在 LaTeX 中进行文学编程

发表于 2015 年 01 月 23 日 | 分类于 LaTeX | 本文共被围观 593 次 | 5条评论

文学编程是 TeX 的作者高德纳提出的编程方式，主张程序员在编写代码的过程中详细地记录自己的思维方式和内在逻辑。

这种编程方式注重编码的逻辑而将编码本身放在更次要的位置，因而不充分的设计在这种变成方式下无所遁形。文学编程的另一个优点是它产生的代码文档能帮助程序员在任意时候重新会想起当时编码的思路。

在 LaTeX 中，可以用 Doc 和 DocStrip 这两个工具来实现文学编程。

对于程序员来说，最重要的有三个部分：

- 代码；
- 代码文档；
- 用户手册。

使用 Doc 和 DocStrip 可以将这三个部分集中到一个 `.dtx` 文件当中。这篇文章将分三个部分讲述如何构建这样一个 `.dtx` 文件。

`.ins` 文件

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，它的结构最为简单，大概是这样的。

1. 作为注释的版权信息
2. **载入 `docstrip.tex`**
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**

这里粗体标记的是必不可少的部分，其他部分或多或少都可以省略。

我们逐步构建一个完整的 `.ins` 文件。

```
%% Copyright (C) 2003--2015
%% CTEX.ORG and any individual authors listed elsewhere in this file.
%% ---------------------------------------------------------------------
%%
%% This work may be distributed and/or modified under the
%% conditions of the LaTeX Project Public License, either
%% version 1.3c of this license or (at your option) any later
%% version. This version of this license is in
%%    http://www.latex-project.org/lppl/lppl-1-3c.txt
%% and the latest version of this license is in
%%    http://www.latex-project.org/lppl.txt
%% and version 1.3 or later is part of all distributions of
%% LaTeX version 2005/12/01 or later.
%%
%% This work has the LPPL maintenance status `maintained'.
%%
%% The Current Maintainers of this work are Leo Liu, Qing Lee and Liam Huang.
%% ---------------------------------------------------------------------
```

这是 `ctex` 宏包的版权声明信息。

- 这里的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。

```
\input docstrip.tex
```

载入 `docstrip.tex`。

```
\keepsilent
```

这是 DocStrip 的控制命令之一，其作用是关闭 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，因此我们关闭它。

更多的控制命令可以参考 DocStrip 的文档。

```
\preamble

    Copyright (C) 2003-2015
    CTEX.ORG and any individual authors listed in the documentation.
---------------------------------------------------------------------------

```

```
    This work may be distributed and/or modified under the
    conditions of the LaTeX Project Public License, either
    version 1.3c of this license or (at your option) any later
    version. This version of this license is in
       http://www.latex-project.org/lppl/lppl-1-3c.txt
    and the latest version of this license is in
       http://www.latex-project.org/lppl.txt
    and version 1.3 or later is part of all distributions of
    LaTeX version 2005/12/01 or later.

    This work has the LPPL maintenance status `maintained'.

    The Current Maintainers of this work are Leo Liu, Qing Lee and Liam Huang.
----------------------------------------------------------------------------

\endpreamble
```

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）。比如上面这段 `ctex` 宏包的版权信息将写入每一个生成的文件。

下面的内容是整个 `.ins` 文件里最重要的部分，它控制着如何生成最终的宏包文件。

```
\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` 命令中的 `\file{⟨filename⟩}{\from{⟨sourcefilename⟩}{⟨optionlist⟩}}` 用来指定宏包文件的生成方式。这里的含义是，从 `⟨sourcefilename⟩` 中抽取含有 `⟨optionlist⟩` 标记的部分，生成 `⟨filename⟩` 这个文件。

一个 `\generate` 里可以有多个 `\file` 命令。一个 `\file` 命令里可以有多个 `\from` 命令。一个 `\from` 命令里的 `⟨option⟩`，组成 `⟨optionlist⟩`。其中 `⟨option⟩` 需要用逗号隔开，`\file` 之间和 `\from` 之间则不需要。

```
\obeyspaces
\Msg{*************************************************************}
\Msg{*                                                           *}
\Msg{* To finish the installation you have to move the following *}
\Msg{* file into proper directories searched by TeX:             *}
\Msg{*                                                           *}
\Msg{* The recommended directory is TDS:tex/latex/ctex           *}
\Msg{*                                                           *}
\Msg{*     ctex.sty                                              *}
\Msg{*     ctexcap.sty                                           *}
\Msg{*     ctexsize.sty                                          *}
\Msg{*     ctexart.cls                                           *}
\Msg{*     ctexbook.cls                                          *}
\Msg{*     ctexrep.cls                                           *}
\Msg{*     ctexspa.def                                           *}
\Msg{*                                                           *}
\Msg{* To produce the documentation run the file ctex.dtx        *}
\Msg{* through XeLaTeX.                                         *}
\Msg{*                                                           *}
\Msg{* Happy TeXing!                                            *}
\Msg{*                                                           *}
\Msg{*************************************************************}
```

这里的内容会在处理完 `.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，第一行的 `\obeyspaces` 则会让 TeX 输出这些空格。

```
\endbatchfile
```

显式地结束安装文件，此后的所有内容都会被忽略。

将上面的代码依次粘贴到一起，就是一个最简单的 `.ins` 文件了。

`.dtx` 文件

`.dtx` 文件比 `.ins` 文件复杂得多。`.ins` 文件在整个过程中会被读取一次，而 `.dtx` 文件却会被读取三次。下面是对此三个步骤的简略描述。如果暂时看不懂也没有关系，下一节我们将会化身人肉编译器，逐行分析示例。

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` 的指示的 `⟨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 `%` 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 `\documentclass{ltxdoc}` 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` 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` 以及一对 `\begin{document}` 和 `\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。

`.dtx` 文件由以下部分组成。

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，这部分代码也应包含在 `% \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，以及间杂在代码说明中间的代码，其中代码说明也应该隐藏在行首的 `%` 之后，而代码本身则不应该被 `%` 保护**

示例 `.dtx` 文件的具体内容，可以参看下一节。

人肉编译器

我们来看两个文件，分别是 `dtxtut.ins` 和 `dtxtut.dtx`。你可以下载这两个文件，然后跟着我的思路一起分析。

首先我们要生成宏包文件，在命令行中运行

```
1  xelatex dtxtut.ins
```

注意，这里的后缀名不可省略。

文件读入之后，直到 12 行都是注释，直接忽略。接着到了第 13 行，读入了 `docstrip.tex` 这个文件。随后进行了一些设置之后来到了第 26 行.

```
1  \generate{\file{\jobname.sty}{\from{\jobname.dtx}{package}}}
```

这里 `\jobname` 是当前文件的名字（不含后缀），即 `dtxtut`。所以这里会从 `dtxtut.dtx` 文件中，抽取 `pac`

kage 的部分，组成名为 dtxtut.sty 这个文件。

现在我们读取 dtxtut.dtx 。前 18 行都是注释，忽略。19 – 21 行是 package 部分，输出。23 行开始了名为 driver 的部分，直到 32 行结束。接下来一直到 81 行都是注释，忽略。82行开始了名为 package 的部分，于是程序将 82 行开始到 100 行中没有注释掉的部分抽出来，输出。这样，输出的内容接在 \preamble 之后，保存为 dtxtut.sty 文件。

接下来我们生成宏包文档，在命令行中运行

```
1  xelatex dtxtut.dtx
```

同样，这里的后缀名不可省略。

前 23 行都是注释，忽略。24 行载入了 ltxdoc 文档类，随后载入了刚刚生成的 dtxtut.sty 宏包。26 行的 \EnableCrossrefs 打开了代码索引的生成（如果你将来不需要，可以用 \DisableCrossrefs 打开）。27 行的 \CodelineIndex 则使得索引指向代码行号，而不是页码。28 行的 \RecordChanges 则会让文档类记录宏包的变化记录。

接下来，30 行的 \DocInput{\jobname.dtx} 重新载入了这个文件本身，但（几乎）所有的 % 符号都被忽略。第 1 行遇到 \iffalse ，直到 16 行的 \fi ，中间的内容都被忽略。17 行是空行。18 行又遇到 \iffalse ，直到 33 行的 \fi ，中间的内容都被忽略。注意，这里正好跳过了 driver 部分。

接下来的 \CheckSum 和 \CharacterTable 是为了检测 .dtx 文件完整性的两个工具。众所周知， .dtx 文件包含了一个宏包的几乎所有信息，如果文件在网络传输的过程中出错，则宏包安装必然失败。因此，检测文件的完整性就变得很有必要。

\CheckSum 采用了一个很简单的方案来检验完整性。它将从 \StopEventually 开始到 \Finale 结尾的，在 macrocode 环境里的反斜杠 \ 计数，将计数的值作为校验和。在生成文档的过程中，程序将会计算这个值，并于 \CheckSum{} 中的值进行比对。若二者不一致，则说明传输过程中可能出现错误。

\CharacterTable 更为直接一点。程序将检查代码中出现的符号均包含在 \CharacterTable 之中。若不然，则认为传输过程中可能出错。

接下来的内容，直到第 76 行都很好理解。77 行出现了 \StopEventually 命令，并在参数中启动了 \PrintIndex 命令。我们刚才说了 \StopEventually 会开启校验和的检查，而 \PrintIndex 则会在此处打印代码索引。

`macrocode` 环境是一个特殊的环境，它有点类似于 LaTeX 原生的 `verbatim` 环境。它在大多数情况下的行为和 `verbatim` 环境相同，大体上是将代码输出到最终的文档当中。

这样持续运行到 103 行，遇到 `\Finale` 命令，校验和计算的终点。此时，程序将会计算出校验和，并与文档中给出的校验和进行比较。如果 `\CheckSum` 不存在，或者给出的值为 `0`，那么程序会给出正确的值，让你填入文档。如果 `\CheckSum` 存在，但值与程序计算的不符，那么程序会报错，并给出正确的值。改正后重新编译才能得到文档。如果 `\CheckSum` 存在，且校验和通过，那么程序会继续运行。

104 行是 `\endinput`，结束整个文件，本次对 `dtxtut.dtx` 的处理结束，回到上一次的断点。

30 行之后，31 行就是 `\end{document}`，文档处理结束，之后的内容全都被忽略。输出文档，退出。

将 `.dtx` 和 `.ins` 合二为一

细心的读者会发现，整个过程中，`.dtx` 文件承担了几乎所有的功能，而 `.ins` 文件只是一个「指路人」。这些读者可能会思考，是不是有办法将 `.ins` 文件并入 `.dtx` 文件呢？答案是肯定的。

对于这类合二为一的 `.dtx` 文件，使用 Plain TeX 格式编译，会启动 DocStrip 工具，得到宏包文件；使用 LaTeX 格式编译，则会生成文档。

这里给出一个来自知乎提问的示例（略有修改），你可以在我的答案里看到对此的具体分析。

最后的最后

今天是 W 的生日。祝你生日快乐~