF 选项同时记录了TF和 _p 的表单. 选项 noTF记录了不是 TF 也不是 T 或 F的表单. 像这种 \prop_get:NN 的函数, 记录了有条件的表单 (\prop_-get:NNTF).

\TestFiles　\TestFiles{⟨*list of files*⟩} 用来指示当前代码使用的测试文件; 它们在文档中被打印.

\UnitTested　在 macro 环境中, 最好标记是否为其定义的命令创建了单元测试. 这是通过 \begin{macro} ...\end{macro} 内的任意位置写入 \UnitTested 来表示的.

如果启用了类选项 checktest , 那么在没有调用 Testfiles 的情况下使用 macro 环境是一个错误.

这适用于大型软件包, 如 expl3, 这些软件包应具有绝对全面的测试套件, 并且其作者可能并不总是像他们应该的那样善于用新代码添加新测试.

\TestMissing　如果函数缺少测试, 则可以通过写入\TestMissing{⟨*explanation of test required*⟩} 来标记. 这些缺失的测试在编译运行结束时的列表中.

variable (*env.*)　在记录变量定义时请使用 variable 环境. 在这里它的行为和 macro 环境相同, 但是如果启用一类选项 checktest , 则不需要变量具有测试文件.

arguments (*env.*)　在 macro 环境中, 你可以使用 arguments 环境来描述函数所采用的参数. 它的行为类似于修改过后枚举环境.

```
% \begin{macro}{\foo:nn, \foo:VV}
% \begin{arguments}
%   \item Name of froozle to be frazzled
%   \item Name of muble to be jubled
% \end{arguments}
%   \begin{macrocode}
... code for \foo:nn and \foo:VV ...
%   \end{macrocode}
% \end{macro}
```

# 第 9 节　编写 dtx 文件的实践

## 9.1　尖角括号

在界定一个宏包所包含的代码时，可以用 <*option> 表示开头，用</option>表示结尾, 在其间的代码会在 docstrip 分离宏包时, 按对应的 option 写入对应的宏包或文档类。

如果几个宏包有共用的代码，则可以作如下设置 <*option1,option2> 表示开头，用 <*option1 , ␣option2> 表示结尾。同时如果在另外一部分，标注了另外的代码 <*option2 , option3> 开头和 <*option2,option3> , 则在使用 latex␣docstrip 分离 dtx 文件时，如果关键字输入option2 , 则所生成的文件将包含这两部分的代码。这是在实践中总结的。

2018 年 5 月 4 号通过对标准文件 classes.dtx 分析, 得到关于尖角号的进一步应用. 在尖角号标注的时候, 可以使用逻辑关系—或 ||, 且 &, 非! . 以几个例子说明: 如果使用 <option1|| option2> , 则分离文件时, 输入选项 option1 和 option2 和 option1,option2 都会输出该尖角号标注的这一行. 如果使用 <option1&option2> , 则分离文件时, 输入选项 option1,option2 才会输出这一行标注的内容. 如果使用 <!option1> , 则在 dtx 文件中, 所有不含 option1 标注的行都会输出. 如果使用 <option1|| !option2> , 则分离文件时, 输入选项中包含 option1 和所有不包含 option2 的选项都会输出到对应的文件中. 使用 “,” 号和或的符号相同意义, 表示 “或”.

# 第 10 节　生成索引和历史变化

这一部分说明如何生成索引，就像利用 latex docstrip 命令来分离 dtx 文件，得到 sty 和 cls 文件一样，关于索引和 changes 需要另个的一个工具 makeindex 来处理，需要先用

XeLaTeX 编译一遍文件，生成\jobname.idx然后再用 makeindex 处理后就生成 ist 文件，然后在文档中使用\PrintIndex就可以生成索引了。

The doc package writes index files to be sorted using MakeIndex with the gind style, so one would then use a command such as

```
makeindex -s gind.ist ltclass.idx
```

and re-run LaTeX.

同理可以打印历史变化，需要在文档开头加入

```
\AtBeginDocument{\RecordChanges}
\AtEndDocument{\PrintChanges}
```

到文件ltxdoc.cfg，然后用 MakeIndex 命令如下

```
makeindex -s gglo.ist -o ltclass.gls ltclass.glo
```

最后，假如你不想列出 source2e.tex 的所有节，你可以在 cfg 文件中使用命令 \includeonly，如下

```
\includeonly{ltvers,ltboxes}
```

## 第 11 节　参考文献


1. 《LaTeX 入门》（刘海洋，电子工业出版社 2013pdf 版）
2. 《LaTeX2e 完全学习手册》（胡伟，清华大学出版社 2011pdf）
3. 《The TeX book》（中文翻译版，鲜鲜）
4. ctxdoc.dtx 文件和 ctxdoc.cls 文件
5. ctex.dtx 文件
6. source2e 源文件
7. 在 LaTeX 中进行文学编程
8. l3doc.dtx 文件