驯服猛兽

—关于 BibTeX 的前生今世

英文名: Tame the BeaST (The B to X of BibTeX)

原作者: Nicolas Markey
markey@lsv.ens-cachan.fr

译作者: LaughCry
laughcry2002@163.com

版本号: 1.3 – October 16, 2005

这本共 53 页的小册子旨在清晰而又详尽地向读者介绍关于 BibTeX 的功能与原理. 之所以写这么个教程, 是因为 BibTeX 手册 (实际上主要是其作者撰写的两个文档 [Pat88a, Pat88b]) 以及几本 LaTeX 书籍 [Lam97, GMS93, MGB+04, ...] 中介绍 BibTeX 的章节大多过于简略, 不能完整地说明其功能与机理.
本手册英文名称中的三个大写字母 “BST” 表示 BibTeX 文献样式文件的标准扩展名, “从 B 到 X” 表示我希望本手册尽可能完整全面 (目标是 “从 A 到 Z”). 欢迎就关于 TeX 方面的技术以及本文档的错误给我发送电子邮件.

Contents

Part 1

LaTeX 中基本的参考文献表

Table of Contents

在 LaTeX 中, BibTeX 和其它各式各样的文献类命令一样, 通常被人们认为是比较复杂深奥的内容. 许多人在使用时, 大多是通过复制/粘贴经典的源代码, 然后加以修改完成自己的要求, 但对这些命令背后的原理并不清楚. 其实, 在 LaTeX 中, 参考文献表仅仅是一个文档中引用到的文献的普通列表而已. 假如我们在对 thebibliography 环境完全不熟悉的情况下手工创建参考文献表的话, 那应该这样书写:

```
这是文档的正文内容，其中提及了 [\ref{doc1}] 与 [\ref{doc2}].
\section*{References}
\begin{enumerate}
  \renewcommand\labelenumi{[\theenumi]}  %% numbers are surronded with brackets
  \item \label{doc1} Michel Goossens, Franck Mittelbach and Alexander
        Samarin, \emph{The \LaTeX{} Companion}, Addison Wesley, 1993.
  \item \label{doc2} Leslie Lamport, \emph{\LaTeX: A Document Preparation
        System}, Addison Wesley, 1997.
\end{enumerate}
```

上述文档经过编译后, 生成的效果为

这是文档的正文内容, 其中提及了 [1] 与 [2].

References

[1] Michel Goossens, Franck Mittelbach and Alexander Samarin, *The LaTeX Companion*, Addison Wesley, 1993.

[2] Leslie Lamport, *LaTeX: A Document Preparation System*, Addison Wesley, 1997.

这个效果与使用 `thebibliography` 环境完全类似 (该环境以一个 `list` 环境开始, 效果类似于 `enumerate`), 而 `\bibitem` 则对应于 `\item` 命令. 二者最大差别之处在于 `\bibitem` 允许使用比 `\item` 和 `\label` 更一般性的交叉引用 (例如, 我们可以这样引用:[GMS93]).

本部分内容正是围绕 "如何在不使用 BibTeX 的情况下书写参考文献表" 展开的, 虽然这并不是这本手册的主要目的, 但它对于深入理解其前后机理非常重要.

1 `thebibliography` 环境

首先应当注意, LaTeX 本身并没有定义 `thebibliography` 环境 (当然 TeX 也没有定义它[1], 它是在文档类文件中定义的 (例如, article.cls 或 book.cls 等). 正如先前所述, 它首先新建一个章节 (根据文档类的不同, 章节可能是 `\section` 或 `\chapter`), 然后在其中定义一个 `list` 类环境[2], 不过为了避免在缩进方面产生问题, 这个列表还是精细地调整. 举例来说, 如果我们在先前例子中改用 "字母与数字" 式的标签, 那么会得到以下不正常的排版效果

References

[GMS93] Michel Goossens, Franck Mittelbach, and Alexander Samarin, *The LaTeX Companion*, Addison Wesley, 1993.

[Lam97] Leslie Lamport, *LaTeX: A Document Preparation System*, Addison Wesley, 1997.

为了避免出现这种不适当的缩进问题, `thebibliography` 提供了一个强制参数, 它通常应取为列表中最长的标签, 目的是以此为标准建立合理空白间距.

下面我们以文档类 article.cls 为例, 来看一下 `thebibliography` 环境的精确定义:

```
1  \newenvironment{thebibliography}[1]
2       {\section*{\refname
3          \@mkboth{\MakeUppercase\refname}{\MakeUppercase\refname}}%
```

正如我们先前预料的那样, `thebibliography` 首先新建了一个节 (或章)[3]. 该环境同时修改了页眉[4].

[1] BibTeX 也可以配合 TeX 使用, 方法是要包含 (input)btxmac.tex 宏包.

[2] 这也就解释了为什么 `thebibliography` 环境需要在文档类文件中定义, 而其它与文献有关的一般命令则都是在标准的 LaTeX 格式中定义的

[3] 事实上, 由于它采用了一个 `\section*` 命令, 所以它不会出现在目录表中. 如果希望它出现在目录表中, 则可以使用 tocbibind.sty 宏包. 其它的解决办法通常会造成页码错乱.

[4] 标准的 apalike.sty 宏包会根据文档类的不同, 将新页眉硬编码地定义为 "REFERENCES" 或 "BIBLIOGRAPHY". 另一

```
4         \list{\@biblabel{\@arabic\c@enumiv}}%
5               {\settowidth\labelwidth{\@biblabel{#1}}%
```

这也是我们预料中的内容: 先创建一个新的 `list` 环境[5], 它有两个强制参数:

- 第一个强制参数 (`\@biblabel{...}`) 用于指定标签格式, 这里是通过 `\@biblabel`[6]命令与 `enumiv` 计数器的配合来实现的, 因此会得到 [1], [2], ... 这样的标签输出. 由于 `\bibitem` 是一种特殊的 `\item`, 因此也允许通过使用可选参数来改变这里所定义的缺省标签的格式.
- 第二个参数 (从上面的第 5 行到下文的第 11 行之间) 是在该环境开始时要运行的一条或多条命令, 通常是一些设置 `list` 环境中各种长度与参数取值的命令. 这里也是为了实现合理缩进而需要我们提供最长标签的地方 ("最长标签" 是 `thebibliography` 环境的强制参数). 人们经常会书写 `\begin{thebibliography}{99}` 这样的语句, 但要注意这种写法仅在被引文献数目在 10 到 99 时才是恰当的 (当然, 这里假定所有数字符号具有相同的宽度, 例如: 在使用 cmr 类字体时就是这种情况).

`thebibliography` 环境定义的其余部分还包括了列表边界的确定以及计数器 `enumiv` 的使用:

```
6               \leftmargin\labelwidth
7               \advance\leftmargin\labelsep
8               \@openbib@code
9               \usecounter{enumiv}%
10              \let\p@enumiv\@empty
11              \renewcommand\theenumiv{\@arabic\c@enumiv}}%
```

使用 `\@openbib@code` 可以允许我们根据需要修改一些参数的取值, 它的缺省定义为空. 传统的样式文件会提供一个 `openbib` 选项, 该选项会使用这条命令对某些参数进行重新设置. 第 9 到 11 行的代码重置了列表计数器的值.

最后, 定义了在参考文献列表排版中所使用的一些间距与分段处理参数:

```
12        \sloppy
13        \clubpenalty4000
14        \@clubpenalty \clubpenalty
15        \widowpenalty4000%
16        \sfcode`\.\@m}
```

以上就是该环境前置内容的所有内容. 而 `thebibliography` 环境的后置内容则更简单: 如果文献列表为空, 则回显一条警告消息, 最后关闭 `list` 环境:

```
17      {\def\@noitemerr
18        {\@latex@warning{Empty `thebibliography' environment}}%
19        \endlist}
```

个同名宏包则将新页眉硬编码为 `\refname` 或 `\bibname`. 因此如果你在实践中遇到关于页眉定义方面的问题, 不妨考虑通过检查一下宏包的源码以确定问题所在.

[5] `\list` 命令等价于 `\begin{list}`, 为了封闭环境, 它需要有一个相匹配的 `\endlist` 命令.

[6] `\@biblabel` 命令是在 LaTeX 中定义的, 它的输出是在其参数串两侧加上方括号. 其精确定义为: `\def\@biblabel#1{[#1]}`.

2 \bibitem 命令

在前述 list 环境的内部, 通常需要插入一些 \item 命令. 这是一种特殊的 \item 命令, 因为它需要能够正确描述每一条参考文献项的特性, 这种特殊的 \item 命令被命名为 \bibitem, 它有两重角色: 一是在列表中写入一项新条目, 二是定义用以引用本条目的交叉引用标签, 该标签的缺省定义是 \@biblabel{\@arabic\c@enumiv}, 效果类似于 [1] 这样, 但也可以通过使用可选参数将其修改成为类似于 [GMS94] 这样的样式, 这一点跟 \item 命令的方式完全相同. \bibitem 命令的精确定义为:

```
\def\bibitem{\@ifnextchar[\@lbibitem\@bibitem}
```

可见, 当使用可选参数时, \bibitem 调用 \@lbibitem, 否则它调用 \@bibitem. 这两个辅助性的命令定义如下:

```
\def\@lbibitem[#1]#2{\item[\@biblabel{#1}\hfill]\if@filesw
      {\let\protect\noexpand
       \immediate
       \write\@auxout{\string\bibcite{#2}{#1}}}\fi\ignorespaces}
```

为了弄清楚它的工作原理, 我们举个例子说明: 假如我们写了 \bibitem[GMS94]{companion}, 那么该命令首先创建一个具有相同可选参数的列表项, 通过使用 \@biblabel 命令, 该列表项两侧会用中括号包围, 同时 \hfill 命令又将它在行中居左对齐. 然后, 又往辅助文件 .aux[7]中写入一条带两个参数的 \bibcite 命令. \bibcite 较为简单, 定义如下:

```
\def\bibcite{\@newl@bel b}
```

\@newl@bel 命令需要三个参数 #1, #2 和 #3, 它定义了一条名称为 #1@#2 的新命令 (当然这里的 #1 和 #2 会用它们相应的取值代替), 该新命令的取值为第三个参数 #3.

这种行为与在文档中定义了一个交叉引用 (通过使用 \label 命令) 时的情形类似: 当 .aux 文件被 LaTeX (namely at the \begin{document} and \end{document}) 读入时, 就会执行这些 \@newl@bel 命令, 从而定义一条 \b@companion 命令, 在本例子中该命令就包含着文本内容: GMS93.

如果没有使用可选参数, 情形也非常类似:

```
\def\@bibitem#1{\item\if@filesw \immediate\write\@auxout
      {\string\bibcite{#1}{\the\value{\@listctr}}}\fi\ignorespaces}
```

新建一个列表项 \item, \bibcite 命令被写入到 .aux 文件中. 这里唯一的新东西是指向 enumiv 的列表计数器 \@listctr, 当在 thebibliography 定义中的 \usecounter 请求时, 该计数器的值就会被取用. 所有出现在 \bibitem (及其参数) 之后的内容都将输出到文档中最近创建的那个列表项中去, 直到出现下一条 \bibitem 命令或到达 thebibliography 环境的末尾为止[8].

我们通过一个小的文献列表的例子作为总结 (在本例中共有两条文献项, 这两条正是正文中出现的那两条):

```
\begin{thebibliography}{GMS93}   %% 最长的标签是 GMS93
```

[7]更准确地说, 是用 \@auxout 命令所指向的文件, 不过通常情况下, 它就是 .aux 文件

[8]有些宏包重新定义了 \bibitem 命令, 从而可能不再符合这种规则, 关于这方面的内容请参看第 4.4 节.

```
\bibitem[GMS93]{companion} Michel Goossens, Franck Mittelbach and Alexander
     Samarin, \emph{The \LaTeX{} Companion}, Addison Wesley, 1993.
\bibitem[Lam97]{lamport} Leslie Lamport, \emph{\LaTeX: A Document Preparation
     System}, Addison Wesley, 1997.
\end{thebibliography}
```

这是编译后的结果:

References

[GMS93] Michel Goossens, Franck Mittelbach and Alexander Samarin, *The LaTeX Companion*, Addison Wesley, 1993.

[Lam97] Leslie Lamport, *LaTeX: A Document Preparation System*, Addison Wesley, 1997.

3 \cite 命令

如果把参考文献表当作是具有交叉引用的列表, 那么 \cite 命令就等价于其中的 \ref 命令. 它有一个强制参数, 在文献项被引用时用作内部标签. 它还有一个可选参数, 可用于对文献进行补充注解, 例如对 BibTeX 引用时, 一种较好的写法是 [GMS93, 第 13 章], 它是通过录入 \cite[第 13 章]{companion} 来实现的.

这里给出它的详细定义[9]:

```
1  \DeclareRobustCommand\cite{%
2    \@ifnextchar [{\@tempswatrue\@citex}{\@tempswafalse\@citex[]}}
```

如果带有可选参数, 则布尔变量 \@tempswa 的值为 true(即记住此时带可选参数的状态和事实), 然后开始调用 \@citex 命令. 否则, 若没有使用可选参数的话, \@tempswa 的值为 false, 然后用一个空可选参数调用 \@citex 命令.

在开始详细解释 \@citex 命令之前, 我们首先快速浏览一下 \@citex 命令中要使用的 \@cite 命令, 这将有助于我们更好地理解 \@tempswa 的使用方法:

```
1  \def\@cite#1#2{[{#1\if@tempswa , #2\fi}]}
```

这是用于将文献输出到文档中的命令. 其第二个参数仅当 \@tempswa 值为 true 时才会使用, 它与第一个参数一道被放进中括号中, 并最终输出到文档中.

于是, \@citex 命令开始充当 \cite 与 \@cite 之间桥梁的角色:

```
1  \def\@citex[#1]#2{%
2    \let\@citea\@empty
3    \@cite{\@for\@citeb:=#2\do
```

这里调用了 \@cite, 当有数条文献被引用时, 该命令的第一个参数需要用 \@for 命令计算得到.

[9]关于 \DeclareRobustCommand 命令的细节参看第 7 页. 如果你对该命令完全陌生的话, 可以暂时将它看作是一个 \newcommand 命令.

```
4       {\@citea\def\@citea{,\penalty\@m\ }%
```

从 `\@for` 循环中的第二项开始, 加入一个逗号和一个换行罚分, 以保证不会在文献引用项之间换行, 缺省行为是在所有文献引用项之间都不会换行.

```
5       \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}%
```

这里重新定义了循环内部变量 `\@citeb`. 整个 `\@for` 命令顺次地将 `\@citeb` 的值设定为所有被引用值. 这里重新定义 `\@citeb` 只是为了去除额外的空白符. 这种处理看起来怪怪的, 不过很有效.

```
6       \if@filesw\immediate\write\@auxout{\string\citation{\@citeb}}\fi
```

这里在 `.aux` 文件中写入一条 `\citation` 命令, 以表明 `\@citeb` 已经在文档中被引用了. 其实这种处理在这里并不那么有用, 但在使用 BibTEX 生成参考文献表时将非常关键 (参看第 5 节).

```
7       \@ifundefined{b@\@citeb}{\mbox{\reset@font\bfseries ?}%
8         \G@refundefinedtrue
9         \@latex@warning
10          {Citation `\@citeb' on page \thepage \space undefined}}%
```

这段代码用来处理所引文献不存在时的情形, 此时被引文献会用一个加粗的问号来代替, 同时一条警告消息将回显到 `.log` 文件中.

```
11        {\@cite@ofmt{\csname b@\@citeb\endcsname}}}{#1}}
```

如果被引文献存在, 则现在使用 `\b@...` 命令将其写入, 注意到这些 `\b@...` 命令是在读取 `.aux` 文件时创建的 (参看第 5 页). `\@cite@ofmt` 命令其实等价于 `\hbox`[10]. 在对所有待引用的文献完成循环处理后, 结果传给 `\@cite` 命令, 同时还附带一个可选的第二参数, 即这里的 `#1`.

由上面描述可知, 这些处理关系似乎有点复杂, 不过不要紧, 使用起来还是挺简单的: 为了得到 [GMS93, Lam97], 你只需要键入 `\cite{companion, lamport}` 即可, 同时还会在文档中生成前一节末尾那样的参考文献表.

4 更多小技巧

4.1 `\DeclareRobustCommand` 命令是什么?

如果一条命令有一个可选参数, 例如 `\cite`, 则称该命令是*脆弱的 (fragile)*: 粗略地说, 这些命令不能直接用在其它命令的参数中 (例如, 在 `\section` 中不能使用 `\cite` 命令). 一种解决办法是在脆弱命令之前写一个 `\protect`, 不过这种写法常会令生手们迷惑不解. 另一种解决办法是将该命令声明为 "健壮的 (robust)", 即在定义时使用 `\DeclareRobustCommand` 而不是 `\newcommand`.

[10] 在较早版本 (早于 2003 年) 的 LATEX 中, `\@citex` 命令是用 `\hbox` 来定义的.

4.2 修改参考文献表的标题

从 thebibliography 环境的定义可知这种需求很容易实现: 只要重新定义 \refname, 将其缺省值从 *References* 修改为所需内容. 不过这种方法只适用于文档类 report.cls, 而文档类 book.cls 和 article.cls 则使用 \bibname, 其缺省值为 *Bibliography*.

例如, 在使用文档类 report.cls 时, 可以这样修改:

```
\renewcommand{\refname}{参考书目}
```

而在使用文档类 book.cls 或 article.cls 时, 则应为:

```
\renewcommand{\bibname}{参考书目}
```

如前文所述, 在处理页眉时, apalike.sty 不使用 \refname, 而是将文献列表标题名进行了硬编码.

4.3 在第一条参考文献表之前增加一些文本内容

仅仅将所需文本放置到 thebibliography 环境起始处的后面是行不通的, 将会发生一个错误, 这是因为在 list 环境中是需要有 \item 命令的. 于是, 我们索性放一个真正的 \item, 然后增加一些负的水平距离, 以便达到恰当的左边距值, 并将我们的文本包装在一个 minipage 环境中 (这是为了避免由于使用列表而带来的缩进):

```
\begin{thebibliography}{GMS93}
\item[]
\hskip-\leftmargin
\begin{minipage}{\textwidth}
Here are some useful references about \LaTeX. They are
available in every worthy bookshop. Many other good documentations
might be found on the web (the FAQ of \textsf{comp.text.tex} for
instance).
\end{minipage}
\bigskip
\bibitem[GMS93]{companion} Michel Goossens, Franck Mittelbach and
     Alexander
     Samarin, \emph{The \LaTeX{} Companion}, Addison Wesley, 1993.
\bibitem[Lam97]{lamport} Leslie Lamport, \emph{\LaTeX: A Document Preparation
     System}, Addison Wesley, 1997.
\end{thebibliography}
```

其效果为:

References

Here are some useful references about LaTeX. They are available in every worthy bookshop. Many other good documentations might be found on the web (the FAQ of comp.text.tex for instance).

[GMS93] Michel Goossens, Franck Mittelbach and Alexander Samarin, *The LaTeX Companion*, Addison Wesley, 1993.

[Lam97] Leslie Lamport, *LaTeX: A Document Preparation System*, Addison Wesley, 1997.

4.4 重新定义 \bibitem

有些样式文件中需要重新定义 \bibitem 命令 (或者更直接地, 修改 \@bibitem 与 \@lbibitem), 以便每个条目项以一个 \par 命令 (或一个空白行) 结束. backref.sty 就是一个这样的例子. 还有一些样式文件则将 \bibitem 命令的可选参数转换成了强制参数, 例如 apalike.sty 即是如此. 了解这些或许可以在调试时避免花费过多时间.

4.5 将方括号换成圆括号

如前文所述, \@biblabel 负责为参考文献列表中的文献项标签两侧添加中括号, 因此通过重新定义该命令就很容易得到圆括号的形式:

```
\makeatletter % @ is now a letter
\def\bibleftdelim{(}
\def\bibrightdelim{)}
\def\@biblabel#1{\bibleftdelim #1\bibrightdelim}
\makeatother % @ is a symbol
```

这里使用了一点小技巧, 以后再需要改变文献项标签两侧的内容时, 只要重新定义 \bibleftdelim 和 \bibrightdelim 即可.

不过要注意, 这并不改变 \@cite 的行为, 它仍然在被引文献标签两侧添加中括号. 因此为改变该行为, 需要重新定义 \@cite:

```
\makeatletter
\def\@cite#1#2{\bibleftdelim{#1\if@tempswatrue , #2\fi}\bibrightdelim}
\makeatother
```

4.6 更多技巧

有许多宏包可以修改参考文献表格式和引用格式, 例如: 宏包 cite.sty 修改了 \cite 命令的效果: 不仅将方括号换成了圆括号, 而且会对同时多个引文序列进行压缩与排序; 又如宏包 overcite.sty 还允许将引文放到上标的位置去.

宏包 splitbib.sty 修改了参考文献列表的输出格式: 它允许将参考文献表分割成多个分类, 并且对列表进行重新排序. 更多细节参看 [Mar05] 的文档.

4.7 内部引用标签可以使用 $ 符号吗?

回答大概是 “不能”, 但我并不确切地知道哪些字符能哪些字符不能用, 不过显然字母与数字是合法的, 这通常就足够了. 此外, 逗号, 尖括号以及反斜杠符是禁用的, 除此之外的其它字符, 我就不能确定了, 通常采用尝试的办法来确定.

结论

至此, 我们已经知道如何写参考文献表了, 似乎可以结束本手册了. 不过手工编排每一条文献既费时又易错, 而且在撰写多篇相关的文章时, 经常会引用同一批文献, 但可能采用不同的文献书写样式. 假设能将大量文献整理成一个*数据库*的形式, 则每次从中选择一部分, 并用 LaTeX 对其格式化及排版, 那将可以大大地简化工作. 这样的假设是存在的, 将在后文加以描述.

Part 2

如何使用 BibTEX?

Table of Contents

5 BibTEX 的工作机理

如前所述, 可将 BibTEX 视为一个数据库管理程序: 它从数据库中提取某些项, 对其进行排序, 并以 `thebibliography` 环境将结果导出, 以供 LATEX 处理.

这种描述有点乐观和粗略, 现实情况则要更复杂些: 为了告诉 BibTEX 哪些条目需要导出, 需要首先对你的文档运行一次 LATEX; 一旦 BibTEX 完成任务, 需要再次运行 LATEX 以便将生成的参考文献表内容一并考虑在内. 具体来说:

- 首先, 需要对文档运行 LATEX. 由于这是第一遍的编译, 没有任何文献引用信息可供利用, 因此参考文献表是空的. 每当 LATEX 遇到文档中的一个文献引用项时, 它将引用的键 (key, 即引用标签) 写入到 `.aux` 文件中[11]. 在编译期间, LATEX 也会在 `.aux` 文件中表明使用了哪个数据库, 以及使用了哪种参考文献样式来排版参考文献表的格式.

- 其次, 执行 BibTEX: 该命令的参数为 `.aux` 文件, 该文件中包含着用于导出文献条目的所有相关信息: 样式文件 (`.bst` 文件) 用来指导文献表的排版, 数据库文件 (`.bib` 文件) 指明了所引文献项的来源. 利用这些信息, BibTEX 就能有效地从数据库中导出所引文献项, 并按照指定排版样式输出到一个 `.bbl` 文件中, 操作过程中的日志信息记入 `.blg` 文件.

[11] 这正是由 `\cite` 命令中的 `\citation` 命令所完成的 (参看第 7 页).

- 第三步, 重新执行一次 LaTeX. 此时, 上一步生成 `.bbl` 文件就会被包含进来, 其中每条文献项中的 `\bibitem` 命令就会被执行. 在这一步中, 会根据需要将交叉引用信息写入到 `.aux` 中去. 不过, 由于这一步中交叉引用项并未正确地定义好, 因此生成的文档中参考文献表仍然是空的.

- 最后, 第三次运行 LaTeX: 这一次编译开始时就会读入 `.aux` 文件, LaTeX 将会将所引文献存储下来, 并在参考文献表中正确地排版列出所引文献.

由此可见, 在最佳情形下, 我们需要对文档 LaTeX 编译 3 遍, 运行 BibTeX 一遍. 有时, 仅这几遍仍然不够, 例如: 当一条文献项条目的内容中又包含了对另一条文献项条目的引用时, 就需要再增加一遍 BibTeX 和 LaTeX. 总而言之, 编译过程的总模式可以归纳成如下的正则表达式:

$$\text{LaTeX (BibTeX LaTeX)}^{+} \text{ LaTeX}.$$

至于其它过程就完全与第 1 部分相同, 这是因为 BibTeX 通常就是创建一个完整的 `thebibliography` 环境而已, 在该环境中, 包含了所有文档中所引用的文献项的 `\bibitem`. 不过, 也有两条 LaTeX 命令是新的, 它们分别用于定义文献样式文件和文献数据库文件:

- `\bibliographystyle` 命令用于声明 BibTeX 所用的文献样式文件. 其定义为:

```
\def\bibliographystyle#1{%
  \ifx\@begindocumenthook\@undefined\else
    \expandafter\AtBeginDocument
  \fi
    {\if@filesw
       \immediate\write\@auxout{\string\bibstyle{#1}}%
     \fi}}
```

  该命令简单地往 `.aux` 文件中写入一条 `\bibstyle` 命令, 其参数是文献样式文件的名称. `\bibstyle` 命令本身带有一个参数, 不过该命令没什么任务. 事实上, 只有 BibTeX 才需要文献样式文件名, LaTeX 并不需要它.

- `\bibliography` 命令用于定义所使用的文献数据库. 与前一条命令相反, `\bibliography` 命令的参数可以一个逗号分隔的文献数据库文件名列表, 这里所说的 “逗号分隔”, 不允许有空白符和换行符. 除此之外, 这条命令的行为与前一条命令的行为方式非常相似: 它也将其参数作为另一条名为 `\bibdata` 的命令 (该命令会由 BibTeX 读取, 但对 LaTeX 无什么意义) 的参数写入到 `.aux` 文件中. 最后, `\bibliography` 命令将 `.bbl` 文件包含进来用于输出参考文献表. 以下是该命令的精确定义:

```
\def\bibliography#1{%
  \if@filesw
    \immediate\write\@auxout{\string\bibdata{#1}}%
  \fi
  \@input@{\jobname.bbl}}
```

  这里正如你所猜测到的, `\jobname` 返回当前正被编译的文档的文件名.

最后再明确一点: `.aux` 文件中也包含着需要导出的条目的键 (key) 列表, 这是通过 `\citation` 命令来实现的, 该命令则是通过 `\cite` 命令回显到 `.aux` 文件中的. `\citation` 命令同 `\bibstyle` 与 `\bibdata` 这两条命令完全一样, 呵, 那就是什么也不干.

6 一些参考文献样式

由于每个出版机构都有自己独特的需求与喜好, 因此存在量大量不同的文献样式. 本节将举一些典型文献样式文件的例子.

6.1 传统的文献样式

以下四种文献格式由 Oren Patashnik 原创, 哦, 忘记交待了, 他也是 BibTEX 的作者. 这四种样式分别是 plain.bst, alpha.bst, unsrt.bst 和 abbrv.bst. 例如, 如果你想使用 plain.bst 样式, 则需要在文档 (任何位置均可) 中书写 `\bibliographystyle{plain}`.

大体上说, 文献样式要管理所有关于文献的排版事务, 可归纳为以下三方面: 一是定义可用的文献条目类型[12]及其相关的字段, 二是对所需文献的提取与排序, 三是对文献表的排版.

因此, 文献样式文件的第一种角色就是定义文献条目类型, 例如在传统文献样式中定义 **@book**, **@article** 以及 **@inproceedings** 等, 对于每种文献条目类型, 还要定义哪些字段是必需的, 哪些字段是可选的, 还有哪些字段会被忽略等等[13]. 下面的表描述了传统的文献样式中所使用的字段的角色与功能. 这里只给出简略的描述, 详细信息可在任何一本 LaTeX 的书中找到.

address	Generally the city or complete address of the publisher.
author	For author names. The input format is quite special, since BibTEX has to be able to distinguish between the first and last names. Section 11 and 18 are completely dedicated to this topic.
booktitle	For the title of a book one part of which is cited.
chapter	The number of the chapter (or any part) of a book being cited. If not a chapter, the `type` field might be used for precising the type of sectioning.
crossref	This one is quite peculiar. It's used to cross-reference within the bibliography. For instance, you might cite a document, and a part of it. In that case, the second one can reference the first one, or at least inherit some of its fields from the first one. This deserves some more comments, see section 12.
edition	The edition number. Or in fact its ordinal, for instance `edition = "First"`. This might raise problems when trying to export a bibliography into another language.
editor	The name of the editor(s) of the entry. The format is the same as for authors.
howpublished	Only used in rare cases where the document being cited is not a classical type such as a **@book**, an **@article** or an **@inproceedings** publication.

[12] 我坚持认为文献条目类型与文献中所使用的字段名应当由文献样式所确定, 而不是由 BibTEX 来固定不变.

[13] 具体规则是, 如果一个字段即不是必需的, 也不是可选的, 那么它就是被忽略的. 因此, 我们可以在文献库中加入一些注释性字段或供个人使用的字段.

institution	For a technical report, the name of the institution that published it.
journal	The name of the journal in which the cited article has been published.
key	Used for defining the label, in case it cannot be computed by BibTEX. It does not force the label, but defines the label when BibTEX needs one but can't compute it.
month	Well... The month during which the document has been published. This also raises the problem of the translation of the bibliography: It's better having a numerical value, or an abbreviation, instead of the complete name of the month. Having the number would also allow BibTEX to sort the entries more precisely (even though, as far as I know, no bibliography style does this at the present time).
note	For any additional data you would want to add. Since classical styles were written in 1985, they don't have a `url` field, and `note` is often used for this purpose, together with the url.sty package.
number	A number... Not whichever, but the number of a report. For volume numbers, a special `volume` field exists.
organization	The organizing institution of a conference.
pages	The relevant pages of the document. Useful for the reader when you cite a huge book; Note that such a precision could be added through the optional argument of `\cite` (see page 6), in which case it would appear in the document but not in the bibliography.
publisher	The institution that published the document.
school	For theses, the name of the school the thesis has been prepared in.
series	The name of a collection of series or books.
title	The title of the document being cited. There are some rules to be observed when entering this field, see section 10.
type	The type. Which type? It depends... The type of publication, if needed. For thesis, for instance, in order to distinguish between a masters thesis and a PhD. Or the type of section being cited (see `chapter` above).
volume	The volume number in a series or collection of books.
year	The publication year.

下表则列出了不同的文献条目类型, 同样更详细的信息可从 LaTeX 文档中获得.

条目类型	必需字段	可选字段
@article : An article published in a journal.	author, title, year, journal.	volume, number, pages, month, note.
@book : Well... A book.	author or editor, title, publisher, year.	volume or number, series, address, edition, month, note.
@booklet : A *small* book, that has no publisher field.	title.	author, howpublished, address, address, month, year, note.
@conference : Article that appeared in the proceedings of a conference, a meeting...	author, title, booktitle, year.	editor, volume or number, series, pages, address, month, organization, publisher, note.
@inbook : Part (generally a chapter) of a book.	author or editor, title, chapter or pages.	volume, number, series, type, address, edition, month, note.
@incollection : Part of a book having its own title.	author, title, booktitle, publisher, year.	editor, volume or number, series, type, chapter, pages, address, edition, month, note.
@inproceedings	Same as **@conference**.	
@manual : A little manual, such as this one for instance.	title.	author, organization, year, address, edition, month, note.
@mastersthesis : Masters thesis, or something equivalent.	author, title, school, year.	type, address, month, note.
@misc : When nothing else fits...	at least one of the optional fields.	author, title, howpublished, year, month, note.
@phdthesis : PhD dissertation, or similar.	author, title, school, year.	type, address, month, note.

条目类型	必需字段	可选字段
@proceedings : Conference proceedings.	`title`, `year`.	`editor`, `volume` or `number`, `series`, `address`, `month`, `organization`, `publisher`, `note`.
@techreport : Technical report, published by a laboratory, a research center, ...	`author`, `title`, `institution`, `year`.	`type`, `address`, `number`, `month`, `note`.
@unpublished : A document that has not been published. Very close to **@misc**, but `author` and `title` are needed here.	`author`, `title`, `note`.	`month`, `year`.

关于文献项内容的次序与排版格式, 这几种传统样式都非常类似. 虽然也跟具体的文献类型有关, 不过大都是先写作者名和文献标题, 然后是文献来源的期刊名称或会议录名称, ... 当然最好的理解是亲手试一试, 或者看一下相关目录下的样式文件内容 (不过如果你并不清楚 BibTEX 样式文件的细节的话, 请先阅读第 4 部分的内容).

以上都是讲四种传统样式文件的共同点, 它们之间的主要差别在于所使用的标签与排版格式方面不同.

- plain.bst 样式对文献表条目按作者名进行排序 (使用字母序[14]), 如果作者完全相同, 则按出版年份排序 (发表时间早的排在前面). 若作者时间均相同, 再比较标题. 如果还分不出先后, 那么在正文中引用在前的文献排在前面. 引用标签为数字, 从 1 开始.
- alpha.bst 样式的命名是由于它采用了字母 - 数字 (alphanumerical) 标签: BibTEX 负责为每个文献条目计算标签, 它取作者名的前三个字母 (若有多个作者, 则使用作者名的首字母), 后面紧接着是表示出版年份的两位数字. 对文献表条目排序时, 先比较标签值, 若标签值相同, 则使用与 plain.bst 相同的规则[15].
- unsrt.bst 样式也很简单: 它不对文献表条目进行特别的排序, 而按它们在正文中首次引用的次序自然排列. 除此之外, 它跟 plain.bst 的处理完全一样;
- abbrv.bst 样式则对作者名以及内置的期刊名称、月份名称都进行了缩略处理. 这里要补充一点, 文献样式最初应用于计算机科学方面的期刊 (Oren Patashnik 本人是一位计算科学家). 因此有许多计算机科学的期刊名缩写被内置到了文献样式文件中, 这是 abbrv.bst 与 plain.bst 的唯一区别.

以上即是四种传统的文献样式. 这些文献样式有许多不足, 例如, 它们都没有 `url` 字段, 也没有多语种支持, 排序机制异常复杂, 等等. 此外, 出版机构经常会针对文献表的排版提出一些具体精确的排版规则. 基于上述原因, 人们又发展了许多新的文献样式, 下面仅举几例.

[14] 标准的字母表包含 26 个字母, 不过有些语种的字母表并非如此, 例如在瑞典语中, "å" 与 "ö" 都会被视为字母, 并排在 "z" 之后.

[15] 通常当然并不希望多个文献表条目具有相同的标签, 因此在计算标签值时, 分两步完成: 第一步使用上述的标准规则求标签值, 第二步对多个相同的标签值追加一个补充字符 ("a", "b", ...). 排序过程就在这两个步骤之间完成了

6.2 部分其它文献样式

6.2.1 apalike.bst 样式

apalike.bst 的作者也是 Oren Patashnik. 它使用了一种特殊的标签格式, 即所谓的作者 - 年代格式. 以下我们用例子来说明:

在 (GMS93) 的第 384 页, 有一个使用 apalike.bst 的完整例子.

References

Goossens, M., Mittelbach, F., and Samarin, A. (1993). *the LaTeX Companion*. Addison-Wesley.

Lamport, L. (1997). *LaTeX: A Document Preparation System*. Addison-Wesley.

使用 apalike.bst 样式时, 必需包含 apalike.sty 宏包. 只要你回忆一下 `\@biblabel` 和 `\cite` 的定义, 你就知道这么做的理由了. 此外, 使用 apalike.bst 创建得到的标签可能会很长, 你通常需要允许 LaTeX 根据需要对标签进行合理断行, 但缺省的 `\cite` 命令中并不允许断行 (参看第 3 节).

另外, 还有一些其它的作者 - 年代的样式文件, 如 authordate1.bst, authordate2.bst, authordate3.bst, authordate4.bst, 它们仅与 apalike.bst 略有不同. 使用它们时, 必需配合地包含 authordate1-4.sty 宏包.

最后, 要特别注意的是: apalike.sty 重新定义了 `\bibitem` 命令, 从而将其可选参数变成了强制参数 (仍然需要放在方括号内). 不过你可能并不需要特别关心这一点, 因为文献样式文件会告诉 BibTeX, 总是要输出可选参数.

6.2.2 natbib.sty 宏包

Patrick W. Daly 写的 natbib.sty 宏包则实现得更彻底: 它通过对 `\cite` 的重新定义, 使得可以用一种更有优雅的方法来使用作者 - 年代引用标签或数字式引用标签.

传统文献样式中, 除了 alpha.bst 外 (因为它本身已经是一种作者 - 年代样式), 均已移植到 natbib.sty 中, 移植后的样式名称为 plainnat.bst, abbrvnat.bst 和 unsrtnat.bst. 此外, 这些样式均增加了 `url` 字段. natbib.sty 还可以跟 apalike.bst, authordate1.bst, ..., authordate4.bst 等配合使用.

最后提醒一下: natbib.sty 有一个非常清晰的文档 [Dal99c], 非常值得一读.

6.2.3 jurabib.sty 宏包

Jens Berger 写的宏包 jurabib.sty, 特别适合于法律学研究方面的文档. 与它关联的有一个文献样式 jurabib.bst, 其中使用 `\bibitem` 命令的可选参数定义了一种非常特别的格式. 此外, 该文献样式也重新定义了 `\cite` 命令, 补充定义了许多有意思的命令. 关于 jurabib.sty 的详细内容请参看 [Ber02].

6.2.4 custom-bib

由于可能有许多种可能的文献样式组合, Patrick W. Daly 写了一个小软件用以自动生成用户定制的文献样式, 该程序大约会提问 20 多个问题, 最后生成立即可用的文献样式文件.

与 Patrick Daly 的一贯风格一样, 其文档 [Dal99b] 写得非常好. 这里不准备展开解释, 仅就如何创建一个自定义的 `.bst` 文件说明如下: 键入 `latex makebst.tex`, 然后依提示回答问题. 随着程序的执行, 最后就生成了所需的文献样式文件, 非常直观方便.

7 问题与解答

本节用一些常见问题来结束本手册关于 LaTeX 的讨论部分, 后面我们将转入中心话题: BibTeX 部分.

7.1 如何得到多个参考文献表?

虽然不是强烈坚持, 但我还是建议每个文档中只使用一个 `\bibliography` 和 `\bibliographystyle` 命令. 那样做的话, 对于 LaTeX 倒不会有任何影响, 但对于 BibTeX 而言, 它将无法确定究竟该使用哪个文献样式或文献库.

因此, 唯一一种可能是写入多个 `.aux` 文件, 每个 `.aux` 文件对应于一个参考文献表, 包含该参考文献表中所有数据. multibib.sty, chapterbib.sty 以及 bibunit.sty 等宏包就是这种做法.

- multibib.sty 宏包的作者为 Thorsten Hansen, 该宏包可以精确化地指定参考文献表的位置, 这是通过使用不同的 `\cite` 来实现的, 每个文献表使用一种 `\cite` 命令. 对于本手册而已, 我就可以通过使用 multibib.sty 宏包来定义特殊的 `\citelatex` 和 `\citebibtex` 命令, 从而可以达到定义两个独立参考文献表的目标[16]. 当然, 这就需要为每个 `.aux` 文件运行一次 BibTeX, 更详细的内容请参看其文档 [Han00b], 这也是非常值得一读的文档. 文档中也同时指出这种方法的一个主要不足: 在不同的参考文献表中可能会为不同的文献项条目使用相同的标签, 当然你可以仔细些避免出现这种问题.

- chapterbib.sty 宏包的主要作者为 Donald Arseneau, 它为长文档中的每一章或每一部分提供了一个参考文献表 (短文章不需要参考文献表). 确实, 一篇长文档通常用若干个 `\include` 命令将各部分内容包含进来. chapterbib.sty 会为每个包含的文件创建一个 `.aux` 文件, 其中包含所需要的 `\bibstyle`, `\bibdata` 与 `\citation` 命令. 其文档可在宏包本身的末尾找到.

- bibunit.sty 宏包的作者也是 Thorsten Hansen, 其处理方式类似于 chapterbib.sty: 可以为每个 "单元"(unit) 创建一个参考文献表, 一个单元可以是文档中的任何一块, 它用 `\begin{bibunit}` 开始, 用 `\end{bibunit}` 结束.

  在每个单元内, 所有出现的 `\cite` 命令均是指当前单元内的参考文献项, 详细文献参看 [Han00a].

还有其它一些宏包也能实现多个参考文献表, 例如: camel.sty, bibtopic.sty 等, 这里不再一一细述.

7.2 如何使参考文献表离引用处更近些?

根据具体目的的不同, 可以使用两种解决办法: 第一种解决办法是将被引文献项放在脚注中, 这可能对于读者比较方便, 因为他们无需在正文与书末的文献表之间来回翻页. footbib.sty 宏包就是为此而设计的, 其详细文档见 [Dom97].

第二种解决办法是在正文中书写被引文献, 即在正文中直接书写 Michel Goossens, Franck Mittelbach et Alexander Samarin, *The LaTeX Companion*, Addison Wesley, 1993. 而不是像 [GMS93] 这样的引用. 这种处理可以增强文档, 但可能会给读者造成迷惑. 如果对这种方式想进行详细的了解, 可以参考宏包 bibentry.sty 及其文档 [Dal99a].

[16] 熟悉能产生多个索引表的 multind.sty 宏包的读者应该比较容易其中的原理.

注意, 上面提到的两个宏包很可能会跟其它一些宏包 (例如 hyperref.sty) 造成冲突, 这是因为它们都希望自己定义新的 `thebibliography` 环境或 `\bibitem` 命令, 通常最后导入的宏包会 “获胜”, 而其余宏包则会 “抱怨”. 对于这种冲突, 似乎没有什么简单的解决办法, 如果说有的话, 那就是手工合并它们各自的新定义.

7.3 如何在参考文献表中增加未被引用的文献?

通过使用 `\nocite` 可以做到这点, 该命令与 `\cite` 的工作原理完全相同, 只是不会向文档写入任何东西, 它只是在 `.aux` 文件中包含了 `\citation` 命令而已.

该命令的一个变种是 `\nocite{*}`: 它相当于是一次性地 `\nocite` 了整个文献数据库中的所有文献条目. 除了真正在正文中被引用的文献外, 其余文献将按它们在 `.bib` 文件中的书写次序依次包含进文档中. 注意 `\cite{*}` 命令也是正确的, 只不过它的实际意义实在有点...

好, 本部分至此也结束了. 下面我们将暂时不再考虑 LaTeX, 而考虑问题的更核心部分. 下一节将介绍如何创建 `.bib` 文件.

Part 3
.bib 文件

Table of Contents

.bib 文件用于存放文献数据库, 其内容一般依赖于将要使用的文献样式文件 (尽管文献样式通常是与相应的数据库兼容的). 这里我们将仅关注标准的文献样式[17]. 不过还要要提醒一下, BibTEX 还有处理文献之外的其它一些事情, 我们将在第 5 篇里看到一些这方面的例子.

8 .bib 文件的结构

我们用一个例子开始:

```
@book{companion,
  author        = "Goossens, Michel and Mittelbach, Franck and Samarin, Alexander",
  title         = "The {{\LaTeX}} {C}ompanion",
  publisher     = "Addison-Wesley",
  year          = 1993,
}
```

[17]因此将主要使用第 13–14 页里的文献项类型与字段.

现在我们只注意这个文献项条目的结构, 而忽略其内容. 其一般形式为[18]:

```
@ 条目类型 {内部键值,
  字段名 1       = " 字段值 1",
  字段名 2       = " 字段值 2",
  ...
  字段名 n       = " 字段值 n"
}
```

几点注释:

- 新条目总是以 @ 开始, 任何出现在以 @ 打头的命令的参数范围之外的内容均被当作注释, 这也给出注释掉一个条目的一种简单办法: 只要去除头部的 @ 即可. 与其它编程语言代码一样, 不要吝惜注释. 与之相反, 所有以 @ 开头的内容都会被视为是一个新的条目[19].

- BibTEX 对于文献项条目及字段名称中的大小写字符不加区分. 如果两个条目具有相同的内部键值 (不区分大小写), 则 BibTEX 就会提出警告, 例如不能为两个文献项条目的内部键值分别取为 Example 与 example.

  同样, 如果你同时引用 example 和 Example, BibTEX 也会抱怨的, 因为它会将同一条目包含两次, 这大约不是你的本意.

- 空白与换行除了影响可读性外, 不会对文献条目有任何影响. 相反, 在两个字段之间使用逗号是强制性要求.

- 字段值 (即每个等号右边的部分) 既可以用大括号也可以用双引号包围, 二者的主要差别在于如果使用大括号, 则字段值中可以使用双引号字符本身, 而后者则不行. 例如, 为了引用 Franck Mittelbach 的 *Comments on "Filenames and Fonts"*, 可以采用如下的写法:

  ```
  title          = "Comments on {"}Filenames and Fonts{"}",
  title          = {Comments on "Filenames and Fonts"},
  ```

  大括号必需配对使用, 因为它们会出现在将被 LATEX 编译的输出文档中. 如果你只需要一个大括号 (比如说左大括号) 就会出现问题, 这时, 仅仅在内容中用 \{ 代替并不行, 因为在这种情况下还是需要与其匹配的右大括号的. 正确的解决办法是使用 LATEX 的命令, 如 \leftbrace. 另一解决办法是在文献项条目中增加一个额外的 \bgroup 命令, 以便让 LATEX 和 BibTEX 都能找到合适数量的括号.

- 当字段值为数值时, 大括号与双引号均可省略.

如前所述, 你可以自定义新的字段, 例如, 关于 LATEX Companion 一书, 可以这样书写:

```
@book{companion,
  author         = "Goossens, Michel and Mittelbach, Franck and Samarin, Alexander",
  title          = "The {{\LaTeX}} {C}ompanion",
```

[18] 最外层的大括号可以用圆括号代替: @ 条目类型 (内部键值, ...).
[19] 有一种特殊的条目类型, 名为 **@comment**, 其主要用途是方便地注释掉一大块内容.

```
  booktitle     = "The {{\LaTeX}} {C}ompanion",
  publisher     = "Addison-Wesley",
  year          = 1993,
  month         = "December",
  ISBN          = "0-201-54199-8",
  library       = "Yes",
}
```

其中包含有该书的额外信息[20], 如该书是否被当地的图书馆收藏等仅供个人用的信息, 等等[21].

9 @string 与 @preamble 条目

它们并不是真正的文献条目类型: **@string** 条目可用来定义词汇缩写. 例如, 假如我们引用了 Addison-Wesley 出版社的两本书, 那么为该出版社名称定义一条缩写就会很方便, 即:

```
@string{AW      = "Addison-Wesley"}

@book{companion,
  author        = "Goossens, Michel and Mittelbach, Franck and Samarin, Alexander",
  title         = "The {{\LaTeX}} {C}ompanion",
  booktitle     = "The {{\LaTeX}} {C}ompanion",
  publisher     = AW,
  year          = 1993,
  month         = "December",
  ISBN          = "0-201-54199-8",
  library       = "Yes",
}
```

这样做不仅仅是为了节约时间, 而且可以避免拼写错误, 同时也便于对数据库进行一致性维护. 我还发现真正有用的是把作者的名字做成缩写, 这样可以保证你总是使用正确的拼写 (或者, 总是使用错误的拼写, 不过在这种情形就会增大发现错误的机会, 从而也增加了纠正的机会).

而 **@preamble** 主要用于在 BibTEX 创建的文件中插入一些命令或文本, 凡是在 **@preamble** 中声明的内容都会被拼接在一起, 统一放在一个名为 `preamble$` 的变量中, 然后就可以在文献样式中使用, 或者更常见地放置到 `.bbl` 文件的开头去 (`thebibliography` 环境之前). 这对于定义一些文献表中将要使用到的新命令而言非常有用, 以下是一个小例子:

```
@preamble{ "\makeatletter" }
@preamble{ "\@ifundefined{url}{\def\url#1{\texttt{#1}}}{}" }
@preamble{ "\makeatother" }
```

[20] 为一本书同时使用 `title` 与 `booktitle` 字段比较有趣, 后面我们在关于交叉引用部分 (第 12 节) 将会看到更详细的原因).

[21] 你可以会发问, "这些自定义字段究竟有多大用途"? 首先, 添加的这些内容都很简短, 其次我们后面将会看到, 通过设计特殊的文献样式文件, 你还可以使用这些新定义的字段.

这样, 你就可以放心地在文献项中使用 `\url` 命令了, 因为如果它没有在文献表的开头定义的话, 就会使用 **@preamble** 中的新定义.

请注意, 决不应该在文献库的 **@preamble** 中定义一些样式选项的内容, 因为那样做会对所有使用该文献库的文档都起作用..

10 title 字段

下面我们来看如何正确书写 "title" 字段. 以 *LaTeX Companion* 一书的标题为例:

```
title          = "The {{\LaTeX}} {C}ompanion"
```

在展开深入研究之前我们先来定义几个概念. *括号深度 (brace depth)* 是指围在两侧的括号的个 (对) 数, 这并不是一个非常正规的定义, 不过较易理解, 例如上例中的标题中 `\LaTeX` 的括号深度为 2, `C` 的括号深度为 1, 而其余内容的括号深度则为 0[22]. *特殊字符 (special character)* 是指字段值中的一个特殊段, 该段以括号深度为 0 的左大括号开始, 其后紧跟着一个反斜杠符, 最后以相配对的右大括号结束. 在上例中没有出现特殊字符, 注意 `\LaTeX` 的括号深度为 2, 故不属于特殊字符. 对于特殊字符, 虽然其两侧也有括号, 但应当将其括号深度视为 0.

有了这些定义, 我们可以展开讨论了. 通常文献样式会对文献的标题进行一些修正处理:

- 首先, 文献的标题可用于排序. 在排序时, BibTeX 会为每个文献条目计算一个名为 `sort.key$` 的字符串值. `sort.key$` 字符串通常是一个很长的串, 用它的值来确定当前文献条目的排列次序. 为了避免二义性, `sort.key$` 只包含字母数字字符. 除了特殊字符外, 传统的非字母非数字的字符[23]都会被 BibTeX 的一个名为 `purify$` 的函数过滤掉. 而对于特殊字符来说, `purify$` 会过滤其中的空白符和 LaTeX 命令 (即以反斜杠开头的字串), 即使它们放在大括号中也会被过滤掉. 其余的所有字符均原样不变. 例如 `t\^ete`, `t{\^e}te` 和 `t{\^{e}}te` 将会转换成 `tete`, 而 `tête` 会变成 `tête`; `Bib{\TeX}` 变成 `Bib`, `Bib\TeX` 则变成 `BibTeX`. 共有 13 个 LaTeX 命令并不遵守上述规则, 它们是: `\i`, `\j`, `\oe` `\OE`, `\ae`, `\AE`, `\aa`, `\AA`, `\o`, `\O`, `\l`, `\L`, `\ss`. 这些命令对应于 ı, ȷ, œ, Œ, æ, Æ, å, Å, ø, Ø, ł, Ł, ß, 如果它们包含在特殊字符内, 则 `purify$` 会把它们分别转换成 `i`, `j`, `oe`, `OE`, `ae`, `AE`, `aa`, `AA`, `o`, `O`, `l`, `L`, `ss`.
- 对标题的第二种转换是将它变成全部小写 (首字符除外), 函数 `change.case$` 就是干这个事的. 不过它只针对括号深度为 0 且不为特殊字符的字母. 在特殊字符内 (其括号深度总是 0), LaTeX 命令不变, 其余字符也要变成小写.

标准文献样式可能会对 `title` 字段施加这两种转换, 因此必需保证标题的书写能同时适应这两种转换处理.

好, 我们马上来实践一下, 仍以 *The LaTeX Companion* 为例, 可以采用如下几种写法:

- `title = "The \LaTeX Companion"`: 这种书写不行, 因为将其转换为小写后得到 `The \latex companion`, LaTeX 不会认识的...
- `title = "The {\LaTeX} {C}ompanion"` : 这种写法转换到小写没问题, 不过使用 `purify$` 处理后得到 `The  Companion`, 从而导致排序错误.

[22] 注意, 将包围字段值的两侧的双引号换成大括号并不改变括号深度.

[23] 例外的情况是: 连字符与波浪符会被换成空白符, 空白符本身会保留. 关于 `purify$` 精确行为的描述参看第 37 页.

- `title = "The {\csname LaTeX\endcsname} {C}ompanion"`: 这种写法也不行, 因为 LaTeX 会变成 latex.
- `title = "The { \LaTeX} {C}ompanion"` : 这种情况下, `{ \LaTeX}` 并不是特殊字符, 而是一组括号深度为 1 的字母. `change.case$` 不会对其进行任何修改. 不过 `purify$` 将会同时保留两处空白, 得到 `The  LaTeX Companion` 的结果, 从而影响了正常排序.
- `title = "The{ \LaTeX} {C}ompanion"`: 这种写法正确, 不过没有下一种写法那么优雅.
- `title = "The {{\LaTeX}} {C}ompanion"` : 这是我采用的正确写法, 它克服了上述的所有困难.

为了排版标题中字符的重音符号, 例如法语单词 *École* 中的大写字符 *É*, 我们可以写为 `{\'{E}}cole`, `{\'E}cole` 或者 `{{\'E}}cole`, 前两个将允许转换成小写, 最后一个在转换成小写时保持不变. 所有三种写法经过 `purify$` 处理后的结果相同. 这里要注意, 第三种写法中并不是一个特殊字符. 假如你用 `text.prefix$` 函数让 BibTEX 提取每个单词串的第一个字符的话, 第一种写法得到 `{\'{E}}`, 第二种得到 `{\'E}`, 而第三种得到 `{{\}}`.

关于标题就介绍至此, 下面我们将讨论更为复杂的作者名字.

11 author 字段

我们用 *LaTeX Companion* 一书的文献条目开始:

```
author  = "Goossens, Michel and Mittelbach, Franck and Samarin, Alexander"
```

有两点要注意: 一是两个作者之间要用 `and`分隔开, 二是人名的书写格式为: 先写 last name, 然后是 first name, 二者之间要用逗号分开. 除此之外, BibTEX 还认识其它一些人名的书写格式.

在深入讨论之前, 再提醒一点: 通常 BibTEX 不得不靠猜测的办法来确定哪部分是 first name, 哪部分是 last name. 更麻烦的是, 有时还需要区分人名中的 “von” 部分 (例如人名: John von Neumann) 以及可能存在的 “Jr” 部分.

下面的说明将显得更加技术化, 我们将把 first name 记作 `First`, last name 记作 `Last`, “von” 记作 `von`, 以及 “Jr” 部分记作 `Jr`.

为了正确区分 `author` 字段中人名中的各部分, BibTEX 只能识别以下三种格式的写法:

- `First von Last`;
- `von Last, First`;
- `von Last, Jr, First`.

一种写法究竟属于哪种格式, 只要数一数其中的逗号个数就行了. 以下是这几种格式的主要特征描述:

- `First von Last`: 假设你录入了 `Jean de La Fontaine`, 由于名字中无逗号, 因此可以辨认出这是 `First von Last` 格式. 除非整个字段值为空, 否则 `Last` 部分不能为空, 因此最简略也应当

写成 Fontaine. 之后, BibTEX 会检查每一个剩余单词的第一个字符[24], 如果在这些 “第一字符” 中存在着小写的字母, 那么从第一个 “小写第一字符” 到最后一个 “小写第一字符” 之间的内容均被视为 von 部分. 而所有位于 von 左边的内容为 First, 位于其右边的内容为 Last. 反之, 如果所有 “第一字符” 均不是小写, 则除了已经归类于 Last 部分之外, 所有内容均当作为 First.

在 Jean de La Fontaine 中, La Fontaine 是 Last 部分, Jean 是 First 部分, 而 de 是 von 部分.

这里给出其它几种组合, 你可以检验一下理解的程度:

Name	First	von	Last
jean de la fontaine		jean de la	fontaine
Jean de la fontaine	Jean	de la	fontaine
Jean {de} la fontaine	Jean de	la	fontaine
jean {de} {la} fontaine		jean	de la fontaine
Jean {de} {la} fontaine	Jean de la		fontaine
Jean De La Fontaine	Jean De La		Fontaine
jean De la Fontaine		jean De la	Fontaine
Jean de La Fontaine	Jean	de	La Fontaine

表中, 只有最后一行是正确的. 当然, 肯定有人要反驳说可以举出人名中的 von 部分含有大写字符的例子, 我们将在第 13 节讨论这个问题.

- von Last, First: 思路是相似的, 只是识别 First 部分更容易些: 逗号之后均为 First 部分. 而在逗号之前, 最后一个单词归于 Last(即使它以小写字母打头也是如此). 如果除此之外还存在词首小写的单词, 则从第一个词首小写的单词开始, 一直到最后一个词首小写的单词为止, 期间的部分都被视为 von 部分, 剩余未归类部分均视为 Last 部分. 还是用几个例子来说明吧:

Name	First	von	Last
jean de la fontaine,[25]		jean de la	fontaine
de la fontaine, Jean	Jean	de la	fontaine
De La Fontaine, Jean	Jean		De La Fontaine
De la Fontaine, Jean	Jean	De la	fontaine

[24] 这里及后文中, “第一个字符” 是指 “第一个括号深度为 0 的非括号字符 (一个特殊字符中的所有内容其括号深度均为 0, 哪怕它用了 15 个大括号包围也是如此).”. If there is no character at depth 0, then the item will go with its neighbour, first and foremost with the First, then with the Last. It will be in the von if, and only if, it is surrounded with two von items. Moreover, two words in the same group (in LATEX sense) will go to the same place. Last, for a LATEX command outside a special character, the backslash is removed and BibTEX considers the remaining word. If you did not understand, please take a while for reading this note anew, since it will be used in the sequel.

[25] 在这种情形下, BibTEX 会报告一个错误说人名不能用逗号结尾, 这是最经常出现的人名没有用 “and” 分隔而是用逗号分隔时才会报告的消息.

Name	First	von	Last
de La Fontaine, Jean	Jean	de	La Fontaine

- `von Last, Jr, First`: 这种情形下辨识各部分也比较简单, 只要先把两个逗号之间部分划分给 `Jr` 即可.

如上所见到的那样, 人名中的各部分通常都是用空白分隔的, 但也有例外, 如在 “Jean-François” 中, 两个 first names 之间用连字符分隔. BibTEX 会自动分隔这样的字串, 如果这两个字串 (及其连字符) 落在 `First` 的位置, 将来如果对其缩写时则写为 “J.-F.” 这样的正确形式. 在人名中, 波浪符也可以充当分隔符, 这主要是为了解决一些特殊的问题, 例如在 `Jean-baptiste Poquelin` 写法中, `baptiste` 会被视为人名的 `von` 部分, 这是因为它的词首错写成了小写字母.

现在重新回到我们先前提到的缩写上来: 你也许跟我一样感受到了人名录入的麻烦和易错特性. 因此，我个人建议为每个作者名定义一个缩写, 在作者字段中使用时, 只要将这些缩写用 ``and'' 和 #相连接起来即可.

例如, 在我的 `.bib` 文件中, 总有以下数行:

```
@string{goossens        = "Goossens, Michel"}
@string{mittelbach      = "Mittelbach, Franck"}
@string{samarin         = "Samarin, Alexander"}
```

以及:

```
@string{AW              = "Addison-Wesley"}
```

而在我的文献库中则使用[26]:

```
@book{companion,
  author        = goossens #" and "# mittelbach #" and "# samarin,
  title         = "The {{\LaTeX}} {C}ompanion",
  booktitle     = "The {{\LaTeX}} {C}ompanion",
  year          = 1993,
  publisher     = AW,
  month         = "December",
  ISBN          = "0-201-54199-8",
  library       = "Yes",
}
```

这样, 当添加一个文献条目时就非常方便, 而且使用第 23 节提到的 `bibexport.sh` 工具也很容易转换成自包含的条目格式.

[26] 这并完全真实, 因为其中的 `September` 仍然跟使用的语种有关, 因此, 我更喜欢使用 `9` 并让文献样式文件根据所使用的语种将其翻译成合适的单词.

12 交叉引用 (crossref)

正如先前所述 (都不记得哪里说过了, 呵, 对了, 在第 13 页), BibTEX 允许使用交叉引用. 这对于引用一本书的部分章节, 或者会议录中的一篇论文等场合都非常有用. 例如, 为了引用 LATEX Companion 一书的第 13 章, 可以这样书写:

```
@incollection{companion-bib,
  crossref      = "companion",
  title         = "Bibliography Generation",
  chapter       = 13,
  pages         = "371-420",
}
```

这就解释了为什么我们要为 `companion` 条目定义一个 `booktitle` 字段: 它不是为 **@book** 类型的条目使用的, 而是为了让 **@incollection** 类型的条目继承而用的. 当然我们可以手工为每个继承条目增加 `booktitle` 字段, 只是必需为所要引用的每一个章节都添加一遍.

另一个比较有意思的特征是, 当多次交叉引用同一个文献条目时, BibTEX 会自动识别, 并将该条目加入到参考文献列表中去, 然后在每个条目中直接引用它. 但另一方面, 如果一个文献条目只被交叉引用了一次, 则自动将其字段继承给交叉引用者条目, 被引用条目就不会放到参考文献列表中去. 以下是这两种行为的例子:

References

[1] Michel Goossens, Franck Mittelbach, and Alexander Samarin. Bibliography generation. In *The LATEX Companion* [GMS93], chapter 13, pages 371–420.

[2] Michel Goossens, Franck Mittelbach, and Alexander Samarin. Higher mathematics. In *The LATEX Companion* [GMS93], chapter 8, pages 215–258.

[3] Michel Goossens, Franck Mittelbach, and Alexander Samarin. Index generation. In *The LATEX Companion* [GMS93], chapter 12, pages 345–370.

[4] Michel Goossens, Franck Mittelbach, and Alexander Samarin. *The LATEX Companion*. Addison-Wesley, December 1993.

或

References

[1] Michel Goossens, Franck Mittelbach, and Alexander Samarin. Bibliography generation. In *The LaTeX Companion*, chapter 13, pages 371–420. Addison-Wesley, December 1993.

[2] Michel Goossens, Franck Mittelbach, and Alexander Samarin. Higher mathematics. In *The LaTeX Companion*, chapter 8, pages 215–258. Addison-Wesley, December 1993.

[3] Michel Goossens, Franck Mittelbach, and Alexander Samarin. Index generation. In *The LaTeX Companion*, chapter 12, pages 345–370. Addison-Wesley, December 1993.

为了精确控制这两种行为, 我们可以告诉 BibTeX 只有被交叉引用多少次后才会将该条目单列到文献表中去, 这要通过 BibTeX 的命令行参数 `-min-crossrefs` 值来确定. 对于前述第一个例子, 我使用的命令为 `bibtex biblio`, 而第二个例子中使用的命令为 `bibtex -min-crossrefs=5 biblio`.

另外一条重要原则是: 被交叉引用的条目必需放在包含相应 `crossref` 字段的条目后面. 此外, 交叉引用不能嵌套, 即不能用 `crossref` 去交叉引用一个已经包含有 `crossref` 的条目.

最后要注意, `crossref` 字段有一种特殊的行为, 即不论文献样式是什么, 它总是存在于每一个文献条目中的. 具体来说, 如果参考文献表中包含了一个被交叉引用的条目 (不管是通过 `\cite` 命令显式地引用, 还是由于被交叉引用的次数足够多而由 BibTeX 加入文献表), 那么那些对它交叉引用的条目中的 `crossref` 字段跟其原始书写一致; 除此之外, 该字段为空.

13 小技巧

13.1 如何得到人名 *Christopher* 缩写为 *Ch.* 的效果?

人名中的 first name 缩写发生在从 `author` 字段中提取人名之时. 如果需要其对 first name 进行缩写, 它将会返回人名中该部分内容各个 "单词" 的首字母缩写, 如将 *Christopher* 缩写成 *C.* 使用特殊字符是解决这一问题的办法: 若键入 `{\relax Ch}ristopher`, 则缩写为 `{\relax Ch}.`, 其结果为 *Ch.*, 而其全写版本为 `{\relax Ch}ristopher`, 即 *Christopher*.

13.2 如何实现大写的 `von` 部分?

注意: 这部分的回答偏重技术性, 为了更好地理解, 你应当阅读并理解 BibTeX 提取人名的规则 (见第 38 页和第 40 页).

由于某种原因, 有时人名的 `von` 部分是以大写字母开头, 标准的例子是 *Maria De La Cruz*.

基本解决办法是写为 `"{\uppercase{d}e La} Cruz, Maria"`. BibTeX 对这个人名进行解析时, 会将 `Cruz` 放到 `Last` 部分, 将 `Maria` 看到 `First` 部分, 而 `{\uppercase{d}e La}` 是一个特殊字符, 其首字母为 `d`, 因此会把它放到 `von` 部分.

然而, 如果此时你使用 "字母数字" 文献样式 (例如 alpha.bst) 的话, BibTeX 就会在标签中使用 `von` 部分的第一个字符[27]. 但这种情形下, 第一个字符为 `{\uppercase{d}e La}`, 因此标签取值为 `{\uppercase{d}e La}C`, 从而得到 `[De LaC]`. 你可能更希望取为 `[DLC]` 或 `[Cru]`, 于是第二种解决办法是

[27] 有人反驳说 `von` 部分不应当用于计算标签值, 但传统样式文件确实是这样做的.

```
author = {\uppercase{d}}e {\uppercase{l}}a Cruz, Maria.
```

此时, 标签值为 {\uppercase{d}}{\uppercase{l}}C, 这是正确的. 另一种更简便些的方法是

```
author = {D}e {L}a Cruz, Maria.
```

这同样可以解决问题, 这是因为 BibTEX 在确定一个单词属于人名的哪一部分时, 只考虑括号深度为 0 的内容, 而在提取第一个字符时则要考虑所有字符的内容.

13.3 如何将人名的 Last 部分变成小写?

这个问题恰好与上个问题相反, 但解决方法却不相同. 假定你要引用著名的西班牙科学家 *Juan de la Cierva y Codorníu* 的文章. 基本想法的书写是

```
author = de la Cierva {\lowercase{Y}} Codorn{\'\i}u, Juan
```

或

```
author = de la Cierva {y} Codorn{\'\i}u, Juan
```

不过, 此时会得到 CYC 或 CyC 这种形式的标签, 而我们更希望产生的标签是 CC.

几种解决办法是:

```
author = de la Cierva{ }y Cordon{\'\i}u, Juan
```

或

```
author = de la {Cierva y} Cordon{\'\i}u, Juan
```

这两种方法均可: 在第一种写法中, BibTEX 将看不到空白符, 并将 y 视为属于前一单词; 第二种写法中, Cierva y 位于括号深度为 1 的层次, 从而进入到了 Last 部分 (该部分比 von 部分具有更高的优先级).

13.4 如何去除人名中 von 与 Last 之间的空白?

这里以人名 *Jean d'Ormesson* 为例, 最好的书写方式是

```
author = "d'\relax Ormesson, Jean"
```

原因是, \relax 命令会吞掉空白, 直到遇到下一个非空字符时为止.

13.5 如何将多作者列表表示成 *et al.* 的形式?

有些特殊的文献样式会自动将多位作者人名列表转换成第一作者人名跟上一个 *et al* 的形式, 不过标准的 BibTEX 不会进行这种转换. 当然, 你也可以通过使用特殊的人名 others 来实现相同的功能, 比如

```
author = "Dupont, Jean and others"
```

在生成的文献列表中将会得到"Jean Dupont *et al.*" 这样的结果.

13.6 key 字段

实践中也会碰到文档没有作者的情况, 对于这样的文献条目, 有些文献样式 (例如 alpha.bst) 在计算 “标签” 时会使用 key 字段 (对于 alpha.bst 来说只需用其前三个字符, 但对于 apalike.bst 而言则需要使用整个字段内容). 如果没有提供 key 字段, 则会使用内部键值的前三个字符.

各位至此感觉还不错吧. 那好, 我们马上进入下一章, 也是本手册最激动人心的一部分: 创建与修改文献样式文件...

Part 4

参考文献表样式 (.bst) 文件

Table of Contents

14 什么是文献样式文件?

.bst 文件的主要功能是创建参考文献表. 有两点需要区分清楚: 一是要分清 BibTEX 与 LATEX 二者的角色, 二是在 BibTEX 内部要分清哪些工作是由文献样式文件完成的, 哪些工作是由 .bib 文件完成的.

粗略地讲, BibTEX 会读入三个文件, 依次分别为: 首先读入 .aux 文件, 从中获得了文献样式文件名, 文献数据库文件名以及所引用文献的清单; 其次是读入 .bst 文件; 最后是读入 .bib 文件.

.bst 文件告诉 BibTEX 该如何处理每一条被引文献, 即

- 文献条目类型是什么?
- 根据条目类型确定哪些字段是强制性的, 哪些是可选的?
- 最主要的, 是如何处理这些条目和字段, 以便产生清晰可被 LATEX 编译的参考文献表.

为了实现功能, BibTEX 使用了一种特殊的语言, 其中包含两种类型的指令: 一种是所谓的命令, 另一中是内部函数. 下一节先介绍命令, 再后面一节将介绍内部函数.

15 .bst 文件的结构组成

这里是一个 .bst 文件的大体结构组成:

```
ENTRY
  {  ...  }
  {  ...  }
  {  ...  }

INTEGERS {  ...  }
STRINGS {  ...  }

MACRO {  ...  }{  ...  }
FUNCTION {  ...  }{  ...  }

READ
EXECUTE {  ...  }
ITERATE {  ...  }
SORT
ITERATE {  ...  }
REVERSE {  ...  }
EXECUTE {  ...  }
```

这个例子中列出了 BibTEX 使用的所有命令. 它们的含义分别为:

ENTRY : 该命令定义了所有可能用到的字段名称列表. 更确切地说, 它有三个参数 (用大括号包围), 分别定义可能使用的 “外部” 条目名称列表 (字串类型的变量)[28], “内部” 整数变量列表, 以及 “内部” 字串变量列表. 这些变量主要供 BibTEX 在进行内部计算时使用, 例如在 alpha.bst 文献样式中建立标签时所进行的计算.

在大括号内部, 多个变量名之间用空白分开. 同时 BibTEX 对于函数和变量的名字并不区分大小写, 我在书写时, 通常将命令写成大写形式, 而将其余都表示成小写.

例如, plain.bst 样式的开头是:

[28] crossref 字段是个例外, 不用定义, 它会被 BibTEX 自动添加.

```
ENTRY
  { address
    author
    booktitle
    ...
    volume
    year
  }
  {}
  { label }
```

这里定义了传统的字段名称以及一个表示标签的变量 `label`. 事实上, plain.bst 样式中是使用数字作为标签的, 不过要注意, 由于必需评估最长标签的长度以便作为参数提供给 `thebibliography` 环境, 因此这里仍需定义 `label` 变量.

注意, 在一个文献样式文件中, `ENTRY` 必需且只能出现一次.

`INTEGERS` : 该命令声明了所用到的整数变量[29]. 命令的参数是用空白符分隔开的变量名列表.

`STRINGS` : 与 `INTEGERS` 命令相类似, 该命令声明用到的字串变量. 与整数变量相比, 字串变量更显得要 "金贵" 些, BibTEX 所允许定义这类变量的最大数目为 20 个. 因此如果打算开发较大型的样式文件时, 一定要仔细规划一下定义哪些字串变量.

`MACRO` : 该命令与前面所述的 **@string** 一样, 用于定义一些缩略语[30]. 如果一个缩略语同时在 `.bst` 文件和 `.bib` 文件中进行了定义, 那么最后只有 `.bib` 文件中的定义才有效. 另外也可以在 **@string** 的定义内容中包含 `MACRO` 的定义, 不过还是不建议这样书写.

在语法上, `MACRO` 需要两个参数, 第一个参数是要定义的缩略名称, 第二个参数则是定义内容本身, 定义内容必需是用双引用包起来的字串, 例如, 在标准的文献样式文件中都有如下的定义:

```
MACRO {jan} {"January"}
MACRO {feb} {"February"}
MACRO {mar} {"March"}
...
```

`FUNCTION` : 这是最有用的一条命令, 用于定义后续处理中需要执行的宏函数. 它的第一个参数是函数名称, 第二个参数是宏函数的定义. 后面的例子将会很多, 这里就不举例了.

`READ` : 该命令的功能是告诉 BibTEX 要将 `.bib` 文件读入. 与 `ENTRY` 命令一样, 这条命令在 `.bst` 文件中也需要且只能出现一次, 这是因为如果读入多次不会带来什么不同, 不读取则无法严重限制了样式文件处理的范围. 因此, 这条命令主要负责将那些在 `.aux` 文件中被引用的文献条目从 `.bib` 文件中提取出来. 注意 `ENTRY` 命令与 `MACRO` 命令应当放在 `READ` 命令之前, 而 `ITERATE` 命令与 `REVERSE` 命令则应当放在其后.

[29] 这些变量与任何文献条目都没有关系, 这与前面所述的 `ENTRY` 的第二个参数中的变量完全不同

[30] 这里注意, **@string** 与 `STRINGS`毫无关联.

EXECUTE : 该命令将会执行其参数指定的函数. 注意该函数必需提前定义好. 另外该命令的参数也可以是一系列 BibTEX 内部函数, 更多解释参看下文中对 **ITERATE** 的描述部分.

ITERATE : 该命令也会执行其参数指定的函数, 但与 **EXECUTE** 不同, 这里指定的函数会执行多次, 其执行次数等于用 **READ** 命令提取得到的文献条目的数目. 被执行的函数可以使用每个文献条目的字段. **EXECUTE** 命令只会执行一次其参数对应的函数, 并且该函数不能使用任何条目中的任何字段.

SORT : 该命令将按变量 **sort.key$** 的取值对所有文献条目进行排序. **sort.key$** 是一个字串型的特殊变量, 每个文献条目都隐式地声明了该变量, 其取值可以在 **ITERATE** 函数中设定, 然后所有文献条目就可以按照该变量取值的字母顺序进行重新排序了. 当然, 有些文献样式并不需要对条目进行排序, 此时文献条目会依它们在文档中出现的次序排列.

REVERSE : 与 **ITERATE** 完全类似, 只是顺序相反.

以上就是所有的命令了, 据此你可以想像一个文献样式如何有效组合这些命令达到预定目标. 这里还没有定义相关的函数 **FUNCTION**, 下两节将涉及这方面: 首先介绍 BibTEX 用到的一些记号, 然后介绍如何定义函数.

16 逆波兰语法

BibTEX 使用一种所谓的逆波兰语法, 这也是人们反映 BibTEX 语言难以理解的主要原因. 它是一种堆栈式语言: 你需要将所有参数推入堆栈, 每个函数从栈顶取走所需数目的参数, 并把处理后的结果推入栈顶. 例如, 对于加法来说, 先从栈顶取出两个元素, 求和后再将结果推入栈顶. 另一个例子[31]: `1 3 5 + 2 3 * -` 的执行过程如下:

- 将 `1` 推入栈顶. 假定堆栈初始内容为空, 则现在堆栈内包含着 `1`;
- 将 `3` 推入栈顶, 则现在堆栈内容为 `1` 和 `3`(其中 `3` 位于栈顶);
- 将 `5` 推入栈顶, 现在堆栈内容 (从底至顶) 依次为 `1`, `3` 和 `5`;
- `+` 是一个二元运算符, 它读取 (并移走) 堆栈顶部的两个元素 `5` 和 `3`, 然后对它们求和, 并将所得的结果 `8` 推入栈顶. 现在堆栈内容为 `1` 和 `8`(`8` 位于栈顶);
- 将 `2` 推入栈顶;
- 将 `3` 推入栈顶. 堆栈内容 (从底至顶) 依次为 `1`, `8`, `2` 和 `3`.
- `*` 运算符取走栈顶两个元素, 并将运算结果[32] $3 \times 2 = 6$ 推入栈. 现在堆栈内容 (从底至顶) 依次为 `1`, `8` 和 `6`.
- `-` 运算符应用于栈顶的 `6` 和 `8`. 按照 BibTEX 对减法的定义[33], 将从第二个值 `8` 中减去第一个值 `6`, 所得结果推入栈顶, 于是堆栈的当前内容为 `1` 和 `2`.

[31] 这一部分很关键, 因此我不厌其烦地举例说明, 当然如果你确实真实理解了其工作机理, 可以跳过这个例子的解释.

[32] 本例中, 我们假定 $*$ 表示乘法, 但实际在 BibTEX 中并非如此, BibTEX 中并没有定义乘法运算符, $*$ 在 BibTEX 中代表两个字串的连接.

[33] This operator is not commutative, and it could have been defined in the other way.

如果你没有理解这个例子, 请多看几遍或通过阅读其它材料, 务求弄懂其原理. 这些是 BibTEX 语言的核心, 如果本例都没有弄懂的话, 那么后面的内容可能更无法理解.

我相信大多数读者已经弄明白了, 我们现在进入最令人激动的核心部分去.

17 内部函数

下表将列出所有的内部函数. 对于每个内部函数, 都包含以下要素:

- 左边是该函数所需要的堆栈顶部内容项, 其中最右边的项位于栈顶;
- 右边是该函数处理完毕后要推入到栈顶的内容.

我们使用以下约定: $\mathcal{I}$ 代表一个整数, $\mathcal{S}$ 代表一个字串, $\mathcal{F}$ 代表一个函数, $\mathcal{N}$ 代表一个变量名字[34], $\mathcal{C}$ 代表一个在 ENTRY 中声明过的字段的名字[35], $\mathcal{E}$ 代表一个或者取整数, 或者取字串的项.

$\mathcal{I}_1$ $\mathcal{I}_2$	+	$(\mathcal{I}_1+\mathcal{I}2)$	整数加法[36];
$\mathcal{I}_1$ $\mathcal{I}_2$	-	$(\mathcal{I}_1-\mathcal{I}_2)$	整数减法;
$\mathcal{I}_1$ $\mathcal{I}_2$	>	$\mathcal{I}$	若 $\mathcal{I}_1$ 严格大于 $\mathcal{I}_2$, 则返回 1, 否则返回 0[37];
$\mathcal{I}_1$ $\mathcal{I}_2$	<	$\mathcal{I}$	若 $\mathcal{I}_2$ 严格大于 $\mathcal{I}_1$, 则返回 1, 否则返回 0;
$\mathcal{I}_1$ $\mathcal{I}_2$	=	$\mathcal{I}$	若两个整数相等, 则返回 1, 否则返回 0;
$\mathcal{S}_1$ $\mathcal{S}_2$	=	$\mathcal{I}$	若两个字串相同, 则返回 1, 否则返回 0;
$\mathcal{S}_1$ $\mathcal{S}_2$	*	$(\mathcal{S}_1\mathcal{S}_2)$	将两个字串连接成一个串[38];
$\mathcal{E}$ $\mathcal{N}$	:=		赋值运算: 假定变量 $\mathcal{N}$ 与 $\mathcal{E}$ 具有相同的数据类型, 则将 $\mathcal{E}$ 的值赋以该变量;
$\mathcal{S}$	add.period$	$\mathcal{S}$	在字串 $\mathcal{S}$ 的末尾添加一个句点, 但当 $\mathcal{S}$ 的末尾已经是句点、叹号、问号三者之一时 (这里指去除右大括号后), 就不再添加了;
	call.type$		执行一个名字为当前文献条目的类型名的函数. 例如, 对于一个类型为 **@book** 的文献条目而言, 就会执行名为 book 的函数. 显然, 该函数不能通过 EXECUTE 命令来调用, 只能用在 ITERATE 命令或 REVERSE 命令中. 这也间接说明了应该如何添加一种新的文献条目类型: 你需要简单地定义一个函数, 该函数的名字就是新文献条目类型. 如果使用了一种文献条目类型, 但在 READ 命令之前却没有定义相应的同名函数, 那么 BibTEX 就会抱怨;

[34] 一个变量名字是指在 STRINGS 或 INTEGERS 或 ENTRY 中声明过的名字. 而且在它左边必需使用一个单引号, 以便让 BibTEX 明确地理解这是在取变量的名字, 而不是取其值. 例如 'label.

[35] 它也可以是被 BibTEX 隐式声明的字段 crossref.

[36] 整数必需在其前加一个 # 号, 例如, 若希望计算 $2+5$, 则应书写为 #2 #5 +. 负数则应书写为形如 #-3 的格式.

[37] 在 BibTEX 中没有布尔型, 通常用整数类型代替: 负数和 0 表示 false, 正数表示 true.

[38] 如前文所述, 在 BibTEX 中没有乘法运算符[39].

[39] 乘法运算还是比较有用的, 我们后面会自定义一个.

$\mathcal{S}$ "t"	change.case$	$\mathcal{S}$	将 $\mathcal{S}$ 转换成小写, 但对于第一个字符以及那些括号深度大于 0 的字符不进行转换. 再次提醒特殊字符的括号深度为 0;
$\mathcal{S}$ "l"	change.case$	$\mathcal{S}$	将 $\mathcal{S}$ 转换成小写, 括号深度为大于 0 的字符除外;
$\mathcal{S}$ "u"	change.case$	$\mathcal{S}$	将 $\mathcal{S}$ 转移成大写, 括号深度为大于 0 的字符除外;
$\mathcal{S}$	chr.to.int$	$\mathcal{I}$	如果 $\mathcal{S}$ 仅包含一个字符 (指传统意义上的字符, 即不考虑特殊字符), 则返回其 ASCII 码值;
	cite$	$\mathcal{S}$	将当前文献条目的内部键值推入堆栈. 当然, 此函数只有在 ITERATE 中 REVERSE 使用才有意义;
$\mathcal{E}$	duplicate$	$\mathcal{E}$ $\mathcal{E}$	将堆栈顶部的一个元素复制一份并推入栈顶;
$\mathcal{E}$	empty$	$\mathcal{I}$	如果 $\mathcal{E}$ 为一空的字串或空字段值 (但该字段仍然是定义过的), 则将 1 推入堆栈, 否则将 0 入栈;
$\mathcal{S}_1$ $\mathcal{I}$ $\mathcal{S}_2$	format.name$	$\mathcal{S}$	提取出字串 $\mathcal{S}_1$ 中的第 $\mathcal{I}$ 个人名 (注意人名是用 and 来分隔的), 并按照字串 $\mathcal{S}_2$ 指定的格式进行格式化处理. 这里无法详细说明, 细节参看下一节;
	global.max$	$\mathcal{I}$	将所允许字串的最大长度值推入堆栈, 这个数值通常比较大 (5000 个字符). 使用该函数可以有效保证在对字串进行连接时不至于产生过长的字串;
$\mathcal{I}$ $\mathcal{F}_1$ $\mathcal{F}_2$	if$		若 $\mathcal{I}$ 取正值, 则执行 $\mathcal{F}_1$, 否则执行 $\mathcal{F}_2$;
$\mathcal{I}$	int.to.chr$	$\mathcal{S}$	按照 ASCII 表将数值 $\mathcal{I}$ 转换成对应的字符. $\mathcal{I}$ 的取值必需为 $0--127$;
$\mathcal{I}$	int.to.str$	$\mathcal{S}$	将整数值转换成等价的字串[40];
$\mathcal{C}$	missing$	$\mathcal{I}$	如果当前文献条目中定义了字段 $\mathcal{C}$, 则返回 1, 否则返回 0;
	newline$		将缓存的输出立即写入到 .bbl 文件中去, 并另起一行 (开始一个新行);
$\mathcal{S}$	num.names$	$\mathcal{I}$	返回字串 $\mathcal{S}$ 中所包含的人名的数目. 计算方法很简单: 计算一下括号深度 0 的 and(其两侧各有一个空白符) 的出现次数, 并将结果加 1 即可;
$\mathcal{E}$	pop$		将栈顶的一个元素移走;
	preamble$	$\mathcal{S}$	把 .bib 文件中所有 preamble 声明中的内容连接在一起, 并推入堆栈顶部;

[40] 学究式的科学家们可能要质疑了: "你还没有对等价关系下个严格的定义啊", 我想只用一个例子便可说明问题: 如果 $\mathcal{I}$ 的值为 147, 则其等价的字串就是 "147".

$\mathcal{S}$	purify$	$\mathcal{S}$	移除字串 $\mathcal{S}$ 中的所有非字母非数字的字符. 更精确地讲, 该函数会保留字母、数字和空格符, 将每一个制表符、连字符、波浪号都转换成空格符, 其余的标准字符 (是指出现在 TEXbook 的附录 C 中的表格内的那些字符) 都会被移除. 特殊字符有些例外: LATEX 命令、空格符、连字符、波浪号都会被移除. 只有出现在 LATEX 命令外部的数字与字母才会保留. 在实践中, 这个函数主要用于对文献条目排序时, 这时一般先对字串进行清理然后再进行比较, 这也是为什么它只保留那些 "已知" 字符的原因. 我坚持认为 "\'e" 与 "é" 是不同的, 由于第一个会被转换成 e, 第二个不会变化, 但在排序时, 第二个应当放在标准的 26 个字母之后.
	quote$	$\mathcal{S}$	将单个字符 " 推入栈顶;
	skip$		空操作 (什么也不干);
	sort.key$	$\mathcal{S}$	该函数实际上只是一个字串变量, 它被隐式地声明, 存在于每一个文献条目中, 并会被 SORT 所使用. 由此可见, 在排序之前它必需要进行适当的定义;
... $\mathcal{E}$ $\mathcal{E}$	stack$		清空堆栈内容, 并将其内容输出到标准输出设备 (而不是输出文件) 中;
$\mathcal{S}$ $\mathcal{I}_1$ $\mathcal{I}_2$	substring$	$\mathcal{S}$	从字串 $\mathcal{S}$ 中提取从位置 $\mathcal{I}_1$ 开始、长度为 $\mathcal{I}_2$ 的子串. 约定第一个字符的位置为 1. 注意这里不考虑特殊字符和大括号, 因此字串 {\LaTeX} 中从位置 2 开始、长度为 3 的子串为 \LaT;
$\mathcal{E}_1$ $\mathcal{E}_2$	swap$	$\mathcal{E}_2$ $\mathcal{E}_1$	将栈顶的两个元素互相交换位置;
$\mathcal{S}$	text.length$	$\mathcal{I}$	返回 $\mathcal{S}$ 中所含字符的个数, 其中特殊字符看作一个字符, 括号忽略.;
$\mathcal{S}$ $\mathcal{I}$	text.prefix$	$\mathcal{S}$	返回字串 $\mathcal{S}$ 的前 $\mathcal{I}$ 个字符, 仍旧将特殊字符看作一个字符, 并省略括号 (但括号会输出). 例如 {{\LaTeX}}1234 的长度为 3 的前缀为 {{\LaT}}, 但 {\LaTeX}1234 的长度为 3 的前缀则为 {\LaTeX}12;
$\mathcal{E}$	top$		将栈顶一个元素回显到标准输出设备 (不是输出文件) 中, 并将其从堆栈中移除;
	type$	$\mathcal{S}$	将当前文献条目的类型推入堆栈;
$\mathcal{S}$	warning$		将字串 $\mathcal{S}$ 作为警告消息写入到标准输出设备中;
$\mathcal{F}_1$ $\mathcal{F}_2$	while$		只要 $\mathcal{F}_1$ 返回正值就连续执行 $\mathcal{F}_1$ 和 $\mathcal{F}_2$, 其中 $\mathcal{F}_1$ 必需返回一个整数值;

$\mathcal{S}$ width$ $\mathcal{I}$	返回字串 $\mathcal{S}$ 的宽度, 度量标准是当使用 *cmr*10 字体 (1987 年 6 月版) 输出字串时所占用的宽度值, 单位是 0.01 磅 (point). 你不必太关心其细节, 该函数主要用于比较标签的宽度值, 其中的标签值会作为参数传递给 thebibliography 环境;
$\mathcal{S}$ write$	将字串 $\mathcal{S}$ 写入到 .bbl 文件.

好了, 以上这些内部函数足以用来定义大量的新函数, 从而完成对文献条目的管理、排序、计算标签、排版等处理. 在正式使用它们之前, 我再详细解释一下 format.name$ 的用法.

18 format.name$ 函数

这个函数比较复杂, 其语法如下:

$\mathcal{S}_1$ $\mathcal{I}$ $\mathcal{S}_2$ format.name$ $\mathcal{S}$

第一个参数 $\mathcal{S}_1$ 通常是字段 author 或 editor 的内容, 是一个用 and 分割的人名列表. 只要人名采用了 BibTEX 所能识别的三种格式 (参看第 11 节) 之一, 那么 BibTEX 就能有效地将人名解析成 last name, first name, "von" 和 "complement" 四个部分. 字串 $\mathcal{S}_2$ 用于定义输出格式. 这里是一个示例:

```
"{ff }{vv }{ll}{, jj}"
```

这里的格式指定描述如下:

- 从整体上看, 它是由一系列字串组成的, 每个字串两侧为大括号;
- ff 代表了人名的 First 部分, 即 first name; ll 代表了人名的 Last 部分, 即 last name, 而 vv 和 jj 则分别代表了人名的 von 部分和 jr 部分;
- format.name$ 会把 $\mathcal{S}$ 中所包含的每个字串中的 ff, ll, vv 和 jj 等部分用相应的取值进行替换. 如果某个取值为空串, 则它所属于的整个字串部分就移除不要了; 否则就会连同其余字面内容一同输出. 例如, 在上述例子中, 若 jr 部分的取值为空, 则逗号及其后的空格也不会输出; 又如: 若 First 部分取值不为空, 则会在其后附着一个空格符.

通过上述介绍, 关于这个函数的用法基本就清楚了. 另外, 整数 $\mathcal{I}$ 表明我们正在处理的是 $\mathcal{S}_1$ 中的第 $\mathcal{I}$ 个人名.

最后要提到的是, 人名还有各种缩略形式, 比如为了只取 first name 的首字母, 我们只要将格式串 {ff } 换成 {f } 即可. 如果在一个人名的某个部分 (如 first name 部分) 中仍然包含着若干个子名字 (例如多个 first name), 则缩略的输出格式是: 取每个子名字的首字母, 每个首字母后面会跟一个句点. 不同的子名字之间必需由空格符、制表符、波浪号或连字符分隔开来, 制表符会转换成空格符, 而波浪号与连字符则原样保留. 注意最后一个子名字缩写后没有句点, 如果需要必需通过 {f. } 来手工指定. 例如:

```
"Goossens, Michel and Mittelbach, Franck and Samarin, Alexander"
#2 "{f. }{vv }{ll}{, jj}" format.name$
```

结果为 *F. Mittelbach.* 再来看几个有趣的例子:

```
"Charles Jean Gustave Nicolas de La Vall{\'e}e Poussin"
#1 "{vv }{ll}{, f}" format.name$
```

产生的结果为 *de La Vallée Poussin, C.J.G.N*, 其中末尾没有句点 (我们没有手工指定).

```
"Doppler, {\relax Ch}ristian Andreas"
#1 "{f. }{vv }{ll}" format.name$
```

结果为 *Ch. A. Doppler.* 又如

```
"Jean-Baptiste Poquelin" #1 "{f. }{vv }{ll}" format.name$
```

输出为 *J.-B. Poquelin.*

似乎还有一个问题没有解决: 为了从作者人名中抽取文献条目的标签, 我们自然也想到应该使用 `format.name$`. 比如一条文献是 La Vallée Poussin 写的一本书, 我们希望使用 `LVP` 作为标签. 第一种能想到的写法是:

```
"Charles Jean Gustave Nicolas de La Vall{\'e}e Poussin"
#1 "{l}" format.name$
```

但很不幸, 其输出为 *L. V. P*, 用它来作标签可不怎么漂亮. 实际上, 子名字之间的句点是缺省取值, 完全可以通过显式指定来更换它. 在本例中, 我们就希望有空串来替换掉它:

```
"Charles Jean Gustave Nicolas de La Vall{\'e}e Poussin"
#1 "{l{}}" format.name$
```

这下可以得到期望的结果 *LVP* 了. 跟在 `l` 后面的一对括号意思就是要在多个子名字 (或其首字母缩写) 之间放上空串. 除此之外, 还可以为子名字左侧或右侧指定其它的文本内容, 其中空格可以直接录入, 而文本与命令则要放在大括号内, 以便让 BibTEX 可以将它们与 `ff`, `ll`, ... 等格式串相区别开. 一个例子胜过千言万语:

```
"Charles Jean Gustave Nicolas de La Vall{\'e}e Poussin"
#1 "{{\scshape\bgroup}ff{ }{\egroup}" format.name$
```

输出为 `{\scshape\bgroup}Charles Jean Gustave Nicolas{\egroup}`, 进而输出 « Charles Jean Gustave Nicolas ». 注意这里用了点小伎俩, 将左大括号用 `\bgroup` 表示, 以便让 BibTeX 与 LaTeX 正确处理, 并要将其以字面输出[41].

好, 实在应用打住了, 因为你已经知道了所有的内容, 起码理论上是如此. 不过, 下一节还是要介绍一些例子与技巧.

[41] 译者注: 这段译得实在别扭, 以下为原文: Note that we had to trick BibTeX and LaTeX regarding the `\bgroup`: braces, which are necessary in BibTeX name specifications, are copied verbatim in the output...

19 一些实践性技巧

实际上, 为了测试 BibTEX 的各种行为, 比如测试各种人名的格式, 最主要的技巧莫过于如何构建一个最小的样式文件了. 这个最小的样式文件只是将其结果输出到标准输出设备上. 以下假定希望测试 BibTEX 是如何处理下面的人名的:

```
de la Cierva y Codorn{\'\i}u, Juan
```

最短的 .bst 文件也必需包含 ENTRY和 READ命令, 哪怕我们在其中根本就用不到任何文献条目时也是如此. 当然, 也还需要有计算输出的函数. 综合这两点, 我们最短的样式文件 min.bst 内容如下:

```
ENTRY {}{}{}

FUNCTION {test}
{"de la Cierva y Codorn{\'\i}u, Juan"
#1 "{ff - }{vv - }{ll}" format.name$ top$}

READ

EXECUTE{test}
```

该样式文件效果是将人名分隔成不同的部分, 期间用连字符分开, 并将结果输出.

现在 .aux 的内容也很简单, 它只定义了所要用到的文献样式文件:

```
\bibstyle{min}
```

在这个名为 min.aux 的文件中, 既没有用于指定文献数据库 .bib 的 \bibdata 命令, 也没有用来声明文献条目的 \citation 命令. BibTEX 对此当然会有抱怨, 不过仍然会顺利运行.

最终结果如下:

```
% bibtex min
This is BibTeX, Version 0.99c (Web2C 7.4.5)
The top-level auxiliary file: min.aux
The style file: min.bst
I found no \citation commands---while reading file min.aux
I found no \bibdata command---while reading file min.aux
Warning--I didn't find any fields--line 1 of file min.bst
Juan - de~la Cierva~y - Codorn{\'\i}u
(There were 2 error messages)
```

其中既有错误消息, 也有输出结果, 输出结果表明我们使用的人名格式指定不正确 (因为 von 部分应当只包含 de la). 至于不间断空白符 ~, 那是 BibTEX 自动加上的.

通过本例, 你就明白了如何测试自己的 BibTEX 函数. 由于 BibTEX 的出错消息难以理解, 因此这个实践过程更显得有价值. 你同样可以测试一下我先前提到的函数 purify$ 的行为.

20 部分简单函数实例

20.1 布尔函数

如前所述, BibTEX 没有布尔类型. 特别地, 没有与、或、非等这些非常有用的布尔运算. 这些运算可以定义为如图 1 的函数.

```
FUNCTION {not}
{   { #0 }
    { #1 }
  if$
}

FUNCTION {and}
{   'skip$
    { pop$ #0 }
  if$
}

FUNCTION {or}
{   { pop$ #1 }
    'skip$
  if$
}
```

Figure 1: 实现基本布尔运算的函数

这里只对前两个进行解释: 对于“非”运算, 先假定当前的栈顶有一个布尔值 (即一个整数). 首先, 函数 `if$` 对这个布尔值进行测试, 如果测试结果为 `true`, 则该函数返回 `#0`(即 `false`), 否则它会返回 `#1`(即 `true`). 这正是我们希望的语义.

对于“与”运算: 假定堆栈中有两个布尔值. 函数 `if$` 先对第一个进行测试, 若其值为 `true`, 则什么也不干 (相当于将第二个布尔值留在栈顶, 作为“与”运算的最终结果). 若第一个值为 `false`, 则“与”运算的最终结果也应为 `false`, 于是将第二个参数移除掉, 然后将 `#0` 推入堆栈.

20.2 乘法

图 2 则是计算两个整数乘法的函数. 我不想涉及过多细节, 代码中的注释已经足够清楚了. 同样你还可以定义除法、求两个整数的最大公约数、素数测试等函数. 不过我无法确定那些是否真的有用.

```
INTEGERS { a b }

FUNCTION {mult}
{
 'a :=                          %% 保存第一个值
 'b :=                          %% 保存第二个值

 b #0 <                         %% 先记住 b 的符号,
    {#-1 #0 b - 'b :=}          %% 然后再考虑其绝对值.
    {#1}                        %%
 if$                            %%

 #0                             %% 将 0 推入堆栈.
 {b #0 >}                       %% 当 b 为正数时,
 {                              %% 把 a 值累加到栈顶元素中
   a +                          %% 并将 b 的值减 1.
   b #1 - 'b :=                 %%
 }                              %%
 while$                         %%

 swap$                          %% 最后, 若 b 为负数
   'skip$                       %% 则取其相反数.
   {#0 swap$ -}                 %%
 if$                            %%
}
```

Figure 2: 两个整数的乘法函数

20.3 将字符串转换成整数

没有一种可将一个字串 (由数字组成) 转换成相应的整数的直接方法, 我们也定义一个函数来实现, 并且当该字串不是一个数字时, 返回一条错误消息. 实现中采用了递归方法, 如图 3 所示.

本例中有几个有趣的特点: 首先, 它需要使用布尔函数和乘法函数, 因此这些函数必需提前定义好, 放在本函数之前. 其次, 函数 `chr.to.value` 还说明了设计函数的一条重要原则: 一条函数必需在任何时间都要行为一致, 即使用相同数目和相同类型的输入参数, 使用相同数目和相同类型的输出. 例如本例中, 当输入参数中的一个字符不是整数时, 虽然会抱怨, 但我们还是返回了一个整数 (就好像没有任何错误一样). 除非你刻意为了调试程序, 否则遵循这条规则很重要.

```
FUNCTION {chr.to.value}       %% 一个字符的 ASCII 码值
{
  chr.to.int$ #48 -           %% 字符 "0" 的 ASCII 码值 = 48
  duplicate$ duplicate$       %% 字符 "1" 的 ASCII 码值 = 49
  #0 < swap$ #9 > or          %% ...
  {                           %% 字符 "9" 的 ASCII 码值 = 57
      #48 + int.to.chr$
      " is not a number..." *
      warning$                %% 若它不是一个数字, 则返回 0.
      pop$ #0                 %%
    }
  {}
  if$
}

FUNCTION {str.to.int.aux}     %% 辅助函数
{
  {duplicate$ empty$ not}     %% 当字串不为空时
    {                         %% 考虑其第一个字符
      swap$ #10 mult 'a :=    %% 并将它``累加'' 到结果的末尾.
      duplicate$ #1 #1 substring$   %%
      chr.to.value a +
      swap$
      #2 global.max$ substring$
    }
  while$
  pop$
}

FUNCTION {str.to.int}
{                             %% 处理负值
  duplicate$ #1 #1 substring$ "-" =
    {#1 swap$ #2 global.max$ substring$}
    {#0 swap$}
  if$
                              %% 初始化, 并开始调用
  #0 swap$ str.to.int.aux     %% 辅助函数.
  swap$
    {#0 swap$ -}              %% 处理符号.
    {}
  if$
}
```


Figure 2: 将字符串转换成整数

20.4 对字符串中的字符进行计数

前面我们在第 37 页看到, `text.length$` 会对一个字串中的字符进行计数, 但这里所说的 “字符” 是 BibTEX 所理解意义下的字符. 事实上, 只有 `substring$` 这一个内部函数使用的是常规意义下标准字符概念. 现在我们利用 `substring$` 函数, 反复计算给定字串的连续前缀, 直到某个前缀串等于原始串本身时为止, 其中重复的次数就是给定字串中所有字符的个数.

```
INTEGERS{ l }
FUNCTION{ string.length }
{
  #1 'l :=
  {duplicate$ duplicate$ #1 l substring$ = not}
    {l #1 + 'l :=}
  while$
  pop$ l
}
```

Figure 4: 计算字符串中所含字符真实数目的函数

作为一个练习, 请读者自己尝试实现另一种算法: 依次移除字串的首字符, 直到所剩余的串为空时为止.

20.5 “查找与替换” 函数

利用上面定义这些函数, 现在可以设计一个用于在给定文献中查找与替换某个字串的算法了. 再次重申, 我们不能使用 `text.length$` 和 `text.prefix$`, 因为它们使用的字符有特殊含义. 我们只使用 `string.length` 和 `substring$` 函数. 算法思想是相继地提取文本的前 n 个字符, 将其与模式进行比较, 然后根据比较的结果决定是否对其进行替换.

该函数会将文本中出现的所有模式都进行替换, 但很容易将其修改成只对第一次出现的模式进行替换. 作为一个练习, 读者可以尝试只使用两个字串变量实现同样的算法[42] (尽可能少占用字串变量, 因为它们的总数非常有限).

[42] 提示: 尝试将 `find` 放到堆栈中去.

```
STRINGS{replace find text}
INTEGERS{find_length}
FUNCTION{find.replace}
{ 'replace :=
  'find :=
  'text :=
  find string.length 'find_length :=
  ""
    { text empty$ not }
    { text #1 find_length substring$ find =
        {
          replace *
          text #1 find_length + global.max$ substring$ 'text :=
        }
        { text #1 #1 substring$ *
          text #2 global.max$ substring$ 'text :=
        }
      if$
    }
  while$
}
```

Figure 5: 在 BibTEX 中进行查找与替换

Part 5

BibTEX 的其它用途

本部分是对 BibTEX 的一些扩展, 主要介绍一下如何使用 BibTEX 完成一些其他任务. 我本人也正在开发一些这样的扩展, 最新消息可参考我的个人主页:

http://www.lsv.ens-cachan.fr/~markey/bibla.php.

敬请提出您的宝贵意见或思路.

21 发表论著清单

这其实不是一个真正的扩展, 因为毕竟它还是用 BibTEX 来处理文献信息的. 在本例中, 为了提取某一个人的论著, 我们仍使用传统的 `.bib` 文件, 但在结果中会标示出其余的合作者人名. 例如:

Books by Michel Goossens

[1] *the LaTeX Companion*. Addison-Wesley, 1993. Joint work with Frank Mittelbach and Alexander Samarin.

[2] *the LaTeX Graphics Companion*. Addison-Wesley, 1999. Joint work with Sebastian Rahtz and Frank Mittelbach.

[3] *the LaTeX Web Companion*. Addison-Wesley, 1999. Joint work with Sebastian Rahtz.

这里有三点值得注意:

- 被引文献条目是通过文献样式文件提取得到的;
- 人名的排版方面有些差别.

第三点差异是关于文献样式文献的配置机制方面的, 它不是靠用户修改 `.bst` 文件完成的, 而是增加一种文献条目类型 **@config**, 一个 **@config** 条目中就指定要提取的论著的作者名字, 以及其它一些配置变量的值.

22 通讯录

一系列的地址信息其实就构成了一个数据库, 对它可排序、分组、排版等处理, 所以 BibTEX 应该能做到这些的.

为此, 必需对字段, 条目类型, 函数等进行重新定义. 字段主要包括 names, addresses, phone numbers, e-mail addresses, ... 以下是输出结果的一个例子:

MARKEY Nicolas	LSV – ENS Cachan 61, avenue du Président Wilson 94235 CACHAN Cédex markey@lsv.ens-cachan.fr	☎ 🖷	01.47.40.75.32 01.47.40.24.64

与前一个例子一样, 为了定义将字符分组功能, 我们定义了一个 **@config** 的条目类型...

23 文献表的导出、提取与整理

有时我们还需要从一个较大的文献库中抽取部分记录, 比如为了把 LaTeX 文档及其相关的文献发送给合作者时就需要这么做. 目前已经有好几种工具来完成这种工作, 不过它们大多不对正确地支持交叉引用 (crossref)、宏以及字串缩写等. 实际上, BibTeX 本身就能完成这种事: 你只需将文献表的输出格式指定为一个 `.bib` 数据库格式即可 (当然, 由于无法告诉 BibTeX 写入文件的名字, 输出结果是一个 `.bbl` 文件). 这种方法有几个好处: 首先它能保证所得到的字段正确无误; 其次它能将缩略写法自动展开; 第三, 它可以很方便地导出某一个指定作者的所有文献 (要知道人名在 `.bib` 数据库中可能有各种各样的编码方式); 等等. 宏包 `bibexport` 中就实现了这种技术, 见: http://www.ctan.org/tex-archive/biblio/bibtex/contrib/bibexport/

24 在一个参考文献表中使用多种语言

传统的文献样式只能处理一种语言 (通常是英语). 如果混合处理多种语言, 则需要使用点技巧: 首先, 在写入 `.bib` 文件时, 必需保证写入的内容都是与具体语种无关的. 例如, 月份要写成数字式的; 又如, BibTeX 在处理每个条目时, 要添加的一些单词, 如最后一个人名前的 "and" 等等.

为了能在一个参考文献表中使用多种语言, 有几种解决办法:

- 为每种语种设计一个样式文件. 这种方案有点麻烦, 但也并不复杂, 目前就有不少这样的翻译版本存在.

  法语版的样式文件可以参看 http://www.ctan.org/tex-archive/biblio/bibtex/contrib/bib-fr/. 在这些样式文件中, 我把每个翻译过的单词放在文件头部, 这样可为其它语种的翻译者提供一些方便.

- 另外一种方案是用 LaTeX 命令把那些与语种相关的单词替换掉. 这方面的例子有 Harald Harders 开发的 babelbib (http://www.ctan.org/tex-archive/biblio/bibtex/contrib/babelbib/). 这些样式文件都需要使用 babel.sty 宏包, 利用它们可以为每一个文献条目定义一种语言 (从而也可以使用恰当的分词模式).

25 词汇术语表

词汇术语表是将文档中出现的科技术语组织到一起形成的词典, 这通常是通过的 LaTeX 的索引机制来实现的. 不过也可以用 BibTeX 来做这件事 (由于它还能从一个更大的词典中提取部分定义, 因此有时它甚至比 `\index` 做得更好些). 此外, 还可以借助 backref.sty 宏包为术语添加其在正文中出现的位置 (页码) 信息. 这方面例子可以参看 Jose Luis Diaz de Arriba 和 Javier Bezos 开发的 gloss.sty 宏包.

References

[Ber02] Jens Berger. *The **jurabib** package*, September 2002.

[Dal99a] Patrick W. Daly. *A LaTeX Package to Place Bibliography Entries in Text*, February 1999.

[Dal99b] Patrick W. Daly. *A Master Bibliographic Style File*, May 1999.

[Dal99c] Patrick W. Daly. *Natural Sciences Citations and References*, May 1999.

[Dom97] Éric Dommenjoud. *The **footbib** package*, March 1997.

[GMS93] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, December 1993.

[Han00a] Thorsten Hansen. *The **bibunits** package*, October 2000.

[Han00b] Thorsten Hansen. *The **multibib** package*, January 2000.

[Lam97] Leslie Lamport. *LaTeX: a Document Preparation System*. Addison-Wesley, March 1997.

[Mar05] Nicolas Markey. *The **splitbib** package*, February 2005.

[MGB+04] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, and Chris Rowley. *The LaTeX Companion — Second Edition*. Addison-Wesley, April 2004.

[Pat88a] Oren Patashnik. BibTeXing, February 1988.

[Pat88b] Oren Patashnik. *Designing* BibTeX *styles*, February 1988.

LaTeX

– R –

– S –

– T –

– U –

BibTEX

– Symbols –

– A –

– B –

– C –

– D –

– E –

– F –

– G –

– H –

– I –

– J –

– K –

– M –

– N –

– O –