

GB/T 7714 BibTeX style

Zeping Lee*

2025/03/11 v2.1.7

摘要

The gbt7714 package provides a BibTeX implementation for the China's national bibliography style standard GB/T 7714. It consists of .bst files for numeric and author-date styles as well as a LaTeX package which provides the citation style defined in the standard. It is compatible with natbib and supports language detection (Chinese and English) for each bibliography entry.

1 简介

GB/T 7714—2015 《信息与文献 参考文献著录规则》^[1]（以下简称“国标”）是中国的参考文献格式推荐标准。国内的绝大部分学术期刊、学位论文都使用了基于该标准的格式。本宏包是国标的 BibTeX^[2] 实现，具有以下特性：

- 兼容 natbib 宏包^[3]。
- 支持“顺序编码制”和“著者-出版年制”两种风格。
- 自动识别语言并进行相应处理。
- 提供了简单的接口供用户修改样式。
- 同时提供了 2005 版的 .bst 文件。

本宏包的主页：<https://github.com/zepinglee/gbt7714-bibtex-style>。

2 版本 v2.0 的重要修改

从 v2.0 版本开始（2020-03-04），用户必须在文档中使用 `\bibliographystyle` 命令选择参考文献样式，如 `gbt7714-numerical` 或 `gbt7714-author-year`。在早期的版本中，选择文献样式的方法是将 `numbers` 或 `super` 等参数传递给 `gbt7714`，而不能使用 `\bibliographystyle`。这跟标准的 LaTeX 接口不一致，所以将被弃用。

*zepinglee AT gmail.com

3 使用方法

以下是 `gbt7714` 宏包的一个简单示例。

```
\documentclass{ctexart}
\usepackage{gbt7714}
\bibliographystyle{gbt7714-numerical}
\begin{document}
  \cite{...}
  ...
  \bibliography{bibfile}
\end{document}
```

按照国标的规定，参考文献的标注体系分为“顺序编码制”和“著者-出版年制”。用户应在导言区调用宏包 `gbt7714`，并且使用 `\bibliographystyle` 命令选择参考文献表的样式，比如：

```
\bibliographystyle{gbt7714-numerical} % 顺序编码制
```

或者

```
\bibliographystyle{gbt7714-author-year} % 著者-出版年制
```

此外还可以使用 2005 版的格式 `gbt7714-2005-numerical` 和 `gbt7714-2005-author-year`。

注意，版本 v2.0 更改了设置参考文献表样式的方法，要求直接使用 `\bibliographystyle`，不再使用宏包的参数，而且更改了 `bst` 的文件名。

```
\citestyle \citestyle{<citation style>}
```

可选：`super`, `numbers`, `author-year`。使用 `\bibliography` 选择参考文献表的样式时会自动设置对应的引用样式。顺序编码制的引用标注默认使用角标式 (`super`)，如“张三^[2]提出”。如果要使用正文模式，如“文献 [3] 中说明”，可以使用 `\citestyle` 命令切换为数字式 (`numbers`)。

```
\citestyle{numbers}
```

著者-出版年制通常不需要修改引用样式。

`sort&compress` 同一处引用多篇文献时，应当将各篇文献的 `key` 一同写在 `\cite` 命令中。如遇连续编号，默认会自动转为起讫序号并用短横线连接（见 `natbib` 的 `compress` 选项）。如果要对引用的编号进行自动排序，需要在调用 `gbt7714` 时加 `sort&compress` 参数，这些参数会传给 `natbib` 处理。

```
\usepackage[sort&compress]{gbt7714}
```

注意国标中要求 2 个或以上的连续编号用连接号，不同于 `natbib` 默认的 3 个或以上。宏包中已经作了修改。

若需要标出引文的页码，可以标在 `\cite` 的可选参数中，如 `\cite[42]{knuth84}`。更多的引用标注方法可以参考 `natbib` 宏包的使用说明^[3]。

使用时需要注意以下几点：

- `.bib` 数据库应使用 UTF-8 编码。
- 使用著者-出版年制参考文献表时，中文的文献必须在 `key` 域填写作者姓名的拼音，才能按照拼音排序，详见第 6 节。

4 文献类型

国标中规定了 16 种参考文献类型，表 1 列举了 `bib` 数据库中对应的文献类型。这些尽可能兼容 `BibTeX` 和 `biblatex` 的标准类型，但是新增了若干文献类型（带 * 号）。

表 1: 全部文献类型

文献类型	标识代码	Entry Type
普通图书	M	book
图书的析出文献	M	incollection
会议录	C	proceedings
会议录的析出文献	C	inproceedings 或 conference
汇编	G	collection*
报纸	N	newspaper*
期刊的析出文献	J	article
学位论文	D	mastersthesis 或 phdthesis
报告	R	techreport
标准	S	standard*
专利	P	patent*
数据库	DB	database*
计算机程序	CP	software*
电子公告	EB	online*
档案	A	archive*
舆图	CM	map*
数据集	DS	dataset*
其他	Z	misc

5 著录项目

由于国标中规定的著录项目多于 Bib_T_EX 的标准域，必须新增一些著录项目（带 * 号），这些新增的类型在设计时参考了 BibLaTeX，如 `date` 和 `urldate`。本宏包支持的全部域如下：

author 主要责任者
title 题名
mark* 文献类型标识
medium* 载体类型标识
translator* 译者
editor 编辑
organization 组织（用于会议）
booktitle 图书题名
series 系列
journal 期刊题名
edition 版本
address 出版地
publisher 出版者
school 学校（用于 @phdthesis）
institution 机构（用于 @techreport）
year 出版年
volume 卷
number 期（或者专利号）
pages 引文页码
date* 更新或修改日期
urldate* 引用日期
url 获取和访问路径
doi 数字对象唯一标识符
langid* 语言
key 拼音（用于排序）

不支持的 Bib_T_EX 标准著录项目有 `annotate`, `chapter`, `crossref`, `month`, `type`。

本宏包默认情况下可以自动识别文献语言，并自动处理文献类型和载体类型标识，但是在少数情况下需要用户手动指定，如：

```
@misc{citekey,  
  langid = {japanese},  
  mark   = {Z},
```

```
medium = {DK},
...
}
```

可选的语言有 `english, chinese, japanese, russian`。

6 文献列表的排序

国标规定参考文献表采用著者-出版年制组织时，各篇文献首先按文种集中，然后按著者字顺和出版年排列；中文文献可以按著者汉语拼音字顺排列，也可以按著者的笔画笔顺排列。然而由于 Bib_TE_X 功能的局限性，无法自动获取著者姓名的拼音或笔画笔顺，所以必须在 `bib` 数据库中的 `key` 域手动录入著者姓名的拼音用于排序，如：

```
@book{capital,
  author = {马克思 and 恩格斯},
  key    = {ma3 ke4 si1 & en1 ge2 si1},
  ...
}
```

对于著者-出版年的样式，如果中文文献较多时更推荐使用 `biblatex` 宏包，其后端 `biber` 可以自动处理中文按照拼音排序，无须手动填写拼音。

7 自定义样式

Bib_TE_X 对自定义样式的支持比较有限，所以用户只能通过修改 `bst` 文件来修改文献列表的格式。本宏包提供了一些接口供用户更方便地修改。

在 `bst` 文件开始处的 `load.config` 函数中，有一组配置参数用来控制样式，表 2 列出了每一项的默认值和功能。若变量被设为 `#1` 则表示该项被启用，设为 `#0` 则不启用。默认的值是严格遵循国标的配置。

若用户需要定制更多内容，可以学习 `bst` 文件的语法并修改^[4-6]，或者联系作者。

8 相关工作

TeX 社区也有其他关于 GB/T 7714 系列参考文献标准的工作。2005 年吴凯^[7]发布了基于 GB/T 7714—2005 的 Bib_TE_X 样式，支持顺序编码制和著者出版年制两种风格。李志奇^[8]发布了严格遵循 GB/T 7714—2005 的 BibLaTeX 的样式。胡海星^[9]提供

表 2: 参考文献表样式的配置参数

参数值	默认值	功能
uppercase.name	#1	将著者姓名转为大写
max.num.authors	#3	输出著者的最多数量
year.after.author	#0	年份置于著者之后
period.after.author	#0	著者和年份之间使用句点连接
italic.book.title	#0	西文书籍名使用斜体
sentence.case.title	#1	将西文的题名转为 sentence case
link.title	#0	在题名上添加 url 的超链接
title.in.journal	#1	期刊是否显示标题
show.patent.country	#0	专利题名是否含国别
space.before.mark	#0	文献类型标识前是否有空格
show.mark	#1	显示文献类型标识
show.medium.type	#1	显示载体类型标识
component.part.label	"slash"	表示析出文献的符号, 可选: "in", "none"
italic.journal	#0	西文期刊名使用斜体
link.journal	#0	在期刊题名上添加 url 的超链接
show.missing.address.publisher	#0	出版项缺失时显示“出版者不详”
space.before.pages	#1	页码与前面的冒号之间有空格
only.start.page	#0	只显示起始页码
wave.dash.in.pages	#0	起止页码使用波浪号
show.urldate	#1	显示引用日期 urldate
show.url	#1	显示 url
show.doi	#1	显示 DOI
show.preprint	#1	显示预印本信息
show.note	#0	显示 note 域的信息
end.with.period	#1	结尾加句点

了另一个 Bib_T_EX 实现, 还给每行 bst 代码写了 java 语言注释。沈周^[10]基于 biblatex-caservector^[11] 进行修改, 以符合国标的格式。胡振震发布了符合 GB/T 7714—2015 标准的 BibLaTeX 参考文献样式^[12], 并进行了比较完善的持续维护。

参考文献

- [1] 中国标准化委员会. 信息与文献 参考文献著录规则: GB/T 7714—2015[S]. 北京: 中国标准出版社, 2015.
- [2] PATASHNIK O. Bib_T_EXing[M/OL]. 1988. <http://mirrors.ctan.org/biblio/bibtex/base/btxdoc.pdf>.

- [3] DALY P W. Natural sciences citations and references[M/OL]. 1999. <http://mirrors.ctan.org/macros/latex/contrib/natbib/natbib.pdf>.
- [4] PATASHNIK O. Designing Bib_T_EX styles[M/OL]. 1988. <http://mirrors.ctan.org/biblio/bibtex/base/btxhak.pdf>.
- [5] MARKEY N. Tame the beast[M/OL]. 2003. http://mirrors.ctan.org/info/bibtex/tamethebeast/ttb_en.pdf.
- [6] MITTELBAACH F, GOOSSENS M, BRAAMS J, et al. The L^AT_EX companion[M]. 2nd ed. Reading, MA, USA: Addison-Wesley, 2004.
- [7] 吴凯. 发布 GBT7714-2005.bst version1 Beta 版 [EB/OL]. 2006. CTeX 论坛（已关闭）.
- [8] 李志奇. 基于 biblatex 的符合 GB/T7714—2005 的中文文献生成工具 [EB/OL]. 2013. CTeX 论坛（已关闭）.
- [9] 胡海星. A GB/T 7714—2005 national standard compliant BibTeX style[EB/OL]. 2013. <https://github.com/Haixing-Hu/GBT7714-2005-BibTeX-Style>.
- [10] 沈周. 基于 caspervector 改写的符合 GB/T 7714—2005 标准的参考文献格式 [EB/OL]. 2016. <https://github.com/szsdk/biblatex-gbt77142005>.
- [11] VECTOR C T. biblatex 参考文献和引用样式: caspervector[M/OL]. 2012. <http://mirrors.ctan.org/macros/latex/contrib/biblatex-contrib/biblatex-caspervector/doc/caspervector.pdf>.
- [12] 胡振震. 符合 GB/T 7714—2015 标准的 biblatex 参考文献样式 [M/OL]. 2016. <http://mirrors.ctan.org/macros/latex/contrib/biblatex-contrib/biblatex-gb7714-2015/biblatex-gb7714-2015.pdf>.

A 宏包的代码实现

兼容过时的接口

```
1 <*package>
2 \newif\ifgbt@legacy@interface
3 \newif\ifgbt@mmxv
4 \newif\ifgbt@numerical
5 \newif\ifgbt@super
6 \newcommand\gbt@obsolete@option[1]{%
7   \PackageWarning{gbt7714}{The option "#1" is obsolete}%
8 }
9 \DeclareOption{2015}{%
10  \gbt@obsolete@option{2015}%
11  \gbt@legacy@interfacetrue
12  \gbt@mmxvtrue
13 }
14 \DeclareOption{2005}{%
15  \gbt@obsolete@option{2005}%
16  \gbt@legacy@interfacetrue
17  \gbt@mmxvfalse
18 }
19 \DeclareOption{super}{%
20  \gbt@obsolete@option{super}%
21  \gbt@legacy@interfacetrue
22  \gbt@numericaltrue
23  \gbt@supertrue
24 }
25 \DeclareOption{numbers}{%
26  \gbt@obsolete@option{numbers}%
27  \gbt@legacy@interfacetrue
28  \gbt@numericaltrue
29  \gbt@superfalse
30 }
31 \DeclareOption{authoryear}{%
32  \gbt@obsolete@option{authoryear}%
33  \gbt@legacy@interfacetrue
34  \gbt@numericalfalse
35 }
36 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{natbib}}
37 \ProcessOptions\relax
```

将选项传递给 natbib

调用宏包，注意只需要 `compress` 不需要 `sort`。

```
38 \RequirePackage{natbib}
```

```
39 \RequirePackage{url}
```

如果将 `compress` 传给 `natbib` 容易导致 `option clash`。这里直接修改内部命令。

```
40 \def\NAT@cmprs{\@ne}
```

`\citestyle` 定义接口切换引用文献的标注法，可用 `\citestyle` 调用 `numerical` 或 `authoryear`，参见 `natbib`。

```
41 \renewcommand\newblock{\space}
```

```
42 \newcommand\bibstyle@super{\bibpunct{[}{]}{,}{s}{,}{\textsuperscript{,}}}
```

```
43 \newcommand\bibstyle@numbers{\bibpunct{[}{]}{,}{n}{,}{,}}
```

```
44 \newcommand\bibstyle@authoryear{\bibpunct{({}{})}{;}{a}{,}{,}}
```

```
45 \newcommand\bibstyle@inline{\bibstyle@numbers}
```

(End of definition for \citestyle. This function is documented on page 2.)

在使用 `\bibliographystyle` 时自动切换引用文献的标注的样式。

```
46 \@namedef{bibstyle@gbt7714-numerical}{\bibstyle@super}
```

```
47 \@namedef{bibstyle@gbt7714-author-year}{\bibstyle@authoryear}
```

```
48 \@namedef{bibstyle@gbt7714-2005-numerical}{\bibstyle@super}
```

```
49 \@namedef{bibstyle@gbt7714-2005-author-year}{\bibstyle@authoryear}
```

`\cite` 下面修改 `natbib` 的引用格式。为了减少依赖的宏包，这里直接重定义命令不使用 `etoolbox` 的 `\patchcmd`。

`Super` 样式的 `\citep` 的页码也为上标。另外加上 `\kern\p@` 去掉上标式引用后与中文之间多余的空格，参考 [tuna/thuthesis#624](#)。

```
50 \renewcommand\NAT@citesuper[3]{%
```

```
51 \ifNAT@swa
```

```
52 \if*#2*\else
```

```
53 #2\NAT@spacechar
```

```
54 \fi
```

```
55 % \unskip\kern\p@\textsuperscript{\NAT@open#1\NAT@close}%
```

```
56 % \if*#3*\else\NAT@spacechar#3\fi\else #1\fi\endgroup}
```

```
57 \unskip\kern\p@
```

```
58 \textsuperscript{%
```

```
59 \NAT@open
```

```
60 #1%
```

```
61 \NAT@close
```

```
62 \if*#3*\else
```

```
63 #3%
```

```
64 \fi
```

```
65 }%
```

```
66 \kern\p@
```

```

67 \else
68   #1%
69 \fi
70 \endgroup
71 }

```

将 numbers 样式的 \citep 的页码置于括号外。

```

72 \renewcommand\NAT@citenum[3]{%
73   \ifNAT@swa
74     \NAT@@open
75     \if*#2*\else
76       #2\NAT@spacechar
77     \fi
78     % #1\if*#3*\else\NAT@cmt#3\fi\NAT@close\else#1\fi\endgroup}
79     #1\NAT@close
80     \if*#3*\else
81       \textsuperscript{#3}%
82     \fi
83   \else
84     #1%
85   \fi
86 \endgroup
87 }

```

Numerical 模式的 \citet 的页码:

```

88 \def\NAT@citexnum[#1][#2]#3{%
89   \NAT@reset@parser
90   \NAT@sort@cites{#3}%
91   \NAT@reset@citea
92   \@cite{\def\NAT@num{-1}\let\NAT@last@yr\relax\let\NAT@nm\@empty
93     \@for\@citeb:=\NAT@cite@list\do
94       {\@safe@activestrue
95         \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}%
96         \@safe@activesfalse
97         \ifundefined{b@\@citeb\@extra@b@citeb}{%
98           {\reset@font\bfseries?}
99           \NAT@citeundefined\PackageWarning{natbib}%
100            {Citation `@\@citeb' on page \thepage \space undefined}}%
101         {\let\NAT@last@num\NAT@num\let\NAT@last@nm\NAT@nm
102           \NAT@parse{\@citeb}%
103           \ifNAT@longnames\ifundefined{bv@\@citeb\@extra@b@citeb}{%
104             \let\NAT@name=\NAT@all@names
105             \global\@namedef{bv@\@citeb\@extra@b@citeb}{}}}{%

```

```

106 \fi
107 \ifNAT@full\let\NAT@nm\NAT@all@names\else
108 \let\NAT@nm\NAT@name\fi
109 \ifNAT@swa
110 \@ifnum{\NAT@ctype>\@ne}{%
111 \@citea
112 \NAT@hyper@{\@ifnum{\NAT@ctype=\tw@}{\NAT@test{\NAT@ctype}}{\NAT@alias}}%
113 }{%
114 \@ifnum{\NAT@cmprs>\z@}{%
115 \NAT@ifcat@num\NAT@num
116 {\let\NAT@nm=\NAT@num}%
117 {\def\NAT@nm{-2}}%
118 \NAT@ifcat@num\NAT@last@num
119 {\@tempcnta=\NAT@last@num\relax}%
120 {\@tempcnta\m@ne}%
121 \@ifnum{\NAT@nm=\@tempcnta}{%
122 \@ifnum{\NAT@merge>\@ne}{\NAT@last@yr@mbbox}%
123 }{%
124 \advance\@tempcnta by\@ne
125 \@ifnum{\NAT@nm=\@tempcnta}{%

```

在顺序编码制下，`natbib` 只有在三个以上连续文献引用才会使用连接号，这里修改为允许两个引用使用连接号。

```

126 % \ifx\NAT@last@yr\relax
127 % \def\NAT@last@yr{\@citea}%
128 % \else
129 % \def\NAT@last@yr{--\NAT@penalty}%
130 % \fi
131 \def\NAT@last@yr{-\NAT@penalty}%
132 }{%
133 \NAT@last@yr@mbbox
134 }%
135 }%
136 }{%
137 \@tempswatruel
138 \@ifnum{\NAT@merge>\@ne}{\@ifnum{\NAT@last@num=\NAT@num\relax}{\@tempswafalse}}{%
139 \if@tempswa\NAT@citea@mbbox\fi
140 }%
141 }%
142 \NAT@def@citea
143 \else
144 \ifcase\NAT@ctype
145 \ifx\NAT@last@nm\NAT@nm \NAT@yrsep\NAT@penalty\NAT@space\else

```

```

146         \@citea \NAT@test{\@ne}\NAT@spacechar\NAT@mbbox{\NAT@super@kern\NAT@@open}%
147         \fi
148         \if*#1*\else#1\NAT@spacechar\fi
149         \NAT@mbbox{\NAT@hyper@{\{\citenumfont{\NAT@num}\}}}%
150         \NAT@def@citea@box
151     \or
152         \NAT@hyper@citea@space{\NAT@test{\NAT@ctype}}%
153     \or
154         \NAT@hyper@citea@space{\NAT@test{\NAT@ctype}}%
155     \or
156         \NAT@hyper@citea@space\NAT@alias
157     \fi
158 \fi
159 }%
160 }%
161 \@ifnum{\NAT@cmprs>\z@}{\NAT@last@yr}{}%
162 \ifNAT@swa\else

```

将页码放在括号外边，并且置于上标。

```

163     % \@ifnum{\NAT@ctype=\z@}{%
164     %   \if*#2*\else\NAT@cmt#2\fi
165     % }{%
166     \NAT@mbbox{\NAT@@close}%
167     \@ifnum{\NAT@ctype=\z@}{%
168         \if*#2*\else
169             \textsuperscript{#2}%
170         \fi
171     }{%
172     \NAT@super@kern
173     \fi
174 }{#1}{#2}%
175 }%

```

Author-year 模式的 \citep 的页码:

```

176 \renewcommand\NAT@cite%
177 [3]{\ifNAT@swa\NAT@@open\if*#2*\else#2\NAT@spacechar\fi
178 #1\NAT@@close\if*#3*\else\textsuperscript{#3}\fi\else#1\fi\endgroup}

```

(End of definition for \cite. This function is documented on page ??.)

Author-year 模式的 \citet 的页码:

```

179 \def\NAT@citex%
180 [#1][#2]#3{%
181 \NAT@reset@parser
182 \NAT@sort@cites{#3}%

```

```

183 \NAT@reset@citea
184 \@cite{\let\NAT@nm\@empty\let\NAT@year\@empty
185 \@for\@citeb:=\NAT@cite@list\do
186 {\@safe@activestru
187 \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}%
188 \@safe@activesfalse
189 \@ifundefined{b@\@citeb\@extra@b@citeb}{\@citea%
190 {\reset@font\bfseries ?}\NAT@citeundefined
191 \PackageWarning{natbib}%
192 {Citation ``\@citeb' on page \thepage \space undefined}\def\NAT@date{}}%
193 {\let\NAT@lastnm=\NAT@nm\let\NAT@lastyr=\NAT@year
194 \NAT@parse{\@citeb}%
195 \ifNAT@longnames\@ifundefined{bv@\@citeb\@extra@b@citeb}{%
196 \let\NAT@name=\NAT@all@names
197 \global\@namedef{bv@\@citeb\@extra@b@citeb}{}}}%
198 \fi
199 \ifNAT@full\let\NAT@nm\NAT@all@names\else
200 \let\NAT@nm\NAT@name\fi
201 \ifNAT@swa\ifcase\NAT@ctype
202 \if\relax\NAT@date\relax
203 \@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}\NAT@date}%
204 \else
205 \ifx\NAT@lastnm\NAT@nm\NAT@yrsep
206 \ifx\NAT@lastyr\NAT@year
207 \def\NAT@temp{?}%
208 \ifx\NAT@temp\NAT@exlab\PackageWarningNoLine{natbib}%
209 {Multiple citation on page \thepage: same authors and
210 year\MessageBreak without distinguishing extra
211 letter,\MessageBreak appears as question mark}\fi
212 \NAT@hyper@{\NAT@exlab}%
213 \else\unskip\NAT@spacechar
214 \NAT@hyper@{\NAT@date}%
215 \fi
216 \else
217 \@citea\NAT@hyper@{%
218 \NAT@nmfmt{\NAT@nm}%
219 \hyper@natlinkbreak{%
220 \NAT@aysep\NAT@spacechar}{\@citeb\@extra@b@citeb
221 }%
222 \NAT@date
223 }%
224 \fi

```

```

225     \fi
226     \or\@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}}%
227     \or\@citea\NAT@hyper@{\NAT@date}%
228     \or\@citea\NAT@hyper@{\NAT@alias}%
229     \fi \NAT@def@citea
230   \else
231     \ifcase\NAT@ctype
232     \if\relax\NAT@date\relax
233       \@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}}%
234     \else
235       \ifx\NAT@last@nm\NAT@nm\NAT@yrsep
236         \ifx\NAT@last@yr\NAT@year
237           \def\NAT@temp{{?}}%
238           \ifx\NAT@temp\NAT@exlab\PackageWarningNoLine{natbib}%
239             {Multiple citation on page \thepage: same authors and
240              year\MessageBreak without distinguishing extra
241              letter,\MessageBreak appears as question mark}\fi
242           \NAT@hyper@{\NAT@exlab}%
243         \else
244           \unskip\NAT@spacechar
245           \NAT@hyper@{\NAT@date}%
246         \fi
247       \else
248         \@citea\NAT@hyper@{%
249           \NAT@nmfmt{\NAT@nm}%
250           \hyper@natlinkbreak{\NAT@spacechar\NAT@@open\if*#1*\else#1\NAT@spacechar\fi}%
251           {\@citeb\@extra@b@citeb}%
252           \NAT@date
253         }%
254       \fi
255     \fi
256     \or\@citea\NAT@hyper@{\NAT@nmfmt{\NAT@nm}}%
257     \or\@citea\NAT@hyper@{\NAT@date}%
258     \or\@citea\NAT@hyper@{\NAT@alias}%
259     \fi
260     \if\relax\NAT@date\relax
261       \NAT@def@citea
262     \else
263       \NAT@def@citea@close
264     \fi
265   \fi
266 } \ifNAT@swa\else

```

将页码放在括号外边，并且置于上标。

```
267 % \if*#2*\else\NAT@cmt#2\fi
268 \if\relax\NAT@date\relax\else\NAT@close\fi
269 \if*#2*\else\textsuperscript{#2}\fi
270 \fi}{#1}{#2}}
```

thebibliography (env.) 参考文献列表的标签左对齐

```
271 \renewcommand\@biblabel[1]{#1\hfill}
```

Patch natbib 内部命令，以支持 \noopsort。参考 <https://tex.stackexchange.com/a/39718/82731>。

```
272 \let\NAT@bare@aux\NAT@bare
273 \def\NAT@bare#1(#2){%
274 \begingroup\edef\x{\endgroup
275 \unexpanded{\NAT@bare@aux#1}(\@firstofone#2)}\x}
```

\url 使用 xurl 宏包的方法，增加 URL 可断行的位置。

```
276 \g@addto@macro\UrlBreaks{%
277 \do0\do1\do2\do3\do4\do5\do6\do7\do8\do9%
278 \doA\doB\doC\doD\doE\doF\doG\doH\doI\doJ\doK\doL\doM
279 \doN\doO\doP\doQ\doR\doS\doT\doU\doV\doW\doX\doY\doZ
280 \do\a\do\b\do\c\do\d\do\e\do\f\do\g\do\h\do\i\do\j\do\k\do\l\do\m
281 \do\n\do\o\do\p\do\q\do\r\do\s\do\t\do\u\do\v\do\w\do\x\do\y\do\z
282 }
283 \Urlmuskip=0mu plus 0.1mu
```

(End of definition for \url. This function is documented on page ??.)

兼容 v2.0 前过时的接口：

```
284 \newif\ifgbt@bib@style@written
285 \@ifpackageloaded{chapterbib}{%
286 \def\bibliography#1{%
287 \ifgbt@bib@style@written\else
288 \bibliographystyle{gbt7714-numerical}%
289 \fi
290 \if@filesw
291 \immediate\write\@auxout{\string\bibdata{\zap@space#1 \@empty}}%
292 \fi
293 \@input@{\jobname.bbl}}
294 \def\bibliographystyle#1{%
295 \gbt@bib@style@writtentrue
296 \ifx\@begindocumenthook\@undefined\else
297 \expandafter\AtBeginDocument
298 \fi
```

```

299     {\if@filesw
300       \immediate\write\@auxout{\string\bibstyle{#1}}%
301     \fi}%
302   }%
303 }
304 \ifgbt@legacy@interface
305   \ifgbt@numerical
306     \ifgbt@super\else
307       \citestyle{numbers}
308     \fi
309     \bibliographystyle{gbt7714-numerical}
310   \else
311     \bibliographystyle{gbt7714-author-year}
312   \fi
313 \fi
314 \end{package}

```

B BibTeX 样式的代码实现

B.1 自定义选项

`bst (env.)` 这里定义了一些变量用于定制样式，可以在下面的 `load.config` 函数中选择是否启用。

```

315 (*author-year | numerical)
316 INTEGERS {
317   citation.et.al.min
318   citation.et.al.use.first
319   bibliography.et.al.min
320   bibliography.et.al.use.first
321   uppercase.name
322   terms.in.macro
323   year.after.author
324   period.after.author
325   italic.book.title
326   sentence.case.title
327   link.title
328   title.in.journal
329   show.patent.country
330   show.mark
331   space.before.mark
332   show.medium.type
333   short.journal
334   italic.journal
335   link.journal
336   bold.journal.volume
337   show.missing.address.publisher
338   space.before.pages

```



```

339 only.start.page
340 wave.dash.in.pages
341 show.urldate
342 show.url
343 show.doi
344 show.preprint
345 show.note
346 show.english.translation
347 end.with.period
348 <*author-year>
349 lang.zh.order
350 lang.ja.order
351 lang.en.order
352 lang.ru.order
353 lang.other.order
354 </author-year>
355 }
356
357 STRINGS {
358 component.part.label
359 }
360

```

下面每个变量若被设为 **#1** 则启用该项，若被设为 **#0** 则不启用。默认的值是严格遵循国标的配置。

```

361 FUNCTION {load.config}
362 {

```

如果姓名的数量大于等于 `et.al.min`，只著录前 `et.al.use.first` 个，其后加“et al.”或“等”。

```

363 <!*lucas>
364 #2 'citation.et.al.min :=
365 #1 'citation.et.al.use.first :=
366 </!lucas>
367 <*ucas>
368 #3 'citation.et.al.min :=
369 #1 'citation.et.al.use.first :=
370 </ucas>
371 #4 'bibliography.et.al.min :=
372 #3 'bibliography.et.al.use.first :=

```

英文姓名转为全大写：

```

373 <*(no-uppercase | thu)>
374 #1 'uppercase.name :=
375 </!(no-uppercase | thu)>
376 <*no-uppercase | thu>
377 #0 'uppercase.name :=
378 </no-uppercase | thu>

```

使用 TeX 宏输出“和”、“等”

```

379 <*(macro |ucas)>
380 #0 'terms.in.macro :=
381 </!(macro |ucas)>
382 <*macro |ucas>

```

```
383 #1 'terms.in.macro :=
384 </macro | ucas)
```

将年份置于著者后面（著者-出版年制默认）

```
385 <*numerical | ucas)
386 #0 'year.after.author :=
387 </numerical | ucas)
388 <*author-year&!ucas)
389 #1 'year.after.author :=
390 </author-year&!ucas)
```

采用著者-出版年制时，作者姓名与年份之间使用句点连接：

```
391 <*numerical)
392 #1 'period.after.author :=
393 </numerical)
394 <*author-year)
395 <*2015&!(period | ustd))
396 #0 'period.after.author :=
397 </2015&!(period | ustd))
398 <*period | 2005 | ustd)
399 #1 'period.after.author :=
400 </period | 2005 | ustd)
401 </author-year)
```

书名使用斜体：

```
402 <*!italic-book-title)
403 #0 'italic.book.title :=
404 </!italic-book-title)
405 <*italic-book-title)
406 #1 'italic.book.title :=
407 </italic-book-title)
```

英文标题转为 sentence case（句首字母大写，其余小写）：

```
408 <*!no-sentence-case)
409 #1 'sentence.case.title :=
410 </!no-sentence-case)
411 <*no-sentence-case)
412 #0 'sentence.case.title :=
413 </no-sentence-case)
```

在标题添加超链接：

```
414 <*!link-title)
415 #0 'link.title :=
416 </!link-title)
417 <*link-title)
418 #1 'link.title :=
419 </link-title)
```

期刊是否含标题：

```
420 <*!no-title-in-journal)
421 #1 'title.in.journal :=
422 </!no-title-in-journal)
423 <*no-title-in-journal)
424 #0 'title.in.journal :=
425 </no-title-in-journal)
```

专利题名是否含专利国别

```
426 <*(show-patent-country | 2005 | ustc | thu)>
427 #0 'show.patent.country :=
428 </!(show-patent-country | 2005 | ustc | thu)>
429 <*(show-patent-country | 2005 | ustc | thu)>
430 #1 'show.patent.country :=
431 </(show-patent-country | 2005 | ustc | thu)>
```

著录文献类型标识（比如“[M/OL]”）：

```
432 <!*no-mark>
433 #1 'show.mark :=
434 </!no-mark>
435 <*no-mark>
436 #0 'show.mark :=
437 </no-mark>
```

文献类型标识前是否有空格：

```
438 <!*space-before-mark>
439 #0 'space.before.mark :=
440 </!space-before-mark>
441 <*space-before-mark>
442 #1 'space.before.mark :=
443 </space-before-mark>
```

是否显示载体类型标识（比如“/OL”）：

```
444 <!*no-medium-type>
445 #1 'show.medium.type :=
446 </!no-medium-type>
447 <*no-medium-type>
448 #0 'show.medium.type :=
449 </no-medium-type>
```

使用“//”表示析出文献

```
450 <*(in-collection | no-slash)>
451 "slash" 'component.part.label :=
452 </!(in-collection | no-slash)>
453 <*in-collection>
454 "in" 'component.part.label :=
455 </in-collection>
456 <*no-slash>
457 "none" 'component.part.label :=
458 </no-slash>
```

期刊名使用缩写：

```
459 <!*short-journal>
460 #0 'short.journal :=
461 </!short-journal>
462 <*short-journal>
463 #1 'short.journal :=
464 </short-journal>
```

期刊名使用斜体：

```
465 <!*italic-journal>
466 #0 'italic.journal :=
```

```
467 </!italic-journal>
468 <*italic-journal>
469 #1 'italic.journal :=
470 </italic-journal>
```

在期刊题名添加超链接:

```
471 <*!link-journal>
472 #0 'link.journal :=
473 </!link-journal>
474 <*link-journal>
475 #1 'link.journal :=
476 </link-journal>
```

期刊的卷使用粗体:

```
477 #0 'bold.journal.volume :=
```

无出版地或出版者时, 著录“出版地不详”, “出版者不详”, “S.l.”或“s.n.”:

```
478 <*!sl-sn>
479 #0 'show.missing.address.publisher :=
480 </!sl-sn>
481 <*sl-sn>
482 #1 'show.missing.address.publisher :=
483 </sl-sn>
```

页码与前面的冒号之间是否有空格:

```
484 <*!no-space-before-pages>
485 #1 'space.before.pages :=
486 </!no-space-before-pages>
487 <*no-space-before-pages>
488 #0 'space.before.pages :=
489 </no-space-before-pages>
```

页码是否只含起始页:

```
490 <*!only-start-page>
491 #0 'only.start.page :=
492 </!only-start-page>
493 <*only-start-page>
494 #1 'only.start.page :=
495 </only-start-page>
```

起止页码使用波浪号:

```
496 <*!wave-dash-in-pages>
497 #0 'wave.dash.in.pages :=
498 </!wave-dash-in-pages>
499 <*wave-dash-in-pages>
500 #1 'wave.dash.in.pages :=
501 </wave-dash-in-pages>
```

是否著录非电子文献的引用日期:

```
502 <*!no-urldate>
503 #1 'show.urldate :=
504 </!no-urldate>
505 <*no-urldate>
506 #0 'show.urldate :=
507 </no-urldate>
```

是否著录 URL:

```
508 <!*no-url>
509   #1 'show.url :=
510 </!no-url>
511 <*no-url>
512   #0 'show.url :=
513 </no-url>
```

是否著录 DOI:

```
514 <*(no-doi | 2005)>
515   #1 'show.doi :=
516 </!(no-doi | 2005)>
517 <*no-doi | 2005>
518   #0 'show.doi :=
519 </no-doi | 2005>
```

是否著录 e-print:

```
520 <!*preprint>
521   #1 'show.preprint :=
522 </!preprint>
523 <*preprint>
524   #0 'show.preprint :=
525 </preprint>
```

在每一条文献最后输出注释 (note) 的内容:

```
526   #0 'show.note :=
```

中文文献是否显示英文翻译

```
527 <!*show-english-translation>
528   #0 'show.english.translation :=
529 </!show-english-translation>
530 <*show-english-translation>
531   #1 'show.english.translation :=
532 </show-english-translation>
```

结尾加句点

```
533 <!*no-period-at-end>
534   #1 'end.with.period :=
535 </!no-period-at-end>
536 <*no-period-at-end>
537   #0 'end.with.period :=
538 </no-period-at-end>
```

参考文献表按照“著者-出版年”组织时, 各个文种的顺序:

```
539 <*author-year>
540   #1 'lang.zh.order :=
541   #2 'lang.ja.order :=
542   #3 'lang.en.order :=
543   #4 'lang.ru.order :=
544   #5 'lang.other.order :=
545 </author-year>
546 }
547
```

B.2 The ENTRY declaration

Like Scribe's (according to pages 231-2 of the April '84 edition), but no fullauthor or editors fields because BibTeX does name handling. The annotate field is commented out here because this family doesn't include an annotated bibliography style. And in addition to the fields listed here, BibTeX has a built-in crossref field, explained later.

```
548 ENTRY
549   { address
550     archivePrefix
551     author
552     booktitle
553     date
554     doi
555     edition
556     editor
557     eprint
558     eprinttype
559     entrysubtype
560     howpublished
561     institution
562     journal
563     journaltitle
564     key
565     langid
566     language
567     location
568     mark
569     medium
570     note
571     number
572     organization
573     pages
574     publisher
575     school
576     series
577     shortjournal
578     title
579     translation
580     translator
581     url
582     urldate
583     volume
584     year
585   }
586   { entry.lang entry.is.electronic is.pure.electronic entry.numbered }
```

These string entry variables are used to form the citation label. In a storage pinch, sort.label can be easily computed on the fly.

```
587   { label extra.label sort.label short.list entry.mark entry.url }
588
```

B.3 Entry functions

Each entry function starts by calling `output.bibitem`, to write the `\bibitem` and its arguments to the `.BBL` file. Then the various fields are formatted and printed by `output` or `output.check`. Those functions handle the writing of separators (commas, periods, `\newblock`'s), taking care not to do so when they are passed a null string. Finally, `fin.entry` is called to add the final period and finish the entry.

A bibliographic reference is formatted into a number of 'blocks': in the open format, a block begins on a new line and subsequent lines of the block are indented. A block may contain more than one sentence (well, not a grammatical sentence, but something to be ended with a sentence ending period). The entry functions should call `new.block` whenever a block other than the first is about to be started. They should call `new.sentence` whenever a new sentence is to be started. The output functions will ensure that if two `new.sentence`'s occur without any non-null string being output between them then there won't be two periods output. Similarly for two successive `new.block`'s.

The output routines don't write their argument immediately. Instead, by convention, that argument is saved on the stack to be output next time (when we'll know what separator needs to come after it). Meanwhile, the output routine has to pop the pending output off the stack, append any needed separator, and write it.

To tell which separator is needed, we maintain an `output.state`. It will be one of these values: `before.all` just after the `\bibitem` `mid.sentence` in the middle of a sentence: `comma` needed if more sentence is output `after.sentence` just after a sentence: `period` needed `after.block` just after a block (and sentence): `period` and `\newblock` needed. Note: These styles don't use `after.sentence`

VAR: `output.state` : INTEGER – state variable for output

The `output.nonnull` function saves its argument (assumed to be `nonnull`) on the stack, and writes the old saved value followed by any needed separator. The ordering of the tests is decreasing frequency of occurrence.

由于专著中的析出文献需要用到很特殊的“//”，所以我又加了一个 `after.slash`。其他需要在特定符号后面输出，所以写了一个 `output.after`。

```
output.nonnull(s) ==
BEGIN
  s := argument on stack
  if output.state = mid.sentence then
    write$(pop() * ",")
    -- "pop" isn't a function: just use stack top
  else
  if output.state = after.block then
```

```

write$(add.period$(pop()))
newline$
write$("\newblock_")
else
if_output.state=_before.all_then
write$(pop())
else_--output.state_should_be_after.sentence
write$(add.period$(pop()))*_""
fi
fi
output.state:=_mid.sentence
fi
push_s_on_stack
END

```

The output function calls output.nonnull if its argument is non-empty; its argument may be a missing field (thus, not necessarily a string)

```

output(s) ==
BEGIN
  if not empty$(s) then output.nonnull(s)
  fi
END

```

The output.check function is the same as the output function except that, if necessary, output.check warns the user that the t field shouldn't be empty (this is because it probably won't be a good reference without the field; the entry functions try to make the formatting look reasonable even when such fields are empty).

```

output.check(s,t) ==
BEGIN
  if empty$(s) then
    warning$("empty_" * t * "_in_" * cite$)
  else output.nonnull(s)
  fi
END

```

The output.bibitem function writes the \bibitem for the current entry (the label should already have been set up), and sets up the separator state for the output functions. And, it leaves a string on the stack as per the output convention.

```

output.bibitem ==
BEGIN
  newline$
  write$("\bibitem[")      % for alphabetic labels,
  write$(label)          % these three lines
  write$("{}")           % are used
  write$("\bibitem{")    % this line for numeric labels
  write$(cite$)
  write$("{}")
  push "" on stack
  output.state := before.all

```



```
END
```

The `fin.entry` function finishes off an entry by adding a period to the string remaining on the stack. If the state is still `before.all` then nothing was produced for this entry, so the result will look bad, but the user deserves it. (We don't omit the whole entry because the entry was cited, and a `bibitem` is needed to define the citation label.)

```
fin.entry ==
BEGIN
  write$(add.period$(pop()))
  newline$
END
```

The `new.block` function prepares for a new block to be output, and `new.sentence` prepares for a new sentence.

```
new.block ==
BEGIN
  if output.state <> before.all then
    output.state := after.block
  fi
END
```

```
new.sentence ==
BEGIN
  if output.state <> after.block then
    if output.state <> before.all then
      output.state := after.sentence
    fi
  fi
END
```

```
589 INTEGERS { output.state before.all mid.sentence after.sentence after.block after.slash }
590
591 INTEGERS { lang.zh lang.ja lang.en lang.ru lang.other }
592
593 INTEGERS { charptr len }
594
595 FUNCTION {init.state.consts}
596 { #0 'before.all :=
597   #1 'mid.sentence :=
598   #2 'after.sentence :=
599   #3 'after.block :=
600   #4 'after.slash :=
601   #3 'lang.zh :=
602   #4 'lang.ja :=
603   #1 'lang.en :=
604   #2 'lang.ru :=
605   #0 'lang.other :=
606 }
607
```

下面是一些常量的定义

```
608 FUNCTION {bbl.anonymous}
609 { entry.lang lang.zh =
610   { "佚名" }
611   { "Anon" }
612   if$
613 }
614
615 FUNCTION {bbl.space}
616 { entry.lang lang.zh =
617   { "\ " }
618   { " " }
619   if$
620 }
621
622 FUNCTION {bbl.and}
623 { "" }
624
625 FUNCTION {bbl.et.al}
626 { entry.lang lang.zh =
627   { "等" }
628   { entry.lang lang.ja =
629     { "他" }
630     { entry.lang lang.ru =
631       { "и др" }
632       { "et~al." }
633       if$
634     }
635     if$
636   }
637   if$
638 }
639
640 FUNCTION {citation.and}
641 { terms.in.macro
642   { "{\biband}" }
643   'bbl.and
644   if$
645 }
646
647 FUNCTION {citation.et.al}
648 { terms.in.macro
649   { "{\bibetal}" }
650   'bbl.et.al
651   if$
652 }
653
654 FUNCTION {bbl.colon} { ": " }
655
656 FUNCTION {bbl.pages.colon}
657 { space.before.pages
658   { ": " }
659   { ":\allowbreak " }
660   if$
661 }
```

```

662
663 <!*2005>
664 FUNCTION {bbl.wide.space} { "\quad " }
665 </!*2005>
666 <*2005>
667 FUNCTION {bbl.wide.space} { "\ " }
668 </2005>
669
670 FUNCTION {bbl.slash} { "//\allowbreak " }
671
672 FUNCTION {bbl.sine.loco}
673 { entry.lang lang.zh =
674   { "[出版地不详]" }
675   { "[S.l.]" }
676   if$
677 }
678
679 FUNCTION {bbl.sine.nomine}
680 { entry.lang lang.zh =
681   { "[出版者不详]" }
682   { "[s.n.]" }
683   if$
684 }
685
686 FUNCTION {bbl.sine.loco.sine.nomine}
687 { entry.lang lang.zh =
688   { "[出版地不详: 出版者不详]" }
689   { "[S.l.: s.n.]" }
690   if$
691 }
692

```

These three functions pop one or two (integer) arguments from the stack and push a single one, either 0 or 1. The 'skip\$ in the 'and' and 'or' functions are used because the corresponding if\$ would be idempotent

```

693 FUNCTION {not}
694 { { #0 }
695   { #1 }
696   if$
697 }
698
699 FUNCTION {and}
700 { 'skip$
701   { pop$ #0 }
702   if$
703 }
704
705 FUNCTION {or}
706 { { pop$ #1 }
707   'skip$
708   if$
709 }
710
711 STRINGS { x y }

```

```

712
713 FUNCTION {contains}
714 { 'y :=
715   'x :=
716   y text.length$ 'len :=
717   x text.length$ len - #1 + 'charptr :=
718     { charptr #0 >
719       x charptr len substring$ y = not
720       and
721     }
722   { charptr #1 - 'charptr := }
723   while$
724   charptr #0 >
725 }
726

```

the variables s and t are temporary string holders

```

727 STRINGS { s t }
728
729 FUNCTION {output.nonnull}
730 { 's :=
731   output.state mid.sentence =
732   { ", " * write$ }
733   { output.state after.block =
734     { add.period$ write$
735       newline$
736       "\newblock " write$
737     }
738     { output.state before.all =
739       'write$
740       { output.state after.slash =
741         { bbl.slash * write$
742           newline$
743         }
744         { add.period$ " " * write$ }
745       }
746     }
747     if$
748   }
749   if$
750   mid.sentence 'output.state :=
751 }
752 if$
753 s
754 }
755
756 FUNCTION {output}
757 { duplicate$ empty$
758   'pop$
759   'output.nonnull
760   if$
761 }
762
763 FUNCTION {output.after}
764 { 't :=

```

```

765 duplicate$ empty$
766 'pop$
767 { 's :=
768   output.state mid.sentence =
769     { t * write$ }
770     { output.state after.block =
771       { add.period$ write$
772         newline$
773         "\newblock " write$
774       }
775       { output.state before.all =
776         'write$
777         { output.state after.slash =
778           { bbl.slash * write$ }
779           { add.period$ " " * write$ }
780         if$
781       }
782       if$
783     }
784     if$
785     mid.sentence 'output.state :=
786   }
787   if$
788   s
789 }
790 if$
791 }
792
793 FUNCTION {output.check}
794 { 't :=
795   duplicate$ empty$
796   { pop$ "empty " t * " in " * cite$ * warning$ }
797   'output.nonnull
798   if$
799 }
800

```

This function finishes all entries.

```

801 FUNCTION {fin.entry}
802 { end.with.period
803   'add.period$
804   'skip$
805   if$
806   write$
807   show.english.translation entry.lang lang.zh = and
808   { ")"
809     write$
810   }
811   'skip$
812   if$
813   newline$
814 }
815
816 FUNCTION {new.block}
817 { output.state before.all =

```

```

818     'skip$
819     { output.state after.slash =
820       'skip$
821       { after.block 'output.state := }
822     if$
823   }
824 if$
825 }
826
827 FUNCTION {new.sentence}
828 { output.state after.block =
829   'skip$
830   { output.state before.all =
831     'skip$
832     { output.state after.slash =
833       'skip$
834       { after.sentence 'output.state := }
835     if$
836   }
837   if$
838 }
839 if$
840 }
841
842 FUNCTION {new.slash}
843 { output.state before.all =
844   'skip$
845   { component.part.label "slash" =
846     { after.slash 'output.state := }
847     { new.block
848       component.part.label "in" =
849         { entry.lang lang.en =
850           { "In: " output
851             writes$
852             ""
853             before.all 'output.state :=
854           }
855           'skip$
856         if$
857       }
858       'skip$
859     if$
860   }
861   if$
862 }
863 if$
864 }
865

```

Sometimes we begin a new block only if the block will be big enough. The `new.block.checka` function issues a `new.block` if its argument is nonempty; `new.block.checkb` does the same if either of its TWO arguments is nonempty.

```

866 FUNCTION {new.block.checka}
867 { empty$

```

```

868     'skip$
869     'new.block
870   if$
871 }
872
873 FUNCTION {new.block.checkb}
874 { empty$
875   swap$ empty$
876   and
877   'skip$
878   'new.block
879   if$
880 }
881

```

The new.sentence.check functions are analogous.

```

882 FUNCTION {new.sentence.checka}
883 { empty$
884   'skip$
885   'new.sentence
886   if$
887 }
888
889 FUNCTION {new.sentence.checkb}
890 { empty$
891   swap$ empty$
892   and
893   'skip$
894   'new.sentence
895   if$
896 }
897

```

B.4 Formatting chunks

Here are some functions for formatting chunks of an entry. By convention they either produce a string that can be followed by a comma or period (using `add.period$`, so it is OK to end in a period), or they produce the null string.

A useful utility is the `field.or.null` function, which checks if the argument is the result of pushing a ‘missing’ field (one for which no assignment was made when the current entry was read in from the database) or the result of pushing a string having no non-white-space characters. It returns the null string if so, otherwise it returns the field string. Its main (but not only) purpose is to guarantee that what’s left on the stack is a string rather than a missing field.

```

field.or.null(s) ==
BEGIN
  if empty$(s) then return ""
  else return s
END

```

Another helper function is `emphasize`, which returns the argument emphasized, if that is non-empty, otherwise it returns the null string. Italic corrections aren't used, so this function should be used when punctuation will follow the result.

```
emphasize(s) ==
BEGIN
  if empty$(s) then return ""
  else return "{\em_} * s * {}"
```

The 'pop\$' in this function gets rid of the duplicate 'empty' value and the 'skip\$' returns the duplicate field value

```
898 FUNCTION {field.or.null}
899 { duplicate$ empty$
900   { pop$ "" }
901   'skip$
902   if$
903 }
904
905 FUNCTION {emphasize}
906 { duplicate$ empty$
907   { pop$ "" }
908   { "\emph{" swap$ * "}" * }
909   if$
910 }
911
912 FUNCTION {format.btitle}
913 { italic.book.title
914   entry.lang lang.en = and
915   'emphasize
916   'skip$
917   if$
918 }
919
```

B.4.1 Detect Language

```
920 INTEGERS { byte second.byte }
921
922 INTEGERS { char.lang tmp.lang }
923
924 STRINGS { tmp.str }
925
926 FUNCTION {get.str.lang}
927 { 'tmp.str :=
928   lang.other 'tmp.lang :=
929   #1 'charptr :=
930   tmp.str text.length$ #1 + 'len :=
931   { charptr len < }
932   { tmp.str charptr #1 substring$ chr.to.int$ 'byte :=
933     byte #128 <
934     { charptr #1 + 'charptr :=
935       byte #64 > byte #91 < and byte #96 > byte #123 < and or
936       { lang.en 'char.lang := }
```



```

937         { lang.other 'char.lang := }
938     if$
939 }
940 { tmp.str charptr #1 + #1 substring$ chr.to.int$ 'second.byte :=
941     byte #224 <

```

俄文西里尔字母: U+0400 到 U+052F, 对应 UTF-8 从 D0 80 到 D4 AF。

```

942     { charptr #2 + 'charptr :=
943     byte #207 > byte #212 < and
944     byte #212 = second.byte #176 < and or
945     { lang.ru 'char.lang := }
946     { lang.other 'char.lang := }
947     if$
948     }
949     { byte #240 <

```

CJK Unified Ideographs: U+4E00–U+9FFF; UTF-8: E4 B8 80–E9 BF BF.

```

950     { charptr #3 + 'charptr :=
951     byte #227 > byte #234 < and
952     { lang.zh 'char.lang := }

```

CJK Unified Ideographs Extension A: U+3400–U+4DBF; UTF-8: E3 90 80–E4 B6 BF.

```

953     { byte #227 =
954     { second.byte #143 >
955     { lang.zh 'char.lang := }

```

日语假名: U+3040–U+30FF, UTF-8: E3 81 80–E3 83 BF.

```

956     { second.byte #128 > second.byte #132 < and
957     { lang.ja 'char.lang := }
958     { lang.other 'char.lang := }
959     if$
960     }
961     if$
962     }

```

CJK Compatibility Ideographs: U+F900–U+FAFF, UTF-8: EF A4 80–EF AB BF.

```

963     { byte #239 =
964     second.byte #163 > second.byte #172 < and and
965     { lang.zh 'char.lang := }
966     { lang.other 'char.lang := }
967     if$
968     }
969     if$
970     }
971     if$
972     }

```

CJK Unified Ideographs Extension B–F: U+20000–U+2EBEF, UTF-8: F0 A0 80 80–F0

AE AF AF. CJK Compatibility Ideographs Supplement: U+2F800–U+2FA1F, UTF-8:

F0 AF A0 80–F0 AF A8 9F.

```

973     { charptr #4 + 'charptr :=
974     byte #240 = second.byte #159 > and
975     { lang.zh 'char.lang := }
976     { lang.other 'char.lang := }
977     if$

```

```

978         }
979         if$
980     }
981     if$
982 }
983 if$
984 char.lang tmp.lang >
985 { char.lang 'tmp.lang := }
986 'skip$
987 if$
988 }
989 while$
990 tmp.lang
991 }
992
993 FUNCTION {check.entry.lang}
994 { author field.or.null
995   title field.or.null *
996   get.str.lang
997 }
998
999 STRINGS { entry.langid }
1000
1001 FUNCTION {set.entry.lang}
1002 { "" 'entry.langid :=
1003   language empty$ not
1004   { language 'entry.langid := }
1005   'skip$
1006   if$
1007   langid empty$ not
1008   { langid 'entry.langid := }
1009   'skip$
1010   if$
1011   entry.langid empty$
1012   { check.entry.lang }
1013   { entry.langid "english" = entry.langid "american" = or entry.langid "british" = or
1014     { lang.en }
1015     { entry.langid "chinese" =
1016       { lang.zh }
1017       { entry.langid "japanese" =
1018         { lang.ja }
1019         { entry.langid "russian" =
1020           { lang.ru }
1021           { check.entry.lang }
1022         if$
1023       }
1024     }
1025   }
1026   if$
1027 }
1028   if$
1029 }
1030 if$
1031 'entry.lang :=
1032 }

```

```

1033
1034 FUNCTION {set.entry.numbered}
1035 { type$ "patent" =
1036   type$ "standard" = or
1037   type$ "techreport" = or
1038   { #1 'entry.numbered := }
1039   { #0 'entry.numbered := }
1040   if$
1041 }
1042

```

B.4.2 Format names

The `format.names` function formats the argument (which should be in BibTeX name format) into First Von Last, Junior, separated by commas and with an and before the last (but ending with `et~al.` if the last of multiple authors is others). This function's argument should always contain at least one name.

```

VAR: nameptr, namesleft, numnames: INTEGER
pseudoVAR: namerresult: STRING      (it's what's accumulated on the stack)

format.names(s) ==
BEGIN
  nameptr := 1
  numnames := num.names$(s)
  namesleft := numnames
  while namesleft > 0
  do
    % for full names:
    t := format.name$(s, nameptr, "{ff~}{vv~}{ll}{,ujj}")
    % for abbreviated first names:
    t := format.name$(s, nameptr, "{f.~}{vv~}{ll}{,ujj}")
  if nameptr > 1 then
    if namesleft > 1 then namerresult := namerresult * ", " * t
    else if numnames > 2
      then namerresult := namerresult * ","
      fi
    if t = "others"
      then namerresult := namerresult * " et~al."
      else namerresult := namerresult * " and " * t
      fi
    fi
  else namerresult := t
  fi
  nameptr := nameptr + 1
  namesleft := namesleft - 1
od
return namerresult
END

```

The `format.authors` function returns the result of `format.names(author)` if the author is present, or else it returns the null string

```
format.authors ==
```

```

BEGIN
  if empty$(author) then return ""
  else return format.names(author)
  fi
END

```

Format.editors is like format.authors, but it uses the editor field, and appends , editor or , editors

```

format.editors ==
BEGIN
  if empty$(editor) then return ""
  else
    if num.names$(editor) > 1 then
      return format.names(editor) * ",_editors"
    else
      return format.names(editor) * ",_editor"
    fi
  fi
END

```

Other formatting functions are similar, so no comment version will be given for them.

```

1043 INTEGERS { nameptr namesleft numnames name.lang }
1044
1045 FUNCTION {format.name}
1046 { "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
1047   t "others" =
1048   { bbl.et.al }
1049   { t get.str.lang 'name.lang :=
1050     name.lang lang.en =
1051     { t #1 "{vv~}{ll}{ f{~}}" format.name$
1052       uppercase.name
1053       { "u" change.case$ }
1054       'skip$
1055       if$
1056       t #1 "{, jj}" format.name$ *
1057     }
1058     { t #1 "{ll}{ff}" format.name$ }
1059   if$
1060 }
1061 if$
1062 }
1063
1064 FUNCTION {format.names}
1065 { 's :=
1066   #1 'nameptr :=
1067   s num.names$ 'numnames :=
1068   ""
1069   numnames 'namesleft :=
1070   { namesleft #0 > }
1071   { s nameptr format.name bbl.et.al =
1072     numnames bibliography.et.al.min #1 - > nameptr bibliography.et.al.use.first > and or

```

```

1073     { ", " *
1074       bbl.et.al *
1075       #1 'namesleft :=
1076     }
1077     { nameptr #1 >
1078       { namesleft #1 = bbl.and "" = not and
1079         { bbl.and * }
1080         { ", " * }
1081         if$
1082       }
1083       'skip$
1084       if$
1085       s nameptr format.name *
1086     }
1087     if$
1088     nameptr #1 + 'nameptr :=
1089     namesleft #1 - 'namesleft :=
1090   }
1091 while$
1092 }
1093
1094 FUNCTION {format.key}
1095 { empty$
1096   { key field.or.null }
1097   { "" }
1098   if$
1099 }
1100
1101 FUNCTION {format.authors}
1102 { author empty$ not
1103   { author format.names }
1104   { "empty author in " cite$ * warning$
1105   <*author-year>
1106     bbl.anonymous
1107   </author-year>
1108   <*numerical>
1109     ""
1110   </numerical>
1111   }
1112   if$
1113 }
1114
1115 FUNCTION {format.editors}
1116 { editor empty$
1117   { "" }
1118   { editor format.names }
1119   if$
1120 }
1121
1122 FUNCTION {format.translators}
1123 { translator empty$
1124   { "" }
1125   { translator format.names
1126     entry.lang lang.zh =
1127     { translator num.names$ #3 >

```

```

1128         { " 译" * }
1129         { ", 译" * }
1130         if$
1131     }
1132     'skip$
1133     if$
1134 }
1135 if$
1136 }
1137
1138 FUNCTION {format.full.names}
1139 {'s :=
1140 #1 'nameptr :=
1141 s num.names$ 'numnames :=
1142 numnames 'namesleft :=
1143 { namesleft #0 > }
1144 { s nameptr "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
1145 t get.str.lang 'name.lang :=
1146 name.lang lang.en =
1147 { t #1 "{vv~}{ll}" format.name$ 't := }
1148 { t #1 "{ll}{ff}" format.name$ 't := }
1149 if$
1150 nameptr #1 >
1151 {
1152     namesleft #1 >
1153     { ", " * t * }
1154     {
1155         numnames #2 >
1156         { ", " * }
1157         'skip$
1158         if$
1159         t "others" =
1160         { " et~al." * }
1161         { " and " * t * }
1162         if$
1163     }
1164     if$
1165 }
1166 't
1167 if$
1168 nameptr #1 + 'nameptr :=
1169 namesleft #1 - 'namesleft :=
1170 }
1171 while$
1172 }
1173
1174 FUNCTION {author.editor.full}
1175 { author empty$
1176     { editor empty$
1177         { "" }
1178         { editor format.full.names }
1179     if$
1180 }
1181 { author format.full.names }
1182 if$

```

```

1183 }
1184
1185 FUNCTION {author.full}
1186 { author empty$
1187   { "" }
1188   { author format.full.names }
1189   if$
1190 }
1191
1192 FUNCTION {editor.full}
1193 { editor empty$
1194   { "" }
1195   { editor format.full.names }
1196   if$
1197 }
1198
1199 FUNCTION {make.full.names}
1200 { type$ "book" =
1201   type$ "inbook" = booktitle empty$ not and
1202   or
1203   'author.editor.full
1204   { type$ "collection" =
1205     type$ "proceedings" =
1206     or
1207     'editor.full
1208     'author.full
1209     if$
1210   }
1211   if$
1212 }
1213
1214 FUNCTION {output.bibitem}
1215 { newline$
1216   "\bibitem[" writes$
1217   label ")" *
1218   make.full.names duplicate$ short.list =
1219   { pop$ }
1220   { duplicate$ "]" contains
1221     { "{" swap$ * "}" * }
1222     'skip$
1223     if$
1224     *
1225   }
1226   if$
1227   "]" * writes$
1228   cite$ writes$
1229   "]" writes$
1230   newline$
1231   ""
1232   before.all 'output.state :=
1233 }
1234

```

B.4.3 Format title

The `format.title` function is used for non-book-like titles. For most styles we convert to lowercase (except for the very first letter, and except for the first one after a colon (followed by whitespace)), and hope the user has brace-surrounded words that need to stay capitalized; for some styles, however, we leave it as it is in the database.

```
1235 FUNCTION {change.sentence.case}
1236 { entry.lang lang.en =
1237   { "t" change.case$ }
1238   'skip$
1239   if$
1240 }
1241
1242 FUNCTION {add.link}
1243 { url empty$ not
1244   { "\href{" url * "}" * swap$ * "}" * }
1245   { doi empty$ not
1246     { "\href{https://doi.org/" doi * "}" * swap$ * "}" * }
1247     'skip$
1248     if$
1249   }
1250   if$
1251 }
1252
1253 FUNCTION {format.title}
1254 { title empty$
1255   { "" }
1256   { title
1257     sentence.case.title
1258     'change.sentence.case
1259     'skip$
1260     if$
1261     entry.numbered number empty$ not and
1262     { bbl.colon *
1263       type$ "patent" = show.patent.country and
1264       { address empty$ not
1265         { address * ", " * }
1266         { location empty$ not
1267           { location * ", " * }
1268           { entry.lang lang.zh =
1269             { " 中国" * ", " * }
1270             'skip$
1271             if$
1272           }
1273         }
1274       }
1275     }
1276     'skip$
1277     if$
1278     number *
1279   }
```



```

1280     }
1281     'skip$
1282     if$
1283     link.title
1284     'add.link
1285     'skip$
1286     if$
1287   }
1288   if$
1289 }
1290

```

For several functions we'll need to connect two strings with a tie (~) if the second one isn't very long (fewer than 3 characters). The `tie.or.space.connect` function does that. It concatenates the two strings on top of the stack, along with either a tie or space between them, and puts this concatenation back onto the stack:

```

tie.or.space.connect(str1,str2) ==
BEGIN
  if text.length$(str2) < 3
    then return the concatenation of str1, "~", and str2
    else return the concatenation of str1, " ", and str2
END

```

```

1291 FUNCTION {tie.or.space.connect}
1292 { duplicate$ text.length$ #3 <
1293   { "~" }
1294   { " " }
1295   if$
1296   swap$ * *
1297 }
1298

```

The `either.or.check` function complains if both fields or an either-or pair are nonempty.

```

either.or.check(t,s) ==
BEGIN
  if empty$(s) then
    warning$(can't use both " t * t * " fields in " * cite$ * warning$)
  fi
END

```

```

1299 FUNCTION {either.or.check}
1300 { empty$
1301   'pop$
1302   { "can't use both " swap$ * " fields in " * cite$ * warning$ }
1303   if$
1304 }
1305

```

The `format.bvolume` function is for formatting the volume and perhaps series name of a multivolume work. If both a volume and a series field are there, we

assume the series field is the title of the whole multivolume work (the title field should be the title of the thing being referred to), and we add an of <series>. This function is called in mid-sentence.

The format.number.series function is for formatting the series name and perhaps number of a work in a series. This function is similar to format.bvolume, although for this one the series must exist (and the volume must not exist). If the number field is empty we output either the series field unchanged if it exists or else the null string. If both the number and series fields are there we assume the series field gives the name of the whole series (the title field should be the title of the work being one referred to), and we add an in <series>. We capitilize Number when this function is used at the beginning of a block.

```

1306 FUNCTION {is.digit}
1307 { duplicate$ empty$
1308   { pop$ #0 }
1309   { chr.to.int$
1310     duplicate$ "0" chr.to.int$ <
1311     { pop$ #0 }
1312     { "9" chr.to.int$ >
1313       { #0 }
1314       { #1 }
1315     if$
1316   }
1317   if$
1318 }
1319 if$
1320 }
1321
1322 FUNCTION {is.number}
1323 { 's :=
1324   s empty$
1325   { #0 }
1326   { s text.length$ 'charptr :=
1327     { charptr #0 >
1328       s charptr #1 substring$ is.digit
1329       and
1330     }
1331     { charptr #1 - 'charptr := }
1332     while$
1333     charptr not
1334   }
1335   if$
1336 }
1337
1338 FUNCTION {format.volume}
1339 { volume empty$ not
1340   { volume is.number
1341     { entry.lang lang.zh =
1342       { " 第 " volume * " 卷" * }
1343       { "Vol." volume tie.or.space.connect }

```

```

1344         if$
1345     }
1346     { volume }
1347     if$
1348 }
1349 { "" }
1350 if$
1351 }
1352
1353 FUNCTION {format.number}
1354 { number empty$ not
1355   { number is.number
1356     { entry.lang lang.zh =
1357       { "第 " number * "册" * }
1358       { "No." number tie.or.space.connect }
1359     if$
1360   }
1361   { number }
1362 if$
1363 }
1364 { "" }
1365 if$
1366 }
1367
1368 FUNCTION {format.volume.number}
1369 { volume empty$ not
1370   { format.volume }
1371   { format.number }
1372 if$
1373 }
1374
1375 FUNCTION {format.title.vol.num}
1376 { title
1377   sentence.case.title
1378   'change.sentence.case
1379   'skip$
1380 if$
1381   entry.numbered
1382   { number empty$ not
1383     { bbl.colon * number * }
1384     'skip$
1385   if$
1386   }
1387   { format.volume.number 's :=
1388     s empty$ not
1389     { bbl.colon * s * }
1390     'skip$
1391   if$
1392   }
1393 if$
1394 }
1395
1396 FUNCTION {format.series.vol.num.title}
1397 { format.volume.number 's :=
1398   series empty$ not

```

```

1399 { series
1400     sentence.case.title
1401     'change.sentence.case
1402     'skip$
1403     if$
1404     entry.numbered
1405     { bbl.wide.space * }
1406     { bbl.colon *
1407     s empty$ not
1408     { s * bbl.wide.space * }
1409     'skip$
1410     if$
1411     }
1412     if$
1413     title *
1414     sentence.case.title
1415     'change.sentence.case
1416     'skip$
1417     if$
1418     entry.numbered number empty$ not and
1419     { bbl.colon * number * }
1420     'skip$
1421     if$
1422     }
1423     { format.title.vol.num }
1424     if$
1425     format.btitle
1426     link.title
1427     'add.link
1428     'skip$
1429     if$
1430 }
1431
1432 FUNCTION {format.booktitle.vol.num}
1433 { booktitle
1434     entry.numbered
1435     'skip$
1436     { format.volume.number 's :=
1437     s empty$ not
1438     { bbl.colon * s * }
1439     'skip$
1440     if$
1441     }
1442     if$
1443 }
1444
1445 FUNCTION {format.series.vol.num.booktitle}
1446 { format.volume.number 's :=
1447     series empty$ not
1448     { series bbl.colon *
1449     entry.numbered not s empty$ not and
1450     { s * bbl.wide.space * }
1451     'skip$
1452     if$
1453     booktitle *

```

```

1454     }
1455     { format.booktitle.vol.num }
1456     if$
1457     format.btitle
1458 }
1459
1460 FUNCTION {remove.period}
1461 { 't :=
1462   "" 's :=
1463   { t empty$ not }
1464   { t #1 #1 substring$ 'tmp.str :=
1465     tmp.str "." = not
1466     { s tmp.str * 's := }
1467     'skip$
1468     if$
1469     t #2 global.max$ substring$ 't :=
1470   }
1471   while$
1472   s
1473 }
1474
1475 FUNCTION {abbreviate}
1476 { remove.period
1477   't :=
1478   t "l" change.case$ 's :=
1479   ""
1480   s "physical review letters" =
1481     { "Phys Rev Lett" }
1482     'skip$
1483   if$
1484   's :=
1485   s empty$
1486     { t }
1487     { pop$ s }
1488   if$
1489 }
1490
1491 FUNCTION {get.journal.title}
1492 { short.journal
1493   { shortjournal empty$ not
1494     { shortjournal }
1495     { journal empty$ not
1496       { journal abbreviate }
1497       { journaltitle empty$ not
1498         { journaltitle abbreviate }
1499         { "" }
1500       }
1501     }
1502     if$
1503   }
1504   if$
1505 }
1506 { journal empty$ not
1507   { journal }
1508   { journaltitle empty$ not

```

```

1509         { journaltitle }
1510         { shortjournal empty$ not
1511           { shortjournal }
1512           { "" }
1513         if$
1514       }
1515     if$
1516   }
1517   if$
1518 }
1519 if$
1520 }
1521
1522 FUNCTION {check.arxiv.preprint}
1523 { #1 #5 substring$ purify$ "l" change.case$ "arxiv" =
1524   { #1 }
1525   { #0 }
1526   if$
1527 }
1528
1529 FUNCTION {format.journal}
1530 { get.journal.title
1531   duplicate$ empty$ not
1532   { italic.journal entry.lang lang.en = and
1533     'emphasize
1534     'skip$
1535     if$
1536     link.journal
1537     'add.link
1538     'skip$
1539     if$
1540   }
1541   'skip$
1542   if$
1543 }
1544

```

B.4.4 Format entry type mark

```

1545 FUNCTION {set.entry.mark}
1546 { entry.mark empty$ not
1547   'pop$
1548   { mark empty$ not
1549     { pop$ mark 'entry.mark := }
1550     { 'entry.mark := }
1551     if$
1552   }
1553   if$
1554 }
1555
1556 FUNCTION {format.mark}
1557 { show.mark
1558   { entry.mark
1559     show.medium.type

```

```

1560     { medium empty$ not
1561       { "/" * medium * }
1562       { entry.is.electronic
1563         { "/OL" * }
1564         'skip$
1565         if$
1566       }
1567     if$
1568   }
1569   'skip$
1570 if$
1571 'entry.mark :=
1572 space.before.mark
1573 { " " }
1574 { "\allowbreak" }
1575 if$
1576 "[" * entry.mark * "]" *
1577 }
1578 { "" }
1579 if$
1580 }
1581

```

B.4.5 Format edition

The `format.edition` function appends `edition` to the `edition`, if present. We lowercase the `edition` (it should be something like `Third`), because this doesn't start a sentence.

```

1582 FUNCTION {num.to.ordinal}
1583 { duplicate$ text.length$ 'charptr :=
1584   duplicate$ charptr #1 substring$ 's :=
1585   s "1" =
1586     { "st" * }
1587   { s "2" =
1588     { "nd" * }
1589     { s "3" =
1590       { "rd" * }
1591       { "th" * }
1592     if$
1593   }
1594   if$
1595 }
1596 if$
1597 }
1598
1599 FUNCTION {format.edition}
1600 { edition empty$
1601   { "" }
1602   { edition is.number
1603     { edition "1" = not
1604       { entry.lang lang.zh =
1605         { edition " 版" * }
1606         { edition num.to.ordinal " ed." * }
1607       if$

```

```

1608     }
1609     'skip$
1610     if$
1611   }
1612   { entry.lang lang.en =
1613     { edition change.sentence.case 's :=
1614       s "Revised" = s "Revised edition" = or
1615         { "Rev. ed." }
1616         { s " ed." * }
1617     if$
1618     }
1619     { edition }
1620   if$
1621 }
1622 if$
1623 }
1624 if$
1625 }
1626

```

B.4.6 Format publishing items

出版地址和出版社会有“[S.l.: s.n.]”的情况，所以必须一起处理。

```

1627 FUNCTION {format.publisher}
1628 { publisher empty$ not
1629   { publisher }
1630   { school empty$ not
1631     { school }
1632     { organization empty$ not
1633       { organization }
1634       { institution empty$ not
1635         { institution }
1636         { "" }
1637     if$
1638     }
1639   if$
1640   }
1641   if$
1642 }
1643 if$
1644 }
1645
1646 FUNCTION {format.address.publisher}
1647 { address empty$ not
1648   { address }
1649   { location empty$ not
1650     { location }
1651     { "" }
1652   if$
1653   }
1654 if$
1655 duplicate$ empty$ not
1656   { format.publisher empty$ not
1657     { bbl.colon * format.publisher * }

```



```

1658     { entry.is.electronic not show.missing.address.publisher and
1659       { bbl.colon * bbl.sine.nomine * }
1660       'skip$
1661     if$
1662   }
1663   if$
1664 }
1665 { pop$
1666   entry.is.electronic not show.missing.address.publisher and
1667   { format.publisher empty$ not
1668     { bbl.sine.loco bbl.colon * format.publisher * }
1669     { bbl.sine.loco.sine.nomine }
1670     if$
1671   }
1672   { format.publisher empty$ not
1673     { format.publisher }
1674     { "" }
1675     if$
1676   }
1677   if$
1678 }
1679 if$
1680 }
1681

```

B.4.7 Format date

The `format.date` function is for the month and year, but we give a warning if there's an empty year but the month is there, and we return the empty string if they're both empty.

期刊需要著录起止范围，其中年份使用“/”分隔，卷和期使用“-”分隔。版本 v2.0.2 前的年份也使用“-”分隔，仅提供兼容性，不再推荐。

```

1682 FUNCTION {extract.before.dash}
1683 { duplicate$ empty$
1684   { pop$ "" }
1685   { 's :=
1686     #1 'charptr :=
1687     s text.length$ #1 + 'len :=
1688     { charptr len <
1689       s charptr #1 substring$ "-" = not
1690       and
1691     }
1692     { charptr #1 + 'charptr := }
1693     while$
1694     s #1 charptr #1 - substring$
1695   }
1696   if$
1697 }
1698
1699 FUNCTION {extract.after.dash}
1700 { duplicate$ empty$
1701   { pop$ "" }

```

```

1702 { 's :=
1703     #1 'charptr :=
1704     s text.length$ #1 + 'len :=
1705     { charptr len <
1706         s charptr #1 substring$ "-" = not
1707         and
1708     }
1709     { charptr #1 + 'charptr := }
1710 while$
1711     { charptr len <
1712         s charptr #1 substring$ "-" =
1713         and
1714     }
1715     { charptr #1 + 'charptr := }
1716 while$
1717     s charptr global.max$ substring$
1718 }
1719 if$
1720 }
1721
1722 FUNCTION {extract.before.slash}
1723 { duplicate$ empty$
1724     { pop$ "" }
1725     { 's :=
1726         #1 'charptr :=
1727         s text.length$ #1 + 'len :=
1728         { charptr len <
1729             s charptr #1 substring$ "/" = not
1730             and
1731         }
1732         { charptr #1 + 'charptr := }
1733     while$
1734     s #1 charptr #1 - substring$
1735 }
1736 if$
1737 }
1738
1739 FUNCTION {extract.after.slash}
1740 { duplicate$ empty$
1741     { pop$ "" }
1742     { 's :=
1743         #1 'charptr :=
1744         s text.length$ #1 + 'len :=
1745         { charptr len <
1746             s charptr #1 substring$ "-" = not
1747             and
1748             s charptr #1 substring$ "/" = not
1749             and
1750         }
1751         { charptr #1 + 'charptr := }
1752     while$
1753     { charptr len <
1754         s charptr #1 substring$ "-" =
1755         s charptr #1 substring$ "/" =
1756         or

```

```

1757         and
1758     }
1759     { charptr #1 + 'charptr := }
1760 while$
1761 s charptr global.max$ substrings$
1762 }
1763 if$
1764 }
1765

```

著者-出版年制必须提取出年份

```

1766 FUNCTION {format.year}
1767 { year empty$ not
1768   { year extra.label * }
1769   { date empty$ not
1770     { date extract.before.dash extra.label * }
1771     { entry.is.electronic not
1772       { "empty year in " cite$ * warning$ }
1773       'skip$
1774       if$
1775       urldate empty$ not
1776         { "[" urldate extract.before.dash * extra.label * "]" * }
1777         { "" }
1778       if$
1779     }
1780   }
1781 }
1782 if$
1783 }
1784
1785 FUNCTION {format.periodical.year}
1786 { year empty$ not
1787   { year extract.before.slash
1788     "--" *
1789     year extract.after.slash
1790     duplicate$ empty$
1791     'pop$
1792     { * }
1793     if$
1794   }
1795   { date empty$ not
1796     { date extract.before.dash }
1797     { "empty year in " cite$ * warning$
1798       urldate empty$ not
1799         { "[" urldate extract.before.dash * "]" * }
1800         { "" }
1801       if$
1802     }
1803   }
1804 }
1805 if$
1806 }
1807

```

专利和报纸都是使用日期而不是年

```

1808 FUNCTION {format.date}
1809 { date empty$ not
1810   { type$ "patent" = type$ "newspaper" = or
1811     { date }
1812     { entrysubtype empty$ not
1813       { type$ "article" = entrysubtype "newspaper" = and
1814         { date }
1815         { format.year }
1816       }
1817     }
1818     { format.year }
1819   }
1820   if$
1821 }
1822 }
1823 { year empty$ not
1824   { format.year }
1825   { "" }
1826 }
1827 if$
1828 }
1829 }
1830

```

更新、修改日期只用于电子资源 `electronic`

```

1831 FUNCTION {format.editdate}
1832 { date empty$ not
1833   { "\allowbreak(" date * ")" * }
1834   { "" }
1835 }
1836 if$
1837 }

```

国标中的“引用日期”都是与 URL 同时出现的，所以其实为 `urldate`，这个虽然不是 BibTeX 标准的域，但是实际中很常见。

```

1838 FUNCTION {format.urldate}
1839 { show.urldate show.url and entry.url empty$ not and
1840   is.pure.electronic or
1841   urldate empty$ not and
1842   { "\allowbreak[" urldate * "]" * }
1843   { "" }
1844 }
1845 if$
1846 }

```

B.4.8 Format pages

By default, BibTeX sets the global integer variable `global.max$` to the BibTeX constant `glob_str_size`, the maximum length of a global string variable. Analogously, BibTeX sets the global integer variable `entry.max$` to `ent_str_size`, the maximum length of an entry string variable. The style designer may change these if necessary (but this is unlikely)

The n.dashify function makes each single '-' in a string a double '--' if it's not already

```

pseudoVAR: pageresult: STRING          (it's what's accumulated on the stack)

n.dashify(s) ==
BEGIN
  t := s
  pageresult := ""
  while (not empty$(t))
    do
      if (first character of t = "-")
        then
          if (next character isn't)
            then
              pageresult:=pageresult*"--"
              t:=twiththe"-"removed
            else
              while_(first_character_of_t="-")
                do
                  pageresult:=pageresult*"- "
                  t:=twiththe"-"removed
                od
              fi
            else
              pageresult:=pageresult*_the_first_character
              t:=twiththe_first_character_removed
              fi
            od
          return pageresult
        END

```

国标里页码范围的连接号使用 hyphen，需要将 dash 转为 hyphen。

```

1847 FUNCTION {hyphenate}
1848 { 't :=
1849   ""
1850   { t empty$ not }
1851   { t #1 #1 substring$ "-" =
1852     { wave.dash.in.pages
1853       { "~" * }
1854       { "-" * }
1855       if$
1856         { t #1 #1 substring$ "-" = }
1857         { t #2 global.max$ substring$ 't := }
1858       while$
1859     }
1860     { t #1 #1 substring$ *
1861       t #2 global.max$ substring$ 't :=
1862     }
1863     if$
1864   }
1865   while$
1866 }
1867

```

This function doesn't begin a sentence so pages isn't capitalized. Other functions that use this should keep that in mind.

```
1868 FUNCTION {format.pages}
1869 { pages empty$
1870   { "" }
1871   { pages hyphenate }
1872   if$
1873 }
1874
1875 FUNCTION {format.extracted.pages}
1876 { pages empty$
1877   { "" }
1878   { pages
1879     only.start.page
1880     'extract.before.dash
1881     'hyphenate
1882     if$
1883   }
1884   if$
1885 }
1886
```

The `format.vol.num.pages` function is for the volume, number, and page range of a journal article. We use the format: `vol(number):pages`, with some variations for empty fields. This doesn't begin a sentence.

报纸在卷号缺失时，期号与前面的日期直接相连，所以必须拆开输出。

```
1887 FUNCTION {format.journal.volume}
1888 { volume empty$ not
1889   { bold.journal.volume
1890     { "\textbf{" volume * "" * }
1891     { volume }
1892     if$
1893   }
1894   { "" }
1895   if$
1896 }
1897
1898 FUNCTION {format.journal.number}
1899 { number empty$ not
1900   { "\allowbreak (" number * "" * }
1901   { "" }
1902   if$
1903 }
1904
1905 FUNCTION {format.journal.pages}
1906 { pages empty$
1907   { "" }
1908   { format.extracted.pages }
1909   if$
1910 }
1911
```

连续出版物的年卷期有起止范围，需要特殊处理

```

1912 FUNCTION {format.periodical.year.volume.number}
1913 { year empty$ not
1914   { year extract.before.slash }
1915   { "empty year in periodical " cite$ * warning$ }
1916   if$
1917   volume empty$ not
1918   { ", " * volume extract.before.dash * }
1919   'skip$
1920   if$
1921   number empty$ not
1922   { "\allowbreak (" * number extract.before.dash * ")" * }
1923   'skip$
1924   if$
1925   "--" *
1926   year extract.after.slash empty$
1927   volume extract.after.dash empty$ and
1928   number extract.after.dash empty$ and not
1929   { year extract.after.slash empty$ not
1930     { year extract.after.slash * }
1931     { year extract.before.slash * }
1932     if$
1933     volume empty$ not
1934     { ", " * volume extract.after.dash * }
1935     'skip$
1936     if$
1937     number empty$ not
1938     { "\allowbreak (" * number extract.after.dash * ")" * }
1939     'skip$
1940     if$
1941   }
1942   'skip$
1943   if$
1944 }
1945

```

B.4.9 Format url and doi

传统的 Bib_TE_X 习惯使用 howpublished 著录 url，这里提供支持。

```

1946 FUNCTION {check.url}
1947 { url empty$ not
1948   { url 'entry.url :=
1949     #1 'entry.is.electronic :=
1950   }
1951   { howpublished empty$ not
1952     { howpublished #1 #5 substring$ "\url{" =
1953       { howpublished 'entry.url :=
1954         #1 'entry.is.electronic :=
1955       }
1956       'skip$
1957     if$
1958   }
1959   { note empty$ not
1960     { note #1 #5 substring$ "\url{" =
1961       { note 'entry.url :=

```

```

1962             #1 'entry.is.electronic :=
1963             }
1964             'skip$
1965             if$
1966             }
1967             'skip$
1968             if$
1969             }
1970             if$
1971             }
1972             if$
1973             }
1974
1975 FUNCTION {output.url}
1976 { show.url is.pure.electronic or
1977   entry.url empty$ not and
1978   { new.block
1979     entry.url #1 #5 substring$ "\url{" =
1980     { entry.url }
1981     { "\url{" entry.url * "}" * }
1982     if$
1983     output
1984   }
1985   'skip$
1986   if$
1987 }
1988

```

需要检测 DOI 是否已经包含在 URL 中。

```

1989 FUNCTION {check.doi}
1990 { doi empty$ not
1991   { #1 'entry.is.electronic := }
1992   'skip$
1993   if$
1994 }
1995
1996 FUNCTION {is.in.url}
1997 { 's :=
1998   s empty$
1999   { #1 }
2000   { entry.url empty$
2001     { #0 }
2002     { s text.length$ 'len :=
2003       entry.url "l" change.case$ text.length$ 'charptr :=
2004         { entry.url "l" change.case$ charptr len substring$ s "l" change.case$ = not
2005           charptr #0 >
2006           and
2007         }
2008         { charptr #1 - 'charptr := }
2009         while$
2010         charptr
2011       }
2012       if$
2013     }
2014     if$

```



```

2015 }
2016
2017 FUNCTION {format.doi}
2018 { ""
2019   doi empty$ not
2020   { "" 's :=
2021     doi 't :=
2022     #0 'numnames :=
2023     { t empty$ not}
2024     { t #1 #1 substring$ 'tmp.str :=
2025       tmp.str "," = tmp.str " " = or t #2 #1 substring$ empty$ or
2026       { t #2 #1 substring$ empty$
2027         { s tmp.str * 's := }
2028         'skip$
2029         if$
2030         s empty$ s is.in.url or
2031         'skip$
2032         { numnames #1 + 'numnames :=
2033           numnames #1 >
2034             { ", " * }
2035             { "DOI: " * }
2036             if$
2037             "\doi{" s * "}" * *
2038           }
2039           if$
2040           "" 's :=
2041         }
2042         { s tmp.str * 's := }
2043         if$
2044         t #2 global.max$ substring$ 't :=
2045       }
2046       while$
2047     }
2048     'skip$
2049   if$
2050 }
2051
2052 FUNCTION {output.doi}
2053 { doi empty$ not show.doi and
2054   show.english.translation entry.lang lang.zh = and not and
2055   { new.block
2056     format.doi output
2057   }
2058   'skip$
2059   if$
2060 }
2061
2062 FUNCTION {check.electronic}
2063 { "" 'entry.url :=
2064   #0 'entry.is.electronic :=
2065   'check.doi
2066   'skip$
2067   if$
2068   'check.url
2069   'skip$

```

```

2070 if$
2071 medium empty$ not
2072   { medium "MT" = medium "DK" = or medium "CD" = or medium "OL" = or
2073     { #1 'entry.is.electronic := }
2074     'skip$
2075     if$
2076     }
2077     'skip$
2078 if$
2079 }
2080
2081 FUNCTION {format.eprint}
2082 { archivePrefix empty$ not
2083   { archivePrefix }
2084   { eprinttype empty$ not
2085     { archivePrefix }
2086     { "" }
2087     if$
2088   }
2089   if$
2090   's :=
2091   s empty$ not
2092   { s ": \eprint{" *
2093     url empty$ not
2094     { url }
2095     { "https://" s "l" change.case$ * ".org/abs/" * eprint * }
2096     if$
2097     * "){" *
2098     eprint * "}" *
2099   }
2100   { eprint }
2101 if$
2102 }
2103
2104 FUNCTION {output.eprint}
2105 { show.preprint eprint empty$ not and
2106   { new.block
2107     format.eprint output
2108   }
2109   'skip$
2110 if$
2111 }
2112
2113 FUNCTION {format.note}
2114 { note empty$ not show.note and
2115   { note }
2116   { "" }
2117 if$
2118 }
2119
2120 FUNCTION {output.translation}
2121 { show.english.translation entry.lang lang.zh = and
2122   { translation empty$ not
2123     { translation }
2124     { "[English translation missing!]" }

```

```

2125     if$
2126     " (in Chinese)" * output
2127     write$
2128     format.doi duplicate$ empty$ not
2129     { newline$
2130     write$
2131     }
2132     'pop$
2133     if$
2134     " \" write$
2135     newline$
2136     "(" write$
2137     ""
2138     before.all 'output.state :=
2139     }
2140     'skip$
2141 if$
2142 }
2143

```

The function `empty.misc.check` complains if all six fields are empty, and if there's been no sorting or alphabetic-label complaint.

```

2144 FUNCTION {empty.misc.check}
2145 { author empty$ title empty$
2146   year empty$
2147   and and
2148   key empty$ not and
2149   { "all relevant fields are empty in " cite$ * warning$ }
2150   'skip$
2151 if$
2152 }
2153

```

B.5 Functions for all entry types

Now we define the type functions for all entry types that may appear in the .BIB file—e.g., functions like ‘article’ and ‘book’. These are the routines that actually generate the .BBL-file output for the entry. These must all precede the READ command. In addition, the style designer should have a function ‘default.type’ for unknown types. Note: The fields (within each list) are listed in order of appearance, except as described for an ‘inbook’ or a ‘proceedings’.

B.5.1 专著

```

2154 FUNCTION {monograph}
2155 { output.bibitem
2156   output.translation
2157   author empty$ not
2158   { format.authors }
2159   { editor empty$ not

```

```

2160         { format.editors }
2161         { "empty author and editor in " cite$ * warnings$
2162 <*author-year>
2163         bbl.anonymous
2164 </author-year>
2165 <*numerical>
2166         ""
2167 </numerical>
2168     }
2169     if$
2170 }
2171 if$
2172 output
2173 year.after.author
2174 { period.after.author
2175     'new.sentence
2176     'skip$
2177     if$
2178     format.year "year" output.check
2179 }
2180 'skip$
2181 if$
2182 new.block
2183 format.series.vol.num.title "title" output.check
2184 "M" set.entry.mark
2185 format.mark "" output.after
2186 new.block
2187 format.translators output
2188 new.sentence
2189 format.edition output
2190 new.block
2191 format.address.publisher output
2192 year.after.author not
2193 { format.year "year" output.check }
2194 'skip$
2195 if$
2196 format.pages bbl.pages.colon output.after
2197 format.urldate "" output.after
2198 output.url
2199 output.doi
2200 new.block
2201 format.note output
2202 fin.entry
2203 }
2204

```

B.5.2 专著中的析出文献

An incollection is like inbook, but where there is a separate title for the referenced thing (and perhaps an editor for the whole). An incollection may CROSSREF a book.

Required: author, title, booktitle, publisher, year

Optional: editor, volume or number, series, type, chapter, pages, address, edi-

tion, month, note

```
2205 FUNCTION {incollection}
2206 { output.bibitem
2207   output.translation
2208   format.authors output
2209   author format.key output
2210   year.after.author
2211   { period.after.author
2212     'new.sentence
2213     'skip$
2214     if$
2215     format.year "year" output.check
2216   }
2217   'skip$
2218   if$
2219   new.block
2220   format.title "title" output.check
2221   "M" set.entry.mark
2222   format.mark "" output.after
2223   new.block
2224   format.translators output
2225   new.slash
2226   format.editors output
2227   new.block
2228   format.series.vol.num.booktitle "booktitle" output.check
2229   new.block
2230   format.edition output
2231   new.block
2232   format.address.publisher output
2233   year.after.author not
2234   { format.year "year" output.check }
2235   'skip$
2236   if$
2237   format.extracted.pages bbl.pages.colon output.after
2238   format.urldate "" output.after
2239   output.url
2240   output.doi
2241   new.block
2242   format.note output
2243   fin.entry
2244 }
2245
```

B.5.3 连续出版物

```
2246 FUNCTION {periodical}
2247 { output.bibitem
2248   output.translation
2249   format.authors output
2250   author format.key output
2251   year.after.author
2252   { period.after.author
2253     'new.sentence
2254     'skip$
```

```

2255     if$
2256     format.year "year" output.check
2257   }
2258   'skip$
2259 if$
2260 new.block
2261 format.title "title" output.check
2262 "J" set.entry.mark
2263 format.mark "" output.after
2264 new.block
2265 format.periodical.year.volume.number output
2266 new.block
2267 format.address.publisher output
2268 year.after.author not
2269   { format.periodical.year "year" output.check }
2270   'skip$
2271 if$
2272 format.urldate "" output.after
2273 output.url
2274 output.doi
2275 new.block
2276 format.note output
2277 fin.entry
2278 }
2279

```

B.5.4 连续出版物中的析出文献

The article function is for an article in a journal. An article may CROSSREF another article.

Required fields: author, title, journal, year

Optional fields: volume, number, pages, month, note

The other entry functions are all quite similar, so no comment version will be given for them.

```

2280 FUNCTION {journal.article}
2281 { output.bibitem
2282   output.translation
2283   format.authors output
2284   author format.key output
2285   year.after.author
2286   { period.after.author
2287     'new.sentence
2288     'skip$
2289     if$
2290     format.year "year" output.check
2291   }
2292   'skip$
2293 if$
2294 new.block
2295 title.in.journal
2296   { format.title "title" output.check
2297     entrysubtype empty$ not

```

```

2298     {
2299         entrysubtype "newspaper" =
2300             { "N" set.entry.mark }
2301             { "J" set.entry.mark }
2302         if$
2303     }
2304     { "J" set.entry.mark }
2305     if$
2306     format.mark "" output.after
2307     new.block
2308 }
2309 'skip$
2310 if$
2311 format.journal "journal" output.check
2312 year.after.author not
2313     { format.date "year" output.check }
2314     'skip$
2315 if$
2316 format.journal.volume output
2317 format.journal.number "" output.after
2318 format.journal.pages bbl.pages.colon output.after
2319 format.urldate "" output.after
2320 output.url
2321 output.doi
2322 new.block
2323 format.note output
2324 fin.entry
2325 }
2326

```

B.5.5 专利文献

number 域也可以用来表示专利号。

```

2327 FUNCTION {patent}
2328 { output.bibitem
2329   output.translation
2330   format.authors output
2331   author format.key output
2332   year.after.author
2333     { period.after.author
2334       'new.sentence
2335       'skip$
2336     if$
2337     format.year "year" output.check
2338   }
2339   'skip$
2340 if$
2341 new.block
2342 format.title "title" output.check
2343 "P" set.entry.mark
2344 format.mark "" output.after
2345 new.block
2346 format.date "year" output.check
2347 format.urldate "" output.after

```

```

2348 output.url
2349 output.doi
2350 new.block
2351 format.note output
2352 fin.entry
2353 }
2354

```

B.5.6 电子资源

```

2355 FUNCTION {electronic}
2356 { #1 #1 check.electronic
2357   #1 'entry.is.electronic :=
2358   #1 'is.pure.electronic :=
2359   output.bibitem
2360   output.translation
2361   format.authors output
2362   author format.key output
2363   year.after.author
2364     { period.after.author
2365       'new.sentence
2366       'skip$
2367       if$
2368       format.year "year" output.check
2369     }
2370   'skip$
2371   if$
2372   new.block
2373   format.series.vol.num.title "title" output.check
2374   "EB" set.entry.mark
2375   format.mark "" output.after
2376   new.block
2377   format.address.publisher output
2378   year.after.author not
2379     { date empty$
2380       { format.date output }
2381       'skip$
2382       if$
2383     }
2384   'skip$
2385   if$
2386   format.pages bbl.pages.colon output.after
2387   format.editdate "" output.after
2388   format.urldate "" output.after
2389   output.url
2390   output.doi
2391   new.block
2392   format.note output
2393   fin.entry
2394 }
2395

```

B.5.7 预印本

```

2396 FUNCTION {preprint}

```



```

2397 { output.bibitem
2398   output.translation
2399   author empty$ not
2400     { format.authors }
2401     { editor empty$ not
2402       { format.editors }
2403       { "empty author and editor in " cite$ * warnings$
2404 <*author-year>
2405       bbl.anonymous
2406 </author-year>
2407 <*numerical>
2408       ""
2409 </numerical>
2410     }
2411   if$
2412 }
2413 if$
2414 output
2415 year.after.author
2416   { period.after.author
2417     'new.sentence
2418     'skip$
2419     if$
2420     format.year "year" output.check
2421   }
2422   'skip$
2423 if$
2424 new.block
2425 title.in.journal
2426   { format.series.vol.num.title "title" output.check
2427 <*2015>
2428     "A" set.entry.mark
2429 </2015>
2430 <!*2015>
2431     "Z" set.entry.mark
2432 </!2015>
2433     format.mark "" output.after
2434     new.block
2435   }
2436   'skip$
2437 if$
2438 format.translators output
2439 new.sentence
2440 format.edition output
2441 new.block
2442 year.after.author not
2443   { date empty$
2444     { format.date output }
2445     'skip$
2446     if$
2447   }
2448   'skip$
2449 if$
2450 format.pages bbl.pages.colon output.after
2451 format.editdate "" output.after

```

```

2452 format.urldate "" output.after
2453 output.eprint
2454 output.url
2455 show.preprint not eprint empty$ or
2456   'output.doi
2457   'skip$
2458 if$
2459 new.block
2460 format.note output
2461 fin.entry
2462 }
2463

```

B.5.8 其他文献类型

A misc is something that doesn't fit elsewhere.

Required: at least one of the 'optional' fields

Optional: author, title, howpublished, month, year, note

Misc 用来自动判断类型。

```

2464 FUNCTION {misc}
2465 { get.journal.title
2466   duplicate$ empty$ not
2467   { check.arxiv.preprint
2468     'preprint
2469     'journal.article
2470     if$
2471   }
2472   { pop$
2473     booktitle empty$ not
2474     'incollection
2475     { publisher empty$ not
2476       'monograph
2477       { eprint empty$ not archivePrefix empty$ not or
2478         'preprint
2479         { entry.is.electronic
2480           'electronic
2481           {
2482             <!*2005>
2483               "Z" set.entry.mark
2484             </!*2005>
2485             <*2005>
2486               "M" set.entry.mark
2487             </2005>
2488             monograph
2489           }
2490         }
2491       }
2492     }
2493   }
2494   if$
2495 }
2496 if$
2497 }

```

```

2498   if$
2499   empty.misc.check
2500 }
2501
2502 FUNCTION {archive}
2503 { "A" set.entry.mark
2504   misc
2505 }
2506
2507 FUNCTION {article} { misc }
2508

```

The book function is for a whole book. A book may CROSSREF another book.

Required fields: author or editor, title, publisher, year

Optional fields: volume or number, series, address, edition, month, note

```

2509 FUNCTION {book} { monograph }
2510

```

A booklet is a bound thing without a publisher or sponsoring institution.

Required: title

Optional: author, howpublished, address, month, year, note

```

2511 FUNCTION {booklet} { book }
2512

```

```

2513 FUNCTION {collection}
2514 { "G" set.entry.mark
2515   monograph
2516 }
2517

```

```

2518 FUNCTION {database}
2519 { "DB" set.entry.mark
2520   electronic
2521 }
2522

```

```

2523 FUNCTION {dataset}
2524 { "DS" set.entry.mark
2525   electronic
2526 }
2527

```

An inbook is a piece of a book: either a chapter and/or a page range. It may CROSSREF a book. If there's no volume field, the type field will come before number and series.

Required: author or editor, title, chapter and/or pages, publisher, year

Optional: volume or number, series, type, address, edition, month, note

原生 BibTeX 的数据模型中 @inbook 不含 booktitle，按照“专著”处理。而 biblatex 的 @inbook 跟 incollection 一样。按照“专著的析出文献”处理。

```

2528 FUNCTION {inbook} {
2529   booktitle empty$
2530   'book
2531   'incollection

```

```
2532   if$
2533 }
2534
```

An inproceedings is an article in a conference proceedings, and it may CROSS-REF a proceedings. If there's no address field, the month (& year) will appear just before note.

Required: author, title, booktitle, year

Optional: editor, volume or number, series, pages, address, month, organization, publisher, note

```
2535 FUNCTION {inproceedings}
2536 { "C" set.entry.mark
2537   incollection
2538 }
2539
```

The conference function is included for Scribe compatibility.

```
2540 FUNCTION {conference} { inproceedings }
2541
2542 FUNCTION {legislation} { archive }
2543
2544
2545 FUNCTION {map}
2546 { "CM" set.entry.mark
2547   misc
2548 }
2549
```

A manual is technical documentation.

Required: title

Optional: author, organization, address, edition, month, year, note

```
2550 FUNCTION {manual} { monograph }
2551
```

A mastersthesis is a Master's thesis.

Required: author, title, school, year

Optional: type, address, month, note

```
2552 FUNCTION {mastersthesis}
2553 { "D" set.entry.mark
2554   monograph
2555 }
2556
2557 FUNCTION {newspaper}
2558 { "N" set.entry.mark
2559   article
2560 }
2561
2562 FUNCTION {online}
2563 { "EB" set.entry.mark
2564   electronic

```

2565 }
2566

A phdthesis is like a mastersthesis.

Required: author, title, school, year

Optional: type, address, month, note

2567 FUNCTION {phdthesis} { mastersthesis }
2568

A proceedings is a conference proceedings. If there is an organization but no editor field, the organization will appear as the first optional field (we try to make the first block nonempty); if there's no address field, the month (& year) will appear just before note.

Required: title, year

Optional: editor, volume or number, series, address, month, organization, publisher, note

2569 FUNCTION {proceedings}
2570 { "C" set.entry.mark
2571 monograph
2572 }

2573
2574 FUNCTION {software}
2575 { "CP" set.entry.mark
2576 electronic
2577 }

2578
2579 FUNCTION {standard}
2580 { "S" set.entry.mark
2581 misc
2582 }

2583

A techreport is a technical report.

Required: author, title, institution, year

Optional: type, number, address, month, note

2584 FUNCTION {techreport}
2585 { "R" set.entry.mark
2586 misc
2587 }
2588

An unpublished is something that hasn't been published.

Required: author, title, note

Optional: month, year

2589 FUNCTION {unpublished} { misc }
2590

We use entry type ‘misc’ for an unknown type; BibTeX gives a warning.

```
2591 FUNCTION {default.type} { misc }  
2592
```

B.6 Common macros

Here are macros for common things that may vary from style to style. Users are encouraged to use these macros.

Months are either written out in full or abbreviated

```
2593 MACRO {jan} {"January"}  
2594  
2595 MACRO {feb} {"February"}  
2596  
2597 MACRO {mar} {"March"}  
2598  
2599 MACRO {apr} {"April"}  
2600  
2601 MACRO {may} {"May"}  
2602  
2603 MACRO {jun} {"June"}  
2604  
2605 MACRO {jul} {"July"}  
2606  
2607 MACRO {aug} {"August"}  
2608  
2609 MACRO {sep} {"September"}  
2610  
2611 MACRO {oct} {"October"}  
2612  
2613 MACRO {nov} {"November"}  
2614  
2615 MACRO {dec} {"December"}  
2616
```

Journals are either written out in full or abbreviated; the abbreviations are like those found in ACM publications.

To get a completely different set of abbreviations, it may be best to make a separate .bib file with nothing but those abbreviations; users could then include that file name as the first argument to the \bibliography command

```
2617 MACRO {acmcs} {"ACM Computing Surveys"}  
2618  
2619 MACRO {acta} {"Acta Informatica"}  
2620  
2621 MACRO {cacm} {"Communications of the ACM"}  
2622  
2623 MACRO {ibmjrd} {"IBM Journal of Research and Development"}  
2624  
2625 MACRO {ibmsj} {"IBM Systems Journal"}  
2626  
2627 MACRO {ieeese} {"IEEE Transactions on Software Engineering"}
```

```

2628
2629 MACRO {ieeetc} {"IEEE Transactions on Computers"}
2630
2631 MACRO {ieeetcad}
2632 {"IEEE Transactions on Computer-Aided Design of Integrated Circuits"}
2633
2634 MACRO {ipl} {"Information Processing Letters"}
2635
2636 MACRO {jacm} {"Journal of the ACM"}
2637
2638 MACRO {jcscs} {"Journal of Computer and System Sciences"}
2639
2640 MACRO {scp} {"Science of Computer Programming"}
2641
2642 MACRO {sicomp} {"SIAM Journal on Computing"}
2643
2644 MACRO {tocs} {"ACM Transactions on Computer Systems"}
2645
2646 MACRO {tods} {"ACM Transactions on Database Systems"}
2647
2648 MACRO {tog} {"ACM Transactions on Graphics"}
2649
2650 MACRO {toms} {"ACM Transactions on Mathematical Software"}
2651
2652 MACRO {toois} {"ACM Transactions on Office Information Systems"}
2653
2654 MACRO {toplas} {"ACM Transactions on Programming Languages and Systems"}
2655
2656 MACRO {tcs} {"Theoretical Computer Science"}
2657

```

B.7 Format labels

The `sortify` function converts to lower case after purifying; it's used in sorting and in computing alphabetic labels after sorting

The `chop.word(w,len,s)` function returns either `s` or, if the first `len` letters of `s` equals `w` (this comparison is done in the third line of the function's definition), it returns that part of `s` after `w`.

```

2658 FUNCTION {sortify}
2659 { purify$
2660   "l" change.case$
2661 }
2662

```

We need the `chop.word` stuff for the dubious `unsorted-list-with-labels` case.

```

2663 FUNCTION {chop.word}
2664 { 's :=
2665   'len :=
2666   s #1 len substring$ =
2667   { s len #1 + global.max$ substring$ }
2668   's

```

```

2669   if$
2670 }
2671

```

The `format.lab.names` function makes a short label by using the initials of the von and Last parts of the names (but if there are more than four names, (i.e., people) it truncates after three and adds a superscripted +; it also adds such a + if the last of multiple authors is others). If there is only one name, and its von and Last parts combined have just a single name-token (Knuth has a single token, Brinch Hansen has two), we take the first three letters of the last name. The boolean `et.al.char.used` tells whether we've used a superscripted +, so that we know whether to include a LaTeX macro for it.

```

format.lab.names(s) ==
BEGIN
  numnames := num.names$(s)
  if numnames > 1 then
    if numnames > 4 then
      namesleft := 3
    else
      namesleft := numnames
    nameptr := 1
    namerresult := ""
    while namesleft > 0
      do
        if (name_ptr = numnames) and
           format.name$(s, nameptr, "{ff}_{vv}_{ll}_{jj}") = "others"
        then namerresult := namerresult * "{\etalchar{+}}"
           et.al.char.used := true
        else namerresult := namerresult *
             format.name$(s, nameptr, "{v}{l}")
           nameptr := nameptr + 1
           namesleft := namesleft - 1
        od
      if numnames > 4 then
        namerresult := namerresult * "{\etalchar{+}}"
        et.al.char.used := true
      else
        t := format.name$(s, 1, "{v}{l}")
        if text.length$(t) < 2 then % there's just one name-token
          namerresult := text.prefix$(format.name$(s, 1, "{ll}"), 3)
        else
          namerresult := t
        fi
      fi
    return namerresult
  END

```

Exactly what fields we look at in constructing the primary part of the label depends on the entry type; this selectivity (as opposed to, say, always looking at author, then editor, then key) helps ensure that ignored fields, as described in the

LaTeX book, really are ignored. Note that MISC is part of the deepest ‘else’ clause in the nested part of calc.label; thus, any unrecognized entry type in the database is handled correctly.

There is one auxiliary function for each of the four different sequences of fields we use. The first of these functions looks at the author field, and then, if necessary, the key field. The other three functions, which might look at two fields and the key field, are similar, except that the key field takes precedence over the organization field (for labels—not for sorting).

The calc.label function calculates the preliminary label of an entry, which is formed by taking three letters of information from the author or editor or key or organization field (depending on the entry type and on what’s empty, but ignoring a leading The in the organization), and appending the last two characters (digits) of the year. It is an error if the appropriate fields among author, editor, organization, and key are missing, and we use the first three letters of the cite\$ in desperation when this happens. The resulting label has the year part, but not the name part, purify\$ed (purify\$ing the year allows some sorting shenanigans by the user).

This function also calculates the version of the label to be used in sorting.

The final label may need a trailing ‘a’, ‘b’, etc., to distinguish it from otherwise identical labels, but we can’t calculate those extra.labels until after sorting.

```

calc.label ==
BEGIN
  if type$ = "book" or "inbook" then
    author.editor.key.label
  else if type$ = "proceedings" then
    editor.key.organization.label
  else if type$ = "manual" then
    author.key.organization.label
  else
    author.key.label
  fi fi fi
  label := label * substring$(purify$(field.or.null(year)), -1, 2)
  % assuming we will also sort, we calculate a sort.label
  sort.label := sortify(label), but use the last four, not two, digits
END

```

```

2672 FUNCTION {format.lab.name}
2673 { "{vv~}{ll}{, jj}{, ff}" format.name$ 't :=
2674   t "others" =
2675   { citation.et.al }
2676   { t get.str.lang 'name.lang :=
2677     name.lang lang.zh = name.lang lang.ja = or
2678     { t #1 "{ll}{ff}" format.name$ }
2679     { t #1 "{vv~}{ll}" format.name$ }
2680   if$
2681   }

```

```

2682   if$
2683 }
2684
2685 FUNCTION {format.lab.names}
2686 { 's :=
2687   #1 'nameptr :=
2688   s num.names$ 'numnames :=
2689   ""
2690   numnames 'namesleft :=
2691   { namesleft #0 > }
2692   { s nameptr format.lab.name citation.et.al =
2693     numnames citation.et.al.min #1 - > nameptr citation.et.al.use.first > and or
2694     { bbl.space *
2695       citation.et.al *
2696       #1 'namesleft :=
2697     }
2698     { nameptr #1 >
2699       { namesleft #1 = citation.and "" = not and
2700         { citation.and * }
2701         { ", " * }
2702         if$
2703       }
2704       'skip$
2705       if$
2706       s nameptr format.lab.name *
2707     }
2708     if$
2709     nameptr #1 + 'nameptr :=
2710     namesleft #1 - 'namesleft :=
2711   }
2712   while$
2713 }
2714
2715 FUNCTION {author.key.label}
2716 { author empty$
2717   { key empty$
2718     { cite$ #1 #3 substring$ }
2719     'key
2720     if$
2721   }
2722   { author format.lab.names }
2723   if$
2724 }
2725
2726 FUNCTION {author.editor.key.label}
2727 { author empty$
2728   { editor empty$
2729     { key empty$
2730       { cite$ #1 #3 substring$ }
2731       'key
2732       if$
2733     }
2734     { editor format.lab.names }
2735     if$
2736   }

```

```

2737     { author format.lab.names }
2738   if$
2739 }
2740
2741 FUNCTION {author.key.organization.label}
2742 { author empty$
2743   { key empty$
2744     { organization empty$
2745       { cite$ #1 #3 substring$ }
2746       { "The " #4 organization chop.word #3 text.prefix$ }
2747       if$
2748     }
2749     'key
2750   if$
2751 }
2752 { author format.lab.names }
2753 if$
2754 }
2755
2756 FUNCTION {editor.key.organization.label}
2757 { editor empty$
2758   { key empty$
2759     { organization empty$
2760       { cite$ #1 #3 substring$ }
2761       { "The " #4 organization chop.word #3 text.prefix$ }
2762       if$
2763     }
2764     'key
2765   if$
2766 }
2767 { editor format.lab.names }
2768 if$
2769 }
2770
2771 FUNCTION {calc.short.authors}
2772 { type$ "book" =
2773   type$ "inbook" = booktitle empty$ not and
2774   or
2775   'author.editor.key.label
2776   { type$ "collection" =
2777     type$ "proceedings" =
2778     or
2779     { editor empty$ not
2780       'editor.key.organization.label
2781       'author.key.organization.label
2782       if$
2783     }
2784     'author.key.label
2785   if$
2786 }
2787 if$
2788 'short.list :=
2789 }
2790

```

如果 `label` 中有中括号“`[`”，分别用大括号保护起来，防止 `\bibitem` 处理出错。另外为了兼容 `bibunits`，“`name(year)fullname`”的每一项都要分别保护起来，参考 [tuna/thuthesis/#630](#)。

```

2791 FUNCTION {calc.label}
2792 { calc.short.authors
2793   short.list "]" contains
2794   { "{" short.list * "]" * }
2795   { short.list }
2796   if$
2797   "("
2798   *
2799   format.year duplicate$ empty$
2800   short.list key field.or.null = or
2801     { pop$ "" }
2802     'skip$
2803   if$
2804   duplicate$ "]" contains
2805   { "{" swap$ * "]" * }
2806   'skip$
2807   if$
2808   *
2809   'label :=
2810 }
2811

```

B.8 Sorting

When sorting, we compute the sortkey by executing `presort` on each entry. The `presort` key contains a number of sortified strings, concatenated with multiple blanks between them. This makes things like `brinch per` come before `brinch hansen per`.

The fields used here are: the `sort.label` for alphabetic labels (as set by `calc.label`), followed by the author names (or editor names or organization (with a leading `The` removed) or key field, depending on entry type and on what's empty), followed by year, followed by the first bit of the title (chopping off a leading `The`, `A`, or `An`). Names are formatted: Von Last First Junior. The names within a part will be separated by a single blank (such as `brinch hansen`), two will separate the name parts themselves (except the von and last), three will separate the names, four will separate the names from year (and from label, if alphabetic), and four will separate year from title.

The `sort.format.names` function takes an argument that should be in BibTeX name format, and returns a string containing `-`separated names in the format described above. The function is almost the same as `format.names`.

```

2812 (*author-year)

```

```

2813 FUNCTION {sort.language.label}
2814 { entry.lang lang.zh =
2815   { lang.zh.order }
2816   { entry.lang lang.ja =
2817     { lang.ja.order }
2818     { entry.lang lang.en =
2819       { lang.en.order }
2820       { entry.lang lang.ru =
2821         { lang.ru.order }
2822         { lang.other.order }
2823         if$
2824       }
2825     }
2826   }
2827   if$
2828 }
2829 if$
2830 #64 +
2831 int.to.chr$
2832 }
2833
2834 FUNCTION {sort.format.names}
2835 { 's :=
2836   #1 'nameptr :=
2837   ""
2838   s num.names$ 'numnames :=
2839   numnames 'namesleft :=
2840   { namesleft #0 > }
2841   {
2842     s nameptr "{vv{ }}{ll{ }}{ ff{ }}{ jj{ }}" format.name$ 't :=
2843     nameptr #1 >
2844     {
2845       " " *
2846       namesleft #1 = t "others" = and
2847       { "zzzz" * }
2848       { numnames #2 > nameptr #2 = and
2849         { "zz" * year field.or.null * " " * }
2850         'skip$
2851         if$
2852         t sortify *
2853       }
2854       if$
2855     }
2856     { t sortify * }
2857     if$
2858     nameptr #1 + 'nameptr :=
2859     namesleft #1 - 'namesleft :=
2860   }
2861   while$
2862 }
2863

```

The `sort.format.title` function returns the argument, but first any leading A 's, An 's, or The 's are removed. The `chop.word` function uses `s`, so we need another

string variable, t

```
2864 FUNCTION {sort.format.title}
2865 { 't :=
2866   "A " #2
2867   "An " #3
2868   "The " #4 t chop.word
2869   chop.word
2870   chop.word
2871   sortify
2872   #1 global.max$ substring$
2873 }
2874
```

The auxiliary functions here, for the presort function, are analogous to the ones for calc.label; the same comments apply, except that the organization field takes precedence here over the key field. For sorting purposes, we still remove a leading The from the organization field.

```
2875 FUNCTION {anonymous.sort}
2876 { entry.lang lang.zh =
2877   { "yi4 ming2" }
2878   { "anon" }
2879   if$
2880 }
2881
2882 FUNCTION {warn.empty.key}
2883 { entry.lang lang.zh =
2884   { "empty key in " cite$ * warning$ }
2885   'skip$
2886   if$
2887 }
2888
2889 FUNCTION {author.sort}
2890 { key empty$
2891   { warn.empty.key
2892     author empty$
2893     { anonymous.sort }
2894     { author sort.format.names }
2895     if$
2896   }
2897   { key }
2898   if$
2899 }
2900
2901 FUNCTION {author.editor.sort}
2902 { key empty$
2903   { warn.empty.key
2904     author empty$
2905     { editor empty$
2906       { anonymous.sort }
2907       { editor sort.format.names }
2908       if$
2909     }
2910     { author sort.format.names }

```

```

2911     if$
2912     }
2913     { key }
2914 if$
2915 }
2916
2917 FUNCTION {author.organization.sort}
2918 { key empty$
2919   { warn.empty.key
2920     author empty$
2921       { organization empty$
2922         { anonymous.sort }
2923         { "The " #4 organization chop.word sortify }
2924         if$
2925       }
2926       { author sort.format.names }
2927     if$
2928   }
2929   { key }
2930 if$
2931 }
2932
2933 FUNCTION {editor.organization.sort}
2934 { key empty$
2935   { warn.empty.key
2936     editor empty$
2937       { organization empty$
2938         { anonymous.sort }
2939         { "The " #4 organization chop.word sortify }
2940         if$
2941       }
2942       { editor sort.format.names }
2943     if$
2944   }
2945   { key }
2946 if$
2947 }
2948
2949 </author-year>

```

顺序编码制的排序要简单得多

```

2950 <numerical>
2951 INTEGERS { seq.num }
2952
2953 FUNCTION {init.seq}
2954 { #0 'seq.num :=}
2955
2956 FUNCTION {int.to.fix}
2957 { "00000000" swap$ int.to.str$ *
2958   #-1 #10 substring$
2959 }
2960
2961 </numerical>

```

There is a limit, `entry.max$`, on the length of an entry string variable (which is

what its `sort.key$` is), so we take at most that many characters of the constructed key, and hope there aren't many references that match to that many characters!

```

2962 FUNCTION {presort}
2963 { set.entry.lang
2964   set.entry.numbered
2965   show.url show.doi check.electronic
2966   #0 'is.pure.electronic :=
2967   calc.label
2968   label sortify
2969   " "
2970   *
2971   <*author-year>
2972   sort.language.label
2973   " "
2974   *
2975   type$ "book" =
2976   type$ "inbook" = booktitle empty$ not and
2977   or
2978   'author.editor.sort
2979   { type$ "collection" =
2980     type$ "proceedings" =
2981     or
2982     'editor.organization.sort
2983     'author.sort
2984     if$
2985   }
2986   if$
2987   *
2988   " "
2989   *
2990   year field.or.null sortify
2991   *
2992   " "
2993   *
2994   cite$
2995   *
2996   #1 entry.max$ substring$
2997   </author-year>
2998   <*numerical>
2999   seq.num #1 + 'seq.num :=
3000   seq.num int.to.fix
3001   </numerical>
3002   'sort.label :=
3003   sort.label *
3004   #1 entry.max$ substring$
3005   'sort.key$ :=
3006 }
3007

```

Now comes the final computation for alphabetic labels, putting in the 'a's and 'b's and so forth if required. This involves two passes: a forward pass to put in the 'b's, 'c's and so on, and a backwards pass to put in the 'a's (we don't want to put in 'a's unless we know there are 'b's). We have to keep track of the longest (in `width$`

terms) label, for use by the thebibliography environment.

```
VAR: longest.label, last.sort.label, next.extra: string
      longest.label.width, last.extra.num: integer

initialize.longest.label ==
BEGIN
  longest.label := ""
  last.sort.label := int.to.chr$(0)
  next.extra := ""
  longest.label.width := 0
  last.extra.num := 0
END

forward.pass ==
BEGIN
  if last.sort.label = sort.label then
    last.extra.num := last.extra.num + 1
    extra.label := int.to.chr$(last.extra.num)
  else
    last.extra.num := chr.to.int$("a")
    extra.label := ""
    last.sort.label := sort.label
  fi
END

reverse.pass ==
BEGIN
  if next.extra = "b" then
    extra.label := "a"
  fi
  label := label * extra.label
  if width$(label) > longest.label.width then
    longest.label := label
    longest.label.width := width$(label)
  fi
  next.extra := extra.label
END
```

```
3008 STRINGS { longest.label last.label next.extra last.extra.label }
3009
3010 INTEGERS { longest.label.width number.label }
3011
3012 FUNCTION {initialize.longest.label}
3013 { "" 'longest.label :=
3014   #0 int.to.chr$ 'last.label :=
3015   "" 'next.extra :=
3016   #0 'longest.label.width :=
3017   #0 'number.label :=
3018   "" 'last.extra.label :=
3019 }
3020
3021 FUNCTION {forward.pass}
3022 {
3023 (*author-year)
```

```

3024 last.label label =
3025 { "" 'extra.label :=
3026 last.extra.label text.length$ 'charptr :=
3027 { last.extra.label charptr #1 substring$ "z" =
3028 charptr #0 > and
3029 }
3030 { "a" extra.label * 'extra.label :=
3031 charptr #1 - 'charptr :=
3032 }
3033 while$
3034 charptr #0 >
3035 { last.extra.label charptr #1 substring$ chr.to.int$ #1 + int.to.chr$
3036 extra.label * 'extra.label :=
3037 last.extra.label #1 charptr #1 - substring$
3038 extra.label * 'extra.label :=
3039 }
3040 { "a" extra.label * 'extra.label := }
3041 if$
3042 extra.label 'last.extra.label :=
3043 }
3044 { "a" 'last.extra.label :=
3045 "" 'extra.label :=
3046 label 'last.label :=
3047 }
3048 if$
3049 </author-year>
3050 number.label #1 + 'number.label :=
3051 }
3052
3053 FUNCTION {reverse.pass}
3054 {
3055 <*author-year>
3056 next.extra "b" =
3057 { "a" 'extra.label := }
3058 'skip$
3059 if$
3060 extra.label 'next.extra :=
3061 extra.label
3062 duplicate$ empty$
3063 'skip$
3064 { "{\natexlab{" swap$ * "}" * }
3065 if$
3066 'extra.label :=
3067 </author-year>
3068 label extra.label * 'label :=
3069 }
3070
3071 FUNCTION {bib.sort.order}
3072 { sort.label 'sort.key$ :=
3073 }
3074

```

B.9 Write bbl file

Now we're ready to start writing the .BBL file. We begin, if necessary, with a \LaTeX macro for unnamed names in an alphabetic label; next comes stuff from the 'preamble' command in the database files. Then we give an incantation containing the command `\begin{thebibliography}{...}` where the '...' is the longest label.

We also call `init.state.consts`, for use by the output routines.

```
3075 FUNCTION {begin.bib}
3076 { preamble$ empty$
3077   'skip$
3078   { preamble$ write$ newline$ }
3079   if$
3080   "\begin{thebibliography}{ number.label int.to.str$ * }" *
3081   write$ newline$
3082   terms.in.macro
3083   { "\providecommand{\biband}{和}"
3084     write$ newline$
3085     "\providecommand{\bibetal}{等}"
3086     write$ newline$
3087   }
3088   'skip$
3089   if$
3090   "\providecommand{\natexlab}[1]{#1}"
3091   write$ newline$
3092   "\providecommand{\url}[1]{#1}"
3093   write$ newline$
3094   "\expandafter\ifx\csname urlstyle\endcsname\relax\else"
3095   write$ newline$
3096   " \urlstyle{same}\fi"
3097   write$ newline$
3098   "\expandafter\ifx\csname href\endcsname\relax"
3099   write$ newline$
3100   " \DeclareUrlCommand\doi{\urlstyle{rm}}"
3101   write$ newline$
3102   " \def\epprint#1#2{#2}"
3103   write$ newline$
3104   "\else"
3105   write$ newline$
3106   " \def\doi#1{\href{https://doi.org/#1}{\nolinkurl{#1}}}"
3107   write$ newline$
3108   " \let\epprint\href"
3109   write$ newline$
3110   "\fi"
3111   write$ newline$
3112   }
3113
```

Finally, we finish up by writing the '`\end{thebibliography}`' command.

```
3114 FUNCTION {end.bib}
3115 { newline$
3116   "\end{thebibliography}" write$ newline$
3117 }
```

3118

B.10 Main execution

Now we read in the .BIB entries.

```
3119 READ
3120
3121 EXECUTE {init.state.consts}
3122
3123 EXECUTE {load.config}
3124
3125 <*numerical>
3126 EXECUTE {init.seq}
3127
3128 </numerical>
3129 ITERATE {presort}
3130
```

And now we can sort

```
3131 SORT
3132
3133 EXECUTE {initialize.longest.label}
3134
3135 ITERATE {forward.pass}
3136
3137 REVERSE {reverse.pass}
3138
3139 ITERATE {bib.sort.order}
3140
3141 SORT
3142
3143 EXECUTE {begin.bib}
3144
```

Now we produce the output for all the entries

```
3145 ITERATE {call.type$}
3146
3147 EXECUTE {end.bib}
3148 </author-year | numerical>
```