The biblatex Package

Programmable Bibliographies and Citations

Philipp Lehman **Version 3.9**
(with Philip Kime, Audrey Boruvka **21/11/2017**
and Joseph Wright)

Biblatex 用户手册[1]

Trans. by: Zhenzhen Hu[2] and Wenbo Sheng [3]

2016-02-08

目录

[1]Doc last revised at 2017-12-15

[2]Email: hzzmail@163.com

[3]Email: wbsheng88@foxmail.com

表格

1 引言

这份文档是系统介绍 biblatex 宏包的参考手册。想获得初步印象，请参考 biblatex 附带的示例文档[4]。想快速入门，请浏览：§§ 1.1、2.1、2.2、2.3、3.1、3.3、3.7、3.8、3.12 节。

1.1 关于 biblatex

biblatex 包提供了一套与 LaTeX 配合使用的高级参考文献工具。它重新实现了 LaTeX 提供的参考文献功能。该包使用后端程序 biber 来处理 BibTeX 格式的数据文件，并完成排序、标签生成 (及更多其它功能)。参考文献的格式化完全由 TeX 宏指令控制。具备良好的 LaTeX 知识就足以设计新的参考文献著录样式和标注样式。

biblatex 也支持参考文献表细分、在一个文档内包含多个参考文献表、以及域缩写等参考文献信息表。参考文献表可以根据主题进行分块或者分段。与参考文献著录样式类似，所有的标注 (引用) 命令也可以自由定义。

提供的功能还包括：文献数据的 Unicode 支持、自定义排序、不同排序方式的多参考文献表、自定义标签和动态数据修改等。biber/biblatex 的版本兼容性见 § 1.5.5 节。该包可完全实现本地化，可与 babel 和 polyglossia 宏包配合使用。该包支持的语言详见表 2。

1.2 许可

[4] http://ctan.org/pkg/biblatex//doc/examples

[5] http://www.ctan.org/tex-archive/macros/latex/base/lppl.txt

1.3 反馈

请使用 Github 上的 biblatex 项目页报告 bug 和提交所需功能[6]。在提出功能需求前，请确保你已经彻底研究过本手册。如果你不想报告 bug 或者请求新功能，而只是需要帮助，可以考虑在 comp.text.tex 新闻组或者 TEX-LATEX Stack Exchange 提交问题。[7]

1.4 致谢

The language modules of this package are made possible thanks to the following contributors: Augusto Ritter Stoffel, Mateus Araújo, Gustavo Barros (Brazilian); Kaloyan Ganev (Bulgarian); Sebastià Vila-Marta (Catalan); Ivo Pletikosić (Croatian); Michal Hoftich (Czech); Jonas Nyrup (Danish); Johannes Wilm (Danish/Norwegian); Alexander van Loon, Pieter Belmans, Hendrik Maryns (Dutch); Benson Muite (Estonian); Hannu Väisänen, Janne Kujanpää (Finnish); Denis Bitouzé (French); Apostolos Syropoulos, Prokopis (Greek); Baldur Kristinsson (Icelandic); Enrico Gregorio, Andrea Marchitelli (Italian); Håkon Malmedal (Norwegian); Anastasia Kandulina, Yuriy Chernyshov (Polish); José Carlos Santos (Portuguese); Oleg Domanov (Russian); Martin Vrábel, Dávid Lupták (Slovak); Tea Tušar and Bogdan Filipič (Slovene); Ignacio Fernández Galván (Spanish); Per Starbäck, Carl-Gustav Werner, Filip Åsblom (Swedish). Sergiy M. Ponomarenko (Ukrainian); Hans Fredrik Nordhaug (Norwegian)

1.5 前提与必备

本节介绍宏包所需的资源，并讨论兼容性问题。

1.5.1 必须资源

如下资源是必须的，否则 biblatex 无法正常工作。

ε-TEX biblatex 宏包依赖于 ε-TEX 。很长时间以来，TEX 发行版就带有 ε-TEX ，并且近来主流的发行版都默认使用。biblatex 宏包会检查是否在 ε-TEX 下运行。只需要像平常一样编译你的文档即可，基本上是可以运行的。如果你得到错误信息，尝试用 elatex 或 pdfelatex 分别代替 latex 或 pdflatex 来编译文档。

biber biber 是biblatex 的后端程序，负责将参考文献源文件中的数据转换为 LaTeX 代码。TEXLive 中带有 biber ，也可以从 SourceForge 获取 [8] 。biber 使用 C 标准库 btparse 解析 BIBTEX 格式文件，这既为了兼容 BIBTEX 的解析规则，也用于修正一些常见问题。详见 Perl 模块 Text::BibTeX 的手册页 [9]。

etoolbox 自动加载，提供biblatex 所需的通用编程工具，可以从 CTAN 下载。[10]

kvoptions 自动加载，用于内部选项处理。可以随 oberdiek 宏包集从 CTAN 下载。[11]

[6] http://github.com/plk/biblatex

[7] http://tex.stackexchange.com/questions/tagged/biblatex

[8] http://biblatex-biber.sourceforge.net/

[9] http://search.cpan.org/~ambs/Text-BibTeX

[10] http://ctan.org/pkg/etoolbox

[11] http://ctan.org/pkg/kvoptions

logreq 自动加载，它提供的前端可用于将机器可读信息写入辅助 log 文件，可以从 CTAN 下载。[12]

xstring 自动加载，提供了一些高级字符串处理宏。可以从 CTAN 下载。[13]

除了上述资源，biblatex 还需要 keyval、ifthen 以及 url 等标准 LaTeX 宏包。常见的 TeX 发行版中都会带有这些宏包，而且本宏包会自动加载。

1.5.2 推荐包

这一节所列出的宏包对于运行 biblatex 不是必须的。不过，它们能提供一些值得推荐的额外功能，或者能增强一些已有的功能。宏包载入的顺序并不重要。

babel/polyglossia babel 和 polyglossia 宏包提供了多语种排版的核心架构。如果你使用美式英语以外的语言写作，那么强烈推荐使用这两个宏包中的一个。[14] 你应当在 biblatex 之前载入 babel 或 polyglossia，这样 biblatex 可以自动检测这两个宏包。

csquotes 如果载入该宏包，biblatex 会使用它的引用语工具给相应标题加上语言相关的引号。如果不载入，那么 biblatex 会使用作为后备的美式英语的引号。当使用美式英语以外的语言写作时，强烈推荐使用 csquotes 宏包。[15]

xpatch xpatch 宏包扩展了 etoolbox 的一些补丁命令，这些补丁命令用在biblatex 的宏、驱动和格式化指令中。[16]

1.5.3 兼容的包

biblatex 宏包专门为本节所列出的文档类和宏包提供了兼容性代码。

hyperref hyperref 宏包将参考文献标注(引用)转化为超链接。详见 § 3.1.2.1 一节中的 hyperref 和 backref 宏包选项。当使用 hyperref 宏包时，最好在 biblatex 之后载入。

showkeys 使用showkeys 宏包能打印出文档中标注和文献表中条目的内部键值。宏包载入的顺序不重要。

memoir 使用 memoir 文档类时，biblatex 会调整默认的参考文献标题，使其与该文档类默认的页面布局协调一致。更多用法请参考 § 3.13.2 一节。

KOMA-Script 使用 scrartcl、scrbook 或 scrreprt 文档类中的任何一个时，biblatex 会调整默认的参考文献标题，使其与这些文档类默认的页面布局协调一致。更多用法请参考 § 3.13.1 一节。

[12] http://ctan.org/pkg/logreq/

[13] http://ctan.org/pkg/xstring/

[14] 这里指的是 babel/polyglossia 支持的其它语言（如一些西方语种）。对于中文文档，出于兼容问题既不应当使用，也没有必要。——译注

[15] http://ctan.org/pkg/csquotes/

[16] http://ctan.org/pkg/xpatch/

1.5.4 不兼容的包

本节列出了与 biblatex 不兼容的宏包。biblatex 从根本上重新实现了 LaTeX 的参考文献功能，因此很自然地与修改这些功能的所有宏包相冲突。这并不是 biblatex 独有的——下面列出的宏包中，出于同样的原因，一些宏包之间也是不兼容的。

babelbib babelbib 宏包为多语种文献提供了支持，这正是 biblatex 的一个典型特点。使用 langid 域和宏包选项 autolang 即可实现类似的功能。请注意，当载入 babel 或 polyglossia 宏包时 biblatex 会自动调整主文档的语言。如果想要在文献中每个条目里切换语言，你只需要以上提到的特性。详见 §§ 2.2.3 和 3.1.2.1 节，另可见 § 3.9 节。

backref backref 宏包可以在参考文献中创建反向引用。类似的功能请参考 § 3.1.2.1 节中的宏包选项 hyperref 和 backref。

bibtopic bibtopic 宏包支持根据主题、类型或者其它标准细分文献。对于按照主题细分文献，可以参考 § 3.7.6 节的类型特征以及 § 3.7.2 节中相应的过滤器。另外，也可以使用 keywords 域结合 keyword 和 notkeyword 过滤器来实现相应功能，详见 §§ 2.2.3 和 3.7.2 节。对于按照类型细分文献，可以使用 type 和 nottype 过滤器。相关例子请参考 § 3.12.4 节。

bibunits bibunits 宏包支持多个（例如每一章的）分文献表。请参考 chapterbib。

chapterbib chapterbib 宏包支持多个分参考文献。使用 refsection 环境和 section 过滤器可以实现相应效果。或者，你也可能需要使用 refsegment 环境和 segment 过滤器。详见 §§ 3.7.4、3.7.5、3.7.2。相关例子请参考 § 3.12.3。

cite cite 可以自动对顺序编码标注进行排序，能将连续的数字压缩为一个区间。它也可以配置标注 (引用) 中的标点符号。关于顺序编码标注的排序和缩写，请参考 § 3.1.2.1 节中的 sortcites 宏包选项和 § 3.3.1 节中的 numeric-comp 引用样式。关于可配置的标点请参考 § 3.10。

citeref 另一个可以创建反向引用的宏包。参考 backref 条目。

inlinebib inlinebib 宏包是为在脚注使用的传统标注设计的。相应的功能请参考 § 3.3.1 节中关于多词标注 (verbose) 样式的说明。

jurabib jurabib 宏包原本用于法学和司法文件（主要是德文）中的标注，它也为人文学科的用户提供了一些功能。在提供的功能方面，jurabib 和 biblatex 有一些类似之处，但是实现的手段是截然不同的。由于 jurabib 和 biblatex 都是那种功能齐备的宏包，限于篇幅这里不再赘述它们的异同之处。

mcite mcite 提供了分组标注的支持，也就是说，多个条目可以由单个标注引用，在参考文献表中作为一个条目组列在一起。标注组由组内的引用项定义。该功能只能用于非排序的参考文献表中。biblatex 宏包同样支持分组标注，在本手册中称之为“条目集”或“参考文献集”。详见 §§ 3.12.5、3.7.11、3.8.10 节。

mciteplus `mcite` 宏包的一个加强版，支持在排序文献表中的分组。参考 `mcite` 宏包条目。

multibib `multibib` 宏包支持根据主题或其它标准的细分文献表。参考 `bibtopic` 宏包条目。

natbib `natbib` 宏包吸收了`cite` 宏包中的合并排序和压缩代码，支持顺序表明和作者年标注样式。它也提供了一些额外的标注 (引用) 命令和数个设置选项。相应的功能请参考 § 3.3.1 节的 `numeric` 和 `author-year` 引用样式及其变种，§ 3.1.2.1 节的 `sortcites` 宏包选项，§ 3.8 节的标注命令，以及 §§ 3.7.7、3.7.8、3.10 节讨论的工具。另见 § 3.8.9 节。

splitbib `splitbib` 宏包支持按照主题细分文献。参考 `bibtopic` 宏包条目。

titlesec `titlesec` 宏包重新定义了一些用户层的文档章节划分命令，例如 `\chapter` 或 `\section`。其方法与 `biblatex` 因设置 `refsection` 和 `refsegment` 选项导致的内部命令变化不兼容，上述两个选项详见 § 3.1.2.1 节。[17]

ucs `ucs` 宏包提供 UTF-8 编码输入的支持。可以使用 `inputenc` 宏包的标准 `utf8` 模块或者 XeTeX、LuaTeX 等支持 Unicode 的编译引擎来实现这一功能。

etextools `etextools`宏包增强了`etoolbox`宏包中处理列表的宏，并提供了少量用于定义命令的工具。该宏包重新定义列表处理宏的方式与`biblatex` 是不兼容的。

1.5.5 biber/biblatex 兼容性

`biber` 的版本与 `biblatex` 的版本有着紧密的联系。你需要二者正确的组合。在处理过程中如果遇到源自不兼容版本 `biblatex` 的信息，`biber` 会抛出异常。表 1 给出了最近一些版本的兼容性矩阵。

2 数据库指南

本节描述 `blx-dm.def` 中定义的默认数据模型。该文件是宏包的一部分。该数据模型的定义由 § 4.5.4 节中的宏实现。因此，可以重新定义 `biblatex` 和 `biber` 所用的数据模型，使得数据源可以包括新的条目类型和域（当然这需要样式文件支持）。数据模型规范还允许定义约束，使得数据源可以根据数据模型进行校验（使用 `biber` 的 `--validate_datamodel` 选项）。若需要定制数据模型，请参考 `blx-dm.def` 文件和 § 4.5.4 节。

2.1 条目类型

本节介绍 `biblatex` 默认数据模型支持的条目类型及每种条目类型支持的域。

2.1.1 常规类型

下面的列表说明了每种条目类型支持的域。注意，每种条目类型对域的使用是由参考文献样式决定的。因此，下面的列表有两个目的，一是说明本宏包标准

[17] 当使用 biblatex 时，不设置 `refsection` 和 `refsegment` 选项，那么 titlesec 就没有兼容性问题，完全可以使用——译注

Biber 版本	biblatex 版本
2.9	3.9
2.8	3.8
2.7	3.7
2.6	3.5, 3.6
2.5	3.4
2.4	3.3
2.3	3.2
2.2	3.1
2.1	3.0
2.0	3.0
1.9	2.9
1.8	2.8
1.7	2.7
1.6	2.6
1.5	2.5
1.4	2.4
1.3	2.3
1.2	2.1, 2.2
1.1	2.1
1.0	2.0
0.9.9	1.7x
0.9.8	1.7x
0.9.7	1.7x
0.9.6	1.7x
0.9.5	1.6x
0.9.4	1.5x
0.9.3	1.5x
0.9.2	1.4x
0.9.1	1.4x
0.9	1.4x

表 1: biber/biblatex 兼容性

样式支持的域，二是作为定制样式的模板。注意，所谓“必选”域并不是在所有情况下都严格必不可少的，详见 § 2.3.2 节。而标记“可选”的域技术上是可选的，不过通常来说，文献格式化规则往往不会仅限于“必选”域。

默认的数据模型为日期域、ISBN 类的域和 `gender` 等特殊域定义了一些约束。但这些约束仅用于校验这些域是否合乎数据模型（通过 `biber` 的 `--validate_datamodel` 选项）。通用域如 `abstract`、`annotation`、`label` 和 `shorthand` 并不在下面的列表中，因为它们独立于条目类型；§ 2.2.3 节讨论的特殊域同样也独立于条目类型，因此也不在下面的列表中。需要了解完整的数据模型规范，详见`biblatex` 附带文件 `blx-dm.def`中的默认数据模型。

article 指从期刊、杂志、报纸或其他周期性刊物中的析出文章，具有自己的标题，是一个独立单元。刊物名在 `journaltitle` 域中给出。如果除刊物名外，某期刊物也有具体的题名，那么该题名在 `issuetitle` 域中给出。注意，`editor` 及相关域指的是期刊，而 `translator` 及其相关域则涉及到文章。

必选域： `author`, `title`, `journaltitle`, `year/date`

可选域： `translator`, `annotator`, `commentator`, `subtitle`, `titleaddon`, `editor`, `editora`, `editorb`, `editorc`, `journalsubtitle`, `issuetitle`, `issuesubtitle`, `language`, `origlanguage`, `series`, `volume`, `number`, `eid`, `issue`, `month`, `pages`, `version`, `note`, `issn`, `addendum`, `pubstate`, `doi`, `eprint`, `eprintclass`, `eprinttype`, `url`, `urldate`

book 单卷本的书籍，有一位或多位作者，其中多位作者名构成一个整体名单作为该著作的责任者。该条目类型也涵盖了传统 BibTeX 的 `@inbook` 类型，详见 § 2.3.1 节。

必选域： `author`, `title`, `year/date`

可选域： `editor`, `editora`, `editorb`, `editorc`, `translator`, `annotator`, `commentator`, `introduction`, `foreword`, `afterword`, `subtitle`, `titleaddon`, `maintitle`, `mainsubtitle`, `maintitleaddon`, `language`, `origlanguage`, `volume`, `part`, `edition`, `volumes`, `series`, `number`, `note`, `publisher`, `location`, `isbn`, `chapter`, `pages`, `pagetotal`, `addendum`, `pubstate`, `doi`, `eprint`, `eprintclass`, `eprinttype`, `url`, `urldate`

mvbook 多卷本书籍。为了向后兼容，多卷书也可用 `@book` 条目类型。然而建议最好使用该专用条目类型 `@mvbook`。

必选域： `author`, `title`, `year/date`

可选域： `editor`, `editora`, `editorb`, `editorc`, `translator`, `annotator`, `commentator`, `introduction`, `foreword`, `afterword`, `subtitle`, `titleaddon`, `language`, `origlanguage`, `edition`, `volumes`, `series`, `number`, `note`, `publisher`, `location`, `isbn`, `pagetotal`, `addendum`, `pubstate`, `doi`, `eprint`, `eprintclass`, `eprinttype`, `url`, `urldate`

inbook 书的一部分。它是一个独立的单元，有自己的标题。注意，该类型的定义不同于标准 BibTeX 给出的定义，见 § 2.3.1 节。

必选域： `author`, `title`, `booktitle`, `year/date`

可选域： bookauthor, editor, editora, editorb, editorc, translator, annotator, commentator, introduction, foreword, afterword, subtitle, titleaddon, maintitle, mainsubtitle, maintitleaddon, booksubtitle, booktitleaddon, language, origlanguage, volume, part, edition, volumes, series, number, note, publisher, location, isbn, chapter, pages, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

bookinbook 类似于 @inbook，但用于原本已经出版的单行本。典型的例子是在一位作者作品集中再版的书籍。

suppbook @book （书）的补充材料，与 @inbook 条目类型密切相关。@inbook 主要用于一本书中带有自身标题的部分，例如一本散文集中相同作者的单独一篇散文；而本条目用于诸如序言、导论、前言、后记等部分，往往只有一个通用标题。一些参考文献著录标准可能会要求该条目类型的著录格式不同于 @inbook。而标准样式将其视为 @inbook 的别名。

booklet 类似于书籍，但没有正式的出版或赞助机构。如果允许，可以使用 howpublished 域提供自由格式的出版信息。或者也可以使用 type 域。

必选域： author/editor, title, year/date

可选域： subtitle, titleaddon, language, howpublished, type, note, location, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

collection 单卷本的文集，由一些具有不同作者和题名的独立稿件构成。作品集作为一个整体没有总体意义上的作者，但通常有一位编者。

必选域： editor, title, year/date

可选域： editora, editorb, editorc, translator, annotator, commentator, introduction, foreword, afterword, subtitle, titleaddon, maintitle, mainsubtitle, maintitleaddon, language, origlanguage, volume, part, edition, volumes, series, number, note, publisher, location, isbn, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

mvcollection 多卷本文集。为了向后兼容，也可用条目类型 @collection 表示。然而，建议最好还是使用专用条目类型 @mvcollection。

必选域： editor, title, year/date

可选域： editora, editorb, editorc, translator, annotator, commentator, introduction, foreword, afterword, subtitle, titleaddon, language, origlanguage, edition, volumes, series, number, note, publisher, location, isbn, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

incollection 文集中的一篇稿件，是一个独立的单元，具有自己的标题。author 指的是 title 的作者，而 editor 指的是 booktitle（即文集标题）的编者。

必选域： author, title, booktitle, year/date

可选域： editor, editora, editorb, editorc, translator, annotator, commentator, introduction, foreword, afterword, subtitle, titleaddon, maintitle, mainsubtitle, maintitleaddon, booksubtitle, booktitleaddon, language, origlanguage, volume, part, edition, volumes, series, number, note, publisher, location, isbn, chapter, pages, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

suppcollection `@collection` 中的补充材料。类似于 `@suppbook` 之于 `@book`。标准样式将其视为 `@incollection` 的别名。

manual 技术或其它文档，不必是印刷形式的。按照 § 2.3.2 一节，`author` 或者 `editor` 是可以省略的。

必选域： author/editor, title, year/date

可选域： subtitle, titleaddon, language, edition, type, series, number, version, note, organization, publisher, location, isbn, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

misc 备选类型，用于无法归入任何其它类别的条目。合适的话，可以使用 `howpublished` 域，可以提供自由格式的出版信息。或者也可以使用 `type` 域。按照 § 2.3.2 节，`author`、`editor` 和 `year` 可以省略。

必选域： author/editor, title, year/date

可选域： subtitle, titleaddon, language, howpublished, type, version, note, organization, location, date, month, year, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

online 在线资源。按照 § 2.3.2 节，`author`，`editor` 和 `year` 可以省略。该条目类型用于网址等固有在线资源。注意：所有条目类型都支持 `url` 域。比如，当增加一篇来自在线期刊的文章时，应优先使用 `@article` 条目和它的 `url` 域。

必选域： author/editor, title, year/date, url

可选域： subtitle, titleaddon, language, version, note, organization, date, month, year, addendum, pubstate, urldate

patent 专利或专利申请。专利号或登记号在 `number` 域中给出。`type` 域用于描述类型，如果专利保护范围与 `type` 域暗指的范围不一致，则可使用`location` 域对专利的保护范围 (权利范围) 进行描述。注意，`location` 在本条目类型中以键值列表的方式处理，详见 § 2.2.1 节。

必选域： author, title, number, year/date

可选域： holder, subtitle, titleaddon, type, version, location, note, date, month, year, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

periodical 周期性刊物中完整的一期，比如某一刊物的某一期特刊。标题在 `title` 域中给出。如果该期除期刊主标题外还有自己的标题，那么在 `issuetitle` 域中给出该信息。根据 § 2.3.2 节，`editor` 域可以省略。

必选域： editor, title, year/date

可选域： editora, editorb, editorc, subtitle, issuetitle, issuesubtitle, language, series, volume, number, issue, date, month, year, note, issn, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

suppperiodical @periodical 的补充材料，类似于 @suppbook 之于 @book。如果你意识到 @article 类型其实就是 @inperiodical，那么本条目类型的作用就显而易见了。该类型应用于仅具有通用标题的信息项，例如固定专栏、讣告、致编者的信等。一些参考文献著录标准会严格要求这些信息的格式不同于 @article。而标准样式则视其为 @article 的别名。

proceedings 单卷本的会议录 (会议文集，汇编)。与 @collection 非常相似。它支持可选的 organization 域用于给出主办机构。根据 § 2.3.2 节，editor 域可以省略。

必选域： title, year/date

可选域： editor, subtitle, titleaddon, maintitle, mainsubtitle, maintitleaddon, eventtitle, eventtitleaddon, eventdate, venue, language, volume, part, volumes, series, number, note, organization, publisher, location, month, isbn, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

mvproceedings 多卷 @proceedings 条目，类似于 @mvbook 之于 @book。

必选域： title, year/date

可选域： editor, subtitle, titleaddon, eventtitle, eventtitleaddon, eventdate, venue, language, volumes, series, number, note, organization, publisher, location, month, isbn, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

inproceedings 会议文集中的一篇文章，与 @incollection 类似。支持 organization 可选域。

必选域： author, title, booktitle, year/date

可选域： editor, subtitle, titleaddon, maintitle, mainsubtitle, maintitleaddon, booksubtitle, booktitleaddon, eventtitle, eventtitleaddon, eventdate, venue, language, volume, part, volumes, series, number, note, organization, publisher, location, month, isbn, chapter, pages, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

reference 单卷本的参考书，诸如百科全书或词典等。它是一般的 @collection 条目类型的一种特殊变体。标准样式将其视为 @collection 的别名。

mvreference 多卷本的 @reference 条目。标准样式将其视为 @mvcollection 的别名。出于向后兼容考虑，也可以使用 @reference 条目。不过，还是建议使用专门的 @mvreference 条目类型。

inreference 参考书中的一篇文章，它是一般的@incollection 条目的一种特殊变体。标准样式将其视为 @incollection 的别名。

report 由大学或其它机构发行的技术报告、研究报告以及白皮书等。使用 type 域来指定报告的类型。主办机构由 institution 域给出。

必选域： author, title, type, institution, year/date

可选域： subtitle, titleaddon, language, number, version, note, location, month, isrn, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

set 条目集，是一种特殊的类型条目，详见 § 3.12.5 节。

thesis 为满足教育机构学位要求而撰写的学位论文。使用 type 域指定学位论文类型。

必选域： author, title, type, institution, year/date

可选域： subtitle, titleaddon, language, note, location, month, isbn, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

unpublished 有作者和标题但是没有正式出版的作品，例如手稿或演讲稿等。允许的话，可使用 howpublished 域和 note 域提供自由格式的附加信息。

必选域： author, title, year/date

可选域： subtitle, titleaddon, language, howpublished, note, location, isbn, date, month, year, addendum, pubstate, url, urldate

xdata 特殊类型，@xdata 条目保存有可以被其它条目用 xdata 域继承的数据。这一类型的条目只是作为数据容器，不可被引用或加入到文献表中，详见 § 3.12.6 节。

custom[a–f] 用于特殊参考文献样式的自定义条目类型，标准样式中不使用。

2.1.2 类型别名

本节中列出的条目类型用于向后兼容传统的 BibTeX 样式。这些别名由后端程序在数据处理时统一处理。参考文献样式仅能见到这些别名所指代的类型，而不是这些别名本身。所有未知的条目类型一般输出为 @misc 条目。

conference BibTeX 遗留的 @inproceedings 的别名。

electronic @online的别名。

mastersthesis 类似于 @thesis，差别在于 type 域是可选的，默认是本地化关键字 ‘Master’s thesis’ 所代表的本地化字符串。用户可以直接使用 type 域进行重新定义。

phdthesis 类似于 @thesis，差别在于 type 域是可选的，默认是本地化关键字 ‘PhD thesis’ 所代表的本地化字符串。用户可以直接使用 type 域进行重新定义。

techreport 类似于@report，差别在于 type 域是可选的，默认是本地化关键字 ‘technical report’ 所代表的本地化字符串。用户可以直接使用 type 域进行重新定义。

www @online 的别名，用于兼容 jurabib 宏包。

2.1.3 不支持的条目类型

本节中的条目类型类似于自定义类型 `@custom[a–f]`，即，标准样式不支持这些类型，若使用标准样式，将会以 `@misc` 条目类型进行处理。

artwork 视觉艺术作品，例如绘画、雕塑和装饰艺术品。

audio 录音作品，典型的有音频 CD、DVD、录音磁带或类似媒介。参考 `@music` 类型。

bibnote 这一特殊条目类型并不像其它类型那样用于 `bib` 文件中。它主要是为第三方宏包提供 `notes2bib` 等，用于将注记并入文献中。注记应该在 `note` 域中。请谨记，`@bibnote` 类型与 `\defbibnote` 命令毫无关系。`\defbibnote` 命令用来在参考文献表的开始或末尾处添加评论，而 `@bibnote` 类型是为那些将尾注作为参考条目处理的宏包准备的。

commentary 法律身份不同于一般书籍的评注，如司法评论等。

image 图像、图画、摄影和类似媒介。

jurisdiction 法庭判决、法庭记录和类似物。

legislation 法律、法案、立法提案和类似物。

legal 协议等法律文书。

letter 私人函件，例如信件、电子邮件、备忘录等。

movie 动画。参考 `@video` 类型。

music 音乐唱片，`@audio` 的一种特殊变体。

performance 音乐或戏剧表演和其它一些表演艺术作品。这一条目类型指的是表演的事件，而不是一种录制品，乐谱或付印的剧本。

review 一些其它工作的回顾总结。这是 `@article` 类型的一种特殊变体。标准样式将其视为 `@article` 的一个别称。

software 电脑软件。

standard 由一个标准组织（例如国际标准组织）发布的国家或国际标准。

video 音像作品，典型的包括 DVD、VHS 录像带或其它类似媒介。参考 `@movie` 类型。

2.2 条目中的域

本节将概略介绍 `biblatex` 默认数据模型支持的域。数据模型规范使用的数据类型简介，见 § 2.2.1 节，实际使用的域的一览表，见 §§ 2.2.2 和 2.2.3 节。

2.2.1 数据类型

在 bib 文件等数据源中，所有的文献数据都在域中给出。其中一些域，例如 author 和 editor，可以包含一个项目列表。在 BibTeX 文件格式中，这种列表结构通过关键词“and”来分隔列表中的每一项。biblatex 宏包实现了三种不同的数据类型来处理文献数据：姓名列表（name list）、文本列表（literal list）和域类型（field）。此外，还有一些列表和域类型的子类，以及一个内容类型（content type），用于从语义上区分那些无法仅根据数据类型进行区分的域（见 § 4.5.4 节）。本节总结了本宏包所支持的数据类型。BibTeX 文件格式中的域与 biblatex 数据类型的对应信息，请参考 §§ 2.2.2 和 2.2.3 节。

姓名列表（name list） 根据分隔词 and 将其解析并划分成独立的项。然后列表中的每一项进一步分解成四个姓名成分：[18] 名（given name，默认值）、姓名前缀（name prefix, 如 von、van、of、da、de、della 等）、姓（family name），以及姓名后缀（name suffix, 如 junior、senior 等）。可以通过调整数据模型的定义来定制有效的姓名成分，见 § 4.2.3 节。在 bib 文件中，姓名列表可以用关键词“and others”来截短。典型的姓名列表是 author 和 editor。

默认的数据模型为每一个姓名列表域自动创建了相应的 \ifuse* 测试命令（见 § 4.6.2 节）。同时也自动创建了一个 ifuse* 选项用以控制姓名的标记和排序行为（见 § 3.1.3.1 节）。biber 支持定制姓名成分组合，不过目前定义的姓名成分组合与传统 BibTeX 支持的相同：

- 姓（family name，即‘last’ 部分）
- 名（given name，即‘first’ 部分）
- 前缀（name prefix，即‘von’ 部分）
- 后缀（name suffix，即‘Jr’ 部分）

在默认数据模型中，支持的姓名成分列表由 \DeclareDatamodelConstant 命令定义为一个固定列表（见 4.5.4 节）。然而，由于姓名成分在参考文献驱动 (driver)[19]和后端处理过程中通常需要硬编码，因此，如果想支持额外的姓名成分，将其简单地添加到姓名成分列表中是不够的。关于如何定义和使用定制姓名成分的细节，可以参考示例文件 93-nameparts.tex。关于如何使用定制姓名成分来消除姓名歧义的信息，参见 § 4.11.4 节中的 \DeclareUniquenameTemplate 命令。

文本列表（literal list） 由分隔词 and 划分成独立的项，但各项不再进一步细分。在 bib 文件中，文本列表可以用关键词“and others”来截短。其中又有两个子类型：

（狭义的）文本列表（literal lists in the strict sense） 按照如上所述进行处理。各独立的项目就简单如实打印。典型的狭义文本列表是 publisher 和 location。

[18] 这是针对西方人名的划分。对于中文来说，姓名无需划分。当然中文名的拼音可以进行对应的划分。——译注

[19] 参考文献驱动是 biblatex 中的特有概念，本质上是一个针对具体条目类型组织参考文献数据输出的宏，由\DeclareBibliographyDriver命令定义——译注

关键字列表（key list） 是文本列表的变体，可以包含可打印的数据或本地化关键字。对于列表中每一项，首先测试它是否为已知的本地化关键字（本地化关键字的默认定义在 § 4.9.2 节中）。如果是，那么打印本地化字符串；否则这些项就按本身内容如实打印。典型的关键字列表是 `language`。

域（field） 通常以整体打印。有如下多种子类型：

文本域（literal field） 会如实打印。典型的文本域是 `title` 和 `note`。

范围域（range field） 包含了一个或更多范围，其中所有的短划线都规范化用 `\bibrangedash` 命令取代。一个范围指的是一个非短划线部分后紧跟一个或多个短划线再紧跟一个非短划线部分（比如 `5–7`）。任意数目的连续短划线都只产生一个表示范围的横线。典型的范围域是 `pages` 域。也可以参考 `\bibrangessep` 命令，它用于定制多重范围间的分隔符。如果不包括范围，那么范围域将被忽略并生成警告信息。如果范围域内容混乱，可以使用 `\DeclareSourcemap` 命令在对其解析之前进行规范化（见 § 4.5.3 节）。

整数域（integer field） 包含的整数打印时会转化为序数或者字符串。典型的例子是 `extrayear` 和 `volume` 域。这些域会按照数字进行排序。出于排序的目的，`biber` 会尝试将非阿拉伯数字的表示（例如罗马数字）转成相应的整数。

日期成分域（datepart field） 保存未格式化的整数，打印时会转化为序数或者字符串。典型的例子是 `month` 域。在数据模型中，对于每一个数据类型为 `date` 的域 X，会自动创建带有如下名称的日期成分域：

⟨*datetype*⟩`year`, ⟨*datetype*⟩`endyear`, ⟨*datetype*⟩`month`, ⟨*datetype*⟩`endmonth`, ⟨*datetype*⟩`day`, ⟨*datetype*⟩`endday`, ⟨*datetype*⟩`hour`, ⟨*datetype*⟩`endhour`, ⟨*datetype*⟩`minute`, ⟨*datetype*⟩`endminute`, ⟨*datetype*⟩`second`, ⟨*datetype*⟩`endsecond`, ⟨*datetype*⟩`timezone`, ⟨*datetype*⟩`endtimezone`

其中，对于任何 `datatype=date` 数据模型域，⟨*datetype*⟩ 是在 'date' 之前的前缀字符串。例如，在默认数据模型中，日期域 `date` 可能的前缀字符串包括'event', 'orig', 'url' 和空字符串 ''。

日期域（date field） 保存形如 `yyyy-mm-ddThh:nn[+-][hh[:nn]Z]` 格式的日期，或者 `yyyy-mm-ddThh:nn[+-][hh[:nn]Z]/yyyy-mm-ddThh:nn[+-][hh[:nn]Z]` 格式的日期范围，和其它 EDTF level 1 允许的格式，见 § 2.3.8 节。日期域的特殊之处在于，日期会被解析并分解成各个日期成分。在数据模型中，当定义了一个数据类型为 `date` 的域后，那么相应的 `datepart` 成分（见上文）会自动定义并识别。典型的例子是 `date` 域。

抄录域（verbatim field） 在抄录模式下处理，可以包含特殊字符。典型的抄录域是 `file` 和 `doi`。

URI 域 在抄录模式下处理，可以包含特殊字符，也可以进行 URL 转义。典型的例子是 `url` 域。

分隔值域（separated value field） 被分隔的文本值列表。例子是 keywords 和 options 域。通过 xsvsep 选项可以将分隔符配置成任何 Perl 正则表达式，其默认值是通常 BibTeX 中的（西文）逗号或者逗号加空格。

模式域（pattern field） 是必须匹配某一特定模式的文本域。例如 § 2.2.3 节的 gender 域。

关键字域（key field） 可以保存可打印数据或本地化关键字。使用时，首先测试是否为已知的本地化关键字（本地化关键字的默认定义在 § 4.9.2 一节中）。如果是，就打印本地化字符串；否则，就按本身内容如实打印。典型的例子是 type 域。

代码域（code field） 保存 TeX 代码。

2.2.2 数据域

本节所列的域是在默认数据模型中保存可打印数据的常规域。下面的列表中，左边的名称是域的默认数据模型名，在 biblatex 和后端使用。名称右侧则是相应的 biblatex 数据类型。不同数据类型的解释请参考 § 2.2.1 节。

一些域标记为“label”域，这表示这些域通常用于缩写标签 (abbreviation labels)，当打印文献列表 (bibliography lists) 时（见 § 3.7.3 节的内容）。biblatex 会自动创建支持这些域的宏，详见 § 3.7.3。

abstract 域 (文本)

该域保存bib 文件中记录的摘要，某些特殊的文献著录样式会将其打印出来。但所有的标准样式中都不使用。

addendum 域 (文本)

在条目末尾打印的杂项文献数据。它与 note 域类似，差别在于它是在文献条目末尾打印。

afterword 列表 (姓名)

后记的作者。如果后记作者与 editor 或 translator 相同，那么在参考文献表中标准样式会自动把这些域关联起来。参考 introduction 域和 foreword 域。

annotation 域 (文本)

该域在实现带注释的参考文献著录样式时很有用。所有的标准样式都不使用。请注意，该域与 annotator 域毫无关系，后者是释文（被引用著作的一部分）的作者。

annotator 列表 (姓名)

释文的作者。如果与 editor 或 translator 相同，那么在参考文献表中标准样式会自动把这些域关联起来。参考 commentator 域。

author 列表 (姓名)

title 的作者。

authortype 域 (关键字)

作者的类型。该域会影响介绍作者的字符串。标准文献样式不使用。

bookauthor 列表 (姓名)

`booktitle` 的作者。

bookpagination 域 (关键字)

如果当前作品是另一个大作品的一部分，该域表示包含当前作品的大作品的分页格式。即，`bookpagination` 与 `pagination` 的关系如同 `booktitle` 之于 `title`的关系。该域的值会影响 `pages` 和 `pagetotal` 域的格式。关键字应当是单数形式。可能的关键字包括 `page`、`column`、`line`、`verse`、`section` 和 `paragraph` 等。参考 `pagination` 域以及 § 2.3.10 节。

booksubtitle 域 (文本)

`booktitle` 的副标题。如果说 `subtitle` 域指的是一个大出版物中一小部分作品的副标题，那么该域则给出了整个大作品的副标题。参考 `subtitle`。

booktitle 域 (文本)

如果 `title` 域指的是一个大出版物中的一小部分作品的标题，那么该域则给出了整个大作品的标题。参考 `title`。

booktitleaddon 域 (文本)

`booktitle` 的附语，会用不同的字体打印。

chapter 域 (文本)

作品的章节或其它单元。

commentator 列表 (姓名)

作品评论的作者。请注意，该域用于那种带评论的作品版本，即，在作者之外还有一位评论作者。如果作品是独立的评论，那么评论作者应该在 `author` 域中给出。如果评论作者与 `editor` 或 `translator` 相同，那么在参考文献中标准样式会自动将这些域关联起来。参考 `annotator` 域。

date 域 (日期)

出版日期。参考 `month` 和 `year` 域以及 § 2.3.8 节。

doi 域 (抄录)

作品的数字对象标识符（Digital Object Identifier, DOI）。

edition 域 (整数或文本)

出版物的版次。这必须是整数而不是序数。不要用 `edition={First}` 或 `edition={1st}`，而要用 `edition={1}`。文献样式会将其转为跟语言相关的序数。也可以用文本字符串表示版次，例如“Third, revised and expanded edition”。

editor 列表 (姓名)

`title`、`booktitle` 或者 `maintitle` 的编者，这取决于条目类型。如果不同于真正的“`editor`”角色，可使用 `editortype` 域来确定具体的角色。更多提示参考 § 2.3.6 节。

editora 列表 (姓名)

次要编者，执行汇集、编校等不同编辑任务。可使用 `editoratype` 域来指定具体的角色。更多提示参考 § 2.3.6 节。

editorb 列表 (姓名)

执行不同任务的另一类次要编者。可使用 `editorbtype` 域来指定具体的角色。更多提示参考 § 2.3.6 节。

editorc 列表 (姓名)

执行不同编辑任务的另一类次要编者。可使用 `editorctype` 域来指定具体的角色。更多提示参考 § 2.3.6 节。

editortype 域 (关键字)

`editor` 执行的编者角色类型。默认支持的角色包括 `editor`、`compiler`、`founder`、`continuator`, `redactor`、`reviser` 和 `collaborator`。默认值是“`editor`”，此时该域可以省略。更多提示参考 § 2.3.6 节。

editoratype 域 (关键字)

类似于 `editortype` 但对应的是 `editora` 域。更多提示参考 § 2.3.6 节。

editorbtype 域 (关键字)

类似于 `editortype` 但对应的是 `editorb` 域。更多提示参考 § 2.3.6 节。

editorctype 域 (关键字)

类似于 `editortype` 但对应的是 `editorc` 域。更多提示参考 § 2.3.6 节。

eid 域 (文本)

`@article` 的电子标识符（electronic identifier）。

entrysubtype 域 (文本)

该域用于指定一个条目类型的子类型。它不会在标准样式中使用，但可用于支持更细化的条目类型集的参考文献样式。

eprint 域 (抄录)

在线出版物的电子标识符。它大致相当于 DOI，但专门针对某个档案、资源库、服务或系统。参考 § 3.12.7 节以及 `eprinttype` 和 `eprintclass` 域。

eprintclass 域 (文本)

域 eprinttype 域指明的资源相关的附加信息。它可以是档案的一部分、标示服务的路径、排序的某个分类等等。参考 § 3.12.7 节以及 eprint 和 eprinttype 域。

eprinttype 域 (文本)

eprint 标识符的类型，例如 eprint 域所指的档案、资源库、服务或系统的名称。参考 § 3.12.7 节以及 eprint 和 eprintclass 域。

eventdate 域 (日期)

@proceedings 和 @inproceedings 条目中的会议、研讨会或其它活动的日期。该域还可用于 § 2.1.3 一节所列的定制类型。参考 eventtitle 和 venue 域以及 § 2.3.8 节。

eventtitle 域 (文本)

@proceedings 和 @inproceedings 条目中的会议、研讨会或其它活动的标题。该域还可以用于在 § 2.1.3 一节所列的定制类型。请注意，该域保存事件的普通标题。而诸如“Proceedings of the Fifth XYZ Conference”之类的信息会归入 titleaddon 或 booktitleaddon 域。参考 eventdate 和 venue 域。

eventtitleaddon 域 (文本)

eventtitle 域的附语。例如可以用于已知活动的首字母缩略词。

file 域 (抄录)

某个作品的 PDF 或其它版本的本地链接。标准样式中不使用。

foreword 列表 (姓名)

作品前言的作者。如果前言的作者与 editor 或 translator 相同，那么在参考文献表中标准样式会自动将其与这些域关联起来。参考 introduction 和 afterword 域。

holder 列表 (名称)

@patent 的持有者（如果与 author 不同的话）。注意，持有者是一个集体 (机构) 时，需要将其放在额外的花括号内，参考 § 2.3.3 节。该域可以用于 § 2.1.3 节所列的定制类型中。

howpublished 域 (文本)

不能划归任何常见类型的非常规出版物的出版通告。

indextitle 域 (文本)

替代常规的 title 域在索引中使用的标题。如果你有一个带有“An Introduction to ...”之类标题的条目，并且想索引为“Introduction to ..., An”，那么就可以使用该域。样式作者需要注意，如果 indextitle 未定义，那么 biblatex 会自动将 title 域的值复制给 indextitle。

institution 列表 (文本)

大学或其它研究机构的名称，这取决于条目类型。而传统上，BibTeX 使用 `school` 域来表示这些信息。本宏包也支持 `school`，但只作为本域的别名。另见 §§ 2.2.5 和 2.3.4 节。

introduction 列表 (姓名)

作品导论的作者。如果导论的作者与 `editor` 或 `translator` 相同，那么在参考文献表中标准样式就会自动将这些域关联起来。参考 `foreword` 和 `afterword` 域。

isan 域 (文本)

音像作品的音像数码国际标准（International Standard Audiovisual Number, ISAN）。不会在标准文献样式中使用。

isbn 域 (文本)

书籍的国际标准书号（International Standard Book Number, ISBN）。

ismn 域 (文本)

乐谱等发行音乐作品的国际标准音乐作品编码（International Standard Music Number, ISMN）。

isrn 域 (文本)

技术报告的国际标准技术报告编码（International Standard Technical Report Number, ISRN）。

issn 域 (文本)

连续出版物的国际标准连续出版物号（International Standard Serial Number, ISSN）。

issue 域 (文本)

期刊的期号。该域适用于期号由“Spring”或“Summer”等名称而不是月份或数字确定的期刊。由于 `issue` 的位置与 `month` 和 `number` 类似，该域也可用于合期或其它特殊场合[20]。参考 `month` 和 `number` 域以及 § 2.3.9 节。

issuesubtitle 域 (文本)

期刊或其它连续出版物中某一期的副标题。

issuetitle 域 (文本)

期刊或其它连续出版物中某一期的标题。

iswc 域 (文本)

音乐作品的国际标准作品号（International Standard Work Code, ISWC）。标准文献样式中不使用。

[20] 例如增刊、特刊等。——译注

journalsubtitle 域 (文本)

期刊、报纸或其它连续出版物的副标题。

journaltitle 域 (文本)

期刊、报纸或其它连续出版物的标题。

label 域 (文本)

在标注样式中，如果生成常规标签的所需数据均缺失，那么该域的内容可用来代替常规标签。例如，当*作者年制*标注样式要生成某个条目的标签，但该条目的作者或年份缺失，那么会使用后备的 `label`。详见 § 2.3.2 节。请注意，与 `shorthand` 域相反，`label` 只是作为后备而使用。另可参见 `shorthand`。

language 列表 (关键字)

作品的语言。语言可以由文本内容或者本地化关键字指定。如果使用本地化关键字，那么前缀 `lang` 可省略。参考 `origlanguage` 域并比较 § 2.2.3 节中的 `langid`。

library 域 (文本)

该域可用于记录图书馆名称或书架号码等信息。某些特殊的参考文献样式可能需要将其打印出来。但在标准样式中不使用。

location 列表 (文本)

出版地，即 `publisher` 或 `institution`（取决于条目类型）的所在地。传统上 BibTeX 使用 `address` 域，本宏包也支持 `address`，但只作为本域的别名。参考 §§ 2.2.5 和 2.3.4 几节。在 `@patent` 条目里，该列表用于表示专利的权利范围。该文本列表可用于 § 2.1.3 中的定制类型。

mainsubtitle 域 (文本)

对应于 `maintitle` 的副标题。参考 `subtitle` 域。

maintitle 域 (文本)

多卷本书籍（例如著作集）的主标题。如果说 `title` 或 `booktitle` 域指的是多卷本中某一单卷的标题，那么该域则给出了全集的标题。

maintitleaddon 域 (文本)

`maintitle` 的附言，会用不同的字体打印。

month 域 (日期成分)

出版月份。必须是整数，而不能是序数或字符串。例如，使用 `month={1}` 而不是 `month={January}`。文献样式会在需要时将它转换为语言相关的字符串或序数。当显式给出数据时 (例如为兼容原始的 BibTeX)，该域转变为一个文本域。然而最好还是使用`date`，因为它能支持更多的功能。参考 `date` 以及 §§ 2.3.9 和 2.3.8。

nameaddon 域 (文本)

参考文献中紧随作者名之后输出的插入语。标准文献样式中不使用。该域可用于添加别名或笔名（或者给出原名，如果常用化名来表示作者话）。

note 域 (文本)

不可归类于其它域的杂项文献数据。`note` 域可以用于记录自由格式的文献数据。`note` 域包含一些典型信息，例如“Reprint of the edition London 1831”这样出版信息。另见 `addendum` 域。

number 域 (整数)

期刊的期数或者 `series` 丛书中某本书的卷数/期数。另见 `issue` 域以及 §§ 2.3.7 和 2.3.9 节。在 `@patent` 条目中，这是专利或专利申请号或登记号。其值应该是一个整数，但不必是阿拉伯数字的形式，因为 `biber` 为了排序会自动将罗马数字或者阿拉伯字符转成整数。

organization 列表 (文本)

出版 `@manual` 或 `@online` 资源，以及主办会议的组织。另可参考 § 2.3.4 节。

origdate 域 (日期)

如果作品是译作、重印或其它类似情况，该域指的是初版日期。在标准文献样式中不使用。另可参考 `date` 域。

origlanguage 列表 (关键字)

如果作品是译作，该域指的是原作使用的语言。另可参考 `language` 域。

origlocation 列表 (文本)

如果作品是译作、重印或其它类似情况，该域指的是初版的 `location`。标准文献样式不使用。另可参考 `location` 域和 § 2.3.4 节。

origpublisher 列表 (文本)

如果作品是译作、重印或其它类似情况，该域指的是初版的 `publisher`。在标准文献样式中不使用。参考 `publisher` 域和 § 2.3.4 节。

origtitle 域 (文本)

如果作品是译作，该域指的是原作的 `title`。标准文献样式不使用。另可参考 `title` 域。

pages 域 (范围)

一个或多个页码数或页码范围。如果当前作品是另一个大作品的一部分，例如期刊或文集析出中的文章，该域指的是当前作品在相关大作品中的页码范围。它也可以用于限定著作中某一特定部分（例如一本书中的一章）。

pagetotal 域 (文本)

作品的总页码数。

pagination 域 (关键字)

作品的页码标记格式。该域的值会影响标注命令 ⟨*postnote*⟩ 参数的格式。关键字应当以单数的形式给出。可能的关键字包括 `page`、`column`、`line`、`verse`、`section` 和 `paragraph`。另可参考 `bookpagination` 域以及 §§ 2.3.10 和 3.13.3 节。

part 域 (文本)

某一分卷的编号。该域只用于书籍而不能用于期刊。它可以用于一个逻辑卷包括两个或更多实际卷的情形。如果这样的话，逻辑卷册的编号由 `volume` 给出，而该逻辑卷的分卷编号由 `part` 给出。另可参考 `volume` 域。

publisher 列表 (文本)

出版者的名称。参考 § 2.3.4 节。

pubstate 域 (关键字)

作品的出版状态，例如“in press”。已知的出版状态请参考 § 4.9.2.11 节。

reprinttitle 域 (文本)

作品重印版的标题。标准样式中不使用。

series 域 (文本)

丛书的名称，例如“Studies in ...”，或者期刊系列的编号。系列出版的丛书通常带有编号。其编号或者卷号由 `number` 域给出。请注意，`@article` 条目类型也使用 `series` 域，但是以一种特别的方式处理。详见 § 2.3.7 节。

shortauthor 列表 (姓名) Label field

作者的缩写形式。该域主要用于集体作者的缩写形式。参考 § 2.3.3 节。

shorteditor 列表 (姓名) Label field

编者的缩写形式。该域主要用于集体编者的缩写形式。参考 § 2.3.3 节。

shorthand 域 (文本) Label field

在标注样式中，用于替代常规标签的特殊标签。如果有定义，那么它会覆盖默认的标签。另可参考 `label` 域。

shorthandintro 域 (文本)

本宏包附带的多词标注样式会在首次引用时使用类似“henceforth cited as [shorthand]”这样的短语来引入 [shorthand]。如果 `shorthandintro` 域有定义，它将覆盖标准短语。请注意，使用的备选短语必须包含 shorthand。

shortjournal 域 (文本) Label field

`journaltitle` 的缩写版本或其首字母缩略语。标准文献样式中不使用。

shortseries 域 (文本) Label field

`series` 的缩写版本或其首字母缩略语。标准文献样式中不使用。

shorttitle 域 (文本) Label field

缩略形式的标题。该域通常不会包括在参考文献表中。它用于 `author-title` 格式的标注。如果该域存在，`author-title` 标注样式使用该域来代替 `title` 域。

subtitle 域 (文本)

作品的副标题。

title 域 (文本)

作品的标题。

titleaddon 域 (文本)

`title` 的附文，会用不同字体打印。

translator 列表 (姓名)

`title` 或 `booktitle` 的译者，具体取决于条目类型。如果译者与 `editor` 相同，在文献表中中标准样式会自动将这些域关联起来。

type 域 (关键字)

`manual`、`patent`、`report` 或 `thesis` 的类型。该域可用于 § 2.1.3 节的定制类型。

url 域 (uri)

在线出版物的 URL。如果它不是 URL-转义的（没有“%”字符），那么会根据 RFC 3987[21] 对其做 URI-转义，也就是说，即使 Unicode 字符也会正确转义。

urldate 域 (日期)

`url` 域中网址的访问日期。见 § 2.3.8 节。

venue 域 (文本)

`@proceedings` 和 `@inproceedings` 条目中的会议、研讨会或其它活动的地点。该域可用于 § 2.1.3 一节所列的定制类型。请注意，`location` 列表指的是出版地点，因此对应于 `publisher` 和 `institution` 列表。而会议活动的会场地点则由 `venue` 域给出。另可参考 `eventdate` 和 `eventtitle` 域。

version 域 (文本)

软件、手册等作品的修订版本号。

[21] 参考 https://tools.ietf.org/html/rfc3987 ——译注

volume 域 (整数)

多卷本或连续出版物中的卷数。其值应当是整数，但不必是阿拉伯数字的形式，因为 biber 为了排序会将罗马数字和阿拉伯字符自动转成整数。另可参考 part 域。

volumes 域 (整数)

多卷本著作的总卷数。根据文献条目类型，该域对应于 title 或 maintitle 域。其值应当是整数，但不必是阿拉伯数字的形式，因为 biber 为了排序会将罗马数字和阿拉伯字符自动转成整数。

year 域 (文本)

出版年份。只有显式给出数据时 (例如为兼容原始的 BibTeX)，该域才是文本类型的域。不过最好使用 date 域，因为它也能兼容显式年份 (plain years) 且支持更多功能[22]。见 § 2.3.8 节。

2.2.3 特殊域

本节中的域不保存可打印数据，而有其它用途。在默认数据模型中，这些域可用于所有条目类型。

crossref 域 (条目关键字)

该域保存的条目关键字可用于交叉引用。带有 crossref 域的子条目可以从由 crossref 域指定的父条目继承数据。如果引用某个父条目的子条目数量达到一个阈值，该父条目就会自动添加到参考文献表中，即使它没有在正文中被显式引用。该阈值可以由 § 3.1.2.1 节中的 mincrossrefs 宏包选项设置。样式作者请注意，在 biblatex 层面上，子条目的 crossref 域是否有定义取决于父条目是否可用。如果父条目可用，那么子条目的 crossref 域将被定义。反之，子条目仍然可以继承父条目的数据但是其 crossref 域是未定义的。父条目是否被添加到文献中（由于阈值隐式地或者由于被引用而显式地被引入）对于该域的定义并不重要。另可参考本节的 xref 域以及 § 2.4.1 节。

entryset 域 (分隔值)

该域是条目集专用的。详见 § 3.12.5 节。在后端程序处理过程中该域会被清除而不出现在 .bbl 中。

execute 域 (代码)

保存任意 TEX 代码的特殊域，这些代码会在获取各条目数据时被执行。这对处理特殊情况很有用。概念上，该域可以类比于 § 4.10.6 节中的钩子命令 \AtEveryBibitem、\AtEveryLositem 和 \AtEveryCitekey，但不同之处在于该域可以基于 bib 文件中的每一条目进行逐条定义。该域中的任何代码都会在这些钩子命令后立即自动执行。

[22] 这里的 plain years 本质上是显式给出年份信息而不是由 date 解析给出的年份，结合 § 2.3.8 节的 explicite year，译为显式年份。而 plain BibTeX 的意义并不明确，可能是采用显式年份方式的 BibTeX 这里暂不深入探究合适的译法——译注

语言	地区/方言	标识符
保加利亚语	保加利亚	bulgarian
加泰罗尼亚语	西班牙、法国、安道尔、意大利	catalan
克罗地亚语	克罗地亚、波黑、塞尔维亚	croatian
捷克语	捷克共和国	czech
丹麦语	丹麦	danish
荷兰语	荷兰	dutch
英语	美国	american, USenglish, english
	英国	british, UKenglish
	加拿大	canadian
	澳大利亚	australian
	新西兰	newzealand
爱沙尼亚语	爱沙尼亚	estonian
芬兰语	芬兰	finnish
法语	法国、加拿大	french
德语	德国	german
	奥地利	austrian
	瑞士	swissgerman
德语（新正字法）	德国	ngerman
	奥地利	naustrian
	瑞士	nswissgerman
希腊语	希腊	greek
意大利语	意大利	italian
挪威语	挪威	norwegian, norsk, nynorsk
波兰语	波兰	polish
葡萄牙语	巴西	brazil
	葡萄牙	portuguese, portuges
俄语	俄罗斯	russian
斯洛伐克语	斯洛伐克	slovak
斯洛文尼亚语	斯洛文尼亚	slovene
西班牙语	西班牙	spanish
瑞典语	瑞典	swedish
乌克兰语	乌克兰	ukrainian

表 2: 支持的语种

gender 域 (匹配 sf、sm、sn、pf、pm、pn、pp 其中之一的模式)

作者或编者（如果没有作者的话）的词性。支持以下标识符：sf （阴性单数，单个女性名）, sm（阳性单数，单个男性名），sn（中性单数，单个中性名），pf（阴性复数，多个女性名），pm（阳性复数，多个男性名），pn（中性复数，多个中性名），pp（复数，不同词性名的组合）。这一信息只在特殊参考文献著录和标注样式，以及某些特定语言中是需要的。例如，某一标注样式会用想“idem”这样的词汇来代替反复出现的作者姓名，如果按照英语或法语的习惯使用这一拉丁词汇，那么就没有必要指定词性。然而在德语出版物中，这样的关键词汇通常用德语给出，此时就会与词性相关。

langid 域 (标识符)

文献条目的语种标识。出于向后兼容性考虑，提供了别名 hyphenation。标识符必须是 babel/polyglossia 宏包中的语言名称。该信息用于在文献表中切换断词模式和本地化字符串。请注意，语言名是大小写敏感的。目前本宏包支持的语言在表 2 中给出。需要注意的是，babel 宏包将标识符 english 当作 british 或 american 的

别名，具体取决于 babel 的版本。而 biblatex 宏包总是将其当作 american 的别名。为了避免可能的混淆，最好使用语言标识符 american 和 british（babel）或者一个语言选项来指定一种变体语言（polyglossia，使用 langidopts 域）。可与 § 2.2.2 节中的 language 域进行比较。

langidopts 域 (文本)

对于 polyglossia 的用户，该域使得每个条目可拥有自己的语言选项。当使用本宏包的选项 autolang=langname 时，该域的值将被传递到 polyglossia 的语言切换工具中。例如，使用如下域:

```
langid         = {english},
langidopts     = {variant=british},
```

会将文献条目置于如下代码块中

```
\english[variant=british]
...
\endenglish
```

ids 域 (条目关键字的分隔值列表)

主要引用关键字的别名。一个条目可以通过别名进行引用，biblatex 会将其视为使用了原始的引用关键字。借助该域，用户可以在改变引用关键字后，仍然可以使用原来使用旧的引用关键字的老文档。该域在后端程序处理过程中会被清楚，不出现在 .bbl 中。

indexsorttitle 域 (文本)

排序索引时使用的标题。与 indextitle 域不同，该域只用于排序。而索引中打印出来的标题是 indextitle 或title 域。当标题中含有与索引排序相冲突的特殊字符或命令时，该域会很有用。考虑如下例子：

```
title          = {The \LaTeX\ Companion},
indextitle     = {\LaTeX\ Companion, The},
indexsorttitle = {LATEX Companion},
```

文献样式作者请注意，当 indexsorttitle 没有定义时，biblatex 会自动将 indextitle 或 title 域的值复制给该域。

keywords 域 (分隔值)

关键词的分隔值列表。这些关键词主要用于文献过滤器，通常不会打印出来 (见 §§ 3.7.2 和 3.12.4 节)。请注意，使用默认的分隔符（西文逗号）时，分隔符左右的空格会被忽略。

options 域 (分隔的 ⟨*key*⟩=⟨*value*⟩ 选项)

⟨*key*⟩=⟨*value*⟩ 形式的条目选项分隔值列表。该域用于设置每一条目的选项，详见 § 3.1.3 节。请注意，标注和著录样式会定义额外的条目选项。

presort 域 (字符串)

用于修改文献排列顺序的特殊域。文献排序时，该域排序程序第一个考虑的项，因此可用于将文献条目分组。这在利用文献过滤器创建文献细分时很有用。更多细节请参考 § 3.5 以及 § 4.5.6 节。该域在后端程序处理过程中被清楚，不出现在 `.bbl` 中。

related 域 (分隔值)

与本条目关联的其它条目的引用关键字。其关联关系由 `relatedtype` 域指定。更多细节请参考 § 3.4 节。

relatedoptions 域 (分隔值)

为关联条目设置类型相关的选项。请注意，这不会设置关联条目本身的选项，而只会影响作为数据源的父条目的临时副本。

relatedtype 域 (标识符)

标识符，为列在 `related` 域中的关键字列表指定关联关系类型。该标识符是本地化字符串，会在来自关联条目列表的数据之前打印。该域也用于为关联条目指明类型相关的格式化指令和参考文献宏。详见 § 3.4 节。

relatedstring 域 (文本)

用于覆盖 `relatedtype` 指定的参考文献字符串。更多细节请参考 § 3.4 节。

sortkey 域 (文本)

用来修改文献排序的域。该域可以认为是最主要的排序键值。当该域存在时，`biblatex` 会在排序时使用它，并且忽略除 `presort` 域之外的所有信息。详见 § 3.5 节。该域在后端处理过程中会被清除，不出现在 `.bbl` 中。

sortname 列表 (姓名)

用于修改文献排序的姓名或姓名列表。如果该域存在，在文献排序时，它会取代 `author` 或 `editor` 域。详见 § 3.5 节。该域在后端程序处理过程中会被清除，不出现在 `.bbl` 中。

sortshorthand 域 (文本)

与 `sortkey` 类似但用于缩略语列表中。如果存在，在缩略语列表排序时，`biblatex` 会用该域取代 `shorthand` 域。当 `shorthand` 域含有格式化命令（如 `\emph` 或 `\textbf`）的缩略语时，该域是很有用的。该域在后端程序过程中会被清除，不出现在 `.bbl` 中。

sorttitle 域 (文本)

用于修改文献排序的域。如果存在，在文献排序时，该域会取代 `title` 域。如果一个条目带有“An Introduction to...”这样的标题，并且你想让它按字母“I”而不是“A”排序，那么 `sorttitle` 域就会派上用场。这时，你就可以在 `sorttitle` 域中填上“Introduction to...”。详见 § 3.5 节。该域在后端程序处理过程中被清除，不出现在 `.bbl` 中。

sortyear 域 (整数)

用于修改文献排序的域。如果存在，在文献排序时，该域会取代 `year` 域。详见 § 3.5 节。该域在后端程序处理过程中被清除，不出现在 `.bbl` 中。

xdata 域 (条目关键字的分隔值列表)

该域从一个或更多 `@xdata` 条目中继承数据。从概念上讲，`xdata` 域与 `crossref` 和 `xref` 域相关：`crossref` 创建一个继承数据的父/子逻辑关系；`xref` 创建一个不继承数据的父/子逻辑关系；而 `xdata` 则继承数据却不创建关系。`xdata` 的值可以是一个单个条目关键字或者条目关键字的分隔值列表。详见 § 3.12.6 节。该域在后端程序处理过程中被清除，不出现在 `.bbl` 中。

xref 域 (条目关键字)

该域可用于代替交叉引用机制。它与 `crossref` 域的不同之处在于，子条目不会从其 `xref` 域所列的父条目中继承数据。如果引用某个父条目的子条目数量达到一个阈值，该父条目就会自动添加到文献表中，即使它并没有显式地被引用。该阈值可以由 § 3.1.2.1 节的 `mincrossrefs` 宏包选项设置。样式作者需要注意，在 `biblatex` 层面上，子条目的 `xref` 域是否有定义取决于父条目是否可用。如果父条目可用，那么子条目的 `xref` 域将被定义。反之，其 `xref` 域是未定义的。父条目是否被添加到文献表中（由于阈值隐式地或者由于被引用而显式地被引入）对于域的定义并不重要。另可参考本节中的 `crossref` 域以及 § 2.4.1 节。

2.2.4 可定制域

本节中的域用于特殊的参考文献样式，标准样式不使用。

name[a–c] 列表 (姓名)

特殊文献样式的定制列表。标准文献样式不使用。

name[a–c]type 域 (关键字)

类似于 `authortype` 和 `editortype` 域，不过对应的是 `name[a–c]` 域。标准文献样式不使用。

list[a–f] 列表 (文本)

特殊文献样式的定制列表。标准文献样式不使用。

user[a–f] 域 (文本)

特殊文献样式的定制域。标准文献样式不使用。

verb[a–c] 域 (文本)

类似于前述的定制域，不过这些是抄录域。标准文献样式不使用。

2.2.5 域的别名

本节列出的别名用于向后兼容传统 BibTeX 以及其它基于传统 BibTeX 样式的应用。请注意，处理 bib 文件时即会解析这些别名。因此所有的参考文献著录和标注样式必须使用它们所指域的名称，而不能这些别名。但在 bib 文件中，既可以使用别名，也可以它们使用所指域的名称，但不能同时使用。

address 列表 (文本)

location 的别名，用于兼容 BibTeX 。传统的 BibTeX 使用这一稍微有些误导性的域 address 来表示出版地点，即出版者的所在地，而 biblatex 使用更一般的 location 域。见 §§ 2.2.2 和 2.3.4 节。

annote 域 (文本)

annotation 的别名，用于兼容 jurabib 宏包。见 § 2.2.2 节。

archiveprefix 域 (文本)

eprinttype 的别名，用于兼容 arXiv 。见 §§ 2.2.2 和 3.12.7 节。

journal 域 (文本)

journaltitle 的别名，用于兼容 BibTeX 。见 § 2.2.2 节。

key 域 (文本)

sortkey 的别名，用于兼容 BibTeX 。见 § 2.2.3 节。

pdf 域 (抄录)

file 的别名，用于兼容 JabRef 。见 § 2.2.2 节。

primaryclass 域 (文本)

eprintclass 的别名，用于兼容 arXiv 。见 §§ 2.2.2 和 3.12.7 节。

school 列表 (文本)

institution 的别名，用于兼容 BibTeX 。传统 BibTeX 中，institution 用于技术报告，而 school 域保存与之相关的研究机构[23]。在这两种情况下，biblatex 宏包都会使用一般的域 institution。见 §§ 2.2.2 和 2.3.4。

[23]原文中，所谓比较的两种情况并没有说清楚，暂不深入——译注

2.3 使用注意事项

对于熟悉 BibTeX 的用户来说，本宏包支持的绝大部分条目类型和域都是很直观的。然而，且不说本宏包额外新增的类型和域，一些熟悉的类型和域的处理方式也需要进一步解释一下。

本宏包考虑到包含一些兼容性代码，用于处理那些由传统 BibTeX 样式生成的 bib 文件。但不幸的是，对所有的老文件都进行自动处理是不可能的，因为 biblatex 的数据模型与传统的 BibTeX 有少许不同。因此，为了能在本宏包中正确使用，这样的 bib 文件也许需要稍作修改。大体上，下列事项是与传统的 BibTeX 样式不同的：

- @inbook 条目类型。详见 §§ 2.1.1 和 2.3.1 节。
- institution、organization、publisher 域以及相应的别名 address 和 school。详见 §§ 2.2.2、2.2.5、2.3.4 节。
- 一些标题类型的处理。详见 § 2.3.5 节。
- series 域。详见 §§ 2.2.2 和 2.3.7 节。
- year 和 month 域。详见 §§ 2.2.2、2.3.8、2.3.9 节。
- edition 域。详见 § 2.2.2 节。
- key 域。详见 § 2.3.2 节。

jurabib 宏包的用户请注意，shortauthor 域被 biblatex 视作姓名列表，详见 § 2.3.3 节。

2.3.1 @inbook 条目类型

@inbook 条目类型用于书籍中有自己标题的独立部分。它与 @book 的关系正如同 @incollection 与 @collection 的关系。示例见 § 2.3.5 节。如果你想要指书中的某一章节，直接使用 book 类型并添加 chapter 或 pages 域即可。参考文献表中究竟是否可以引用章节是有争议的，因为章并不是文献实体。

2.3.2 缺失和可忽略数据

在 § 2.1.1 节中标记为“required”的域并不一定在所有情况下都是严格需要的。对于本宏包附带的参考文献样式，绝大部分条目类型，即便只包含 title 域也能使用。就参考文献表而言，匿名出版的书籍、没有明确编者的周期性出版物、或者没有明确作者的软件手册都应当不会有问题。但是，标注样式也许会有不同的要求。例如，author-year 标注样式就明确要求 author/editor 域和 year 域。

一般来说，可以使用 label 域代替标注所要求的任意缺失数据。label 域的使用方式取决于标注样式。如果 author/editor 域或 year 域缺失，本宏包所带的 author-year 标注样式会将 label 域作为备用信息。另一方面，顺序编码制样式根本不会用到这些，因为顺序编码格式与该数据无关。此外，author-title 样式也会忽略这些，因为单靠 title 域已经足以生成惟一的标注，而标题几乎在所有情形中都是存在的。在顺序字母 (alphabetic) 标注样式中，label 域也可以用于覆盖自动生成的 labelalpha 域中的非数值部分。详见 § 4.2.4 节。

请注意，当 author 和 editor 域都缺失时，传统的 BibTeX 样式支持 key 域用于依字母排序。biblatex 宏包将 key 视为 sortkey 的别名。此外，biblatex 还提供了非常细化的排序控制，详见 §§ 2.2.3 和 3.5 节。natbab 宏包使用 key 域作为备用的标注标签，而biblatex 则使用 label 域来代替。

2.3.3 集体作者和集体编者

集体作者和集体编者分别在 author 和 editor 域中给出。请注意，他们必须再用花括号括起来，以防被认为是个人姓名进而被分解成姓名成分。如果你想在标注时给出简称或首字母缩写的形式，请使用 shortauthor 域。

```
author       = {{National Aeronautics and Space Administration}},
shortauthor  = {NASA},
```

默认的标注样式会在所有标注里使用短名称而在参考文献表中打印全名。对于集体编者，则使用 editor 和 shorteditor 域。由于这些域都被视作姓名列表，因此，只要把所有的集体作者和单位用花括号括起来，就可以将个人姓名与集体名称混合使用。

```
editor       = {{National Aeronautics and Space Administration}
                and Doe, John},
shorteditor  = {NASA and Doe, John},
```

从 jurabib 宏包转到 biblatex 宏包的用户需要注意，shortauthor 域被视作姓名列表。

2.3.4 文本列表

按照 § 2.2 节，institution、organization、publisher 和 location 等域是文本列表。origlocation、origpublisher，以及作为别名的 address 和 school 域也是如此。所有的这些域都可以包含一个由关键词“and”分隔的项列表。如果它们本身带有“and”文本，那么必须用花括号括起来。

```
publisher    = {William Reid {and} Company},
institution  = {Office of Information Management {and} Communications},
organization = {American Society for Photogrammetry {and} Remote Sensing
                and
                American Congress on Surveying {and} Mapping},
```

请注意以上例子中作为文本和作为列表分隔符的“and”之间的区别。你也可以把整个名称用括号括起来：

```
publisher    = {{William Reid and Company}},
institution  = {{Office of Information Management and Communications}},
organization = {{American Society for Photogrammetry and Remote Sensing}
                and
                {American Congress on Surveying and Mapping}},
```

对于一些老文件，即使没有针对 biblatex 宏包做更新，即这些域中不含“and”文本，仍然可以使用。然而需要注意，这种情况下，你会丢失那些属于文本列表的额外特性，例如配置格式和自动截短。

2.3.5 标题

以下例子展示了如何处理不同类型的标题。首先是一个作为整体的五卷本作品：

```
@MvBook{works,
  author     = {Shakespeare, William},
  title      = {Collected Works},
  volumes    = {5},
  ...
```

多卷本作品的每一卷通常有自己的标题。假设该多卷文选的第四卷是莎士比亚的十四行诗，并且我们要单独引用该卷：

```
@Book{works:4,
  author     = {Shakespeare, William},
  maintitle  = {Collected Works},
  title      = {Sonnets},
  volume     = {4},
  ...
```

如果单卷没有标题，我们在 title 域中给出主标题，并标明卷数：

```
@Book{works:4,
  author     = {Shakespeare, William},
  title      = {Collected Works},
  volume     = {4},
  ...
```

在下个例子里，我们引用一卷的某一部分，但是该部分是一个独立作品且有自己的标题。相应的卷也有一个标题，并且整个作品有一个主标题：

```
@InBook{lear,
  author     = {Shakespeare, William},
  bookauthor = {Shakespeare, William},
  maintitle  = {Collected Works},
  booktitle  = {Tragedies},
  title      = {King Lear},
  volume     = {1},
  pages      = {53-159},
  ...
```

假设多卷文选的第一卷是由一位著名学者写的再版评论文章。这不是常见的由编者写的简介，而是一份独立的作品。且多卷文选另有编者：

```
@InBook{stage,
  author     = {Expert, Edward},
  title      = {Shakespeare and the Elizabethan Stage},
  bookauthor = {Shakespeare, William},
  editor     = {Bookmaker, Bernard},
  maintitle  = {Collected Works},
  booktitle  = {Tragedies},
  volume     = {1},
  pages      = {7-49},
  ...
```

更多例子请参考 § 2.3.7 节。

2.3.6 编者角色

编者域（包括 editor、editora、editorb、editorc等）中编者角色类型可以由相应的editor...type 域指定。biblatex 默认支持下述多种角色，其中“editor”是默认缺省角色，当采用缺省角色时，editortype 域可省略。

editor 主要编者。这是最普通的编者角色，也是默认值。

compiler 类似于 editor，但适用于编者主要进行编纂工作的情况。

founder 诸如“多卷文选”或连续的法律评论等连续的或综合的出版项目的创始编者。

continuator 继续创立者（founder）工作的编者。创立者的工作由现任编辑（editor）所接替。

redactor 从事编修工作的次要编者。

reviser 从事校订工作的次要编者。

collaborator 次要编者或者主编的顾问。

organizer 类似于 editor，但适用于当编者主要进行组织整理工作的情况。

例如，如果编者的任务是编纂的话，你可以在相应的 editortype 域中指明：

```
@Collection{...,
  editor      = {Editor, Edward},
  editortype  = {compiler},
  ...
```

除主编之外可以有次要编者：

```
@Book{...,
  author      = {...},
  editor      = {Editor, Edward},
```

```
  editora     = {Redactor, Randolph},
  editoratype = {redactor},
  editorb     = {Consultant, Conrad},
  editorbtype = {collaborator},
  ...
```

期刊或长期连续的出版项目通常有不同阶段的编者。例如，除现任编者之外还可以有一位创始编者：

```
@Book{...,
  author      = {...},
  editor      = {Editor, Edward},
  editora     = {Founder, Frederic},
  editoratype = {founder},
  ...
```

请注意，在正文标注中以及在文献表排序时，只有 `editor` 域会起作用。如果一个条目要特地引用创始编者（并且据此在文献中排列），那么创始编者应在 `editor` 域中给出，而现任编者则移动到 `editor...` 域中：

```
@Collection{...,
  editor      = {Founder, Frederic},
  editortype  = {founder},
  editora     = {Editor, Edward},
  ...
```

可以通过初始化和定义新的本地化关键字来增加更多的角色，关键字的名称对应于 `editor...type` 域中的标识符。详见 §§ 3.9 和 4.9.1 节。

2.3.7 出版物和期刊系列

在传统的 BibTeX 样式中，`series` 域既用于多卷本作品的主标题 (main title)，也用于出版物系列，例如同一出版者的针对大致相同的一个方向或者同一个研究领域的关系较松散的一系列书籍。这种用法是容易导致模糊的。因此，本宏包引入了 `maintitle` 域来表示多卷本作品，而 `series` 只用于出版物系列。系列中某一本书的卷号或序号由 `number` 域给出：

```
@Book{...,
  author        = {Expert, Edward},
  title         = {Shakespeare and the Elizabethan Age},
  series        = {Studies in English Literature and Drama},
  number        = {57},
  ...
```

@article 条目类型也使用 series 域，但是使用方式比较特殊。首先，会执行一个测试来确定该域的值是否是整数。如果是的话，它会以序数的形式打印；反之，会执行另一个测试来确定它是否是本地化关键字。如果是的话，会打印本地化字符串；反之则按照本身内容如实打印。考虑下面这个以数字系列出版的期刊例子：

```
@Article{...,
  journal         = {Journal Name},
  series          = {3},
  volume          = {15},
  number          = {7},
  year            = {1995},
  ...
```

该条目会打印成“*Journal Name.* 3rd ser. 15.7 (1995)”。一些期刊也会使用“旧系列”(“old series”) 和“新系列”(“new series”) 等标识来代替数字。这样的标识也可以由 series 域给出，或者是一个文本字符串，或者是一个本地化关键字。考虑如下这个使用本地化关键字 newseries 的例子：

```
@Article{...,
  journal         = {Journal Name},
  series          = {newseries},
  volume          = {9},
  year            = {1998},
  ...
```

该条目会打印成“*Journal Name.* New ser. 9 (1998)”。默认定义的本地化关键字列表请参考 § 4.9.2 节。

2.3.8 日期和时间规范

日期域，例如默认数据模型的日期域 date、origdate、eventdate 和 urldate等，遵循扩展日期时间格式[24]（Extended Date/Time Format, EDTF）规范标准 0 和标准 1。此外也支持 ISO8601-2 当前草案第 4.5 节中的开放式范围规范[25]。相比于ISO8601v2004允许的有些混乱的格式，EDTF 是其中一个更严格的子集，更适用于参考文献数据。

除了 EDTF 空日期范围标记外，还通过给定范围分隔符并省略结束或开始日期的方式（例如 YYYY/、/YYYY）来指定无末端或无开端的日期范围。表 3 列出了一些有效的日期规范以及由 biblatex 自动生成的日期格式。日期格式与语言有关，因此会自动调整。如果条目中没有 date 域，biblatex 还会考虑 year 和 month 域。不过这仅仅出于对传统 BibTeX 的向后兼容性考虑，并不鼓励使用。因为显式的 year 和 month 域不能解析为日期的元信息标记，只能原样使用。

样式作者需要注意，date 或 origdate 等日期域只在 bib 文件中有效。随着 bib 文件的处理，所有的日期都被解析并分解为各个日期成分。样式使用日期和

[24] https://www.loc.gov/standards/datetime/pre-submission.html

[25] http://www.loc.gov/standards/datetime/iso-tc154-wg5_n0039_iso_wd_8601-2_2016-02-16.pdf

日期规格	日期格式（例）	
	短格式/12 小时格式	长格式/24 小时格式
1850	1850	1850
1997/	1997–	1997–
/1997	–1997	–1997
1997/unknown	1997–	1997–
1997/*	1997–	1997–
unknown/1997	–1997	–1997
*/1997	–1997	–1997
1997/open	1997–	1997–
open/1997	–1997	–1997
1967-02	02/1967	February 1967
2009-01-31	31/01/2009	31st January 2009
1988/1992	1988–1992	1988–1992
2002-01/2002-02	01/2002–02/2002	January 2002–February 2002
1995-03-30/1995-04-05	30/03/1995–05/04/1995	30th March 1995–5th April 1995
2004-04-05T14:34:00	05/04/2004 2:34 PM	5th April 2004 14:34:00

表 3: 日期规范

时间成分主要通过通过 § 4.2.4.3 节讨论的特殊域。更多信息请参考该节和 147 页的表 10。

EDTF 日期是天文学日期，其中第“0”年是存在的。当输出公元前年代（BCE/BC era）的日期时（见下面的 `dateera` 选项），请注意它们通常要早一年，因为公元前年代没有第 0 年（第 0 年就是公元前 1 年）。该转换是自动完成的，见表 5 中的例子。

如同默认日期域，日期域的名称必须以字符串“date”结尾。当需要在数据模型中添加新的日期域时（见 § 4.5.4 节）必须记住这一点。`biblatex` 在读入日期模型后会检查所有的日期域，如果发现有日期域不遵循这一命名约定就会报错并退出。

EDTF 通过负日期格式支持公元前（before common era, BCE/BC）日期，此外还支持“近似”（circa）和不确定的日期。这样的日期格式设置可以检测的内部标记，进而可以插入合适的本地化标记（例如 `circa` 或 `beforecommonera`）。另外，不确定日期（EDTF 5.2.2）会根据⟨*datetype*⟩`dateunspecified` 域指定的未定数据间隔尺寸，自动展开成合适的日期范围。参考文献样式可以使用该信息构造合适的日期格式，但标准样式不会使用。37 页的表 4 列出了允许的 EDTF 未定日期格式、范围展开和 ⟨*datetype*⟩`dateunspecified` 域的值（§ 4.2.4.1 节）。

表 5 展示了使用恰当测试和格式化的日期格式。参考 § 4.6.2 节的日期元信息测试以及 § 4.9.2.21 节的本地化字符串。关于测试和本地化字符串使用的完整例子请参考 `96-dates.tex` 示例文件。

在标准样式或没有定制内部宏 `\mkdaterange*` 的定制样式中，‘circa’、不确定信息和纪元信息的输出由 § 3.1.2.1 节中的宏包选项 `datecirca`、`dateuncertain`、`dateera` 和 `dateeraauto` 控制。38 页中的表 5 列出了使用全部这些选项的例子。

日期规范	扩展范围	元信息
199u	1990/1999	yearindecade
19uu	1900/1999	yearincentury
1999-uu	1999-01/1999-12	monthinyear
1999-01-uu	1999-01-01/1999-01-31	dayinmonth
1999-uu-uu	1999-01-01/1999-12-31	dayinyear

表 4: EDTF 5.2.2 未定日期解析

2.3.9 月份和期刊的期号

`month` 是整数域。文献样式按照要求将月份转化成不同语言的字符串。出于向后兼容考虑，你也可以在 `month` 域中使用以下的三字母缩写形式：`jan`、`feb`、`mar`、`apr`、`may`、`jun`、`jul`、`aug`、`sep`、`oct`、`nov`、`dec`。请注意，这些缩写词是 BibTeX 字符串，不能带有任何括号或引号。即，不要用 `month={jan}` 或 `month="jan"`，而直接使用 `month=jan`。不可以像 `month={8/9}` 这样指定月份，而可以使用 `date` 域来表示日期范围。季刊通常由“Spring”或“Summer”等标识指定，这些标识应在 `issue` 域中给出。在 `@article` 条目中，`issue` 域的位置与 `month` 域类似，并且会覆盖后者。

2.3.10 标记页码

当在条目的 `pages` 域中或标注命令的 ⟨*postnote*⟩ 选项中指明页码或页码范围时，可以很方便地使用 `biblatex` 自动添加“p.”或“pp.”等前缀，而这也确实是本宏包的默认方式。然而，一些作品或许使用不同的页码标记格式，或者不是按页码而是按诗节或者行号引用。此时 `pagination` 和 `bookpagination` 就可以起作用了。例如，考虑如下条目：

```
@InBook{key,
  title          = {...},
  pagination     = {verse},
  booktitle      = {...},
  bookpagination = {page},
  pages          = {53--65},
  ...
```

`bookpagination` 域会影响文献列表中 `pages` 和 `pagetotal` 的格式。由于 `page` 是默认的，因此在上面这个例子中该域可以省略。此时页码范围的格式是“pp. 53–65”。假设引用该作品时习惯使用韵节号而不是页码数，这可以通过 `pagination` 域反映出来，进而影响任何标注命令的 ⟨*postnote*⟩ 参数的格式。引用命令如果是 `\cite[17]{key}`，注记 (postnote) 的格式就会是“v. 17”。若设置 `pagination` 域为 `section`，那么就会产生“§ 17”。用法的进一步说明，请参考 § 3.13.3 节。

`pagination` 和 `bookpagination` 都是关键字域。如果关键字是已定义的，本宏包会尝试使用这些域的值作为本地化关键字。在 `bib` 文件中要使用关键字名的单数形式，复数形式是自动形成的。预定义的关键字有 `page`、`column`、`line`、`verse`、

日期规范	格式化日期（例）	
	输出格式	输出格式注记
0000	1 BC	dateera=christian 打印本地化字符串 beforechrist
-0876	877 BCE	dateera=secular 打印本地化字符串 beforecommonera
-0877/-0866	878 BC–867 BC	使用 \ifdateera 测试和本地化字符串 beforechrist
0768	0768 CE	dateeraauto 设置为 1000，并使用本地化字符串 commonera
-0343-02	344-02 BCE	
0343-02-03	343-02-03 CE	以及 dateeraauto=400
0343-02-03	343-02-02 CE	以及 dateeraauto=400 和 julian
1723~	circa 1723	使用 \ifdatecirca 测试
1723?	1723?	使用 \ifdateuncertain 测试
1723?~	circa 1723?	使用 \ifdateuncertain 和 \ifdatecirca 测试
2004-22	2004	另外，season 设置为本地化字符串 ‘summer’
2004-24	2004	另外，season 设置为本地化字符串 ‘winter’

表 5: 增强的日期规范

section 和 paragraph，其中 page 是默认值。在使用 pagination 和 bookpagination 时，字符串“none”有特殊意义，它将取消相应条目页码标记的前缀。如果某一条目页码标记格式使用的本地化关键字未定义，你可以直接添加它们。参考 § 3.9 节中的 \NewBibliographyString 和 \DefineBibliographyStrings 命令。你需要定义两个本地化字符串来对应附加的页码标记格式：单数形式（本地化关键字对应于 pagination 域的值）和复数形式（本地化关键字必须是单数形式加上字母“s”）。具体例子可以参考 § 4.9.2 节的预定义关键字。

2.4 提示与警告

本节提供了一些关于本宏包的数据接口的额外提示，另外也讨论了一些常见问题。

2.4.1 交叉引用

biber 的一大特色是高度可定义的交叉引用机制以及灵活的数据继承规则。因此不再需要从父条目复制一些域或者向子条目添加一些空白域，而可以用很自然的方式指定条目：

```
@Book{book,
  author        = {Author},
  title         = {Booktitle},
  subtitle      = {Booksubtitle},
  publisher     = {Publisher},
  location      = {Location},
  date          = {1995},
}
@InBook{inbook,
  crossref      = {book},
  title         = {Title},
```

```
  pages          = {5--25},
}
```

父条目的 title 和 subtitle 会分别复制给子条目的 booktitle 和 booksubtitle。父条目的 author 会成为子条目的 bookauthor，并且由于子条目没有提供 author 域，它也会复制给子条目的 author 域。继承数据之后子条目会大致如下：

```
author          = {Author},
bookauthor      = {Author},
title           = {Title},
booktitle       = {Booktitle},
booksubtitle    = {Booksubtitle},
publisher       = {Publisher},
location        = {Location},
date            = {1995},
pages           = {5--25},
```

默认的映射规则列表请参考附录 B。请注意，所有这一切都是可以定制的。关于如何配置 biber 的交叉引用机制请参考 § 4.5.11 以及 § 2.2.3 节。

2.4.1.1 xref 域 除了 crossref 域之外，biblatex 也支持一种基于 xref 域的简化交叉引用机制。如果你想在两个关联条目间创建父/子关系，但又希望保持它们之间的数据独立性，那么该域会很有用。xref 域与 crossref 的不同之处在于子条目不会从父条目继承任何数据。如果一个父条目被一定数量的子条目引用，那么它将被 biblatex 自动添加到参考文献表中。关联子条目数量的阈值由 § 3.1.2.1 节的 minxrefs 宏包选项所控制。另可参考 § 2.2.3 节。

2.4.2 排序和编码问题

biber 能处理 Ascii 编码，Latin 1 等 8 比特编码，以及UTF-8 。它支持真正的 Unicode，并能以一种鲁棒的方式对bib 数据进行即时重新编码。对于排序，biber 使用 perl 实现的 Unicode 排序算法（Unicode Collation Algorithm，UCA），该算法见 Unicode 技术标准 #10[26]。另外也支持基于 Unicode 通用区域数据库（Common Locale Data Repository，CLDR）的排序裁剪 [27] 。

支持 Unicode 不仅意味着能处理 UTF-8 输入。Unicode 是一个复杂的标准，不仅涵盖了它最著名的部分——Unicode 字符编码和 UTF-8 等传输编码。它同样对字符串排序等方面做了标准化，用于语言相关的排序。例如，使用 Unicode 排序算法（UCA），biber 可以处理字符“ß”，而不需要任何人工干预。要做本地化排序，你只需要指定本地化设置：

```
\usepackage[sortlocale=de]{biblatex}
```

[26] http://unicode.org/reports/tr10/

[27] http://cldr.unicode.org/

如果通过 babel 或者 polyglossia 等宏包将德语设置为主文档语言，设置方式为：

```
\usepackage[sortlocale=auto]{biblatex}
```

这时，biblatex 会将 babel/polyglossia 主文档语言作为本地化语言传递进来，biber 会将其映射为合适的默认本地化语言。biber 不会尝试从操作环境中获取本地化信息，因为这会使得文本处理依赖于文档以外的东西，而这有悖于 TEX 要求可移植性的精神。由于 babel/polyglossia 实际上提供了文档的相关环境，这种处理方式也是合理的。

请注意，这对于 Latin 9 等 8 比特编码也是有效的，也就是说，你可以利用基于 Unicode 的排序，即使你没有使用 UTF-8 输入。关于如何正确指定输入和数据编码，请参考 § 2.4.2.1。

2.4.2.1 指定编码 当在 bib 中使用非 Ascii 编码时，理解biblatex 能做什么以及哪些还需要进行人工干预很重要。本宏包能满足 LATEX 需要，只要 bibencoding 宏包选项设置正确，就能确保从 bbl 文件导入的数据能被正确解析。所有这一切都会自动处理，除bibencoding 选项设置为某些特定值的情况外，不需要额外的步骤。以下给出了一些典型的使用场景以及文件导言区中的相关行：

- tex 和 bib 文件都使用 Ascii 编码，使用 pdfTEX 或传统的 TEX 编译：

  ```
  \usepackage{biblatex}
  ```

- tex 使用 Latin 1 编码（ISO-8859-1），bib 文件使用 Ascii 编码，用 pdfTEX 或传统的 TEX 编译：

  ```
  \usepackage[latin1]{inputenc}
  \usepackage[bibencoding=ascii]{biblatex}
  ```

- tex 和 bib 文件中都使用 Latin 9 编码（ISO-8859-15），用 pdfTEX 或传统的 TEX 编译：

  ```
  \usepackage[latin9]{inputenc}
  \usepackage[bibencoding=auto]{biblatex}
  ```

  由于 bibencoding=auto 是默认设置，因此该选项可以省略。如下设置具有相同效果：

  ```
  \usepackage[latin9]{inputenc}
  \usepackage{biblatex}
  ```

- tex 文件中使用 UTF-8 编码，bib 文件中使用 Latin 1（ISO-8859-1）编码，用 pdfTEX 或传统的 TEX 编译：

```
\usepackage[utf8]{inputenc}
\usepackage[bibencoding=latin1]{biblatex}
```

在原生 UTF-8 模式下使用 XETEX 或 LuaTEX 编译的相同场景：

```
\usepackage[bibencoding=latin1]{biblatex}
```

biber 可以处理 Ascii 记法、Latin 1 等 8 比特编码，以及UTF-8 。它也能在运行中对bib数据进行实时重新编码（取代biblatex 在宏层面有限的重新编码功能）。如果你能正确指定 bib 文件的编码，这将会在需要时自动处理。除了以上讨论的场景外，biber 还能够处理以下情况：

- 直接的 UTF-8 工作流，即，在 tex 和 bib 文件中都使用 UTF-8 编码并使用 pdfTEX 或传统的 TEX 编译：

```
\usepackage[utf8]{inputenc}
\usepackage[bibencoding=auto]{biblatex}
```

由于 bibencoding=auto 是默认设置，因此该选项可以省略：

```
\usepackage[utf8]{inputenc}
\usepackage{biblatex}
```

在原生 UTF-8 模式下使用 XETEX 或 LuaTEX 编译的相同场景：

```
\usepackage{biblatex}
```

- 甚至可以在 tex 文件中使用 8 比特编码，而在 bib 文件中使用 UTF-8 编码，只要 bib 文件中的所有字符都能被所选择的 8 比特编码覆盖：

```
\usepackage[latin1]{inputenc}
\usepackage[bibencoding=utf8]{biblatex}
```

当对 UTF-8 编码使用传统的 TEX 或 pdfTEX 时，可能需要一些变通处理，因为 inputenc 的 utf8 模块并不能覆盖所有的 Unicode。粗略地讲，它只覆盖了西欧字符的 Unicode 范围。当载入带有 utf8 选项的 inputenc 宏包时，biblatex 通常会指示 biber 将 bib 数据重新编码为 UTF-8 。如果 bib 文件中的字符超出了 inputenc 支持的 Unicode 范围，这可能会导致 inputenc 报错。

- 如果你受到这个问题的影响，尝试设置 safeinputenc 选项：

```
\usepackage[utf8]{inputenc}
\usepackage[safeinputenc]{biblatex}
```

如果该选项被启用，biblatex 会忽略 inputenc 的 utf8 选项而使用 Ascii。biber 随后会尝试将 bib 数据转化为 Ascii 记法。例如，它将 Ş 转化为 \k{S}。该选项类似于设置 texencoding=ascii 但是只影响这一特定场合（带有 UTF-8 的 inputenc/inputenx 宏包）。这一变通处理利用了一个事实：Unicode 和 UTF-8 传输编码都向后兼容 Ascii。

如果 bib 文件中的数据主要是 Ascii，仅含有很少部分会导致问题的字符串（例如一些作者的名字），那么这一变通方法式可以接受的。然而，需要记住的是，它不会奇迹般地让传统的 TEX 或 pdfTEX 支持 Unicode。当使用重音命令（例如用 \d{S} 取代 Ṣ）时，如果遇到零星一些inputenc 不支持字符，但仍需要用 TEX 处理，这种方式会有所帮助。然而，如果你需要完全的 Unicode 支持，请使用 XETEX 或 LuaTEX 。

inputenc 不能处理某一特定 UTF-8 字符时典型的错误是：

```
! Package inputenc Error: Unicode char <char> (U+<codepoint>)
(inputenc)                not set up for use with LaTeX.
```

但也可能不那么明显，如：

```
! Argument of \UTFviii@three@octets has an extra }.
```

3 用户使用手册

本部分介绍了 biblatex 宏包的用户接口。该使用指南涵盖了要使用 biblatex 自带标准样式所需的所有信息。所以无论怎样都应该首先阅读该使用指南。如果你进一步想编写你自己的引用和文献样式，请继续阅读随后的作者指南。

3.1 宏包选项

所有的宏包选项都以 ⟨*key*⟩=⟨*value*⟩ 记法给出。对于所有的布尔型键值，true 是可以省略的。例如，给出不带值的 sortcites 等价于 sortcites=true。

3.1.1 载入时选项

以下的选项必须在 biblatex 载入时使用，即作为 \usepackage 的可选项。

backend=bibtex, bibtex8, biber　　default: biber

指定数据库后端程序。支持以下后端：

`biber` biber，biblatex 的默认后端程序，支持 Ascii、8 比特编码、UTF-8、实时重新编码、本地化定制排序，以及许多其它特性。本地化定制排序，大小写敏感排序和大小写优先排序分别由选项 `sortlocale`、`sortcase` 和 `sortupper` 控制。

`bibtex` 遗留下来的 BibTeX。传统的 BibTeX 只支持 Ascii 编码，并且排序总是大小写不敏感的。

`bibtex8` bibtex8 是 BibTeX 的 8 比特实现，支持 Ascii 和 Latin 1 等 8 比特编码。

使用 BibTeX 作为后端详见 § 3.14 节。

`style=`⟨*file*⟩ default: `numeric`

加载文献样式文件 ⟨*file*⟩`.bbx` 和引用样式文件 ⟨*file*⟩`.cbx`。关于标准样式的概览请参考 § 3.3 节。

`bibstyle=`⟨*file*⟩ default: `numeric`

加载文献样式文件 ⟨*file*⟩`.bbx`。关于标准文献样式的概览见 § 3.3.2 节。

`citestyle=`⟨*file*⟩ default: `numeric`

加载引用样式文件 ⟨*file*⟩`.cbx`。关于标准引用样式的概览见 § 3.3.1 节。

`natbib=true, false` default: `false`

加载兼容性模块，以提供 `natbib` 宏包引用命令的同名替代命令。详见 § 3.8.9 节。

`mcite=true, false` default: `false`

加载引用模块，以提供 `mcite`/类 `mciteplus` 的引用命令。详见 § 3.8.10 节。

3.1.2 导言区选项

3.1.2.1 一般选项 下列选项既可以作为 `\usepackage` 的可选项，也可以在配置文件和导言区中使用。右边列出的是本宏包的默认值。请注意，文献和引用样式可以在载入时修改默认设置，详见 § 3.3 节。

`sorting=nty, nyt, nyvt, anyt, anyvt, ynt, ydnt, none, debug, `⟨*name*⟩ default: `nty`

参考文献的排序方式。除非另加说明，文献条目总是按升序排列。有以下预设可选值。

`nty` 按照姓名、标题、年份排序。

`nyt` 按照姓名、年份、标题排序。

`nyvt` 按照姓名、年份、卷数、标题排序。

`anyt` 按照标签的字母顺序、姓名、年份、标题排序。

`anyvt` 按照标签的字母顺序、姓名、年份、卷数、标题排序。

`ynt` 按照年份、姓名、标题排序。

ydnt	按照年份（降序）、姓名、标题排序。
none	不进行排序。所有的条目按照引用顺序处理。
debug	按照条目键值排列。该选项只用于程序调试。
⟨*name*⟩	使用随 \DeclareSortingScheme（§ 4.5.6）定义的 ⟨*name*⟩。

只有与打印相应标签的文献样式相结合，使用“字母顺序”排序格式才有意义。请注意，一些文献样式会将宏包选项从宏包的默认值（nty）初始化为另外一个值。详见 § 3.3.2 节。关于以上排序选项的深度解读以及排序过程中涉及的域，请参考 § 3.5 节。关于如何调整预定义格式或者定义新格式也可参考 § 4.5.6。

sortcase=true, false　　default: true

文献和缩略语列表的排序是否是大小写敏感的。

sortupper=true, false　　default: true

对应于 biber 的 --sortupper 命令行选项。当该选项激活时，文献会按照“大写先于小写”的顺序排列。关闭该选项则使用“小写先于大写”的顺序。

sortlocale=auto, ⟨*locale*⟩

该选项设置全局的排序区域语言。如果排序格式没有使用 \printbibliography 命令的 ⟨*locale*⟩ 选项，那么就会继承该区域语言设置。如果该选项值设置为 auto，那么需要使用 babel/polyglossia 宏包并设置主文档语言标识符，否则将设置为 en_US。biber 会将 babel/polyglossia 语言标识符映射为有意义的本地化标识符（参考 biber 文档）。因此，你可以指定常规的本地化标识符，例如 de_DE_phonebook、es_ES；另外如果对 biber 的语言映射满意的话，还可以指定支持的 babel/polyglossia 语言标识符。

related=true, false　　default: true

是否使用相关条目中的信息。参考 § 3.4 节。

sortcites=true, false　　default: false

当多个条目关键字传给引用命令时，是否对引用进行排序。如果该选项激活，就会根据当前的文献排序格式（见 § 3.7.10 节）对引用进行排序。该特征对所有的引用样式都有效。

maxnames=⟨*integer*⟩　　default: 3

影响所有名称列表（author、editor 等）的阈值。如果某个列表长度超过了该阈值，即，它包括的姓名数量超过 ⟨*integer*⟩，那么就会根据 minnames 选项的设置进行自动截断。maxnames 是设置 maxbibnames 和 maxcitenames 的主选项。

minnames=⟨*integer*⟩　　default: 1

影响所有名称列表（author、editor 等）的限制值。如果某个列表包含的姓名数量超过 ⟨*maxnames*⟩，那么就会根据 minnames 选项的设置自动截断。⟨*minnames*⟩ 的

值必须小于或等于 ⟨*maxnames*⟩。minnames 是设置 minbibnames 和 mincitenames 的主选项。

maxbibnames=⟨*integer*⟩ default: ⟨*maxnames*⟩

类似于 maxnames 但只影响参考文献。

minbibnames=⟨*integer*⟩ default: ⟨*minnames*⟩

类似于 minnames 但只影响参考文献。

maxcitenames=⟨*integer*⟩ default: ⟨*maxnames*⟩

类似于 maxnames 但只影响正文中的引用。

mincitenames=⟨*integer*⟩ default: ⟨*minnames*⟩

类似于 minnames 但只影响正文中的引用。

maxitems=⟨*integer*⟩ default: 3

类似于 maxnames 但影响所有的文本列表（publisher、location等）。

minitems=⟨*integer*⟩ default: 1

类似于 minnames 但影响所有的文本列表（publisher、location等）。

autocite=plain, inline, footnote, superscript, ...

该选项控制 § 3.8.4 节中讨论的 \autocite 命令的行为。plain 选项使得 \autocite 相当于 \cite；inline 选项使得它相当于 \parencite；footnote 选项使得它相当于 \footcite；superscript 选项使得它相当于 \supercite。选项 plain、inline 和 footnote 总是可行的，而 superscript 只能用于本宏包所带的数值引用样式中。引用样式也可以定义其它选项值。该选项的默认设置取决于所选的引用样式，参考 § 3.3.1 节。

autopunct=true, false default: true

该选项控制引用命令是否先扫描标点。详见 § 3.8 节和 § 4.7.5 节中的命令 \DeclareAutoPunctuation。

language=autobib, autocite, auto, ⟨*language*⟩ default: autobib

该选项控制多语种支持功能。当其设置为 autobib、autocite 或 auto 时，biblatex 会尝试从 babel/polyglossia 宏包中获取文档的主语言（如果babel/polyglossia 宏包不可用则设置为后备的英语）。也可以手动选择文档的语言，此时选择的语言会覆盖条目的 langid 域，并且仍需要使用 autolang 选项选择语言切换环境以确定如何处理手动选择语言的切换方式。关于所支持的语言和相应的标识符请参考表 2。autobib 使用条目中的 langid 域和 autolang 选项确定的语言环境为条目切换语言。autocite 使用条目中的 langid 域和 autolang 选项确定的语言环境为引用切换语言。而 auto 是同时设置 autobib 和 autocite 的缩略语。

clearlang=true, false default: true

如果激活该选项，biblatex 会自动清除那些与文档的 babel/polyglossia 语言（或者由 language 选项显式确定的语言）相匹配的所有条目的 language 域，以便略去冗余的语言设定。该特性所需的语言映射由 § 4.9.1 节的 \DeclareRedundantLanguages 命令提供。

autolang=none, hyphen, other, other*, langname default: none

如果载入了 babel/polyglossia 宏包并且文献条目包含 langid 域（见 § 2.2.3 节），那么该选项可以控制使用哪种 babel 语言环境[28]。请注意，当载入 babel/polyglossia 宏包时 biblatex 会自动适应主文档的语言。在多语言文档中，只要涉及到引用和文献的默认语言，本宏包也会连续地调整以适应当前语言。切换语言的效果取决于该选项选择的语言环境，可用的选择有：

none 关闭该特性，也就是不使用任何语言环境。

hyphen 将条目装入 hyphenrules 环境中。如果可用的话，会为条目的 hyphenation 域指定的语言导入断词模式。

other 将条目装入 otherlanguage 环境中。这将导入特定语言的断词模式，激活 babel/polyglossia 和 biblatex 为相应语言提供的所有额外定义，并翻译“editor”和“volume”等键项。这些额外定义包括日期格式、序数和其它类似项目的本地化。

other* 将条目装入 otherlanguage* 环境中。请注意，此时 biblatex 将 otherlanguage* 环境视为 otherlanguage 环境但其它宏包不会。

langname 只用于 polyglossia 宏包。将条目装入 ⟨*languagename*⟩ 环境中。该选项值对 polyglossia 用户的好处是注意了 langidopts 域，这样可以为每一条目增加语言选项（类似于选择语言变种）。当使用 babel 时，该选项值与 other 选项值相同。

block=none, space, par, nbpar, ragged default: none

该选项控制块（block）之间的额外空白，即文献条目的更大的分段。可用的选项值有：

none 不添加任何东西。

space 在块之间添加额外的水平间距，类似于标准 LaTeX 文档类的默认行为。

par 每一块都另起一段，类似于标准 LaTeX 文档类的 openbib 选项。

nbpar 类似于 par 选项但是不允许在块的边界和条目内部分页。

ragged 插入一个小的负惩罚项以鼓励在块的边界处断行并设置左对齐。

也可以直接重新定义 \newblockpunct 命令实现不同的效果，见 § 3.10.1 节。更多信息见 § 4.7.1 节。

[28] polyglossia 宏包也可以理解 babel 语言环境，因此本选项可以同时控制 babel/polyglossia 语言环境。

notetype=foot+end, footonly, endonly　　default: foot+end

该选项控制 § 4.10.4 节中 \mkbibfootnote、\mkbibendnote 和类似封装的行为。可用的选项值有：

foot+end　同时支持脚注和尾注，即，\mkbibfootnote 会生成脚注而 \mkbibendnote 会生成尾注。

footonly　强制脚注，即，\mkbibendnote 也生成脚注。

endonly　强制尾注，即，\mkbibfootnote 也生成尾注。

hyperref=true, false, auto　　default: auto

是否将引用和向后引用转化为可点击的超链接。这一特性需要 hyperref 宏包。它同样需要引用样式的支持。本宏包所带的所有标准样式都支持超链接。hyperref=auto 会自动检测 hyperref 宏包是否载入。

backref=true, false　　default: false

是否在文献中打印出反向引用。反向引用是一组用于标明引用相应文献的页码数。如果在文档中有 refsection 环境，反向引用在每个引用章节中是局部的。严格地讲，该选项只控制 biblatex 是否收集所需引用的数据。该特性也需要文献样式的支持。本宏包所带的所有标准样式都支持该特性。

backrefstyle=none, three, two, two+, three+, all+　　default: three

该选项控制反向引用中的连续页码的格式。可用样式有：

none　不启用该特性，即，不压缩页码列表。

three　将任意连续三页或更多页缩写为页码范围，例如，“1, 2, 11, 12, 13, 21, 22, 23, 24”会压缩成“1, 2, 11–13, 21–24”。

two　将任意连续两页或更多页缩写成页码范围，例如上面的例子会变成“1–2, 11–13, 21–24”。

two+　概念类似于 two，但是连续两页的打印格式为开始页和本地化字符串 sequens，例如上面的例子会变成“1 sq., 11–13, 21–24”。

three+　概念类似于 two+，但是连续三页的打印格式为开始页可本地化字符串 sequentes，例如上面的例子会变成“1 sq., 11 sqq., 21–24”。

all+　概念类似于 three+，但是任意连续页码将打印成末端不封闭的形式，例如上面的例子会变成“1 sq., 11 sqq., 21 sqq.”。

所有这些样式都同时支持阿拉伯和罗马数字。为了避免可能的歧义，不同数字集在生成数字范围时不会混在一起，例如，“iii, iv, v, 6, 7, 8”将缩写为“iii–v, 6–8”。

backrefsetstyle=setonly, memonly, setormem, setandmem, memandset, setplusmem　　default: setonly

该选项控制怎样处理指向 @set 条目及其成员的反向引用。可用选项有：

`setonly` 所有的反向引用都添加到 `@set` 条目中。而其成员的 `pageref` 列表为空。

`memonly` 条目集成员的引用添加到各自成员中。`@set` 条目的引用添加到所有成员中。`@set` 条目的 `pageref` 列表为空。

`setormem` `@set` 条目的引用添加到 `@set` 条目中。其成员的引用添加到各自成员中。

`setandmem` `@set` 条目的引用添加到 `@set` 条目中。其成员的引用添加到各自成员和 `@set` 条目中。

`memandset` `@set` 条目的引用添加到 `@set` 条目和所有成员中。其成员的引用添加到各自成员中。

`setplusmem` `@set` 条目的引用添加到 `@set` 条目和所有成员中。其成员的引用添加到各自成员和 `@set` 条目中。

`indexing=true, false, cite, bib` default: `false`

该选项控制文献和引用中的索引。更准确地说，它影响 § 4.6.2 节的 `\ifciteindex` 和 `\ifbibindex` 命令。该选项可以在全局、基于每一类型或每一条目设置。可用的选择有：

`true` 全局激活索引。

`false` 全局不激活索引。

`cite` 只在引用中激活索引。

`bib` 只在文献中激活索引。

该特性需要引用样式的支持。本宏包所带的所有标准样式都支持引用和文献条目中的索引。请注意，为了得到索引仍然需要用 `\makeindex` 命令全局激活索引模式。

`loadfiles=true, false` default: `false`

该选项控制是否载入通过 `\printfile` 命令所需的外部文件。另参考 § 3.12.8 和 § 4.4.1 节中的 `\printfile` 命令。请注意，出于性能原因该特性默认不激活。

`refsection=none, part, chapter, section, subsection` default: `none`

该选项自动在文档分段处（例如一章或一节）开始一个新的参考文献分节。这等价于 `\newrefsection` 命令，参考 § 3.7.4 节。可用的文档分段如下：

`none` 不启用该特性。

`part` 在每个 `\part` 命令处开始一个参考文献分节。

`chapter` 在每个 `\chapter` 命令处开始一个参考文献分节。

`section` 在每个 `\section` 命令处开始一个参考文献分节。

`subsection` 在每个 `\subsection` 命令处开始一个参考文献分节。

这些命令对应带星号的版本不会开始新的参考文献分节。

refsegment=none, part, chapter, section, subsection default: none

类似于 refsection 选项，但是开始一个新的参考文献片段。这等价于 \newrefsegment 命令，详见 § 3.7.5 节。当两个选项都使用时，请注意该选项只能应用到比 refsection 选项应用的更低水平的文档分段，同时，嵌套的参考文献片段相对于所附的参考文献分节来讲是局部的。

citereset=none, part, chapter, section, subsection default: none

该选项在文档分段处（例如一章或一节）自动执行 § 3.8.8 节介绍的 \citereset 命令。可用的文档分段有：

none 不启用该特性。

part 在每个 \part 命令后执行重置。

chapter 在每个 \chapter 命令后执行重置。

section 在每个 \section 命令后执行重置。

subsection 在每个 \subsection 命令后执行重置。

这些命令对应带星号的版本不会引起重置。

abbreviate=true, false default: true

是否在引用和文献中使用较长或者缩写的字符串。该选项会影响本地化模块。如果启用该选项，“editor”等键值项会缩写，反之则会全部写出。

date=year, short, long, terse, comp, ymd, edtf default: comp

该选项控制日期规范的基本格式。有以下选择：

year 只使用年份，例如：

2010

2010–2012

short 使用短格式和详细的日期范围，例如：

01/01/2010

21/01/2010–30/01/2010

01/21/2010–01/30/2010

long 使用长格式和详细的日期范围，例如：

1st January 2010

21st January 2010–30th January 2010

January 21, 2010–January 30, 2010

terse 使用短格式和紧凑的日期范围，例如：

21–30/01/2010

01/21–01/30/2010

`comp` 使用长格式和紧凑的日期范围，例如：

21st–30th January 2010

January 21–30, 2010

`edtf` 使用严格的扩展日期/时间格式[29] (`yyyy-mm-dd`)，例如：

2010-01-01

2010-01-21/2010-01-30

`ymd` 不同于严格的 EDTF 格式，使用可以被其它选项修改的年-月-日格式，例如：

2010-1-1

2010-1-21/2010-1-30

注意，`edtf` 格式会强制开启 `dateera=astronomical`, `datezeros=true`, `timezeros=true`, `seconds=true`, ⟨*datetype*⟩`time=24h` 以及 `julian=false` 等键值。

从以上例子可以看出，实际的日期格式是与语言相关的。请注意，在所有长格式中月份名称与 `abbreviate` 宏包选项相对应。所有短格式中月和日的首位零可以另外由 `datezeros` 宏包选项控制。所有短格式中时分秒的首位零可以另外由 `timezeros` 宏包选项控制。如果要输出时刻，那么秒和时区的打印分别由 `seconds` 和 `timezones` 选项控制。

`julian` 和 `gregorianstart` 选项可以用于控制何时输出儒略历日期。

`labeldate`=`year`, `short`, `long`, `terse`, `comp`, `ymd`, `edtf` default: `year`

类似于 `date` 选项但是控制由 `\DeclareLabeldate` 选择的日期域的格式。

⟨*datetype*⟩`date`=`year`, `short`, `long`, `terse`, `comp`, `ymd`, `edtf` default: `comp`

类似于 `date` 选项但是控制数据模型中 ⟨*datetype*⟩`date` 域的格式。

`alldates`=`year`, `short`, `long`, `terse`, `comp`, `edtf`

将数据模型中所有日期的选项设置为相同值。默认数据模型中的日期域为 `date`、`origdate`、`eventdate` 和 `urldate`。

`julian`=`true`, `false` default: `false`

该选项控制是否将 `gregorianstart` 选项指定日期之前的日期自动转换为儒略历。改变的日期在 `\ifdatejulian` 和 `\if`⟨*datetype*⟩`datejulian` 测试下会返回“true”（见 § 4.6.2 节）。请谨记，只包含年份的日期不会转换为儒略历，例如“1565”，这是因为没有月日的日期在儒略历表示下会引起混乱[30] 例如，在“1565”的例子中，在公历（格里高利历）“1565 年 1 月 10 日”之后的日期是儒略历“1565”年，而之前的日期是儒略历“1564”年。

[29] https://www.loc.gov/standards/datetime/pre-submission.html

[30] 缺失时刻的日期也有可能出现这一问题，不过对于文献作品问题不大。

gregorianstart=⟨*YYYY-MM-DD*⟩

该选项控制在哪一日期之前的日期可以转换为儒略历。选项值有严格的字符串格式：4 位年份、2 位月份和日期，并且由短划线分隔（或者具有“Dash”属性的任何有效 Unicode 字符）。默认值是“1582-10-15”，即标准公历（格里高利历）的颁布日期。只有 julian 设置为“true”时本选项才起作用。

datezeros=true, false default: true

该选项控制 short 和 terse 日期成分当没有覆盖特定格式时打印是否首位补零。

timezeros=true, false default: true

该选项控制没有覆盖特定格式时，时刻成分打印是否首位补零。

timezones=true, false default: false

该选项控制打印时刻时是否输出时区。

seconds=true, false default: false

该选项控制打印时刻时是否输出秒。

dateabbrev=true, false default: true

该选项控制 long 和 comp 日期格式打印时带有完整还是缩写的月份名。该选项类似于一般的 abbreviate 选项但是只针对日期格式。

datecirca=true, false default: false

该选项控制关于日期是否输出“circa”信息。如果设置为 true，那么日期将由 \datecircaprint 宏的展开来引导，见 § 3.10.1 节。

dateuncertain=true, false default: false

该选项控制是否输出日期的不确定信息。如果设置为 true，那么日期将由 \dateuncertainprint 宏的展开来引导，并由 \enddateuncertainprint 宏结束，见 § 3.10.1 节。

dateera=astronomical, secular, christian default: astronomical

该选项控制如何打印日期纪元信息。选项值“astronomical”使用 \dateeraprintpre 命令在起止日期之前打印纪元信息。而选项值“secular”和“christian”使用 \dateeraprint 命令在起止日期之后打印纪元信息。缺省情况下，使用“astronomical”效果是在公元前日期之前使用负号，而使用“‘secular’”或“‘christian’”的效果是在公元前日期之后加上“BCE”或“BC”等相关的本地化字符串。见 § 3.10.1 节的有关评论以及 § 4.9.2.21 节的本地化字符串。

dateeraauto=⟨*integer*⟩ default: 0

该选项设置天文学年份，使得在之前的年份自动添加纪元本地化字符串。只有当 dateera 设置为“secular”或者“christian”时本选项才起作用。

`time`=12h, 24h, 24hcomp　　default: 24h

该选项控制时刻规范的基本格式。可用的选择有：

`24h`	24 小时格式，例如：
	14:03:23
	14:3:23
	14:03:23+05:00
	14:03:23Z
	14:21:23–14:23:45
	14:23:23–14:23:45
`24hcomp`	带有缩写范围的 24 小时格式，例如：
	14:21–23 （小时相同）
	14:23:23–45 （小时和分钟相同）
`12h`	带有本地化上下午标识符的 12 小时格式，例如：
	2:34 PM
	2:34 PM–3:50 PM

从以上例子可以看出，实际的时刻格式是与语言相关的。请注意，如果与指定的区域不同，那么上下午（AM/PM）字符串对应于 `abbreviate` 宏包选项。24 小时格式首位补零的话可以单独用 `timezeros` 宏包选项控制。此外与语言相关并可以单独定制的还有时刻成分之间的分隔符 `\bibtimesep`、`\bibtzminsep`，以及时刻与时区之间的分隔符 `\bibtimezonesep`，见 § 3.10.3 节。还有一些全局的宏包选项可以控制是否打印秒和时区（分别是 `seconds` 和 `timezones`，见 § 3.1.2.1 节）。如果有时区的话，要么是字符 'Z'，要么是表示正负偏移量的数值。默认样式不打印时刻信息。定制样式可以使用 `\print`⟨*datetype*⟩`time` 命令打印时刻，见 § 4.4.1 节。

`labeltime`=12h, 24h, 24hcomp　　default: 24h

类似于 `time` 选项但是控制由 `\DeclareLabeldate` 选择的域中得到的时刻部分格式。

⟨*datetype*⟩`time`=12h, 24h, 24hcomp　　default: 24h

类似于 `time` 选项但是控制数据模型中 ⟨*datetype*⟩`date` 域中得到的时刻部分格式。

`alltimes`=12h, 24h, 24hcomp

为数据模型中所有的时刻设置相同的 `labeltime` 和 ⟨*datetype*⟩`time` 值。默认数据模型中支持时刻部分的日期域有 `date`、`origdate`、`eventdate` 和 `urldate`。

`dateusetime`=true, false　　default: false

确定在日期域的日期成分后是否打印时刻成分。日期和时刻成分的分隔符是 `\bibdatetimesep`，见 § 3.10.3 节。如果使用紧凑的日期格式（见 § 3.1.2.1 节），那么该选项则不起作用，否则会引起混乱。

labeldateusetime=true, false default: false

类似于 dateusetime 选项，但是控制是否打印 \DeclareLabeldate 选择的域中的时刻成分。

⟨*datetype*⟩dateusetime=true, false default: false

类似于 dateusetime 选项，但是控制是否打印数据模型中 ⟨*datetype*⟩date 域的时刻成分。

alldatesusetime=true, false default: false

为数据模型中所有的 ⟨*datetype*⟩date 域设置 labeldateusetime 和 ⟨*datetype*⟩dateusetime 选项值。

defernumbers=true, false default: false

与标准 LaTeX 不同，本宏包生成的数字标签一般在文档正文的一开始就分配给引用列表全体。如果该选项被激活，数字标签（也就是 § 4.2.4 中讨论的 labelnumber 域）在参考文献第一次打印出某条目时就被分配。进一步解释见 § 3.13.5 节。该选项在后端将数据导出到 bbl 文件后仍然需要两次 LaTeX 运行（在由分页变化等要求的编译之外）。需要注意的一件重要的事是，如果你在文档中改变了该选项值（或者那些依赖于本选项的选项值，例如与 \printbibliography 宏相关的选项，§ 3.7.2 节），那么很可能需要删除当前的 aux 文件然后重新运行 LaTeX 来获得正确的数字编号，见 § 4.1 节。

punctfont=true, false default: false

该选项激活一个可选机制，用来处理不同字体打印的域（例如斜体的标题）之后的单位标点。详见 § 4.7.1 节中的 \setpunctfont。

arxiv=abs, ps, pdf, format default: abs

arXiv 链接的路径选择。如果启用超链接支持，该选项会控制 arXiv 的 eprint 域会指向该文件的哪个版本。以下是可用的选择：

abs	链接到摘要页面。
ps	链接到 PostScript 版本。
pdf	链接到 PDF 版本。
format	链接到格式选择页面。

关于 arXiv 和电子出版信息的支持详见 § 3.12.7 节。

texencoding=auto, ⟨*encoding*⟩ default: auto

指定 tex 文件的编码。该选项影响从后端转向 biblatex 的数据。该选项对应于 biber 的 --output-encoding 选项。可用的选择有：

auto　尝试自动识别输入的编码。如果有 inputenc/inputenx/luainputenc 等宏包，biblatex 会从宏包中获取主要编码。否则，当探测到 XeTeX 或 LuaTeX 引擎时设定为 UTF-8 编码，其余情况设为 Ascii。

⟨*encoding*⟩　显式指定编码为 ⟨*encoding*⟩。少数情况下自动检测失败，或你出于某种原因想强制为某个编码，那么此时可以使用该选项。

请注意如果 bibencoding = auto，那么设置 texencoding = ⟨*encoding*⟩ 也会影响 bibencoding 选项。

bibencoding=auto, ⟨*encoding*⟩　default: auto

指定 bib 文件的编码。该选项对应于 biber 程序的 --output-encoding 选项。可用选择有：

auto　当工作流是透明时，即，当 bib 文件的编码与 tex 文件的编码相同时使用该选项。

⟨*encoding*⟩　如果 bib 文件的编码与 tex 不同，你需要显式指定编码。

默认情况下，biblatex 假定 tex 和 bib 文件使用相同的编码（bibencoding=auto）。

safeinputenc=true, false　default: false

如果启动该选项，biblatex 会在载入 inputenc/inputenx 宏包并且输入代码是 UTF-8 时自动加入 texencoding=ascii，也就是说，它会忽略任何基于宏的 UTF-8 支持而只是用 Ascii。然后 biber 会尝试将 bib 文件中的任意非 Ascii 数据转化为 Ascii。例如，它会将 Ş 转化为 \d{S}。关于为什么需要启用该选项的原因，请参考 § 2.4.2.1 节。

bibwarn=true, false　default: true

默认情况下，biblatex 会报告后端产生的关于 bib 文件数据的警告，就像 LaTeX 警告一样。使用该选项会取消此警告。

mincrossrefs=⟨*integer*⟩　default: 2

当需要后端运行时设置交叉引用的最小数目为 ⟨*integer*⟩[31]。该选项也影响 xref 域的处理。详见 § 2.2.3 节以及 § 2.4.1 节对该域的描述。

minxrefs=⟨*integer*⟩　default: 2

类似于 mincrossrefs 但针对于 xref 域。

[31] 如果一个条目被 bib 文件中的其它条目引用的数目达到这个阈值，它就会载入到参考文献中，即使没有显式地被引用。这是 BibTeX 格式的标准特性，并不是 biblatex 特有的。更多信息见 § 2.2.3 节中关于 crossref 域的描述。

3.1.2.2 特定样式选项

下列选项由标准样式（而不是宏包内核）提供。技术上讲，它们和 § 3.1.2.1 中的选项一样也是导言区选项。

`isbn=true, false` default: `true`

该选项控制是否打印 `isbn/issn/isrn` 等域。

`url=true, false` default: `true`

该选项控制是否打印 `url` 域并获取日期。该选项只影响 `url` 信息是可选的那些条目类型。而 `@online` 条目的 `url` 域总是打印出来的。

`doi=true, false` default: `true`

该选项控制是否打印 `doi` 域。

`eprint=true, false` default: `true`

该选项控制是否打印 `eprint` 信息。

3.1.2.3 内部选项

下列导言区选项的默认设置由文献和引用样式控制。除了 `pagetracker` 和 ⟨*name*⟩`inits` 是你可能想调整的之外，一般没有必要显式地设置。

`pagetracker=true, false, page, spread` default: `false`

该选项控制 § 4.6.2 节的 `\ifsamepage` 和 `\iffirstonpage` 测试所需的页码跟踪器。可用选择有：

`true`	在自动模式下激活。该选项在 LaTeX 处于双面模式时类似于 `spread`，否则类似于 `page`。
`false`	不激活跟踪器。
`page`	在页面模式下激活。此时跟踪器基于每一页。
`spread`	在跨页模式下激活。此时跟踪器基于每一页面（双页）。

注意所有的浮动体都禁用该跟踪器，见 § 4.11.5 节。

`citecounter=true, false, context` default: `false`

该选项控制 § 4.6.2 节的 `citecounter` 所需的引用计数器。可用的选择有：

`true`	在全局模式下启用引用计数器。
`false`	禁用引用计数器。
`context`	在内容相关模式下启用引用计数器。此时，脚注和正文中的引用将独立计数。

`citetracker=true, false, context, strict, constrict` default: `false`

该选项控制 § 4.6.2 节的 `\ifciteseen` 和 `\ifentryseen` 测试所需的引用跟踪器。可用选择有：

`true` 在全局模式下启用跟踪器。

`false` 禁用跟踪器。

`context` 在内容相关模式下启用跟踪器。此时，脚注和正文中的引用将独立跟踪。

`strict` 在严格模式下启用跟踪器。此时，跟踪器只考虑独立的引用，即，引用命令只传递单个的条目键。

`constrict` 该模式是 `context` 和 `strict` 的结合。

注意所有的浮动体都禁用该跟踪器，见 § 4.11.5 节。

`ibidtracker=true, false, context, strict, constrict` default: `false`

该选项控制 § 4.6.2 节的 `\ifciteibid` 测试所需的“如前所述”（ibidem）跟踪器。可用的选择有：

`true` 在全局模式下启用跟踪器。

`false` 禁用跟踪器。

`context` 在内容相关模式下启用跟踪器。此时，脚注和正文中的引用将独立跟踪。

`strict` 在严格模式下启用跟踪器。此时将不考虑那些模糊不清的参考文献。如果当前引用（包含“ibidem”）或者之前引用（“ibidem”的指向）包含多个文献时，就认为是模糊不清的。[32]

`constrict` 该模式是 `context` 和 `strict` 的结合。它也保持对脚注数量的跟踪，不过检测脚注中含义不清的文献时比 `strict` 更加严格。除了 `strict` 选项的条件外，只有当前引用和之前引用都在同一个或者连续的脚注中时，脚注中的文献才认为是含义清晰的。

注意所有的浮动体都禁用该跟踪器，见 § 4.11.5 节。

`opcittracker=true, false, context, strict, constrict` default: `false`

该选项控制 § 4.6.2 节的 `\ifopcit` 测试所需的“opcit”跟踪器。该特性类似于“ibidem”跟踪器，不同之处在于它跟踪的是基于某一作者或编辑的引用，即，如果引用项目与之前项目的作者或编辑相同，那么 `\ifopcit` 的结果为 `true`。可用选择有：

`true` 在全局模式下启用该跟踪器。

`false` 禁用该跟踪器。

`context` 在内容相关模式下启用该跟踪器。此时，脚注和正文中的引用会独立跟踪。

[32] 例如，假设一开始的引用是“Jones, *Title*; Williams, *Title*”，而接下来的是“ibidem”。从技术角度来看，“ibidem”指向“Williams”是相对清晰的，因为这是之前引用命令处理的最后文献。然而对于用户而言，“ibidem”也可以同时指向这两个标题，因此含义不清。严格模式就避免这种含义不清的文献。

`strict` 在严格模式下启用该跟踪器。此时将不跟踪那些含义不清的引用。详见 `ibidtracker=strict` 选项。

`constrict` 该模式是 `context` 和 `strict` 的结合。详见 `ibidtracker=constrict` 选项的解释。

注意所有的浮动体都禁用该跟踪器，见 § 4.11.5 节。

`loccittracker=true, false, context, strict, constrict` default: `false`

该选项控制 § 4.6.2 节的 `\ifloccit` 测试所需的“loccit”跟踪器。该特性类似于“opcit”跟踪器，不同之处在于它也会检查 ⟨*postnote*⟩ 选项是否匹配，即，如果引用项目与之前引用指向的页数相同，那么 `\ifloccit` 的结果为 `true`。可用选择有：

`true` 在全局模式下启用该跟踪器。

`false` 禁用该跟踪器。

`context` 在内容相关模式下启用该跟踪器。此时，脚注和正文中的引用会独立跟踪。

`strict` 在严格模式下启用该跟踪器。此时将不跟踪那些含义不清的引用。详见 `ibidtracker=strict` 选项。此外，该模式也会检查 ⟨*postnote*⟩ 选项是否是数值型的（基于 § 4.6.2 节的 `\ifnumerals` 命令）。

`constrict` 该模式是 `context` 和 `strict` 的结合。详见 `ibidtracker=constrict` 选项的解释。此外，该模式也会检查 ⟨*postnote*⟩ 选项是否是数值型的（基于 § 4.6.2 节的 `\ifnumerals` 命令）。

注意所有的浮动体都禁用该跟踪器，见 § 4.11.5 节。

`idemtracker=true, false, context, strict, constrict` default: `false`

该选项控制 § 4.6.2 节的 `\ifciteidem` 测试所需的“idem”跟踪器。可用选择有：

`true` 在全局模式下启用该跟踪器。

`false` 禁用该跟踪器。

`context` 在内容相关模式下启用该跟踪器。此时，脚注和正文中的引用会独立跟踪。

`strict` 该选项是 `true` 的别名。提供该选项只是为了和其它跟踪器保持一致。由于“idem”不会像“ibidem”或“op. cit.”那样引起歧义，因此不必使用 `strict` 跟踪模式。

`constrict` 该模式类似于 `context`，只有一个额外条件：对于脚注中的引用，只有当前引用和之前引用位于同一个或相连的脚注中才会认为是含义明确的。

注意所有的浮动体都禁用该跟踪器，见 § 4.11.5 节。

`parentracker=true, false` default: `true`

该选项控制括号跟踪器，用于对嵌套的圆括号和方括号的跟踪。所得信息用于 § 3.8.5 节的 `\parentext` 和 `\brackettext` 命令、§ 4.10.4 节的 `\mkbibparens` 和

\mkbibbrackets 命令，以及同样来自于 § 4.10.4 节的 \bibopenparen, \bibcloseparen, \bibopenbracket, \bibclosebracket 等命令。

maxparens=⟨*integer*⟩ default: 3

圆括号和方括号嵌套的最大层级。如果嵌套深度大于该值则会报错。

⟨*namepart*⟩inits=true, false default: false

启用该选项时所有的 ⟨*namepart*⟩ 姓名部分都会用首字母表示。该选项会影响 § 4.6.2 节的 \if⟨*namepart*⟩inits 测试。在数据模型中有效的姓名部分由 \DeclareDatamodelConstant 命令定义，见 § 4.2.3 节。

terseinits=true, false default: false

该选项控制 biblatex 生成的首字母格式。当启用时，首字母缩写将采用没有点号和空格的紧凑形式。例如 Donald Ervin Knuth 的首字母缩写默认是“D. E.”，而在此选项启用时会是“DE”。该选项会影响 § 4.6.2 中的 \ifterseinits 测试。该选项会重新定义一些控制首字母格式的宏。详见 § 3.13.4 节。

labelalpha=true, false default: false

是否提供特殊的域 labelalpha 和 extraalpha，详见 § 4.2.4 节。该选项可以基于每一条目设置。另见 maxalphanames 和 minalphanames 选项。表 7 总结了各种 extra* 歧义消除的计数器以及它们所跟踪的选项。

maxalphanames=⟨*integer*⟩ default: 3

类似于 maxnames 但用于定制 labelalpha 域的格式。

minalphanames=⟨*integer*⟩ default: 1

类似于 minnames 但用于定制 labelalpha 域的格式。

labelnumber=true, false default: false

是否提供特殊域 labelnumber，详见 § 4.2.4 节。该选项可基于每一类型而设置。

labeltitle=true, false default: false

是否提供特殊域extratitle，详见 § 4.2.4 节。请注意，特殊域 labeltitle 总是提供的，而该选项控制是否利用 labeltitle 生成 extratitle 信息。该选项也可基于每一类型而设置。表 7 总结了各种 extra* 消除歧义的计数器以及所跟踪的选项。

labeltitleyear=true, false default: false

是否提供特殊域 extratitleyear，详见 § 4.2.4 节。请注意，特殊域 labeltitle 总是提供的，而该选项控制是否利用 labeltitle 生成 extratitleyear 信息。该选项也可基于每一类型而设置。表 7 总结了各种 extra* 消除歧义的计数器以及所跟踪的选项。

选项	测试	跟踪的域
singletitle	\ifsingletitle	labelname
uniquetitle	\ifuniquetitle	labeltitle
uniquebaretitle	\ifuniquebaretitle	labeltitle （当 labelname 为空时）
uniquework	\ifuniquework	labelname+labeltitle

表 6: 惟一性选项

labeldateparts=true, false default: false

是否提供特殊域 labelyear, labelmonth, labelday, labelendyear, labelendmonth, labelendday, labelhour, labelendhour, labelminute, labelendminute, labelsecond, labelendsecond, labelseason, labelendseason, labeltimezone, labelendtimeone 以及 extrayear，详见 § 4.2.4 节。该选项也可基于每一类型而设置。表 7 总结了各种 extra* 消除歧义的计数器以及所跟踪的选项。

singletitle=true, false default: false

是否提供 \ifsingletitle 测试所需的数据，详见 § 4.6.2 节。关于该测试中数据的控制因素详见 表 6。该选项也可基于每一类型而设置。

uniquetitle=true, false default: false

是否提供 \ifuniquetitle 测试所需的数据，详见 § 4.6.2 节。关于该测试中数据的控制因素详见 表 6。该选项也可基于每一类型而设置。

uniquebaretitle=true, false default: false

是否提供 \ifuniquebaretitle 测试所需的数据，详见 § 4.6.2 节。关于该测试中数据的控制因素详见 表 6。该选项也可基于每一类型而设置。

uniquework=true, false default: false

是否提供 \ifuniquework 测试所需的数据，详见 § 4.6.2 节。关于该测试中数据的控制因素详见 表 6。该选项也可基于每一类型而设置。

uniqueprimaryauthor=true, false default: false

是否提供 \ifuniqueprimaryauthor 测试所需的数据，详见 § 4.6.2 节。

uniquename=true, false, init, full, allinit, allfull, mininit, minfull default: false

是否更新 uniquename 计数器，详见 § 4.6.2 节。该特性会消除 labelname 列表中各个姓名的歧义。该选项也可基于每一类型而设置。可用的选择有：

- true　full 的别称。
- false　禁用该特性。
- init　只使用首字母消除歧义。
- full　根据要求使用首字母或全名消除歧义。

选项	激活域	激活计数器	计数器跟踪
labelalpha	labelalpha	extraalpha	label
labeldateparts	labelyear	extrayear	labelname+
	labelmonth		labelyear
	labelday		
	labelendyear		
	labelendmonth		
	labelendday		
	labelhour		
	labelminute		
	labelsecond		
	labelendhour		
	labelendminute		
	labelendsecond		
	labelseason		
	labelendseason		
	labeltimezone		
	labelendtimezone		
labeltitle	—	extratitle	labelname+labeltitle
labeltitleyear	—	extratitleyear	labeltitle+labelyear

表 7: 歧义消除计数器

allinit　类似于 init，但是会对 labelname 列表中所有姓名消除歧义，即便超出了 maxnames/minnames/uniquelist 选项。

allfull　类似于 full，但是会对 labelname 列表中所有姓名消除歧义，即便超出了 maxnames/minnames/uniquelist 选项。

mininit　init 的变种，只对列表中有同一姓（last name）的姓名消除歧义。

minfull　full 的变种，只对列表中有同一姓（last name）的姓名消除歧义。

请注意，uniquename 选项也会影响 uniquelist 选项、\ifsingletitle 测试，以及 extrayear 域。更多细节和实例见 § 4.11.4 节。

uniquelist=true, false, minyear　default: false

是否更新 uniquelist 计数器，详见 § 4.6.2 节。如果 labelname 列表在 maxnames/minnames 截断后含义不清，那么该特性会消除 labelname 列表中的歧义。本质上，该选项会覆盖基于每一域的 maxnames/minnames 设置。该选项也可基于每一类型而设置。可用的选择有：

true　消除 labelname 列表的歧义。

false　禁用该特性。

minyear　只有当被截断的列表与带有相同 labelyear 的另一列表相同时才会消除 labelname 列表的歧义。该操作模式适用于作者–年份样式中，如果要求 labeldateparts=true 的场合。

请注意，uniquelist 选项也会影响 \ifsingletitle 测试和 extrayear 域。更多细节和实例见 § 4.11.4 节。

3.1.3 条目选项

条目选项是控制参考文献数据条目处理的包选项。可以在以下不同尺度上设置。

3.1.3.1 导言区/类型/条目选项 下列选项可以基于类型或条目在 options 域中设置。此外还可以在 \usepackage 的可选项以及配置文件和导言区中使用。这可用于全局改变默认样式。

useauthor=true, false default: true

是否在标签和排序中使用 author 域。如果一个条目包含 author 域但出于某种原因通常不以作者引用，该选项是很有用的。设置 useauthor=false 并不意味着 author 被完全忽略了，而只是在标签和排序中不使用。该条目将按照 editor 或 title 域的字母顺序排列。在标准样式中，此时 author 会在标题后打印。参考 § 3.5 节。该选项可以基于每一类型设置。

useeditor=true, false default: true

在标签和排序中是否用 editor 域来代替缺失的 author 域。如果一个条目包含 editor 域但是通常不以作者引用，那么该选项是很有用的。设置 useeditor=false 并不意味着 editor 被完全忽略了，而只是在标签和排序中不用 editor 来代替缺失的 author。该条目会以 title 的字母顺序排列。在标准样式中，此时 editor 会在标题之后打印。参考 § 3.5 节。该选项可以基于每一类型或条目而设置。

usetranslator=true, false default: false

在标签和排序中是否用 translator 代替缺失的 author/editor。设置 usetranslator = true 并不意味着 translator 会覆盖 author/editor，而只是当 author/editor 缺失或者 useauthor 和 useeditor 选项设置为 false 时作为后备。也就是说，如果要按译者而不是作者引用一本书，你需要设置如下选项：

```
@Book{...,
  options    = {useauthor=false,usetranslator=true},
  author     = {...},
  translator = {...},
  ...
```

该选项也可以基于每一类型或条目而设置。在标准样式中，translator 默认在标题之后打印，参考 § 3.5 节。

use⟨*name*⟩=true, false default: true

按照 useauthor、useeditor 和 usetranslator，数据模型中定义的所有的姓名列表都有一个选项用于控制自动定义的排序和标签行为。此时会自动创建全局、基于类型和基于条目选项而调用的 use⟨*name*⟩。

useprefix=true, false default: false

是否在以下几种情况下考虑默认数据模型中的姓名前缀部分（von、van、of,、da、de、della 等）：

- 在引用中打印姓
- 排序
- 生成标签的某些类型
- 生成姓名惟一性信息。
- 参考文献的格式方面

如果激活该选项，biblatex 总是在姓氏前面加上该前缀。例如 Ludwig van Beethoven 将引作“van Beethoven”并按照“Van Beethoven, Ludwig”排序，而如果未激活该选项（默认情况），将引作“Beethoven”并按照“Beethoven, Ludwig van”排序。该选项也可基于每一类型设置。当使用 biblatexml 数据源以及 biber 支持的 BibTeX 扩展名格式时，还可以基于每一姓名列表或姓名而设置。

indexing=true, false, cite, bib

indexing 选项也可以基于每一类型或条目设置。详见 § 3.1.2.1 节。

3.1.3.2 类型/条目选项 下列选项只能基于类型或条目在 options 域中设置，而不能全局设置。

skipbib=true, false default: false

如果激活该选项，该条目在参考文献中将被排除在外但仍然可以引用。该选项也可以基于每一类型设置。

skipbiblist=true, false default: false

如果激活该选项，该选项将在文献列表中被排除在外，但仍然包含在参考文献中并可以被 shorthand 引用。该选项也可以基于每一类型设置。

skiplab=true, false default: false

如果激活该选项，biblatex 不会给该条目分配标签。正常操作不需要该选项。要小心使用。当激活时，biblatex 不能保证相应条目有唯一的引用，而且那些需要标签的引用样式可能不能为该条目创建有效的引用。该选项也可以基于每一类型设置。

dataonly=true, false default: false

设置该选项等价于设置 uniquename = false、uniquelist = false、skipbib、skipbiblist 和 skiplab。正常操作不需要该选项。要小心使用。该选项也可以基于每一类型设置。

3.1.3.3 条目选项

下列选项只能基于条目在 options 域中而不能全局或基于类型设置。

labelnamefield=⟨*fieldname*⟩

指定搜索 labelname 时首先考虑的域。本质上，只在该条目中该域会放到 \DeclareLabelname 创建的搜索列表之前。

labeltitlefield=⟨*fieldname*⟩

指定搜索 labeltitle 时首先考虑的域。本质上，只在该条目中该域会放到 \DeclareLabeltitle 创建的搜索列表之前。

3.1.4 遗留选项

下面的遗留选项可以全局地在 \documentclass 的可选项中使用，也可以局部地在 \usepackage 的可选项中使用：

openbib 该选项用于向后兼容标准 LaTeX 文档类。openbib 类似于 block=par。 Deprecated

3.2 全局定制

除了编写新的引用和文献样式，本宏包中还有很多定制样式的方法。定制通常在导言区中进行，但也可以在配置文件中进行以便长期使用。配置文件也可以用于将宏包选项从默认值初始化为不同的值。

3.2.1 配置文件

当可用时，本宏包会导入配置文件 biblatex.cfg。该文件会在宏包的末尾，紧跟在引用和文献样式导入之后立即被读取。

3.2.2 设置宏包选项

§ 3.1.1 节中的实时载入宏包选项必须在 \usepackage 的可选项中给出。§ 3.1.2 节中的宏包选项同样可以在导言区中给出。以下命令用于执行选项：

\ExecuteBibliographyOptions[⟨*entrytype, ...*⟩]{⟨*key=value, ...*⟩}

该命令也可以在配置文件中使用，以修改宏包选项的默认设置。某些选项还可以基于每一条目而设置。此时，可选的 ⟨*entrytype*⟩ 选项用来确定条目类型。⟨*entrytype*⟩ 选项可以是逗号分隔的值列表。

3.3 标准样式

本节简要地描述了本宏包所带的所有文献和引用样式。如果你想自己写样式文件，请参考 § 4 节。

3.3.1 标注样式

本宏包所带的引用样式实现了一些常见的引用格式。所有的标准样式都支持 shorthand 域，并且支持超链接和索引。

numeric 该样式实现与 LaTeX 的标准文献工具类似的数值式引用格式。它应当与某种能在参考文献中打印出相应标签的数值式文献样式一起使用，并用于文内引用。该样式在宏包载入时设置如下的宏包选项：autocite=inline、labelnumber=true。该样式还额外提供了一个导言区选项 subentry，这会影响条目集的处理。如果该选项被禁止，指向条目集中某一成员的引用会指向整个的条目集。如果该选项被激活，该样式会支持类似于“[5c]”这样指向条目集中的子条目的引用（这个例子是第三个条目）。详见样式例子。

numeric-comp numeric 样式紧凑形式的变种，会将两个以上的连续数字打印成一个区间。该样式类似于 cite 宏包和数值模式中的 natbib 宏包的 sort&compress 选项。例如，“[8, 3, 1, 7, 2]”会变成“[1–3, 7, 8]”。它用于文中引用。该样式在宏包载入时设置如下的宏包选项：autocite=inline、sortcites=true、labelnumber=true。它也提供了 subentry 选项。

numeric-verb numeric 样式详细形式的变种。不同之处在于对一组引用的处理，并且只当不同的条目键值传递给单个引用命令时才会显示差异。例如，“[2, 5, 6]”会变成“[2]; [5]; [6]”。它用于文中引用。该样式在宏包载入时设置如下的宏包选项：autocite=inline、labelnumber=true。它也提供了 subentry 选项。

alphabetic 该样式实现的字母顺序引用格式类似于传统 BibTeX 的 alpha.bst 样式。字母标签某种程度上类似于紧凑的作者–年份样式，但是使用的方式类似于数字引用格式。例如，“Jones 1995”会是“[Jon95]”；而“Jones and Williams 1986”会缩写为“[JW86]”。该样式应当与一种字母顺序文献样式一起使用，从而可以在参考文献中打印出相应的标签。它用于文内引用。该样式在载入时设置如下的宏包选项：autocite=inline、labelalpha=true。

alphabetic-verb alphabetic 样式详细格式的变种。不同之处在于对一组引用的处理，并且只当不同的条目键值传递给单个引用命令时才会显示差异。例如“[Doe92; Doe95; Jon98]”会变成“[Doe92]; [Doe95]; [Jon98]”。它用于文内引用。该样式在载入时设置如下的宏包选项：autocite=inline、labelalpha=true。

authoryear 该样式实现了作者–年份引用格式。如果参考文献中包含多个由同一作者同一年份发表的作品，那么年份后会附加一个字母用以区分。例如该样式会打印出“Doe 1995a; Doe 1995b; Jones 1998”这样的引用。该样式应当与一种作者–年份文献样式一起使用，从而可以在参考文献中打印出相应的标签。它起初用于文内引用，但也可以用在脚注中。该样式在载入时设置如下的宏包选项：autocite=inline、labeldate=true、uniquename=full、uniquelist=true。

authoryear-comp authoryear 样式紧凑格式的变种。如果传递给单个引用命令的一列文献作者相同，那么该作者只会打印一次。如果它们年份也相同，那么年份也只会打印一次。例如，“Doe 1995b; Doe 1992; Jones 1998; Doe 1995a”在该样式下会变成“Doe 1992,

1995a,b; Jones 1998”。它起初用于文中引用，但也可以用在脚注中。该样式在载入时设置如下的宏包选项：autocite=inline、sortcites=true、labeldate=true、uniquename=full、uniquelist=true。

authoryear-ibid　authoryear 样式的变种，会用缩略语 *ibidem* 替代重复的引用，除非该引用在当前页或跨页是第一次出现，或者 *ibidem* 在宏包选项 ibidtracker=constrict 的意义下表意不清。该样式在载入时设置如下的宏包选项：autocite=inline、labeldate=true、uniquename=full、uniquelist=true、ibidtracker=constrict、pagetracker=true。该样式还额外提供了一个导言区选项 ibidpage。详见样式例子。

authoryear-icomp　一个结合了 authoryear-comp 和 authoryear-ibid 的样式。该样式在载入时设置如下的宏包选项：autocite=inline、labeldate=true、uniquename=full、uniquelist=true、ibidtracker=constrict、pagetracker=true、sortcites=true。该样式还额外提供了一个导言区选项 ibidpage。详见样式例子。

authortitle　该样式实现了一个简单的作者–标题引用格式。如果可用的话，它会使用 shorttitle 域。它用于脚注中给出的引用。该样式在载入时设置如下的宏包选项：autocite=footnote、uniquename=full、uniquelist=true。

authortitle-comp　authortitle 样式紧凑格式的变种。如果传递给单个引用命令的一列文献作者相同，那么该作者只会打印一次。例如，“Doe, *First title*; Doe, *Second title*”在此样式下会变成“Doe, *First title*, *Second title*”。它用于脚注中给出的引用。该样式在载入时设置如下的宏包选项：autocite=footnote、sortcites=true、uniquename=full、uniquelist=true。

authortitle-ibid　authortitle 样式的变种，会用缩略语 *ibidem* 替代重复的引用，除非该引用在当前页或跨页是第一次出现，或者 *ibidem* 在宏包选项 ibidtracker=constrict 的意义下表意不清。它用于脚注中给出的引用。该样式在载入时设置如下的宏包选项：autocite=footnote、uniquename=full、uniquelist=true、ibidtracker=constrict、pagetracker=true。该样式还额外提供了一个导言区选项 ibidpage。详见样式例子。

authortitle-icomp　结合了 authortitle-comp 和 authortitle-ibid 特性的样式。该样式在载入时设置如下的宏包选项：autocite = footnote、uniquename = full、uniquelist = true、ibidtracker=constrict、pagetracker=true、sortcites=true。该样式还额外提供了一个导言区选项 ibidpage。详见样式例子。

authortitle-terse　authortitle 样式简明格式的变种，如果文献中包含多个相应作者/编辑的作品，那么只会打印出标题。如果可用的话，该样式会使用 shorttitle 域。它在文中引用和脚注中引用都适用。该样式在载入时设置如下的宏包选项：autocite=inline、singletitle=true、uniquename=full、uniquelist=true。

authortitle-tcomp　结合了 authortitle-comp 和 authortitle-terse 特性的样式。如果可用的话，该样式会使用 shorttitle 域。它在文内引用和脚注中引用都适用。该样式在载入时设置如下的宏包选项：autocite=inline、sortcites=true、singletitle=true、uniquename=full、uniquelist=true。

authortitle-ticomp 结合了 authortitle-icomp 和 authortitle-terse 特性的样式。换句话说就是带有 *ibidem* 特性的 authortitle-tcomp 样式变种。它在文中引用和脚注中引用都适用。该样式在载入时设置如下的宏包选项：autocite=inline、ibidtracker=constrict、pagetracker=true、sortcites=true、singletitle=true、uniquename=full、uniquelist=true。该样式还额外提供了一个导言区选项 ibidpage。详见样式例子。

verbose 详细的引用样式，在第一次引用某条目时会打印出类似于参考文献那样的长引用格式，并且在之后打印出短格式。如果可用的话，shorttitle 域会用在所有的短格式中。如果 shorthand 域有定义，该 shorthand 会在第一次引用时被引入并在之后作为短格式被使用。由于在第一次引用时提供了所有的文献数据，因此该样式的使用不需要参考文献和 shorthand 列表。它用于脚注中给出的引用。该样式在载入时设置如下的宏包选项：autocite=footnote、citetracker=context。该样式还额外提供了一个导言区选项 citepages。详见样式例子。

verbose-ibid verbose 样式的变种，会用缩略语 *ibidem* 替代重复的引用，除非该引用在当前页或跨页是第一次出现，或者 *ibidem* 在宏包选项 ibidtracker=strict 的意义下表意不清。它用于脚注中给出的引用。该样式在载入时设置如下的宏包选项：autocite=footnote、citetracker=context、ibidtracker=constrict、pagetracker=true。该样式还额外提供了导言区选项 ibidpage 和 citepages。详见样式例子。

verbose-note 该样式与 verbose 样式类似，会在第一次引用某条目时打印出类似于参考文献那样的长格式，并且在之后打印出短格式。与 verbose 样式不同的是，短格式会指向带有长格式的脚注。如果文献包含了多个同一作者/编辑的作品，该短格式会带有标题。如果可用的话，所有的短格式会使用 shorttitle 域。如果 shorthand 域有定义，它会被 verbose 样式处理。由于在第一次引用时提供了所有的文献数据，因此该样式的使用不需要参考文献和 shorthand 列表。该样式仅仅用于脚注中给出的引用。该样式在载入时设置如下的宏包选项：autocite=footnote、citetracker=context、singletitle=true。该样式还额外提供了导言区选项 pageref 和 citepages。详见样式例子。

verbose-inote verbose"=note 样式的变种，会用缩略语 *ibidem* 替代重复的引用，除非该引用在当前页或跨页是第一次出现，或者 *ibidem* 在宏包选项 ibidtracker=strict 的意义下表意不清。该样式仅仅用于脚注中给出的引用。该样式在载入时设置如下的宏包选项：autocite=footnote、citetracker=context、ibidtracker=constrict、singletitle=true、pagetracker=true。该样式还额外提供了导言区选项 ibidpage、pageref 和 citepages。详见样式例子。

verbose-trad1 该样式实现了传统的引用格式。与 verbose 样式类似，它会在第一次引用某条目时打印出类似于参考文献那样的长格式，并且在之后打印出短格式。此外，它在重复的引用中使用学术性缩略语 *ibidem*、*idem*、*op. cit.* 和 *loc. cit.* 来代替重复的作者、标题、页码数。如果 shorthand 域有定义，那么会在第一次引用时被引入并在之后作为短格式被使用。由于在第一次引用时提供了所有的文献数据，因此该样式的使用不需要参考文献和 shorthand 列表。它用于脚注中给出的引用。该样式在载入时设置如下的宏包选项：autocite=footnote、citetracker=

context、ibidtracker=constrict、idemtracker=constrict、opcittracker=context、loccittracker=context。该样式还额外提供了导言区选项 ibidpage、strict 和 citepages。详见样式例子。

verbose-trad2 另外一种传统引用格式。它同样类似于 verbose 样式但是在重复的引用中使用 *ibidem* 和 *idem* 等学术性缩略语。与 verbose-trad1 样式不同的是，*op. cit.* 缩略语的逻辑有所不同，并且不使用 *loc. cit.*。事实上它更类似于 verbose-ibid 和 verbose-inote 而不是 verbose-trad1。该样式在载入时设置如下的宏包选项：autocite=footnote、citetracker=context、ibidtracker=constrict、idemtracker=constrict。该样式还额外提供了导言区选项 ibidpage、strict 和 citepages。详见样式例子。

verbose-trad3 仍然是一种传统的引用格式。它类似于 verbose-trad2 样式，但是使用缩略语 *ibidem* 和 *op. cit.* 的方式稍有不同。该样式在载入时设置如下的宏包选项：autocite=footnote、citetracker=context、ibidtracker=constrict、loccittracker=constrict。该样式也额外提供了导言区选项 strict 和 citepages。详见样式例子。

reading 一个同名的文献样式所带的引用样式，会载入 authortitle 样式。

下列样式具有特殊目的，不用于文档的最终版本。

draft 在引用中使用条目键的草稿样式。该样式在载入时设置如下的宏包选项：autocite=plain。

debug 该样式会打印出条目键而不是标签。它只用于调试，在载入时设置如下的宏包选项：autocite=plain。

3.3.2 参考文献样式

本宏包所带的所有文献样式对于每一文献条目都使用相同的基本格式。不同之处仅仅在于参考文献中打印的标签种类和文献列表的总体格式。每一个引用样式都有一个对应的文献样式。请注意，一些文献样式仅仅载入了另外更一般的样式，因此这里没有提及。例如，文献样式 authortitle-comp 会载入 authortitle 样式。

numeric 该样式打印出类似于 LaTeX 标准文献功能的数值标签。它应与数值引用样式结合使用。请注意，shorthand 域会覆盖默认标签。该样式在载入时设置如下的宏包选项：labelnumber=true。该样式还额外提供了一个导言区选项 subentry，这会影响条目集的处理。如果该选项被激活，条目集中的所有成员都会用一个字母标记，这可用于集成员的引用而不是整个条目集。详见样式例子。

alphabetic 该样式打印的字母顺序标签类似于传统 BibTeX 的 alpha.bst 样式。它应与字母顺序引用样式结合使用。请注意，shorthand 域会覆盖默认标签。该样式在载入时设置如下的宏包选项：labelalpha=true、sorting=anyt。

authoryear 该样式不同于其它样式之处在于，发表日期不是在条目的末尾而是在作者/编辑之后。它应与一个作者–年份引用样式结合使用。重复的作者和编辑名会用短横线代替，除非该条目是当前页或跨页的第一个。该样式额外提供了导言区选项 dashed

来控制该特征。此外还额外提供了导言区选项 mergedate。详见样式例子。该样式在载入时设置如下的宏包选项：labeldate=true、sorting=nyt、pagetracker=true、mergedate=true。

authortitle 该样式不会打印出任何标签。它应与一个作者–年份引用样式结合使用。重复的作者和编辑名会用短横线代替，除非该条目是当前页或跨页的第一个。该样式额外提供了一个导言区选项 dashed 来控制该特征。详见样式例子。该样式在载入时设置如下的宏包选项：pagetracker=true。

verbose 该样式类似于 authortitle 样式。该样式额外提供了一个导言区选项 dashed。详见样式例子。该样式在载入时设置如下的宏包选项：pagetracker=true。

reading 这一特殊的文献样式是为个人阅读列表、带有注释的文献和类似应用而设计的。它选择性地在参考文献中包含 annotation、abstract、library 和 file 等域。如果需要的话，它还会在参考文献中添加不同种类的短标题。该样式还额外提供了导言区选项 entryhead、entrykey、annotation、abstract、library 和 file 来控制是否在参考文献中打印相应的项目。详见样式例子。见 § 3.12.8 节。该样式在载入时设置如下的宏包选项：loadfiles=true、entryhead=true、entrykey=true、annotation=true、abstract=true、library=true、file=true。

下列样式具有特殊目的，不用于文档的最终版本。

draft 草稿样式会在参考文献中包含条目键。文献会按照条目键排序。该样式在载入时设置如下的宏包选项：sorting=debug。

debug 该样式会以表格形式打印出所有的文献数据。它只用于调试，在载入时设置如下的宏包选项：sorting=debug。

3.4 关联条目

几乎所有的文献样式都需要作者去确定条目之间的某些关系类型，例如“Reprint of”、“Reprinted in”等等。当然不可能通过提供数据域来覆盖所有的关系，为此，biblatex 通过使用条目域 related、relatedtype 和 relatedstring 提供了一种一般性的机制。被关联的条目不需要被引用，本身也不会出现在参考文献中（当然，除非它自己另外单独被引用），而是作为数据源被拷贝一份副本。relatedtype 域需要确定在相关联条目的信息前打印的本地化字符串，例如“Orig. Pub. as”。relatedstring 域可以用于覆盖那些通过 relatedtype 确定的字符串。一些例子如下：

```
@Book{key1,
  ...
  related     = {key2},
  relatedtype = {reprintof},
  ...
}
```

```
@Book{key2,
  ...
}
```

这里我们指定条目 key1 是条目 key2 的重印本。在 Book 条目的文献驱动里，当为条目 key1 而调用 \usebibmacro{related} 时：

- 如果本地化字符串“reprintof”有定义，那么将以 relatedstring:reprintof 格式打印出来。如果该格式指令没有定义，这些字符串将以 relatedstring:default 格式打印。
- 如果宏 related:reprintof 有定义，那么将用于确定条目 key2 包含的信息的格式，否则将使用宏 related:default。
- 如果 related:reprintof 格式有定义，那么将用于确定本地化字符串和数据的格式；如果该格式没有定义，将使用 related 格式。

也支持串联或者循环关系：

```
@Book{key1,
  ...
  related     = {key2},
  relatedtype = {reprintof},
  ...
}

@Book{key2,
  ...
  related     = {key3},
  relatedtype = {translationof},
  ...
}

@Book{key3,
  ...
  related     = {key2},
  relatedtype = {translatedas},
  ...
}
```

也可以实现同一条目的多重关系：

```
@MVBook{key1,
  ...
  related     = {key2,key3},
```

```
  relatedtype = {multivolume},
  ...
}

@Book{key2,
  ...
}

@Book{key3,
  ...
}
```

请注意，多重关联条目列表中的顺序是很重要的。多重关联条目的数据将按照该域中所列的顺序打印。关于该特征背后的机制请参考 § 4.5.1 节。可以通过 § 3.1.2.1 中的宏包选项 related 来关闭该特征。

可以使用 relatedoptions 域来设置关联条目数据克隆的选项。如果你需要覆盖默认设置的关于所有关联条目克隆的 dataonly 选项，那么该域是很有用的。例如，如果你想在文档中展示一些相关联克隆体的名称，同时想要它们不与其它条目的名称相混淆，但是正常情况下这不会发生的，因为相关联克隆体由基于每一条目的 dataonly 选项设置，这反过来又设置了 uniquename=false 和 uniquelist=false。此时，你只需设置 relatedoptions 为 skiplab, skipbib。

3.5 排序选项

本宏包支持多种文献排序格式。排序格式由 § 3.1.2.1 节中的 sorting 宏包选项确定。除了常规的数据域之外，还有一些特殊域也可用于优化文献排序。附录 C.1 和 C.2 大致概述了 biblatex 支持的字母顺序排序格式。而年代顺序排序格式则列在附录 C.3 中。以下依次是这些格式的一些解释。

在排序过程中首先要考虑的事项总是条目的 presort 域。如果该域没有定义，biblatex 会使用缺省值“mm”作为预排序字符串。其次考虑的是 sortkey 域。如果该域有定义，它将作为主要的排序关键字。此时除了 presort 域，将不考虑其它信息。如果 sortkey 域没有定义，排序将使用姓名信息。本宏包将依次尝试使用 sortname、author、editor 和 translator 等域。考虑哪些域也取决于 useauthor、useeditor 和 usetranslator 选项的设置。如果这三个选项都没有启用，那么 sortname 也将被忽略。请注意，所有的名称域都与 maxnames 和 minnames 有关。如果没有名称域是合适的，或者由于它们没有定义、或者由于 use⟨*name*⟩ 域都未启用，那么 biblatex 将采用 sorttitle 和 title 作为最后的备选。余下考虑的诸项依次是：sortyear 域（如果给出的话），否则考虑 year 域的前四个数字；sorttitle 域（如果给出的话），否则考虑 title 域；volume 域。请注意，附录 C.2 展示的排序格式包括了额外一项：labelalpha 域是“alphabetic”文献样式所使用的标签。严格地讲，用于排序的字符串是 labelalpha + extraalpha。附录 C.2 中的排序格式只可以与字母顺序样式联合使用。

附录 C.3 展示的年代排序格式同样使用域 presort 和 sortkey（如果有定义的话）。其次考虑的是 sortyear 或者 year 域，这当然取决于是否可用。ynt 格式将从该域中提取前四个数字。如果这两个域都没有定义，那么将使用后备值 9999。这意味着没有年份的条目都会移动到列表末尾。ydnt 格式从概念上也是类似的，不过是用降序排列年份。与 ynt 格式一样，后备值是 9999。余下考虑的项与上面讨论的字母排序格式类似。请注意，ydnt 排序格式只对日期按照降序排列。其它项仍和平常一样按照升序排列。

通常来说不需要使用 sortkey、sortname 或 sorttitle 等特殊域。biblatex 宏包通过使用条目常规域的数据就很容易得到所需的排列顺序。只有当你想手动修改文献排序或者所需的数据缺失时，你才需要使用这些特殊域。关于特殊域的可能用法请参考 § 2.2.3 节中的描述。

3.6 数据注解

理想状态下，文献数据文件中不应当有格式信息。然而，有时只有通过这种有争议的做法才能实现想要的结果。数据注解（data annotations）就是一种解决该问题的方法。通过允许用户在文献数据源中添加某种语义信息（而不是排版标记），使得文献样式可以在标记时使用该信息。例如，如果想要按照如下规则高亮某些作品中的姓名：学生作者在文献中用上标星号表示，而通讯作者用粗体表示；那么，可以尝试如下方法：

```
@MISC{Article1,
  AUTHOR = {Last1\textsuperscript{*}, First1 and \textbf{Last2}, \textbf{
    ↪ First2} and Last3, First3}
}
```

这一做法有一些问题。首先，它会打断 BibTeX 脆弱的姓名解析程序指令，可能根本不能编译。其次，数据与标记的混合是硬编码的：其它样式不易共享和使用该数据。当然，在样式或者文件中使用 biblatex 内部指令可能实现该格式，但是这一做法比较复杂而且不可靠，很多用户不愿意使用。

为了处理这些问题，biblatex 提供了一般性的数据注解功能，使得可以向数据域、数据域列表中的项目（例如姓名），以及某些项目的一部分（例如姓、名等姓名部分）中附加逗号分隔列表作为注解。此外还提供了一些宏来检查可以用于格式指令的注解。

数据注解有三种“尺度”，按照特性增加的顺序依次为：

- field—用于数据源条目中的顶层域
- item—用于数据源条目中列表域中的项目
- part—用于数据源条目中列表域中项目的一部分

BibTeX 和 biblatexml 数据源都支持数据注解。

在 biblatexml 数据源中添加数据注解是很容易的，因为可以通过简单的 XML 属性来指定。继续上面的例子，我们有：

```
<bltx:entries xmlns:bltx="http://biblatex-biber.sourceforge.net/
    ↪ biblatexml">
  <bltx:entry id="test" entrytype="misc">
    <bltx:names type="author">
      <bltx:name>
        <bltx:namepart type="given" initial="F">First1</bltx:namepart>
        <bltx:namepart type="family" initial="L" annotation="student">
    ↪ Last1</bltx:namepart>
      </bltx:name>
      <bltx:name annotation="corresponding">
        <bltx:namepart type="given" initial="F">First2</bltx:namepart>
        <bltx:namepart type="family" initial="L">Last2</bltx:namepart>
      </bltx:name>
      <bltx:name>
        <bltx:namepart type="given" initial="F">First3</bltx:namepart>
        <bltx:namepart type="family" initial="L">Last3</bltx:namepart>
      </bltx:name>
    </bltx:names>
  </bltx:entry>
</bltx:entries>
```

这里，向数据项目中添加注解的方式是很显然的。而在 BibTeX 数据源中，注解的格式就没有那么直观了：

```
@MISC{ann1,
  AUTHOR    = {Last1, First1 and Last2, First2 and Last3, First3},
  AUTHOR+an = {1:family=student;2=corresponding},
}
```

这里域姓名后缀 `+an` 可以由用户定义[33]，用于标记某个数据域为去掉后缀的域的注解。BibTeX 注解域的格式如下：

```
<annotationspecs> ::= <annotationspec> [ ";" <annotationspec> ]
<annotationspec>  ::= [ <itemcount> [ ":" <part> ] ] "=" <annotations>
<annotations>     ::= <annotation> [ "," <annotation> ]
<annotation>      ::= (string)
```

也就是说，多个特性之间由分号分隔。每一特性是一个等号后跟一个逗号分隔的注解关键字列表。为了为列表中某一项作注解，需要将该列表项的编号放在等号前面（列表从 1 开始编号）。如果需要为列表项的某一部分做注解，需要将该部分名放在编号之后，并且前接一个冒号。姓名部分的名称在数据模型中有定义，见 § 4.2.3 节。以下是一些例子：

[33] 见 `biber` 的 `--annotation-marker` 选项。

```
AUTHOR      = {Last1, First1 and Last2, First2 and Last3, First3},
AUTHOR+an   = {3:given=annotation1, annotation2},
TITLE       = {A title},
TITLE+an    = {=a title annotation, another title annotation},
LANGUAGE    = {english and french},
LANGUAGE+an = {1=annotation3; 2=annotation4}
}
```

为了在文献格式中获取注解信息，这里提供了三个宏，分别对应与相应的注解尺度：

\iffieldannotation{⟨*annotation*⟩}{⟨*true*⟩}{⟨*false*⟩}

如果当前数据域有注解，那么执行 ⟨*true*⟩，否则为 false。

\ifitemannotation{⟨*annotation*⟩}{⟨*true*⟩}{⟨*false*⟩}

如果当前数据域的当前项目有注解，那么执行 ⟨*true*⟩，否则为 false。

\ifpartannotation{⟨*part*⟩}{⟨*annotation*⟩}{⟨*true*⟩}{⟨*false*⟩}

如果当前数据域中当前项目中名为 ⟨*part*⟩ 的部分有注解，那么执行 ⟨*true*⟩，否则为 false。

这些宏的使用场合与 `\currentfield`, `\currentlist` 和 `\currentname` 等命令相同（见 § 4.4.2 节），即，在格式指令内部。它们自动确定当前被处理的数据域的名称，以及能够确定列表域中当前项目的 `listcount` 值。姓名部分等项目部分需要显式地指明。下面的例子可以用于姓名格式指令，说明如何使用注解信息来解决本节之前提出的问题：在所有注解为“student”的姓之后加上星号：

```
  \ifpartannotation{family}{student}
    {\textsuperscript{*}}
    {}%
```

将标记为“corresponding”的姓名列表项中的姓和名加粗：

```
\renewcommand*{\mkbibnamegiven}[1]{%
  \ifitemannotation{corresponding}
    {\textbf{#1}}
    {#1}}

\renewcommand*{\mkbibnamefamily}[1]{%
  \ifitemannotation{corresponding}
    {\textbf{#1}}
    {#1}}
```

3.7 参考文献命令

3.7.1 数据源

`\addbibresource`[⟨*options*⟩]{⟨*resource*⟩}

将 ⟨*resource*⟩ 添加到默认资源列表中，例如 `.bib` 文件。该命令只能在导言区中使用。它取代了过时的 `\bibliography` 命令。请注意，文件名包括扩展名，所以不要省略文件名中的 `.bib` 扩展名。另外要注意的是，⟨*resource*⟩ 只能是一个单独的数据源。添加更多的资源需要多次调用 `\addbibresource` 命令，例如：

```
\addbibresource{bibfile1.bib}
\addbibresource{bibfile2.bib}
\addbibresource[location=remote]{http://www.citeulike.org/bibtex/group
    ↪ /9517}
\addbibresource[location=remote,label=lan]{ftp://192.168.1.57/~user/file.
    ↪ bib}
```

由于 ⟨*resource*⟩ 字符串的读取类似于抄录模式，因此它可以包含任意的字符。唯一的限制是其中任何的花括号必须左右匹配。可用的 ⟨*options*⟩ 如下：

`label`=⟨*identifier*⟩

给该数据源分配一个标签。⟨*identifier*⟩ 可以用于在 `refsection` 环境的可选参数中以取代该数据源的全名（见 § 3.7.4 节）。

`location`=⟨*location*⟩ default: `local`

数据源的地址。⟨*location*⟩ 可以是 `local` 或者 `remote`，分别对应本地数据和在线 URL 数据。远程资源需要 `biber` 程序。支持 HTTP 和 FTP 协议。远程的 URL 必须是 `bib` 文件的合法路径全称或者是返回 `bib` 文件的 URL。

`type`=⟨*type*⟩ default: `file`

资源的类型。目前唯一支持的类型是 `file`。

`datatype`=⟨*datatype*⟩ default: `bibtex`

资源的数据类型（格式）。目前支持以下格式：

`bibtex` BibTeX 格式。

`biblatexml` 针对 `biblatex` 的实验性 XML 格式。见附录 D。

`\addglobalbib`[⟨*options*⟩]{⟨*resource*⟩}

该命令不同于 `\addbibresource` 之处在于将 ⟨*resource*⟩ 添加到全局数据源列表中。不过，只有当文档中有参考文献章节并且使用 `refsection` 环境的可选参数（见 § 3.7.4 节）作为确定代替默认资源列表的备选资源时，考虑默认数据源和全局数据源的不同才是有意义的。任何全局资源将被添加到所有的参考文件章节中。

`\addsectionbib`[⟨*options*⟩]{⟨*resource*⟩}

该命令与 `\addbibresource` 的不同之处在于，会记录数据源的 ⟨*options*⟩ 但是 ⟨*resource*⟩ 没有添加到任何数据源列表中。有该需求的场合是 (1) 该数据源仅仅用于 `refsection` 环境的可选参数中（§ 3.7.4 节）；(2) 该数据源需要不同于默认设置的选项。此时，`\addsectionbib` 会在导言区中设置合适的 ⟨*options*⟩，从而在其使用前声明 ⟨*resource*⟩。`label` 选项可以用于分配给该资源一个简短的名称。

`\bibliography`{⟨*bibfile, ...*⟩} Deprecated

添加文献资源的过时命令，仅处于向后兼容性而支持。类似 `\addbibresource`，该命令只能在导言区中使用，并将资源添加到默认资源列表中。它的选项是逗号分隔的 `bib` 文件列表。文件名中的 `.bib` 扩展名可以省略。也可以通过多次调用该命令来添加更多文件。该命令已过时，请考虑使用 `\addbibresource` 来取代。

3.7.2 参考文献

`\printbibliography`[⟨*key=value, ...*⟩]

该命令可以打印出参考文献。它的可选参数是以 ⟨*key*⟩=⟨*value*⟩ 形式给出的一列选项。可用的选项如下：

`env`=⟨*name*⟩ default: `bibliography/shorthands`

可以用 `\defbibenvironment` 定义的环境来控制参考文献和 shorthands 列表的高层次布局。该选项选择了一个环境。⟨*name*⟩ 对应于用 `\defbibenvironment` 定义环境时的标识符。缺省状态下，`\printbibliography` 命令使用标识符 `bibliography`；而 `\printshorthands` 使用 `shorthands`。另见 §§ 3.7.3 和 3.7.7 节。

`heading`=⟨*name*⟩ default: `bibliography/shorthands`

参考文献和 shorthand 列表通常有一个章标题或者节标题。该选项选择由 `\defbibheading` 定义的标题名 ⟨*name*⟩。缺省状态下，`\printbibliography` 命令使用标题名 `bibliography`；而 `\printshorthands` 使用 `shorthands`。另见 §§ 3.7.3 和 3.7.7 节。

`title`=⟨*text*⟩

如果标题定义支持的话，该选项覆盖由 `heading` 选项提供的缺省标题名。详见 § 3.7.7 节。

`prenote`=⟨*name*⟩

前注是打印在标题之后、文献列表之前的任意文本片段。该选项选择由 `\defbibnote` 所定义的前注 ⟨*name*⟩。缺省状态下不打印任何前注。该注记使用标准正文字体。它不受 `\bibsetup` 和 `\bibfont` 的影响但可以包含自己的字体声明。详见 § 3.7.8 节。

postnote=⟨*name*⟩

后注是打印在参考文献列表之后的任意文本片段。该选项选择由 `\defbibnote` 所定义的后注 ⟨*name*⟩。缺省状态下不打印任何后注。该注记使用标准正文字体。它不受 `\bibsetup` 和 `\bibfont` 的影响但可以包含自己的字体声明。详见 § 3.7.8。

section=⟨*integer*⟩ default: `current section`

只打印在第 ⟨*integer*⟩ 文节中引用的条目。该参考文献节从 1 开始编号。所有在 `refsection` 环境外给出的引用标记为第零节。详见 § 3.7.4 和 § 3.12.3 节中的使用例子。

segment=⟨*integer*⟩ default: `0`

只打印在第 ⟨*integer*⟩ 文献段中引用的条目。参考文献段从 1 开始编号。所有在 `refsection` 环境外给出的引用标记为第零段。详见 § 3.7.4 和 § 3.12.3 节中的使用例子。请注意，一节内部的片段是在该节中局部编号的，故而需要的片段是被查询（或者当前激活的）节的第 n 段。

type=⟨*entrytype*⟩

只打印类型为 ⟨*entrytype*⟩ 的条目。

nottype=⟨*entrytype*⟩

只打印类型不为 ⟨*entrytype*⟩ 的条目。该选项可以使用多次。

subtype=⟨*subtype*⟩

只打印域 `entrysubtype` 定义为 ⟨*subtype*⟩ 的条目。

notsubtype=⟨*subtype*⟩

只打印域 `entrysubtype` 没有定义或者不为 ⟨*subtype*⟩ 的条目。该选项可以使用多次。

keyword=⟨*keyword*⟩

只打印域 `keywords` 包括 ⟨*keyword*⟩ 的条目。该选项可以使用多次。

notkeyword=⟨*keyword*⟩

只打印域 `keywords` 不包括 ⟨*keyword*⟩ 的条目。该选项可以使用多次。

category=⟨*category*⟩

只打印属于 ⟨*category*⟩ 类型的条目。该选项可以使用多次。

notcategory=⟨*category*⟩

只打印不属于 ⟨*category*⟩ 类型的条目。该选项可以使用多次。

filter=⟨*name*⟩

使用由 \defbibfilter 定义的 filter ⟨*name*⟩ 来过滤条目。详见 § 3.7.9 节。

check=⟨*name*⟩

使用由 \defbibcheck 定义的 check ⟨*name*⟩ 来过滤条目。详见 § 3.7.9 节。

resetnumbers=⟨*true,false,number*⟩

该选项只用于数值引用/参考文献样式，并且要求 § 3.1.2.1 中的 defernumbers 选项全局启用。如果启用的话，它将重新设置分配给相应文献中条目的数值标签，即，编号会重新从 1 开始。此外还可以传递数值给该选项以重置编号为给定的数值，例如 resetnumbers=10，这样可以改进整个文档中编号的连续性。请小心使用本选项，因为在手动重新设置下，biblatex 不能保证标签是全局唯一的。

omitnumbers=true, false

该选项只用于数值引用/参考文献样式，并且要求 § 3.1.2.1 中的 defernumbers 选项全局启用。如果启用的话，biblatex 不会为相应文献中的条目分配数值标签。当数值型子文献和其它不同格式（例如作者-标题或者作者-年份）的子文献相混合时，这是很有用的。

\bibbysection[⟨*key=value, ...*⟩]

该命令会自动遍历所有的参考文献节。这等价于为每一节给出一个 \printbibliography 命令，不过会有额外好处：自动跳过不含参考文献的节。请注意，\bibbysection 一开始寻找第 1 节中的文献。它会忽略 refsection 外给出的文献，因为它们被分配给第零节。使用例子请参考 § 3.12.3 节。选项可以是由 \printbibliography 支持的一个子集。有效选项是 env、heading、prenote 和 postnote。当前文献内容排序格式会应用在所有的节中（见 § 3.7.10 节）。

\bibbysegment[⟨*key=value, ...*⟩]

该命令会自动遍历所有的参考文献段。这等价于为当前 refsection 的每一段给出一个 \printbibliography 命令，不过会有额外好处：自动跳过不含参考文献的片段。请注意，\bibbysection 一开始寻找第 1 段中的文献。它会忽略 refsection 外给出的文献，因为它们被分配给第 0 段。使用例子请参考 § 3.12.3。选项可以是由 \printbibliography 支持的一个子集。有效选项是 env、heading、prenote 和 postnote。当前文献内容排序格式会用于所有的段中（见 § 3.7.10 节）。

\bibbycategory[⟨*key=value, ...*⟩]

该命令遍历所有的文献类型。这等价于为每一类型给出一个 \printbibliography 命令，不过会有额外好处：自动跳过空类型。类型按照声明的顺序处理。示例见 § 3.12.3 节。选项可以是由 \printbibliography 支持的一个子集。有效选项是 env、heading、prenote 和 postnote。请注意，heading 对于该命令是无效的。当前类型的名字会自动作为标题名。这等价于传递 heading=⟨*category*⟩ 给 \printbibliography，并且意味着对于每一类型都必须有一个匹配的标题定义。当前文献内容排序格式会用于所有的类型中（见 § 3.7.10 节）。

`\printbibheading`[⟨*key=value, ...*⟩]

该命令打印出由 `\defbibheading` 定义的参考文献标题。它有一个可选项，是用 ⟨*key*⟩=⟨*value*⟩ 记号给出的选项列表。选项是 `\printbibliography` 支持的一个小子集。有效选项是 `heading` 和 `title`。缺省情况下，该命令使用标题 `bibliography`。详见 § 3.7.7 节。实例也可见 §§ 3.12.3 和 3.12.4 节。

如果想要在参考文献中使用非全局排序格式的另外一种排序格式，使用 § 3.7.10 节提供的文献内容切换命令。

3.7.3 参考文献列表

`biblatex` 除了可以打印常规参考文献之外，还能根据文献数据打印任意文献信息列表，例如，与特定条目或者期刊标题缩写有关的速记缩写列表。

文献列表与常规参考文献不同的是，使用同一文献驱动打印所有条目，而不是根据条目类型使用特定于条目的驱动。

`\printbiblist`[⟨*key=value, ...*⟩]{⟨*biblistname*⟩}

该命令用于打印文献列表。其可选项是 ⟨*key*⟩=⟨*value*⟩ 形式的一列选项。除了 `resetnumbers` 和 `omitnumbers`，`\printbibliography` 命令（见 § 3.7.2 节）支持的其它选项在这里都是有效的。如果文档中有任何 `refsection` 环境，那么文献列表只针对于这些环境，详见 § 3.7.4 节。默认情况下该命令使用标题 `biblist`，详见 § 3.7.7 节。

必选项 ⟨*biblistname*⟩ 是文献列表的标题，用于确定如下项目：

- 用于打印列表条目的默认文献驱动。
- 使用 `\DeclareBiblistFilter` 声明的默认 filter（见 § 4.5.7 节），用于过滤 `biber` 返回的条目。
- 使用 `\defbibcheck` 命令声明的默认 check（见 § 3.7.9 节），用于后置处理列表条目。
- 默认使用的 bib 环境。
- 默认使用的排序格式名称。

在列表的排序方面，默认使用与该文献列表同名的排序格式（如果存在的话）。只有当未定义时才会切换到备选的当前内容排序格式（见 § 3.7.10 节）。

最常用的文献列表是关于某些条目的速记列表，出于向后兼容性专门有一个别名 `\printshorthands[…]`，定义如下：

```
\printbiblist[...]{shorthand}
```

`biblatex` 自动支持默认数据模型中标记为“Label fields”的数据域（见 § 2.2.2 节）。这些域已经自动为其定义了如下项目：

- 默认的 bib 环境（见 § 3.7.7 节）。

- 文献列表 filter（见 § 4.5.7 节）
- 一些支持的格式和长度（见 § 4.10.5 和 § 4.10.4 节）。

因此，打印带有这些域的文献列表只需要很少的设置。例如，想要打印出期刊标题缩写列表，只需要将如下一小段代码放在导言区中：

```
\DeclareBibliographyDriver{shortjournal}{%
  \printfield{journaltitle}}
```

然后在正文中想要打印列表的地方使用如下代码：

```
\printbiblist[title={Journal Shorthands}]{shortjournal}
```

由于默认数据模型将 `shortjournal` 定义为“标签域”，因此在这个例子中：

- 使用自动创建的“shortjournal”bib 环境。
- 使用自动创建的“shortjournal”文献列表 filter，返回 `.bbl` 文件中只带有 `shortjournal` 域的条目。
- 使用定义的“shortjournal”文献驱动来打印条目。
- 使用默认的“biblist”标题，但是这里用“Journal Shorthands”来代替。
- 如果没有名为 `shortjournal` 的格式，那么使用当前文献内容排序格式。

很多情况下想要根据列表中标签域进行排序。由于根据列表名可以自动获取排序格式，因此此时可以简单地使用如下代码：

```
\DeclareSortingScheme{shortjournal}{
  \sort{
        \field{shortjournal}
  }
}
```

自然地，`\printbiblist` 命令的选项以及环境、filters 等的定义可以覆盖所有的默认设置。因此通过这种方法可以从文献数据中打印任意类型的文献列表，并且包含各式信息。

文献列表通常用于打印各类 shorthand 列表。如果多个条目有相同的 shorthand 就会导致重复的条目。例如，如果有几篇论文在同一期刊上，那么期刊缩写列表中就会出现重复条目。不过，这样的列表会自动获取与列表同名的 `\bibcheck`，进而定义相应的 check 来删除重复项目。如果使用 `shortjournal` 域来定义打印所有期刊缩写的列表，那么需要定义如下的 `\bibcheck`：

```
\defbibcheck{shortjournal}{%
   \iffieldundef{shortjournal}{\skipentry}{%
```

```
\iffieldundef{journal}{\skipentry}{%
 \ifcsdef{\strfield{shortjournal}=\strfield{journal}}
   {\skipentry}
   {\savefieldcs{journal}{\strfield{shortjournal}=\strfield{journal
↪ }}}}}}
```

3.7.4 参考文献分节

在文档中，refsection 环境用于标记参考文献分节。该环境主要用于在文档的每一章、节或其它部分中实现各自独立的参考文献和 shorthand 列表。在一个文献分节内部，所有引用文献分配的标签都局部在该环境中。技术上，尽管文献分节通常在每一章或每一节中使用，但它们与 \chapter 和 \section 等文档划分是完全独立的。关于自动实现这一功能请参考 § 3.1.2.1 中的 refsection 宏包选项。使用例子也可以参见 § 3.12.3。

\begin{refsection}[⟨*resource, ...*⟩]

\end{refsection}

可选项是特定于该参考文献分节的逗号分隔资源列表。如果省略了该选项，参考文献节会使用缺省的数据源列表，由导言区的 \addbibresource 指定。如果提供了该选项，它会替代缺省的资源列表。不过，由 \addglobalbib 指定的全局文献资源总是包含在内的。refsection 环境不可以相互嵌套，但是可以在 refsection 环境内使用 refsegment 环境来进一步分段。当打印参考文献时，使用 \printbibliography 的 section 选项来选择节；同样地当打印文献列表时使用 \printbiblist 对应的选项。参考文献分节从 1 开始编号。当前节的编号也被写入副本文件中。所有在 refsection 环境外给出的引用都归到第 0 节中。如果在 refsection 内部使用 \printbibliography 环境，它会自动选择当前节。此时不需要 section 选项。这也适用于 \printbiblist。

\newrefsection[⟨*resource, ...*⟩]

该命令类似于 refsection 环境，不同之处在于它是单独命令而不是一个环境。它会自动结束之前的文献分节（如果有的话）并立即开始新的一节。请注意，文档中由最后一个 \newrefsection 开始的文献节会延续到文档的最后。如果你想提前终止的话可以使用 \endrefsection。

3.7.5 参考文献分段

在文档中，refsegment 环境用来标记参考文献片段。该环境用于实现在文档的每一章、节或其它部分中将全局的参考文献分成片段。技术上，尽管文献分段通常在每一章或每一节中使用，但它们与 \chapter 和 \section 等文档划分是完全独立的。关于自动实现这一功能请参考 § 3.1.2.1 中的 refsegment 宏包选项。使用例子也可以参见 § 3.12.3。

\begin{refsegment}

\end{refsegment}

regsection 与 refsegment 环境的不同之处在于，前者创建局部于该环境的标签而后者仅为 \printbibliography 命令的 segment filter 提供目标而不影响标签。在整个文档中它们是唯一确定的。refsegment 环境不可以嵌套，但是你可以将其与 refsection 环境结合使用来将文献节细分为段。此时，这些文献分段是局部于被包含的 refsection 环境的。当打印参考文献时，使用 \printbibliography 的 segment 选项来选择文献分段。在一节内，文献段从 1 开始编号，并且当前段的编号会被写入到一个副本文件中。所有在 refsegment 环境之外的引用都归到第 0 段。与 refsection 环境相反，当 \printbibliography 在一个 refsegment 环境内使用时，当前文献分段并不自动选定。

\newrefsegment 该命令类似于 refsegment 环境，不同之处在于它是单独命令而不是一个环境。它会自动结束之前的文献分段（如果有的话）并立即开始新的一段。请注意，由最后一个 \newrefsegment 开始的文献分段会延续到文档结束。如果你想提前终止的话可以使用 \endrefsegment。

3.7.6 参考文献分类

参考文献分类允许将参考文献针对不同主题或不同文献类型分成若干部分，例如分成主要文献和次要文献。使用例子参见 § 3.12.4 节。

\DeclareBibliographyCategory{⟨*category*⟩}

声明一个新的 ⟨*category*⟩，可以和 \addtocategory 以及 \printbibliography 的 category、notcategory filter 结合使用。该命令在导言区中使用。

\addtocategory{⟨*category*⟩}{⟨*key*⟩}

将 ⟨*key*⟩ 关键字分配给 ⟨*category*⟩ 类，可以和 \addtocategory 以及 \printbibliography 的 category、notcategory filter 结合使用。该命令可以在导言区和正文中使用。⟨*key*⟩ 可以是一个单独条目关键字或者逗号分隔的键值列表。该分配是全局的。

3.7.7 参考文献标题与环境

\defbibenvironment{⟨*name*⟩}{⟨*begin code*⟩}{⟨*end code*⟩}{⟨*item code*⟩}

该命令定义参考文献环境。其中 ⟨*name*⟩ 是标识符，当选择该环境时会传递给 \printbibliography 和 \printshorthands 的 env 选项。⟨*begin code*⟩ 是该环境开始时执行的 LaTeX 代码；而 ⟨*end code*⟩ 在该环境结束时执行；⟨*item code*⟩ 是在参考文献或者 shorthand 列表的每一条目开始时执行的代码。如下是基于 LaTeX 标准 list 环境定义的例子。

```
\defbibenvironment{bibliography}
  {\list{}
```

```
    {\setlength{\leftmargin}{\bibhang}%
     \setlength{\itemindent}{-\leftmargin}%
     \setlength{\itemsep}{\bibitemsep}%
     \setlength{\parsep}{\bibparsep}}}
  {\endlist}
  {\item}
```

如上述例子所示，`\defbibenvironment` 的使用大体类似于 `\newenvironment`，不同之处在于有一个额外的必选项 ⟨*item code*⟩。

`\defbibheading{`⟨*name*⟩`}[`⟨*title*⟩`]{`⟨*code*⟩`}`

该命令定义参考文献标题。其中 ⟨*name*⟩ 是标识符，当选择该标题时会传递给 `\printbibliography` 和 `\printshorthands` 的 `env` 选项。⟨*code*⟩ 是能生成完整标题的 LaTeX 代码，包括页眉和目录中的条目（如果必要的话）。如果 `\printbibliography` 或 `\printshorthands` 带有 `title` 选项，那么 `title` 将作为 `#1` 传递给标题定义；否则由可选的 ⟨*title*⟩ 确定的标题将作为 `#1` 传递给标题定义。⟨*title*⟩ 选项通常是 `\bibname`、`\refname` 或者 `\biblistname` （见 § 4.9.2.1 节）。如果在导言区中改变文档标题时，那么之后通常需要该命令。如下是一个简单标题定义的例子：

```
\defbibheading{bibliography}[\bibname]{%
  \chapter*{#1}%
  \markboth{#1}{#1}}
```

以下预定义的标题与 `\printbibliography` 和 `\printbibheading` 结合使用：

`bibliography`

如果没有给出 `heading` 选项，那么这是 `\printbibliography` 使用的默认标题。缺省定义取决于文档类。如果文类提供 `\chapter` 命令，那么该标题就类似于标准 LaTeX 的 `book` 文类的参考文献标题，即使用 `\chapter*` 来创建不带编号的章，并且不包含在目录中。如果没有 `\chapter` 命令，那么它将类似于标准 LaTeX 的 `article` 文类的参考文献标题，即使用 `\section*` 来创建不带编号的节，并且不包含在目录中。标题中使用的字符串也取决于文档类。`book` 文档类使用本地化字符串 `bibliography`，在其它文档类中则是 `references`（见 § 4.9.2 节）。关于文档类的提示也可以见 §§ 3.13.1 和 3.13.2 节。

`subbibliography`

类似于 `bibliography`，但是标题格式低一级。即，在 `book` 文档类中使用 `\section*` 而不是 `\chapter*`，其它情况使用 `\subsection*` 而不是 `\section*`。

`bibintoc`

类似于 `bibliography` 但是在目录中添加条目。

`subbibintoc`

类似于 `subbibliography` 但是在目录中添加条目。

`bibnumbered`

类似于 `bibliography` 但是使用 `\chapter` 或 `\section` 来创建带编号的条目，同时也添加到目录中。

`subbibnumbered`

类似于 `bibliography` 但是使用 `\section` 或 `\subsection` 来创建带编号的条目，同时也添加到目录中。

`none`

空白的标题定义，用来取消标题。

以下预定义的标题与 `\printshorthands` 结合使用：

`biblist`

如果没有给出 `heading` 选项，那么这是 `\printbiblist` 使用的缺省标题。类似于上面的 `bibliography`，不过是使用本地化字符串 `shorthands` 而不是 `bibliography` 或 `references` （见 § 4.9.2 节）。关于文档类的提示另见 §§ 3.13.1 和 3.13.2 节。

`biblistintoc`

类似于 `shorthands` 但是在目录中添加条目。

`biblistnumbered`

类似于上面的 `biblist` 但是使用 `\chapter` 或 `\section` 来创建带编号的标题，同时也添加到目录中。

3.7.8 参考文献注记

`\defbibnote{`⟨*name*⟩`}{`⟨*text*⟩`}`

定义名为 ⟨*name*⟩ 的参考文献注记，通过 `\printbibliography` 和 `\printbiblist` 的 `prenote` 和 `postnote` 选项使用。⟨*text*⟩ 可以是任意文本片段，通常包含若干段落和字体声明。另见 § 3.13.6 节。

3.7.9 参考文献过滤和检查

`\defbibfilter{`⟨*name*⟩`}{`⟨*expression*⟩`}`

定义一个可定制的文献过滤 ⟨*name*⟩，可以通过 `\printbibliography` 的 `filter` 选项使用。⟨*expression*⟩ 是复合测试，基于逻辑运算符 `and`、`or`、`not`，组运算符 `(...)`，以及以下的基本测试：

segment=⟨*integer*⟩

匹配所有在参考文献分段 ⟨*integer*⟩ 中引用的条目。

type=⟨*entrytype*⟩

匹配所有类型为 ⟨*entrytype*⟩ 的条目。

subtype=⟨*subtype*⟩

匹配所有 entrysubtype 域为 ⟨*subtype*⟩ 的条目。

keyword=⟨*keyword*⟩

匹配所有 keywords 域包含 ⟨*keyword*⟩ 的条目。如果 ⟨*keyword*⟩ 包含空格，那么需要用括号括起来。

category=⟨*category*⟩

匹配所有由 \addtocategory 归入 ⟨*category*⟩ 类的条目。

如下是一个 filter 表达式的例子：

```
\defbibfilter{example}{%
  ( type=book or type=inbook )
  and keyword=abc
  and not keyword={x y z}
}
```

该 filter 匹配的条目规则是，条目类型是 @book 或 @inbook，keywords 域包含关键词“abc”但不包含“x y z”。从以上例子可以看出，所有的元素由空白分开（空格、制表符或者换行）。等号周围没有空白。逻辑运算使用 etoolbox 宏包的 \ifboolexpr 执行。关于该语法详见 etoolbox 手册。biblatex 旧版本中使用的 ifthen 宏包的 \ifthenelse 语法这里仍然支持。如下是相同的测试，使用 ifthen 样式的语法：

```
\defbibfilter{example}{%
  \( \type{book} \or \type{inbook} \)
  \and \keyword{abc}
  \and \not \keyword{x y z}
}
```

请注意，定制的 filter 对于所在的参考文献分节是局部的。使用 \printbibliography 的 section filter 来选择不同的分节。这在定制 filter 中是不可能的。

\defbibcheck{⟨*name*⟩}{⟨*code*⟩}

定义了可定制的参考文献 check ⟨*name*⟩，可以通过 \printbibliography 的 check 选项使用。\defbibcheck 从概念上类似于 \defbibfilter 不过更加低层。与高层次

表达式不同，⟨*code*⟩ 是 LaTeX 代码，更像是驱动定义中使用的代码，可以执行任意测试来决定是否打印某个给定的条目。当执行 ⟨*code*⟩ 时，相应条目的文献数据是可用的。在 ⟨*code*⟩ 中使用 `\skipentry` 命令会跳过当前条目。例如，下面的 check 只会输出带有 `abstract` 域的条目：

```
\defbibcheck{abstract}{%
  \iffieldundef{abstract}{\skipentry}{}}
...
\printbibliography[check=abstract]
```

下面的 check 会排除所有在 2000 年之前出版的条目：

```
\defbibcheck{recent}{%
  \iffieldint{year}
    {\ifnumless{\thefield{year}}{2000}
       {\skipentry}
       {}}
    {\skipentry}}
```

更多细节请参见作者指南，特别是 §§ 4.6.2 和 4.6.3 节。

3.7.10 著录文境

参考文献列表中文献的引用和打印都处于某个著录文境（context）内。对于某一条目，著录文境决定了实际用于引用或者提供文献信息的数据。一个著录文境包括以下信息[34]：

- 排序格式
- 构建姓名排序关键字的格式
- 使用字母或数值标签的引用格式的前缀字符串

著录文境具有双重意义。首先，会用于设置影响打印参考文献的选项；其次，设置的选项还可以影响引用命令打印的数据。前一应用场景是很常见的，例如，打印多个带有不同排序格式的参考文献表。

```
\usepackage[sorting=nyt]{biblatex}
\begin{document}
\cite{one}
\cite{two}
\printbibliography
\newrefcontext[sorting=ydnt]
\printbibliography
```

[34] 设计“著录文境”这一概念的目的在于，使其在未来具有可扩展性。

这里我们打印两个参考文献表。其中一个带有默认的“nyt”排序格式，另一个则使用“ydnt”排序格式。

为了说明著录文境的第二种类型应用，我们必须意识到这一点：条目的实际数据可以基于不同的著录文境而变化。在以下的情况中这一点尤其明显：由后端生成的 extra* 域（例如 extrayear）与条目在排序之后的顺序有关，这样出来的结果就是预期的“a, b, c”的顺序。这就表明，条目的数据在不同排序格式下可以不一样。如果文档中包含多个带有不同排序格式的参考文献列表，那么 .bbl 文件中可能出现多个排序列表，它们带有同一条目但是其数据不同（例如 extrayear 的值可以不同）。著录文境的目的就在于将这些事项封装在一个语境内部，这样 biblatex 就可以使用正确的条目数据。以下的例子展示了使用与全局排序格式不同的另一格式打印参考文献列表，使得同一条目的 extra* 域在不同排序列表中是不同的：

```
\usepackage[sorting=nyt,style=authoryear]{biblatex}
\DeclareSortingScheme{yntd}{
  \sort{
    \field[strside=left,strwidth=4]{sortyear}
    \field[strside=left,strwidth=4]{year}
    \literal{9999}
  }
  \sort{
    \field{sortname}
    \field{author}
    \field{editor}
  }
  \sort[direction=descending]{
    \field{sorttitle}
    \field{title}
  }
}
\begin{document}
\cite{one}
\cite{two}
\printbibliography
\newrefcontext[sorting=yntd]
\cite{one}
\cite{two}
\printbibliography
```

这里，第二次使用引用命令和 \printbibliography 命令时会使用在定制“yntd”排序格式的著录文境中的数据，这与默认的“nyt”格式相关联的数据可能会不相同。也就是说，对于同一条目，不同著录文境中的引用标签（在使用 extrayear 的 authoryear 样式中）可以不同，这样对其引用就可以不一致。

引用文境可以使用 \DeclareRefcontext 命令进行声明，然后通过文境的名称进行使用，见以下说明。

默认情况下，用于引用的数据来自于打印该条目的最后一个参考文献列表所在的引用文境。例如：

```
\DeclareRefcontext{ap}{labelprefix=A}
\begin{document}

\cite{book, article, misc}

\printbibliography[type=book]

\newrefcontext{ap}
\printbibliography[type=article]

\newrefcontext[sorting=ydnt]
\printbibliography[type=misc]

\end{document}
```

这个例子同时展示了引用文境的声明和使用。在该例子中假设条目类型就是条目的键名，文献引用就对应与用户通常预料到的默认场景。

- 条目 book 的引用会从全局引用文境中提取数据，因为打印该条目的最后的参考文献列表位于全局引用文境中。
- 条目 article 的引用会从带有 labelprefix=A 的引用文境中提取数据，因此引用时会带有前缀“A”。
- 条目 misc 的引用会从带有 sorting=ydnt 的引用文境中提取数据。

有这样一种情况，条目在多个参考文献列表中并且有不同的形式或者可能带有不同的标签（例如，带有不同 labelprefix 值的数字格式）。此时需要告诉 biblatex 希望从哪个引用文境中提取引用信息。如上所述，这可以通过显式地将引用置于引用文境中而实现。但是在大文档中这种方式会很繁重，因此提供了将引用以程序化的方式分配到引用文境的功能，见下面的 \assignrefcontext* 宏命令。

\DeclareRefcontext{⟨*name*⟩}{⟨*key=value, ...*⟩}

声明一个名称为 ⟨*name*⟩ 的引用文境。⟨*key=value*⟩ 选项定义该文境的属性。所有的文境属性都是可选的，缺省为全局设置。有效的选项为：

sorting=⟨*name*⟩

指定由之前的 \DeclareSortingScheme 命令定义的排序格式。对于在该文境内引用命令中的条目，该格式用于确定检索和打印的数据。

sortingnamekeyscheme=⟨*name*⟩

指定由之前的 \DeclareSortingNamekeyScheme 命令定义的排序姓名关键字格式。该格式用于为文境内的姓名构建排序关键字。

labelprefix=⟨*string*⟩

该选项只用于数字型引用和文献样式，需要全局开启 § 3.1.2.1 节中的 defernumbers 选项。设置改选项也会为该文境范围内的任意 \printbibliography 启用 resetnumbers 选项（除非 resetnumbers 被用户指定的值覆盖）。该选项将 ⟨*string*⟩ 作为前缀分配到该引用文境中的所有条目。例如，如果 ⟨*string*⟩ 是 A，那么打印出来的数值标签就会形如 [A1], [A2], [A3] 等。特别适用的场合是，将参考文献列表划分成带有不同前缀的子列表。⟨*string*⟩ 可以用于所有有关条目中 labelprefix 域中的样式。详见 § 4.2.4.2 节。

\begin{refcontext}[⟨*key=value, ...*⟩]{⟨*name*⟩}

\end{refcontext}

将引用文境封装在一个环境内。可能的 ⟨*key*⟩=⟨*value*⟩ 可选项和 \DeclareRefcontext 中的相同，并且覆盖名为 ⟨*name*⟩ 的引用文境的选项。⟨*name*⟩ 也可以省略成 {}，甚至空的括号也可以省略[35]

refcontext 环境不可以相互嵌套，如果这样的话 biblatex 会报错。

\newrefcontext[⟨*key=value, ...*⟩]{⟨*name*⟩}

该命令类似于 refcontext 环境，不同之处在于这是单独的命令而不是环境。它会自动结束任何之前以 \newrefcontext 开始的引用文境片段（如果有的话），并立即开启新的引用文境。注意，文档中最后的 \newrefcontext 命令开启的引用文境会一直持续到文档的最后。如果想要提前终止，那么需要使用 \endrefcontext 命令。

在文档的开始，总会有一个全局的文境，其中为每一引用文境选项进行了全局设置。这里的例子总结了引用文境的不同设置：

```
\usepackage[sorting=nty]{biblatex}

\DeclareRefcontext{testrc}{sorting=nyt}

% Global reference context:
%   sorting=nty
%   sortingnamekeyscheme=global
%   labelprefix=

\begin{document}
```

[35] 这种有点怪异的句法是出于对 biblatex <3.5 的向后兼容性。

```
\begin{refcontext}{testrc}
% reference context:
%   sorting=nyt
%   sortingnamekeyscheme=global
%   labelprefix=
\end{refcontext}

\begin{refcontext}[labelprefix=A]{testrc}
% reference context:
%   sorting=nyt
%   sortingnamekeyscheme=global
%   labelprefix=A
\end{refcontext}

\begin{refcontext}[sorting=ydnt,labelprefix=A]
% reference context:
%   sorting=ydnt
%   sortingnamekeyscheme=global
%   labelprefix=A
\end{refcontext}

\newrefcontext}[labelprefix=B]
% reference context:
%   sorting=nty
%   sortingnamekeyscheme=global
%   labelprefix=B
\endrefcontext

\newrefcontext}[sorting=ynt,labelprefix=C]{testrc}
% reference context:
%   sorting=ynt
%   sortingnamekeyscheme=global
%   labelprefix=C
\endrefcontext
```

\assignrefcontextkeyws[⟨*key=value, ...*⟩]{⟨*keyword1,keyword2, ...*⟩}
\assignrefcontextkeyws*[⟨*key=value, ...*⟩]{⟨*keyword1,keyword2, ...*⟩}
\assignrefcontextcats[⟨*key=value, ...*⟩]{⟨*category1, category2, ...*⟩}
\assignrefcontextcats*[⟨*key=value, ...*⟩]{⟨*category1, category2, ...*⟩}
\assignrefcontextentries[⟨*key=value, ...*⟩]{⟨*entrykey1, entrykey2, ...*⟩}
\assignrefcontextentries*[⟨*key=value, ...*⟩]{⟨*entrykey1, entrykey2, ...*⟩}
\assignrefcontextentries[⟨*key=value, ...*⟩]{⟨***⟩}
\assignrefcontextentries*[⟨*key=value, ...*⟩]{⟨***⟩}

当默认行为不充分时，这些命令会自动将引用置于引用文境中。默认行为是指，从最后打印条目的参考文献列表所在的引用文境中提取引用数据。对于没有在参考文献列表中打印但是以某种方式使用的引用，默认会从文档一开始建立的全局引用文境中提取数据。为了覆盖这一行为，可以手动将引用命令放置在 `refcontext` 环境内，但是这样容易出错并且很繁琐。除此之外，可以登记一个关于 ⟨*keywords*⟩、⟨*categories*⟩ 和 ⟨*entrykeys*⟩ 的逗号分隔列表。这样，任何带有指定关键字的条目、任何指定类别的条目（见 § 3.12.4 节）、任何指定引用关键字的条目都会分别从由 ⟨*refcontext key/values*⟩ 指定的特定引用文境中提取数据，并按照对应的 `refcontext` 环境选项进行解析。这样的引用文境自动分配方式特定于当前的参考文献分节。你可以在任意的这些命令中指定相同的引用键，但需要注意的是，分配方式按照 ⟨*keywords*⟩, ⟨*categories*⟩, ⟨*entrykeys*⟩ 的顺序，后面的规范会覆盖之前的规范。`\assignrefcontextentries` 命令可以接受单个的星号作为选项以代替一列条目键，这样可以将某一参考文献分节中的所有条目键都分配给某个引用文境，而不必显式列出。例如：

```
\assignrefcontextentries[labelprefix=A]{key2}
\cite{key1}
\begin{refcontext}[labelprefix=B]
\cite{key2}
\end{refcontext}
```

这里 `key2` 的引用数据会从引用文境 `labelprefix=A` 中提取，而不是 `labelprefix=B`。即，标签的前缀是“A”不是“B”。带星号的版本不会覆盖局部的引用文境，也就是：

```
\assignrefcontextentries*[labelprefix=A]{key2}
\cite{key1}
\begin{refcontext}[labelprefix=B]
\cite{key2}
\end{refcontext}
```

`key2` 的引用数据会从 `labelprefix=B` 引用文境中提取。注意，这些命令大部分情况下都不必使用，除非多个参考文献表中有相同的文献引用，并且 `biblatex` 按照

默认设置不知道引用应该指向哪一个文献列表。详见文件 94-labelprefix.tex 中的例子。

3.7.11 动态条目集

除了 @set 条目类型之外，biblatex 也支持基于文档或参考文献分节定义的动态条目集。下面的命令定义了 ⟨*key*⟩ 集合，可以用在导言区或正文中：

\defbibentryset{⟨*key*⟩}{⟨*key1,key2,key3, ...*⟩}

⟨*key*⟩ 是集合的条目键，像其它条目键一样用于指向该集合。⟨*key*⟩ 必须是唯一的，并且不能与其它条目键名冲突。第二个选项是组成该集合的逗号分隔条目键列表。\defbibentryset 也蕴含了与 \nocite 命令的等价性，即所有声明的集合也都添加到参考文献表中。当多次声明相同集合时，只有第一次调用的 \defbibentryset 会定义该集合。接下来的对于相同 ⟨*key*⟩ 的定义将被忽略并如同 \nocite⟨*key*⟩ 一样处理。在正文中定义的动态条目集如果包含在 refsection 环境中，那么是局部的。否则它们会归到第 0 文献分节中。在导言区中定义的动态条目集也归到第 0 文献分节中。详见 § 3.12.5 节。

3.8 引用命令

大体上，所有的引用命令都有一个必选参数和两个可选参数。前注 ⟨*prenote*⟩ 是引用开始时打印的文本，通常是“see”或“compare”等提示。而后注 ⟨*postnote*⟩ 是引用结束时打印的文本，通常是页码数。如果只给出一个可选参数，那么将视作后缀。如果想给出前注但不要后注，那么需要将第二个可选项设置为空，例如 \cite[see][]{key}。所有的引用命令中选项 ⟨*key*⟩ 都是必须的，其内容是 bib 文件中的条目键或者对应于条目键的逗号分隔列表。总的来说，以下所有的基本引用命令都具有如下的句法结构：

\command[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*keys*⟩}⟨*punctuation*⟩

如果启用了 § 3.1.2.1 节中的 autopunct 宏包选项，这些命令会首先扫描紧跟在最后选项的 ⟨*punctuation*⟩ 标点符号。这可用于避免引用之后出现错误的标点符号。该特性由 \DeclareAutoPunctuation 命令配置，详见 § 4.7.5 节。

3.8.1 标准命令

下列命令由引用样式定义。引用样式可以提供任意特殊命令，但是这些是由通用样式提供的标准命令。

\cite[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}
\Cite[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}

基本引用命令。只打印出引用，而不带括号等任何附加物。不过数值和字母样式仍然会将标签放在方括号里，因为否则的话参考文献可能含糊不清。\Cite 与 \cite 类似，不同之处仅仅在于，如果开启了 useprefix 选项，并且引用样式会打印全名，那么引用中名部分（first name）的前缀要大写。

\parencite[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}
\Parencite[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

这些命令的格式类似于 \cite，不过将引用全部放入圆括号内。不过数值和字母样式仍然使用方括号。\Parencite 与 \parencite 类似，不过，如果有姓名前缀并且引用样式打印全名，并且同时开启了 useprefix 选项，那么 \Parencite 会使得该姓名前缀首字母大写。

\footcite[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}
\footcitetext[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

这些命令的格式类似于 \cite，不过将引用的全部放入脚注内并在末尾加上句号。在脚注中，如果有姓名前缀并且引用样式打印全名，同时开启了 useprefix 选项的话，那么该姓名前缀的首字母会自动大写。\footcitetext 与 \footcite 的不同之处在于它使用 \footnotetext 而不是 \footnote 命令。

3.8.2 样式相关命令

下列额外的引用命令只由本宏包所带的某些引用样式提供。

\textcite[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}
\Textcite[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

这些引用命令在本宏包所带的所有样式中都有提供。它们用于正文中代替句子成分。它们打印出作者或编辑，后面接着在括号中括起来的引用标签。取决于引用样式，标签可以是数字、出版年份、缩写版本的标题等等。数值和字母样式会使用方括号来代替圆括号。在详细样式中，标签在脚注中提供。作者或编辑名与脚注标记的空格将被移除。\Textcite 与 \textcite 类似，不同之处在于如果有前缀名并且启用 useprefix 选项的话，引用中的前缀名是大写的。

\smartcite[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}
\Smartcite[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

在脚注中与 \parencite 相似而在正文中与 \footcite 类似。

\cite*[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

该命令由所有的*作者–年份*和*作者–标题*样式提供。它类似于常规的 \cite 命令，但是只打印出年份或者标题。

\parencite*[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

该命令由所有的*作者–年份*和*作者–标题*样式提供。它类似于常规的 \parencite 命令，但是只打印出年份或者标题。

\supercite{⟨*key*⟩}

该命令只在数值样式中提供，会以不带括号的上标形式打印出引用编号。它使用 \supercitedelim 而不是 \multicitedelim 作为引用定界符。请注意，任何的 ⟨*prenote*⟩ 和 ⟨*postnote*⟩ 选项都会被忽略。如果给出的话也会弃掉它们并且显示警告信息。

3.8.3 合格的引用列表

本宏包支持一类特殊的称之为“多重引用”的引用命令。这些命令的特点是，它们的选项是一列引用，其中每一项都具有完整的前注/后注形式。这对于带括号或脚注中给出的引用特别有用。也可以将一个前注或后注分配给整个列表。这种多重引用命令构建在 \parencite 和 \footcite 等后端命令之上。引用样式通过命令 \DeclareMultiCiteCommand（见 § 4.3.1 节）提供了多重引用的定义。下面的例子展示了多重引用命令的语法。

```
\parencites[35]{key1}[88--120]{key2}[23]{key3}
```

选项的格式与常规引用命令类似，不过只用给出一个引用命令。如果对于列表中的某一项只给出一个可选参数，那么将视为后注。如果只想要前注而不要后注，那么需要将相应项的第二个可选参数设置为空：

```
\parencites[35]{key1}[chapter 2 in][]{key2}[23]{key3}
```

此外，整个的引用列表也可以有一个共同的前注或后注。与其他可选参数的语法不同，这些全局注记在圆括号而不是通常的方括号中：

```
\parencites(and chapter 3)[35]{key1}[78]{key2}[23]{key3}
\parencites(Compare)()[35]{key1}[78]{key2}[23]{key3}
\parencites(See)(and the introduction)[35]{key1}[78]{key2}[23]{key3}
```

请注意，多重引用命令会一直扫描选项，直到遇到一个不是可选或必选参数的字符记号。如果文本中接着多重引用命令的是一个左圆括号或方括号，那么需要手动在最后一个有效参数后添加 \relax 命令或者控制空格（跟在反斜线后的空格）作为标记。这样才会停止该命令的扫描。

```
\parencites[35]{key1}[78]{key2}\relax[...]
\parencites[35]{key1}[78]{key2}\␣{...}
```

默认情况下，本宏包提供了如下一些多重引用命令，分别对应于 §§ 3.8.1 和 3.8.2 节中常规命令：

\cites(⟨*multiprenote*⟩)(⟨*multipostnote*⟩)[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}...[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}

\Cites(⟨*multiprenote*⟩)(⟨*multipostnote*⟩)[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}...[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}

\cite 和 \Cite 的多重引用版本。

\parencites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}...[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}
\Parencites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}...[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

\parencite 和 \Parencite 的多重引用版本。

\footcites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}...[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}
\footcitetexts(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}...[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

\footcite 和 \footcitetext 的多重引用版本。

\smartcites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}...[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}
\Smartcites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}...[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

\smartcite 和 \Smartcite 的多重引用版本。

\textcites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}...[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}
\Textcites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}...[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

\textcite 和 \Textcite 的多重引用版本。

\supercites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}...[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

\supercite 的多重引用版本。该命令只由数值样式提供。

3.8.4 样式无关命令

有时我们需要源文件中的引用格式不依赖于某种特定的引用样式，而是可以在导言区中全局修改。通过导入不同的引用样式我们可以很容易地改变引用格式。但是，当使用 \parencite 或 \footcite 等命令时，引用和正文结合的方式仍然是硬编码的（hard-coded）。\autocite 命令的想法是提供高层次上的引用标记，使得可以全局地切换行内引用和脚注引用（或上标引用）。\autocite 命令构建在 \parencite 和 \footcite 等后端命令之上。由引用样式通过 \DeclareAutoCiteCommand （见 § 4.3.1 节）来定义。该定义可以通过 § 3.1.2.1 节中的 autocite 宏包选项启用。引用样式通常会将该宏包选项初始化为适合该样式的值，详见 § 3.3.1。请注意，这种高层次引用标记有一些限制。例如，行内的作者-年份引用格式通常和正文结合得很紧密，事实上不可能自动转化为脚注。只有当正常地使用 \parencite 或 \footcite（或者数值样式中的 \supercite）时，\autocite 命令才是可用的。引用应当在句子或其部分的末尾、标点符号之前给出，并且在语法上不应当是句子的一部分（例如 \textcite）。

\autocite[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}
\Autocite[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

与其它引用命令不同，\autocite 命令不仅扫描跟在最后一个选项的标点符号来避免双重标点，还在必要时挪动标点。例如，后面的标点在 autocite=footnote 选项下会被移动使得脚注标记打印在标点之后。\Autocite 类似于 \autocite，不同之处在于，如果引用样式打印全名并且有姓名前缀，并且启用了 useprefix 选项，那么引用中的姓名前缀会大写。

\autocite*[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}
\Autocite*[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

\autocite 带星号的变种没有什么不同。星号只是传递给后端命令。例如，如果 \autocite 配置使用 \parencite，那么 \autocite* 就会执行 \parencite*。

\autocites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}...[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}
\Autocites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}...[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

\autocite 的多重引用版本。它同样会在必要时检测和移动标点。请注意它没有带星号的变种。\Autocites 类似于 \autocites，不同之处在于，如果引用样式打印全名并且有姓名前缀，并且启用了 useprefix 选项，那么引用中的姓名前缀会大写。

3.8.5 文本命令

下列命令由 biblatex 内核提供，在文本流中使用。请注意，这里所有的文本命令都与引用追踪无关。

\citeauthor[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}
\citeauthor*[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}
\Citeauthor[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}
\Citeauthor*[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

这些命令打印出作者。严格地讲，打印出的是 labelnames 列表，这可以是 author、editor 或者 translator 等域。\Citeauthor 类似于 \citeauthor，不同之处在于，如果有姓名前缀并且启用 useprefix 选项的话，引用中的名字前缀采用大写形式。带星号的版本会强制设置 maxcitenames 为 1，这样只会打印标签名称列表中第一个姓名（如果有更多名字的话会后接“et al”字符串）。当涉及单一作者的文章时，这会使行文更加自然；否则的话（多位作者时），使用 \citeauthor 会产生名称列表（当然是复数的）。

\citetitle[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}
\citetitle*[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

该命令会打印出标题。它会在可用时使用 shorttitle 域中的短标题，否则的话会使用备用的 title 域中的标题全称。带星号的版本总会打印出标题全称。

\citeyear[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}
\citeyear*[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

该命令会打印出年份（year 域或者 date 域中的年份成分）。带星号的版本会包括可能有的 extrayear 信息。

\citedate[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}
\citedate*[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

该命令会打印出日期（date 或 year 域）。带星号的版本会包括可能有的 extrayear 信息。

\citeurl[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

该命令打印 url 域。

\parentext{⟨text⟩}

该命令将 ⟨text⟩ 装入上下文相关的圆括号中。

\brackettext{⟨text⟩}

该命令将 ⟨text⟩ 装入上下文相关的方括号中。

3.8.6 特殊命令

以下特殊命令同样由 biblatex 内核提供。

\nocite{⟨key⟩}
\nocite{*}

该命令类似于 LaTeX 中的 \nocite 命令。它将没有引用的条目 ⟨key⟩ 添加到参考文献中。如果 ⟨key⟩ 是星号，bib 文件中的所有可用条目都会被添加到参考文献中。与其它引用命令一样，正文中的 \nocite 是在可能的 refsection 环境中的局部命令。与标准的 LaTeX 不同的是，\nocite 也可以在导言区中使用，此时参考文献将归到第零参考文献分节中。

\fullcite[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

该命令针对相应的条目类型使用参考文献驱动，从而创建类似于文献条目的完整引用格式。当然，这样的话就与参考文献样式相关而不是与引用样式相关。

\footfullcite[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

类似于 \fullcite，但是会将整个引用放在脚注中并在末尾添加句号。

\volcite[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
\Volcite[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}

这些命令类似于 \cite 和 \Cite，不过用于多卷作品以卷数和页码样式的引用。与 ⟨postnote⟩ 不同，命令中有一个必选的 ⟨volume⟩ 和一个可选的 ⟨page⟩ 选项。这些命令实际上是样式无关的，因为 ⟨volume⟩ 和 ⟨page⟩ 选项构成后注并且将其以 ⟨postnote⟩ 选项传递给由引用样式提供的 \cite 命令。卷数部分的格式由域格式指令 volcitevolume 直接控制，而页码或文本部分的格式由域格式指令 volcitepages （§ 4.10.4）直接控制。卷数部分与页码部分之间的定界符可用通过重新定义 \volcitedelim 宏（见 § 4.10.1 节）来修改。

\volcites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
...[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
\Volcites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
...[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}

\volcite 和 \Volcite 命令的多重引用版本。

\pvolcite[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
\Pvolcite[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}

类似于 \volcite，不过基于 \parencite 命令。

\pvolcites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
...[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
\Pvolcites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
...[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}

\pvolcite 和 \Pvolcite 相应的多重引用版本。

\fvolcite[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
\ftvolcite[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}

类似于 \volcite 但是分别基于 \footcite 和 \footcitetext。

\fvolcites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
...[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
\Fvolcites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
...[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}

\fvolcite 和 \Fvolcite 相应的多重引用版本。

\svolcite[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
\Svolcite[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}

类似于 \volcite 但是基于 \smartcite。

\svolcites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
...[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
\Svolcites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
...[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}

\svolcite 和 \Svolcite 相应的多重引用版本。

\tvolcite[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
\Tvolcite[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}

类似于 \volcite 但是基于 \textcite。

\tvolcites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
...[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
\Tvolcites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
...[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}

\tvolcite 和 \Tvolcite 相应的多重引用版本。

\avolcite[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
\Avolcite[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}

类似于 \volcite 但是基于 \autocite。

\avolcites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
...[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
\Avolcites(⟨multiprenote⟩)(⟨multipostnote⟩)[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}
...[⟨prenote⟩]{⟨volume⟩}[⟨page⟩]{⟨key⟩}

\avolcite 和 \Avolcite 相应的多重引用版本。

\notecite[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}
\Notecite[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

这些命令打印出 ⟨*prenote*⟩ 和 ⟨*postnote*⟩ 选项但是没有引用部分。与之不同的是，\nocite 命令用于每个 ⟨*key*⟩。主要的应用场合是，用户想要隐式地包含引用条目，但只给出没有在之前行文中提到的信息，同时仍想利用自动的 ⟨*postnote*⟩ 格式以及隐式的 \nocite 功能。这是一般的、样式无关的引用命令。一些特殊的引用样式可以针对相同目的提供更加智能的工具。其中，\Notecite 命令会强制大写（请注意，只有当注记以 biblatex 标点追踪器相关的命令开始时，该命令才是可用的）。

\pnotecite[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}
\Pnotecite[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

类似于 \notecite 但是注记打印在括号中。

\fnotecite[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}

类似于 \notecite 但是注记打印在脚注中。

3.8.7 底层命令

以下命令同样由 biblatex 内核提供，向所有列表和域提供底层接口。

\citename[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}[⟨format⟩]{⟨name list⟩}

⟨*format*⟩ 是由 \DeclareNameFormat 定义的格式指令。格式指令的讨论在 § 4.4.2 节。如果忽略可选参数，该命令调用备选的 citename 格式。最后一个选项是 ⟨*name list*⟩ 的名称（其意义在 § 2.2 节中解释）。

\citelist[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}[⟨*format*⟩]{⟨*literal list*⟩}

⟨*format*⟩ 是由 \DeclareListFormat 定义的格式指令。格式指令的讨论在 § 4.4.2 节。如果忽略可选参数，该命令调用备选的 citelist 格式。最后一个选项是 ⟨*literal list*⟩ 的名称（其意义在 § 2.2 节中解释）。

\citefield[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}[⟨*format*⟩]{⟨*field*⟩}

⟨*format*⟩ 是由 \DeclareFieldFormat 定义的格式指令。格式指令的讨论在 § 4.4.2 一节。如果忽略可选参数，该命令调用备选的 citefield 格式。最后一个选项是 ⟨*field*⟩ 的名称（其意义在 § 2.2 节中解释）。

3.8.8 其它命令

本节中的命令是一些与引用相关的小工具。

\citereset 该命令重新设置引用样式。典型应用的场景为，样式使用 *ibidem*、*idem*、*op. cit.* 等缩略语代替重复引用，而你想在新的章节开始或别的地方强制改为引用全称。该命令执行由 § 4.3.1 节的 \InitializeCitationStyle 命令所定义的样式相关初始化钩子（hook）。它同样重新设置了本宏包的内部引用追踪器。该重置会影响 § 4.6.2 节讨论的 \ifciteseen、\ifentryseen、\ifciteibid 和 \ifciteidem 等测试。当在 refsection 环境内部使用时，引用追踪器的重置仅局部在当前 refsection 环境中。另见 § 3.1.2.1 节的 citereset 宏包选项。

\citereset* 类似于 \citereset 但只执行样式初始化钩子，而不重置内部引用追踪器。

\mancite 如果你想混合自动生成引用和手工引用，可以使用该命令来标记手工插入的引用。如果引用样式使用 *ibidem* 等可能含糊不清或引起歧义的缩略语代替重复引用，那么该命令特别有用。应当总是在相同的场合使用 \mancite 作为手工引用，例如，如果引用在脚注中给出，那么在脚注中加入 \mancite。\mancite 命令会执行 § 4.3.1 节的 \OnManualCitation 命令定义的样式相关重置钩子。它也会重置本宏包的内部“ibidem”和“idem”追踪器。该重置会影响 § 4.6.2 节讨论的 \ifciteibid 和 \ifciteidem 等测试。

\pno 该命令在引用命令的 ⟨*postnote*⟩ 选项中强制开始单页前缀。更多细节和使用说明见 § 3.13.3 节。请注意，该命令只能用于引用和参考文献中局部。

\ppno 类似于 \pno 但是强制为区间前缀。更多细节和使用说明见 § 3.13.3 节。请注意，该命令只能用于引用和参考文献中局部。

\nopp 类似于 \pno 但是抑制所有前缀。更多细节和使用说明见 § 3.13.3 节。请注意，该命令只能用于引用和参考文献中局部。

\psq 在引用命令的 ⟨*postnote*⟩ 选项中，该命令在只给出开始页之处表示两页的范围。更多细节和使用说明见 § 3.13.3 节。打印出的后缀是本地化字符串 sequens（见 § 4.9.2 节）。在后缀和页码之间的空格可以通过重定义 \sqspace 修改。默认值是一个不可打断的词间空格。请注意，该命令只能用于引用和参考文献中局部。

\psqq 类似于 \psq 但表示一个不包括结束页的页码范围。更多细节和使用说明见 § 3.13.3 节。打印出的后缀是本地化字符串 sequens（见 § 4.9.2 节）。请注意，该命令只能用于引用和参考文献中局部。

\RN{⟨*integer*⟩}

该命令将一个整数打印为大写罗马数字形式。可以通过重定义 \RNfont 宏来修改用于该数值的格式。

\Rn{⟨*integer*⟩}

类似于 \RN 但是打印出小写罗马数字。可以通过重定义 \Rnfont 宏来修改用于该数值的格式。

3.8.9 natbib 兼容命令

本宏包的 natbib 选项会导入 natbib 兼容性模块。该模块定义了由 natbib 宏包提供的引用命令的别名。其中包括核心引用命令 \citet 和 \citep 以及 \citealt 和 \citealp 等变形命令。另外也支持这些命令带星号的变种，从而可以打印出完整的作者列表。\cite 命令在 natbib 宏包中经过了特殊处理，但在这里不会专门处理。同时，也支持文本命令（\citeauthor、\citeyear等）以及所有首字母大写的命令（\Citet、\Citep、\Citeauthor 等）。另外，\defcitealias、\citetalias 和 \citepalias 也是可以使用的。请注意，这些兼容性命令不会模仿 natbib 宏包的引用格式，而仅仅是 biblatex 宏包中功能上等价的 natbib 命令别称。引用格式取决于主引用样式。不过，兼容性样式会调整 \nameyeardelim 命令以匹配 natbib 宏包的默认样式。

3.8.10 mcite 引用命令

宏包的 mcite 选项导入了一个特殊的引用模块，提供了类似于 mcite/mciteplus 的引用命令。严格地讲，该模块提供的是主引用样式命令的封装。例如，以下命令：

```
\mcite{key1,setA,*keyA1,*keyA2,*keyA3,key2,setB,*keyB1,*keyB2,*keyB3}
```

本质上等价于

```
\defbibentryset{setA}{keyA1,keyA2,keyA3}%
\defbibentryset{setB}{keyB1,keyB2,keyB3}%
\cite{key1,setA,key2,setB}
```

由于 \cite 后端命令和通常一样由主引用样式控制，因此 \mcite 命令可以和任何样式兼容。mcite 模块提供了 §§ 3.8.1 和 3.8.2 中标准命令的封装，见表 9。它也支持前注和后注以及所有命令带星号的版本，其参数会传递到后端命令中。例如：

```
\mcite*[pre][post]{setA,*keyA1,*keyA2,*keyA3}
```

标准命令	mcite 命令
\cite	\mcite
\Cite	\Mcite
\parencite	\mparencite
\Parencite	\Mparencite
\footcite	\mfootcite
\footcitetext	\mfootcitetext
\textcite	\mtextcite
\Textcite	\Mtextcite
\supercite	\msupercite

表 8: mcite 命令

会执行：

```
\defbibentryset{setA}{keyA1,keyA2,keyA3}%
\cite*[pre][post]{setA}
```

请注意，mcite 模块不是完全兼容的。它提供的命令与 mcite 宏包的命令在语法和功能上十分相似但并不完全等价。当从 mcite/mciteplus 迁移到 biblatex 时必须更新旧文件。在 mcite 中，引用组的第一个成员也是该组整个的标识符。举个 mcite 手册中的例子，以下的组：

```
\cite{glashow,*salam,*weinberg}
```

包括三个条目，且第一个条目键也同样是整个组的标识符。与此相反，biblatex 条目集是一个实体。因此，它需要分配给该集的唯一的条目键值：

```
\mcite{set1,*glashow,*salam,*weinberg}
```

一旦定义之后，条目集的处理与其它 bib 文件中的常规条目相同。当在 biblatex 中使用 numeric 样式并且启动了 subentry 选项时，甚至可以指向集成员。参考表 9 中的一些例子。也可以重启条目集的原始定义，但这没有必要。不过与 mciteplus 不同的是，重启原始定义的一部分是无效的，需要使用集合的条目键。

3.9 本地化命令

biblatex 宏包提供了“edition”和“volume”等关键词的翻译，并定义日期格式和序数等语言相关的特征。这些定义是自动导入的，可以通过本节所介绍的命令在导言区或配置文件中修改或扩展。

\DefineBibliographyStrings{⟨*language*⟩}{⟨*definitions*⟩}

该命令用于定义本地化字符串。⟨*language*⟩ 选项必须是 babel/polyglossia 可知的语言名，即 26 页的表 2 所列出的标识符。⟨*definitions*⟩ 是 ⟨*key*⟩=⟨*value*⟩ 对，将表达式分配给标识符：

输入	输出	评注
\mcite{set1,*glashow,*salam,*weinberg}	[1]	定义并引用该集合
\mcite{set1}	[1]	该集合随后的引用
\cite{set1}	[1]	常规的 \cite
\mcite{set1,*glashow,*salam,*weinberg}	[1]	冗余，但是允许的
\mcite{glashow}	[1a]	引用集成员
\cite{weinberg}	[1c]	又一次常规的 \cite

表 9: mcite 语法（使用 style=numeric 和 subentry 选项时的样例输出）

```
\DefineBibliographyStrings{american}{%
  bibliography  = {Bibliography},
  shorthands    = {Abbreviations},
  editor        = {editor},
  editors       = {editors},
}
```

默认支持的所有关键字列表在 § 4.9.2 节中给出。请注意，所有的表达式在句子中间时也要像本来一样首字母大写。biblatex 宏包会在必要时自动在句首处将首字母大写。用于标题的表达式的大写方式要与标题匹配。与 \DeclareBibliographyStrings 相反，\DefineBibliographyStrings 覆盖了字符串及其缩写两个版本。详见 § 4.9.1 节。

\DefineBibliographyExtras{⟨*language*⟩}{⟨*code*⟩}

该命令用于调整日期格式和序数等语言相关的特征。⟨*language*⟩ 必须是 babel/polyglossia 可知的语言名。⟨*code*⟩ 可以是任意代码，通常包括 § 3.10.3 中格式命令的重定义。

\UndefineBibliographyExtras{⟨*language*⟩}{⟨*code*⟩}

该命令用于存储被 \DefineBibliographyExtras 修改的命令的原始定义。如果重定义的命令在 § 3.10.3 中，那么没有必要存储之前的定义，因为这些命令总归会被所有的语言模块调整的。

\DefineHyphenationExceptions{⟨*language*⟩}{⟨*text*⟩}

这是 TEX 的 \hyphenation 命令的 LATEX 前端，用于定义断词例外。⟨*language*⟩ 必须是 babel/polyglossia 可知的语言名。⟨*text*⟩ 是空格分开的单词列表，断词点用短横线标记：

```
\DefineHyphenationExceptions{american}{%
  hy-phen-ation ex-cep-tion
}
```

\NewBibliographyString{⟨*key*⟩}

该命令声明了新的本地化字符串，即为命令 \DefineBibliographyStrings 的 ⟨*definitions*⟩ 初始化新的 ⟨*key*⟩。选项 ⟨*key*⟩ 也可以是逗号分隔的键值名列表。默认的键名在 § 4.9.2 节中列出。

3.10 格式命令

本节介绍的命令和工具可以用于调整引用和参考文献的格式。

3.10.1 一般命令和钩子

本小节的命令可以在导言区用 \renewcommand 重定义。页边标记为“Context Sensitive”的命令还可以用 \DeclareDelimFormat 进行定制，并使用 § 3.10.2 节的 \printdelim 打印。请注意，所有以 \mk… 开头的命令都带有一个选项。所有这些命令的定义在 biblatex.def 中。

\bibsetup 在参考文献开始执行的任意代码，用于影响参考文献的页面布局的命令。

\bibfont 在参考文献中用于设置字体的任意代码。类似于 \bibsetup 但用于切换字体。

\citesetup 在引用命令开始执行的任意代码。

\newblockpunct 插入在‘blocks’（其意义在 § 4.7.1 节中解释）之间的分隔符。缺省定义由宏包选项 block（见 § 3.1.2.1）所控制。

\newunitpunct 插入在‘units’（其意义在 § 4.7.1 节中解释）之间的分隔符。这通常是句号或者逗号加上一个词间距。缺省定义是句号加一个空格。

\finentrypunct 在每条文献条目最后打印的标点，通常是句号。缺省定义是句号。

\entrysetpunct 一个条目集中文献子条目之间的标点。默认定义是分号和一个空格。

\bibnamedelima 该定界符控制组成姓名成分之间的空白。它将自动插入在名（first name）之后（如果该成分少于三个字符长度）和最后的姓名成分之前。缺省值是一个词间距加上由计数器 highnamepenalty（§ 3.10.4）的值控制的惩罚项。详见 § 3.13.4 节。

\bibnamedelimb 该定界符插入在 \bibnamedelima 不可用的姓名成分之间。缺省值是一个词间距加上由计数器 lownamepenalty（§ 3.10.4）的值控制的惩罚项。详见 § 3.13.4 节。

\bibnamedelimc 该定界符控制姓名成分之间的空白。如果 useprefix=true，那么它插入在名前缀（name prefix）和姓（last name）之间。缺省值是一个词间距加上由计数器 highnamepenalty（§ 3.10.4）的值控制的惩罚项。详见 § 3.13.4 节。

\bibnamedelimd 该定界符插入在所有的 \bibnamedelimc 不可用的姓名成分之间。缺省值是一个词间距加上由计数器 lownamepenalty（§ 3.10.4）的值控制的惩罚项。详见 § 3.13.4 节。

\bibnamedelimi 该定界符在首字符缩写中取代 \bibnamedelima/b。请注意该情况只用于既定的首字母缩写（例如在 bib 文件中给出的），而不是由 biblatex 自动生成的首字母缩写，后者会使用自己的定界符集。

\bibinitperiod 当没有使用 \bibinithyphendelim 时插入在首字母缩写之后的标点。缺省值是句点（\adddot）。详见 § 3.13.4 节。

\bibinitdelim 当没有使用 \bibinithyphendelim 时多重首字母缩写之间的空白。缺省值是一个不可打断的词间距。详见 § 3.13.4 节。

\bibinithyphendelim 带连字符的姓名成分首字母缩写之间插入的标点，用以替代 \bibinitperiod 和 \bibinitdelim。缺省定义是一个句点后接一个不可打断的连字符。详见 § 3.13.4 节。

\bibindexnamedelima 在索引中代替 \bibnamedelima。

\bibindexnamedelimb 在索引中代替 \bibnamedelimb。

\bibindexnamedelimc 在索引中代替 \bibnamedelimc。

\bibindexnamedelimd 在索引中代替 \bibnamedelimd。

\bibindexnamedelimi 在索引中代替 \bibnamedelimi。

\bibindexinitperiod 在索引中代替 \bibinitperiod。

\bibindexinitdelim 在索引中代替 \bibinitdelim。

\bibindexinithyphendelim 在索引中代替 \bibinithyphendelim。

\revsdnamepunct 当名字反写（姓在前）时姓和名之间的标点。如下是一个使用缺省的逗号作为 \revsdnamedelim 的例子：

```
Jones, Edward
```

对于姓名列表，该命令应当与格式指令中的姓名反写分隔符 \bibnamedelimd 一起使用。详见 § 3.13.4 节。

\bibnamedash 用于替代文献中连续重复的作者或编辑的破折号。缺省值是一个 'em' 或 'en' 长度横线，取决于文献列表的缩进。

\labelnamepunct 在参考文献按字母排序时用在名字之后的分隔符（author，在其未定义时是 editor）。对于缺省样式，该分隔符在此位置代替 \newunitpunct。缺省定义是 \newunitpunct，即对其处理与常规单元标点没有不同之处。

\subtitlepunct 域 title 和 subtitle、booktitle 和 booksubtitle，以及 maintitle 和 mainsubtitle 之间的分隔符。对于缺省样式，该分隔符在此位置代替 \newunitpunct。缺省定义是 \newunitpunct，即对其处理与常规单元标点没有不同之处。

\intitlepunct 在 @article, @inbook, @incollection 等条目类型中单词“in”与之后的标题之间的分隔符。缺省值是一个冒号加上一个词间距（例如，“Article, in: *Journal*”或者“Title, in: *Book*”）。请注意，这是分隔字符串而不仅是标点。如果你不想在“in”后加冒号，\intitlepunct 仍然应当插入一个空格。

\bibpagespunct 域 pages 之前的分隔符。缺省值是逗号加上一个词间距。

\bibpagerefpunct 域 pageref 之前的分隔符。缺省值是一个词间距。

\multinamedelim 在 author 或 editor 等姓名列表中各项之间的定界符（如果有两个以上姓名）。缺省值是一个逗号加上一个词间距。例子请参考 \finalnamedelim。[36]

\finalnamedelim 在姓名列表的最后一个姓名之前用以代替 \multinamedelim 的定界符。缺省值是用词间距分隔的本地化项“and”。这里是一个例子：

```
Michel Goossens, Frank Mittelbach and Alexander Samarin
Edward Jones and Joe Williams
```

第一个例子中的逗号是 \multinamedelim，而这两个例子中的字符串“and”是 \finalnamedelim。另见 § 3.10.3 节中的 \finalandcomma。

\revsdnamedelim 当姓名反序时打印在名（first name）后面的额外定界符。缺省值是空字符串，即没有额外的定界符。这里是一个设置 \revsdnamedelim 为逗号的姓名列表例子：

```
Jones, Edward, and Joe Williams
```

在本例中，“Edward”之后的逗号是 \revsdnamedelim 而字符串“and”是前者之后的 \finalnamedelim。

\andothersdelim 当 author 或 editor 等姓名列表被截断时本地化字符串“andothers”之前的定界符。缺省值是一个词间距。

\multilistdelim 在 publisher 或 location 等文本列表中诸项之间的定界符（如果列表中有两个以上项）。缺省值是一个逗号加上一个词间距。进一步解释参见 \multinamedelim。

\finallistdelim 在文本列表的最后一项之前代替 \multilistdelim 的定界符。缺省值是用词间距分隔的本地化字符串“and”。进一步解释参见 \finalnamedelim。

\andmoredelim 当 publisher 或 location 等文本列表被截断时打印在本地化字符串“andmore”之前的定界符。缺省值是一个词间距。

\multicitedelim 当多个条目键传递给单个引用命令时打印在引用之间的定界符。缺省值是一个分号加上一个词间距。

\supercitedelim 类似于 \multicitedelim 但只用在 \supercite 中。缺省值是一个逗号。

\compcitedelim 类似于 \multicitedelim 但只用于某些“压缩”的多重引用样式。缺省值是一个逗号加上一个词间距。

\datecircadelim 开启全局选项 datecirca 时，在日期格式中本地化“circa”项之后的定界符。缺省值是一个词间距。 Context Sensitive

\dateeradelim 设置全局选项 dateera 时，在日期格式中本地化纪年项之前的定界符。缺省值是一个词间距。 Context Sensitive

\dateuncertainprint 开启全局选项 dateuncertain 并且 \ifdateuncertain 测试为真时打印的日期不确定信息。缺省打印语言相关的 \bibdateuncertain 字符串（§ 3.10.3 节）。

\enddateuncertainprint 开启全局选项 dateuncertain 并且 \ifenddateuncertain 测试为真时打印的日期不确定信息。缺省打印语言相关的 \bibdateuncertain 字符串（§ 3.10.3 节）。

\datecircaprint 开启全局选项 datecirca 并且 \ifdatecirca 测试为真时打印日期约数信息。缺省打印本地化的“circa”项（§ 4.9.2.21 节）和 datecircadelim 定界符。

\enddatecircaprint 开启全局选项 datecirca 并且 \ifenddatecirca 测试为真时打印日期约数信息。缺省打印本地化的“circa”项（§ 4.9.2.21 节）和 datecircadelim 定界符。

\datecircaprintedtf 开启全局选项 datecirca 并且 \ifdatecirca 测试为真时打印 EDTF 格式的日期约数信息。缺省打印 \textasciitilde。

\enddatecircaprintedtf 开启全局选项 datecirca 并且 \ifenddatecirca 测试为真时打印 EDTF 格式的日期约数信息。缺省打印 \textasciitilde。

\dateeraprint yearfield 当全局选项 dateera 设置为 secular 或 christian 时打印的日期纪年信息。缺省情况下打印 dateeradelim 定界符和合适的本地化纪年词语（§ 4.9.2.21 节）。如果设置了 dateeraauto，那么将测试传递过来的 ⟨*yearfield*⟩（year、origyear、endeventyear 等年份域的名称）以确定对应的值是否早于 dateeraauto 阈值。如果更早的话还会输出本地化的 BCE/CE 词语。dateeraauto 的缺省设置为 0，这样只会输出 BCE/BC 本地化字符串。通过探测传递来的 ⟨*yearfield*⟩ 名称确定是否打印开始和结束年份的纪年信息。

\dateeraprintpre 当全局选项 dateera 设置为 astronomical 时打印的日期约数信息。缺省值是 bibdataeraprefix。通过探测传递来的 ⟨*yearfield*⟩ 名称确定是否打印开始和结束年份的纪年信息。

\textcitedelim 类似于 \multicitedelim 但用于 \textcite 和相关命令（§ 3.8.2 节）。缺省值是一个逗号加上一个词间距。标准样式会修改这一临时定义以确保最后一个引用之前的定界符是词间距分隔的本地化字符串“and”。另见 § 3.10.3 节的 \finalandcomma 和 \finalandsemicolon。

\nametitledelim 作者–标题和其它一些详细引用样式中作者/编辑和标题之间的定界符。缺省定义是一个逗号加上一个词间距。 Context Sensitive

\nameyeardelim 作者–年份引用样式中作者/编辑和年份之间的定界符。缺省定义是一个词间距。 Context Sensitive

\namelabeldelim 字母样式和数值样式中姓名/标题和标签之间的定界符。缺省定义是一个词间距。 Context Sensitive

\nonameyeardelim 在作者–年份引用样式中，当某一标签名不存在（标准样式中通常是标签或者标题）时其替代者与年份之间的定界符。仅当没有标签名时使用。这是因为标签名存在时会使用 \nameyeardelim。缺省值是一个词间距。 Context Sensitive

[36]请注意，如果列表中只有两个姓名，那么不会使用 \multinamedelim，此时缺省样式使用 \finalnamedelim。

\labelalphaothers 当作者/编辑数目超过了 maxalphanames 阈值或者 author/editor 列表在 bib 文件中由关键词“and others”截断时，接在 labelalpha 域的非数值部分之后的字符串（即，该域包含了由字母引用样式使用的引用标签）。这通常是加号或者星号等单字符。缺省值是加号。该命令可以通过重定义为空字符串来关闭该特性。任何情况下必须在导言区中重定义。

\sortalphaothers 类似于 \labelalphaothers 但使用在排序过程中。如果 \labelalphaothers 中包含了格式命令，建议将本命令设置为另外一个值，例如：

```
\renewcommand*{\labelalphaothers}{\textbf{+}}
\renewcommand*{\sortalphaothers}{+}
```

如果 \sortalphaothers 没有定义，其缺省值为 \labelalphaothers。

\prenotedelim 引用命令的 ⟨*prenote*⟩ 选项之后打印的定界符。详见 § 3.8 节。默认为一个词间距。

\postnotedelim 引用命令的 ⟨*postnote*⟩ 选项之前打印的定界符。详见 § 3.8 节。默认为一个词间距。

\extpostnotedelim 在引用命令中，当后注出现在引用括号外时，引用和带括号的 ⟨*postnote*⟩ 选项之间的定界符。在标准样式中，如果引用使用条目的 shorthand 域就会出现这一现象。详见 § 3.8 节。默认为一个词间距。

\mkbibname⟨*namepart*⟩{⟨*text*⟩}该命令带有一个选项，用于姓名列表域的姓名成分 ⟨*namepart*⟩ 的格式。默认数据模型定义了姓名成分 ‘family’, ‘given’, ‘prefix’ 和 ‘suffix’，因此自动定义了以下宏命令：

```
\mkbibnamefamily
\mkbibnamegiven
\mkbibnameprefix
\mkbibnamesuffix
```

出于对传统 BibTeX 姓名成分的向后兼容性也定义以下宏命令。这些宏会生成警告并设置正确的宏：

```
\mkbibnamelast
\mkbibnamefirst
\mkbibnameaffix
```

\relatedpunct 在 relatedtype 文献的本地化字符串和第一个相应条目数据之间的分隔符。如下是将 \relatedpunct 设置为短横线的例子：

```
A. Smith. Title. 2000, (Orig. pub. as-Origtitle)
```

`\relateddelim` 多重相关条目数据之间的分隔符。缺省定义是一个可选的点号加上一个断行。如下是一个五卷作品的例子，其中卷 A-E 是相关条目：

```
Donald E. Knuth. Computers & Typesetting. 5 vols. Reading, Mass.:
    ↪ Addison-
Wesley, 1984-1986.
Vol. A: The TEXbook. 1984.
Vol. B: TEX: The Program. 1986.
Vol. C: The METAFONTbook. By. 1986.
Vol. D: METAFONT: The Program. 1986.
Vol. E: Computer Modern Typefaces. 1986.
```

`\relateddelim`⟨*relatedtype*⟩ ⟨*relatedtype*⟩ 类型的相关联条目中，多重关联条目数据之间的分隔符。没有缺省值。如果这样的类型相关的定界符不存在，那么将使用 `\relateddelim`。

3.10.2 Context-sensitive 定界符

§ 3.10.1 节介绍的定界符是全局定义的。也就是说，无论在哪里使用，结果都是相同的。然而，如果想要在不同语境（context）中打印不同的效果，这些定界符就不可取了。这里“context”可以是“引用文本内部”或者“参考文献项”等。基于此，biblatex 还提供了更为智能的定界符规范和用户接口（通过 `\newcommand` 定义的常规宏）。

`\DeclareDelimFormat`[⟨*context, ...*⟩]{⟨*name, ...*⟩}{⟨*code*⟩}
`\DeclareDelimFormat*`[⟨*context, ...*⟩]{⟨*name, ...*⟩}{⟨*code*⟩}

在逗号分隔列表 ⟨*names*⟩ 中使用替代测试 ⟨*code*⟩ 声明定界符宏。如果给出了可选的逗号分隔列表 ⟨*context*⟩，那么只为相应的 context 声明 ⟨*names*⟩。如果没有可选的 ⟨*context*⟩，那么被定义的 ⟨*names*⟩ 与由 `\newcommand` 给出的全局定界符的定义是一致的——仅仅是一个简单的宏，用于替代 `\name`。不过，通过使用能探测 context 的 `\printdelim` 命令，仍然可以调用以这种方式定义的定界符宏。带星号的命令会首先清楚所有 ⟨*names*⟩ 中的 ⟨*context*⟩ 声明。

`\printdelim`[⟨*context*⟩]{⟨*name*⟩}

打印名为 ⟨*name*⟩ 的定界符，首先建立可选的局部 ⟨*context*⟩。如果没有可选的 ⟨*context*⟩，那么 `\printdelim` 会使用当前活动的定界符 context。

定界符文境是一个简单的字符串，内部宏 `\blx@delimcontext` 的值。后者可以通过 `\delimcontext` 命令手动设置。

`\delimcontext`{⟨*context*⟩}

设置定界符文境为 ⟨*context*⟩。该设置不是全局的，因此定界符 context 可以使用通常的 LaTeX 组方法进行嵌套。

\DeclareDelimcontextAlias{⟨*alias*⟩}{⟨*name*⟩}

基于传递给 \DeclareCiteCommand 的引用命令（“parencite”、“textcite”等）的名称，定界符系统会创建定界符 context。在某些情况下引用命令存在嵌套定义，此时 \DeclareCiteCommand 会调用自己（相关示例见 authoryear-icomp 中 \textcite 的定义）。此时的 context 一般不准确，从而无法使用定界符规范。例如，\textcite 的定义实际上定义并使用了 \cbx@textcite，因此当打印引用时 context 会自动设置为 cbx@textcite。因此期望 context 为 textcite 的定界符定义就无法运行。为此，该命令可以将 ⟨*alias*⟩ 作为 ⟨*name*⟩ 的别称，例如：

```
\DeclareDelimcontextAlias{cbx@textcite}{textcite}
```

这会确保 context 在 \cbx@textcite 引用命令内部时就是预期的 textcite。

biblatex 在不同位置自动创建了若干个默认的 context：

none 文档开始处。

bib 以 \printbibliography 开始的参考文献内部或者 \usedriver 内部。

biblist 以 \printbiblist 开始的参考文献列表内部

‘citecommand’ 使用 \DeclareCiteCommand 定义的 \citecommand 引用命令内部

例如，\nametitledelim 的默认设置为：

```
\DeclareDelimFormat{nametitledelim}{\addcomma\space}
\DeclareDelimFormat[textcite]{nametitledelim}{\addspace}
```

这意味着对于每个标准定界符接口，\nametitledelim 全局定义为 \addcomma\space。此外，总体上使用 \printdelim 可以打印该定界符为 \nametitledelim，不过在 \textcite 内部会打印 \addspace，这对于行文更适合。如果需要的话，可以强制添加 context 作为\printdelim 的可选项，例如：

```
\printdelim[textcite]{nametitledelim}
```

无论 \printdelim 周围的内容怎样，总会打印出 \addspace。由于 context 是任意字符串，因此可以在任何时刻使用 \delimcontext 构建。如果 \printdelim 在当前 context 没有找到定界符 ⟨*name*⟩ 的特定值，那么就直接打印出 \name。这意味着样式作者可以使用 \printdelim。同时希望使用 \renewcommand 重定义定界符的用户也可以这样做。不过前提是，该定义不能改变任何 context 相关的定界符，如下所示：

```
\DeclareDelimFormat{delima}{X}
\DeclareDelimFormat[textcite]{delima}{Y}
\renewcommand*{\delima}{Z}
```

这里，\delima 总会打印“Z”。同时，\printdelim{delima} 在任何“textcite”之外的 context 中也会打印 \delima 也就是“Z”，而在“textcite”中则打印“Y”。更多信息参考 biblatex 附带的示例文件 04-delimiters.tex。

3.10.3 语言相关命令

本节中的命令是与语言相关的。因此重定义时需要将新的定义包裹在 .lbx 文件的 \DeclareBibliographyExtras 命令或者用户文件的 \DefineBibliographyExtras 命令中，详见 § 3.9 节。请注意所有以 \mk… 开头的命令都带有一个或更多选项。

\bibrangedash 用于数字范围的横线。默认值为 \textendash。

\bibrangessep 用于多重范围之间的分隔符。默认为一个逗号加一个空格。

\bibdatesep 在短日期格式用于日期成分之间的分隔符。默认为 \hyphen。

\bibdaterangesep 用于日期范围的分隔符。对于 ymd 格式默认为 \slash，对于其它日期格式默认为 \textendash。日期格式选项 edtf 则硬编码为 \slash，因为这是标准兼容格式。

\mkbibdatelong 以对应于三个日期成分（以年/月/日的顺序）的三个域名作为选项，并在长日期格式中使用相应值来打印日期。

\mkbibdateshort 类似于 \mkbibdatelong 但是使用短日期格式。

\mkbibtimezone 修改传入的时区字符串作为唯一选项。默认情况会将“Z”改为 \bibtimezone 的值。

\bibdateuncertain 当启用全局选项 dateuncertain 时用于不确定日期之后的标识符。默认为空格加一个问号。

\bibdateeraprefix 当设置 dateera 选项为 astronomical 时，在日期范围中公元前日期的前缀标识符。命令 \textminus 如果有定义则为默认值，否则 \textendash 为默认值。

\bibdateeraendprefix 当设置 dateera 选项为 astronomical 时，在日期范围中公元前日期结束的标识符。当 \bibdaterangesep 选项设置为短横线时默认值为窄间距（thin space）加上 \bibdateeraprefix，否则默认值为 \bibdateeraprefix。

\bibtimesep 分隔时间成分的标识符，默认为冒号。

\bibtimezonesep 分隔时间与可选的时区信息的标识符。默认为无。

\bibtzminsep 偏移时区的小时和分钟信息直接的分隔符。默认值为 \bibtimesep。

\bibdatetimesep 当打印时间成分与日期成分时二者之间的分隔符（见 § 3.1.2.1 节的 ⟨*datetype*⟩dateusetime 选项）。对于非 EDTF 输出格式，默认值为空格；对于 EDTF 输出格式，默认值为“T”。

\finalandcomma 当在对应语言中可用时在列表的最后一个“and”之前打印逗号。这里是一个例子：

```
Michel Goossens, Frank Mittelbach, and Alexander Samarin
```

\finalandcomma 是单词“and”之前的逗号。另见 § 3.10.1 节的 \multinamedelim、\finalnamedelim、\textcitedelim 和 \revsdnamedelim。

\finalandsemicolon 当在对应语言中可用时在列表的最后一个“and”之前打印分号。例如：

```
Goossens, Mittelbach, and Samarin; Bertram and Wenworth; and Knuth
```

\finalandsemicolon 是单词“and”之前的分号。另见 § 3.10.1 节的 \textcitedelim。

\mkbibordinal{⟨*integer*⟩}

该命令将一个整数作为选项，打印出一个序数。

\mkbibmascord{⟨*integer*⟩}

类似于 \mkbibordinal，但在对应语言中可用时打印出阳性序数。

\mkbibfemord{⟨*integer*⟩}

类似于 \mkbibordinal，但在对应语言中可用时打印出阴性序数。

\mkbibneutord{⟨*integer*⟩}

类似于 \mkbibordinal，但在对应语言中可用时打印出中性序数。

\mkbibordedition{⟨*integer*⟩}

类似于 \mkbibordinal，但与单词“edition”一起使用。

\mkbibordseries{⟨*integer*⟩}

类似于 \mkbibordinal，但与单词“series”一起使用。

3.10.4 长度和计数器

本节中的长度寄存器和计数器可以在导言区中分别用 \setlength 和 \setcounter 来修改。

\bibhang 参考文献的悬挂缩进（当使用时）。该长度在导入时初始化为 \parindent。

\biblabelsep 参考文献中条目和相应标签之间的水平距离。这只应用于 numeric 和 alphabetic 等打印标签的参考文献样式。该长度在导入时初始化为 \labelsep 值的两倍。

\bibitemsep 参考文献中每一条目之间的垂直间距。该长度在导入时初始化为 \itemsep。请注意 \bibitemsep、\bibnamesep 和 \bibinitsep 服从 \addvspace 的规则，也就是，当这些命令中任何一个直接在另外一个之后引入垂直间距时，所得到的总间距是其中的最大值。

\bibnamesep 参考文献中插入在两条姓名不同的条目之间的垂直间距。缺省值是零。将该值设为大于 \bibitemsep 会使得参考文献按照作者/编辑名分组。请注意 \bibitemsep、\bibnamesep 和 \bibinitsep 服从 \addvspace 的规则，也就是，当这些命令中任何一个直接在另外一个之后引入垂直间距时，所得到的总间距是其中的最大值。

\bibinitsep 参考文献中插入在两条首字母不同的条目之间的垂直间距。缺省值是零。将该值设置为大于 \bibitemsep 会使得参考文献按字母分组。请注意 \bibitemsep、\bibnamesep 和 \bibinitsep 服从 \addvspace 的规则，也就是，当这些命令中任何一个直接在另外一个之后引入垂直间距时，所得到的总间距是其中的最大值。

\bibparsep 参考文献中条目内部的段间距。缺省值为零。

abbrvpenalty 该计数器在本地化模块中使用，用于设定本地化字符串中缩写和短语中使用的惩罚值。例如，“et al”或“ed. by”等短语中的断行是不美观的，但为了防止盒子溢出仍然应当可以使用。该计数器在导入时初始化为 \hyphenpenalty。断行考虑的原则是，使得 TEX 将整个语句看做是单个可以用连字符断行的单词。如果你不喜欢相应的断行，可以设置为更高的值。如果你不介意这些效果，可以设置为零。如果你想无条件地取消这种效果，可以设置为“无穷”（10 000 或更高）。[37]

highnamepenalty 该计数器设定了影响姓名中断行的惩罚值。其解释见 §§ 3.13.4 和 3.10.1 节。该计数器在导入时初始化为 \hyphenpenalty。如果你不喜欢相应的断行，可以设置为更高的值。如果你不介意这些效果，可以设置为零。如果你更喜欢传统的 BIBTEX 样式（在 highnamepenalty 处没有断行），可以设置为“无穷”（10 000 或更高）。

lownamepenalty 类似于 highnamepenalty。其解释见 §§ 3.13.4 和 3.10.1 节。该计数器在导入时初始化为 \hyphenpenalty 的一半。如果你不喜欢相应的断行，可以设置为更高的值。如果你不介意这些效果，可以设置为零。

3.10.5 通用命令

本节中的命令是通用文本命令，除了引用和参考文献之外在一般情况下都可以使用。

\bibellipsis 带有方括号的省略号：“[...]”。

\noligature 在该位置取消连字并增加一些空格。使用该命令来打断标准中“fi”、“fl”等连字。类似于 babel/polyglossia 宏包中一些语言模块提供的 "| 缩写。

\hyphenate 条件连字号。与标准的 \- 命令不同，该命令允许在单词的剩余部分使用连字号。类似于 babel/polyglossia 宏包中一些语言模块提供的 "- 缩写。

\hyphen 用于复合词的显式可断连字号。与文本“-”不同，该命令允许在单词的剩余部分使用连字号。类似于 babel/polyglossia 宏包中一些语言模块提供的 "= 缩写。

[37] 这里很慎重地将 abbrvpenalty、lownamepenalty 和 highnamepenalty 的缺省值设定得非常低以防止盒子溢出。这意味着，如果文本设置合理，那么你几乎不会注意到断行的影响。如果你将这些值设置为 10 000 以取消相应断点，那么你就会注意到它们的影响，不过同时你也许要面对盒子溢出现象。需要注意的是，参考文献中的断行往往比正文中更困难，而且你不能通过换一种表达方式来解决。在某些情况下，在整个参考文献中设置 \raggedright 来阻止非最佳的断行往往更好。此时，即使是相对低的缺省惩罚项也会造成不同效果。

\nbhyphen 用于复合词的显式不可断连字号。与文本“-”不同，该命令不允许在连字号处断行但仍然允许在单词的剩余部分使用连字号。类似于 babel/polyglossia 宏包中一些语言模块提供的 "~ 缩写。

\nohyphenation 局部抑制连字号的一般切换命令。正常情况下其作用范围应被限定在一个组内。

\textnohyphenation{⟨*text*⟩}

类似于 \nohyphenation 但是限制在 ⟨*text*⟩ 选项中。

\mknumalph{⟨*integer*⟩}

将 1–702 间的整数作为选项并将其转化为如下的字符串：1=a, ..., 26=z, 27=aa, ..., 702=zz。这用于 extrayear 和 extraalpha 等域的格式设置。

\mkbibacro{⟨*text*⟩}

在可用时使用当前字体的小型大写变体排版首字母缩写的一般性命令，否则依原样排版。首字母缩写应当以大写字母形式给出。

\autocap{⟨*character*⟩}

如果 biblatex 的标点追踪能够在当前位置将本地化字符串大写，该命令自动将 ⟨*character*⟩ 转化为大写形式。该命令是鲁棒的。在条目的某些字符串需要给定条件下的大写时该命令是很有用的。请注意，⟨*character*⟩ 选项是以小写形式给出的单字符。例如：

```
\autocap{s}pecial issue
```

将产生合适的“Special issue”或者“special issue”。如果被大写的字符串以 Ascii 记号给出的变体字符开始，包括如下 ⟨*character*⟩ 选项中的重音命令：

```
\autocap{\'e}dition sp\'eciale
```

这会生成“Édition spéciale”或者“édition spéciale”。如果大写的字符串以能打印出字符的命令开始，例如 \ae 或 \oe，只要该命令放入 ⟨*character*⟩ 选项即可：

```
\autocap{\oe}uvres
```

这会生成“Œuvres”或“œuvres”。

3.11 特定语言注记

本节讨论的功能特定于某些本地化模块。

3.11.1 美式英文

美式英文本地化模块使用 § 4.7.5 节的 \uspunctuation 来激活“美式”标点。如果启用该特性，所有在 \mkbibquote 之后的逗号和句号会前移到引号内。如果想要关闭该特性，使用如下的 \stdpunctuation：

```
\DefineBibliographyExtras{american}{%
  \stdpunctuation
}
```

缺省情况下，“美式标点”特性只由 american 本地化模块启用。以上代码只在你想要不带美式标点的美式英文本地化时需要。由于标准的标点是宏包缺省的，对于其它语言这会是多余的。

为了避免可能的混淆，在导入 babel/polyglossia 宏包时，强烈建议总是指明 american、british、australian 等而不是 english。老版本的 babel 宏包过去将 english 作为 british 的别名；而更近的版本中将其作为 american 的别名。除了以上只在需要显式指明 american 时才启用的特性，biblatex 宏包本质上将 english 作为 american 的别名。

3.11.2 西班牙文

在本宏包中，处理西班牙文的单词“and”比其它语言更困难，因为可以是“y”或“e”，这取决于下个单词的第一个音节。因此，西班牙文的本地化模块不使用本地化字符串“and”，而是使用特殊的内部“智能 and”命令。该命令的行为由 smartand 计数器控制。

smartand 该计数器控制内部“智能 and”命令的行为。当设置为 1 时，取决于语境会打印“y”或“e”。当设置为 2 时，总是打印“y”。当设置为 3 时，总是打印“e”。当设置为 0 时则禁用“智能 and”特性。该计数器在导入时初始化为 1，可以在导言区中修改。请注意，将该计数器设置为一个正整数则说明西班牙文的本地化模块会忽略 \finalnamedelim 和 \finallistdelim。

\forceE 如果 biblatex 在某个错误的名字前得到“and”，那么可以在 bib 文件中使用该命令。如同该命令的名字，该命令会强制输出“e”。为了获得正确的 BibTeX 数据文件格式，必须以特定方式使用该命令。例如：

```
author = {Edward Jones and Eoin Maguire},
author = {Edward Jones and {\forceE{E}}oin Maguire},
```

请注意，相应姓名成分的首字母作为 \forceE 的选项，然后整个地放在额外一对花括号中。

\forceY 类似于 \forceE 但是强制输出“y”。

3.11.3 希腊文

希腊文本地化模块需要 UTF-8 支持，与其它编码不兼容。一般来说，biblatex 宏包与 inputenc 宏包以及 XeLaTeX 都兼容。而 ucs 宏包不可用。由于 inputenc 的标准 utf8 模块缺失一部分希腊语字形映射，因此希腊文用户可以选择 XeLaTeX 。请注意，用户仍需要载入额外宏包来设置希腊文字体。根据经验，常规希腊文的文档设置一般也应当可以使用 biblatex 。然而有一个根本性限制：biblatex 不支持切换语言。参考文献中可以出现希腊文标题，但英文和其它标题可能会渲染为希腊字母。如果需要多语言的参考文献，使用 XeLaTeX 是明智的选择。

3.11.4 俄文

与希腊文模块类似，俄文模块同样需要 UTF-8 支持。与其它编码不兼容。

3.12 使用注记

以下几节讨论了 biblatex 宏包的基本概述以及一些典型的使用场合。

3.12.1 概述

使用 biblatex 宏包与传统的 BibTeX 样式和相关宏包稍有不同。因此，在讨论具体的使用场景之前，我们首先要看一下典型的文件结构：

```
\documentclass{...}
\usepackage[...]{biblatex}
\addbibresource{bibfile.bib}
\begin{document}
\cite{...}
...
\printbibliography
\end{document}
```

在传统的 BibTeX 下，\bibliography 命令提供了两个目的：标记文献的位置并且确定 bib 文件。文件扩展名是省略的。而在 biblatex 下则在导言区通过 \addbibresource 使用文件名全称（带有 .bib 后缀）来确定文献资源。文献的打印则使用 \printbibliography 命令，而且该命令可以使用多次（详见 § 3.7 节）。文档正文可以包含任意多个引用命令（§ 3.8 节）。处理示例文件需要以下若干步骤。假设我们的示例文件叫 example.tex，参考文献数据在 bibfile.bib 中，那么过程如下：

1. 对 example.tex 运行 latex 命令。如果该文件包含引用，biblatex 会将有关命令写入辅助文件 example.bcf，进而从 biber 调用相关数据。
2. 对 example.bcf 运行 biber 命令。biber 会从 bibfile.bib 中检索数据，并将其写入辅助文件 example.bib 中，写入的格式可以被 biblatex 处理。

3. 对 example.tex 运行 latex 命令。biblatex 会从 example.bbl 中读取数据并打印所有的引用及参考文献。

3.12.2 辅助文件

biblatex 宏包只使用一个 bcf 辅助文件。即便文件中通过 \include 包含引用命令，你也只需在主 bcf 文件上运行 biber 。biber 需要的全部信息都在 bcf 文件中，包括当使用多重 refsection 环境（见 § 3.12.3 节）时关于所有参考文献分节的信息。

3.12.3 多重文献

在由多位作者所写文章的合集中，例如会议文集的一卷，非常常见的做法是对每篇文章而不是对整本书分别做文献索引。在以下的例子中，每篇文章是不同的一章 \chapter，并带有自己的文献索引。

```
\documentclass{...}
\usepackage{biblatex}
\addbibresource{...}
\begin{document}
\chapter{...}
\begin{refsection}
...
\printbibliography[heading=subbibliography]
\end{refsection}
\chapter{...}
\begin{refsection}
...
\printbibliography[heading=subbibliography]
\end{refsection}
\end{document}
```

如果 \printbibliography 在 refsection 环境内部使用，它会自动将文献列表范围限制在 refsection 环境内。对于在一本书末尾列出但是按照每一章划分的累积参考文献，使用 \printbibliography 的 section 选项来选择参考文献分节，如下面的例子所示。

```
\documentclass{...}
\usepackage{biblatex}
\defbibheading{subbibliography}{%
  \section*{References for Chapter \ref{refsection:\therefsection}}}
\addbibresource{...}
\begin{document}
\chapter{...}
```

```
\begin{refsection}
...
\end{refsection}
\chapter{...}
\begin{refsection}
...
\end{refsection}
\printbibheading
\printbibliography[section=1,heading=subbibliography]
\printbibliography[section=2,heading=subbibliography]
\end{document}
```

请注意上面例子中文献标题的定义。该定义考虑到了参考文献中的子标题。主标题由普通的 `\chapter` 生成。`biblatex` 宏包会自动在每个相应的 `refsection` 环境开始处用标准 `\label` 命令分别设置标签。其标识符是字符串 `refsection:` 接上 `refsection` 环境的序数。当前节的序号可以通过 `refsection` 计数器获得。当使用 `\printbibliography` 的 `section` 选项时，该计数器也被设置为局部的。这意味着你可以在标题定义中使用该计数器来打印类似于上面例子中“References for Chapter 3”这样的子标题。你也可以通过载入 `nameref` 宏包和使用 `\nameref` 代替 `\ref` 来使用相应的章名作为子标题：

```
\usepackage{nameref}
\defbibheading{subbibliography}{%
  \section*{\nameref{refsection:\therefsection}}}
```

为参考文献的每一个子部分都给出 `\printbibliography` 是很繁琐的，所以 `biblatex` 提供了一个缩写语。`\bibbysection` 命令会自动遍历所有的参考文献分节。这等价于为每节给出一个 `\printbibliography` 命令，此外还会自动跳过没有文献的节。在上面的例子中，参考文献可以按如下方式生成：

```
\printbibheading
\bibbysection[heading=subbibliography]
```

当使用按章（或其它文档单元）划分的累积参考文献格式时，使用 `refsegment` 比 `refsection` 环境更合适一些。不同之处在于 `refsection` 环境生成的标签是环境局部的，而 `refsegment` 环境不影响标签生成，因此在整个文档中是唯一的。下面的例子也可以在 § 3.12.4 节中给出，因为它看起来创建了一个划分为多重片段的全局参考文献。

```
\documentclass{...}
\usepackage{biblatex}
\defbibheading{subbibliography}{%
```

```
  \section*{References for Chapter \ref{refsegment:
    ↪ \therefsection\therefsegment}}}
\addbibresource{...}
\begin{document}
\chapter{...}
\begin{refsegment}
...
\end{refsegment}
\chapter{...}
\begin{refsegment}
...
\end{refsegment}
\printbibheading
\printbibliography[segment=1,heading=subbibliography]
\printbibliography[segment=2,heading=subbibliography]
\end{document}
```

refsegment 的使用类似于 refsection，也有对应于 \printbibliography 的 segment 选项。biblatex 宏包自动在每个 refsegment 环境开始用字符串 refsegment: 后接相应 regsegment 环境的序号来设置标签作为标识符。如前所述，有一个匹配的 refsegment 计数器可以用在标题定义中。对于文献节，也有缩写名可以自动遍历所有的文献片段：

```
\printbibheading
\bibbysegment[heading=subbibliography]
```

这等价于为当前 refsection 的每个片段分别给出 \printbibliography 命令。

3.12.4 文献表划分

依照某一标准进行文献划分是非常普遍的。例如，你也许需要分别列出印刷和网络资源，或者将参考文献分为主要和次要类型。前一种情况比较简单，因为可以使用条目类型作为 \printbibliography 的 type 和 nottype filter 的标准。下面的例子也演示了如何为参考文献的两部分生成匹配的子标题。

```
\documentclass{...}
\usepackage{biblatex}
\addbibresource{...}
\begin{document}
...
\printbibheading
\printbibliography[nottype=online,heading=subbibliography,
                   title={Printed Sources}]
```

```
\printbibliography[type=online,heading=subbibliography,
                   title={Online Sources}]

\end{document}
```

也可以使用两个以上的划分：

```
\printbibliography[type=article,...]
\printbibliography[type=book,...]
\printbibliography[nottype=article,nottype=book,...]
```

甚至可以给出一组的不同类型的 filter：

```
\printbibliography[section=2,type=book,keyword=abc,notkeyword=xyz]
```

这会打印出所有在第二参考分节中条目类型为 `@book` 并且 `keywords` 域包括关键词“abc”但是不包括“xyz”的作品。关于结合数值样式使用文献 filter 见 § 3.13.5 节。如果你需要带有条件表达式的复杂 filter，可以使用 `filter` 选项结合由 `\defbibfilter` 定义的定制 filter。关于定制 filter 详见 § 3.7.9 节。

```
\documentclass{...}
\usepackage{biblatex}
\addbibresource{...}
\begin{document}
...
\printbibheading
\printbibliography[keyword=primary,heading=subbibliography,%
                   title={Primary Sources}]
\printbibliography[keyword=secondary,heading=subbibliography,%
                   title={Secondary Sources}]
\end{document}
```

如上例所示，将参考文献分为主要和次要部分可以通过 `keyword` filter 实现。在该情况下（只分成两部分），使用一个关键词作为 filter 标准就足够了：

```
\printbibliography[keyword=primary,...]
\printbibliography[notkeyword=primary,...]
```

由于 `biblatex` 无法知道文献中的某一条是否被认为是主要或者次要文献，我们需要在 `bib` 文件中为每一条目增加 `keywords` 域来提供文献 filter 所需的数据。如上例所示，这些关键词可以用于 `keyword` 和 `notkeyword` filter 的目标。在建立 `bib` 文件时就添加这样的关键词是一个不错的办法。

```
@Book{key,
  keywords      = {primary,some,other,keywords},
  ...
```

另外一种划分文献列表的方法是使用参考文献类别。这与上述例子中使用的基于关键词的方法的不同之处在于，它们在文档水平处理而并不需要修改 bib 文件。

```
\documentclass{...}
\usepackage{biblatex}
\DeclareBibliographyCategory{primary}
\DeclareBibliographyCategory{secondary}
\addtocategory{primary}{key1,key3,key6}
\addtocategory{secondary}{key2,key4,key5}
\addbibresource{...}
\begin{document}
...
\printbibheading
\printbibliography[category=primary,heading=subbibliography,%
                   title={Primary Sources}]
\printbibliography[category=secondary,heading=subbibliography,%
                   title={Secondary Sources}]
\end{document}
```

在这个例子中，只使用一个类别也是可以的：

```
\printbibliography[category=primary,...]
\printbibliography[notcategory=primary,...]
```

不过，显式地声明参考文献中使用的所有类别仍然是个不错的主意，因为有一个 \bibbycategory 命令能自动遍历所有的类别。这等价于为每一类别按照所声明的顺序依次给出 \printbibliography 命令。

```
\documentclass{...}
\usepackage{biblatex}
\DeclareBibliographyCategory{primary}
\DeclareBibliographyCategory{secondary}
\addtocategory{primary}{key1,key3,key6}
\addtocategory{secondary}{key2,key4,key5}
\defbibheading{primary}{\section*{Primary Sources}}
\defbibheading{secondary}{\section*{Secondary Sources}}
\addbibresource{...}
\begin{document}
```

```
...
\printbibheading
\bibbycategory
\end{document}
```

在这个例子中，标题的处理与 \bibbysection 和 \bibbysegment 是不同的。\bibbycategory 使用当前类别的名字作为标题名。这等价于将 heading=⟨*category*⟩ 传递给 \printbibliography，从而意味着你需要为每一类别提供相匹配的标题。

3.12.5 条目集

条目集是用单个引用并在参考文献中作为一项列出的一组条目。条目集中每一项用 \entrysetpunct 分隔（§ 4.10.1 节）。biblatex 宏包支持两种类型的条目集。静态条目集在 bib 文件中定义，这与其它条目类似。而动态条目集在文档导言区或者正文中用 \defbibentryset（§ 3.7.11）定义，并且基于文档或参考文献分节。本节讨论条目集的定义问题；样式作者对于更多信息也可以参考 § 4.11.1 节。

3.12.5.1 静态条目集 静态条目集如同其它条目一样在 bib 文件中定义。定义这样的条目集只需添加一个类型为 @set 的条目。该条目有一个 entryset 域，其中使用条目键值的逗号分隔列表定义了条目集的元素：

```
@Set{set1,
  entryset = {key1,key2,key3},
}
```

条目可以是文档或参考文件分节中一个集合的一部分，或者是另外一个条目集中的孤立文献，这取决于 @set 条目。如果 @set 条目被引用，其中的成员自动分成一组。否则它们就像其它的常规条目一样。

3.12.5.2 动态条目集 动态条目集的设置和运行和静态条目集很相似。主要的区别是，它们是在导言区或者实时地在文档中使用 § 3.7.11 节的 \defbibentryset 命令来定义的：

```
\defbibentryset{set1}{key1,key2,key3}
```

正文中的动态条目集在其所在的 refsection 环境中是局部的（如果有的话）。否则它们被分配给第零文献分节。定义在导言区的动态条目集也被分在第零文献节。

3.12.6 数据容器

作为数据容器，@xdata 条目类型可以包含一个或更多域。这些域可以被其它条目使用 xdata 来继承。@xdata 条目可以不被引用或者打印在参考文献中，它们只为其它条目提供数据源。这种数据继承机制常用于 publisher/location 这样的固定域组合或者其它常用数据：

```
@XData{hup,
  publisher  = {Harvard University Press},
  location   = {Cambridge, Mass.},
}
@Book{...,
  author     = {...},
  title      = {...},
  date       = {...},
  xdata      = {hup},
}
```

一个条目通过在 xdata 域中使用分隔键列表，可以继承若干个 @xdata 条目的数据。@xdata 条目的串联也是支持的，即，一个 @xdata 条目可以涉及到一个或更多其它 @xdata 条目：

```
@XData{macmillan:name,
  publisher  = {Macmillan},
}
@XData{macmillan:place,
  location   = {New York and London},
}
@XData{macmillan,
  xdata      = {macmillan:name,macmillan:place},
}
@Book{...,
  author     = {...},
  title      = {...},
  date       = {...},
  xdata      = {macmillan},
}
```

另见 §§ 2.1.1 和 2.2.3 节。

3.12.7 电子出版信息

biblatex 宏包为电子出版信息提供了三种域：eprint、eprinttype 和 eprintclass。eprint 域类似于 doi，是一种保持项目标识符的抄录模式域。eprinttype 域保存资源名称，即网址或电子档案的名称。可选的 eprintclass 域用于标明特定于 eprinttype 域所指资源的额外信息。这可以是章节、路径、分类信息等。如果 eprinttype 可用，标准样式会将其当做文本标签使用。在以下例子中，它们会打印“Resource: identifier”而不是一般的“eprint: identifier”：

```
eprint     = {identifier},
eprinttype = {Resource},
```

标准样式对一些在线资源提供了专门支持。对于 arXiv 文献，将标识符放在 eprint 域中，将字符串 arxiv 放在 eprinttype 域中：

```
eprint     = {math/0307200v3},
eprinttype = {arxiv},
```

对于使用新标识格式的文章（2007 年四月之后），将主分类放在 eprintclass 域中：

```
eprint      = {1008.2849v1},
eprinttype  = {arxiv},
eprintclass = {cs.DS},
```

为了方便 arXiv 条目的整合专门设置了两个别称。archiveprefix 是 eprinttype 的别称；而 primaryclass 是 eprintclass 的别称。如果启用超链接，eprint 标识符将转换为指向 arxiv.org 的链接。更多信息可参见 § 3.1.2.1 节中的宏包选项 arxiv。

对于 JSTOR 资源，将稳定的 JSTOR 号放在 eprint 域中，将字符串 jstor 放在 eprinttype 域中：

```
eprint     = {number},
eprinttype = {jstor},
```

当使用 JSTOR 的导出功能来导出 BibTeX 格式引用时，JSTOR 缺省使用 url 域（当 ⟨*number*⟩ 是唯一稳定标识符时）：

```
url = {http://www.jstor.org/stable/number},
```

尽管这样可以运行，但整个的 URL 会使参考文献变得杂乱无章。而使用 eprint 域，标准样式会使用更加可读的“JSTOR: ⟨*number*⟩”格式而且同样支持超链接。当启用 hyperref 支持时，⟨*number*⟩ 会变成可以点击的链接。

对于 PubMed 资源，将稳定的 PubMed 标识符放在 eprint 域中，将字符串 pubmed 放在 eprinttype 域中。也就是

```
url = {http://www.ncbi.nlm.nih.gov/pubmed/pmid},
```

会变成：

```
eprint     = {pmid},
eprinttype = {pubmed},
```

并且标准样式会打印出“PMID: ⟨*pmid*⟩”来取代冗长的 URL。如果启用 hyperref 支持，⟨*pmid*⟩ 会变成指向 PubMed 的可点击的链接。

对于句柄系统[38]（HDL），将句柄放在 eprint 域中，将字符串 hdl 放在 eprinttype 域中：

[38]参考 http://www.handle.net/——译注

```
eprint     = {handle},
eprinttype = {hdl},
```

对于 Google Books 资源，将 Google 标识符放在 eprint 域中，将字符串 googlebooks 放在 eprinttype 域中。如下例。

```
url = {http://books.google.com/books?id=XXu4AkRVBBoC},
```

会变成：

```
eprint     = {XXu4AkRVBBoC},
eprinttype = {googlebooks},
```

并且标准样式会打印出“Google Books: XXu4AkRVBBoC”代替整个 URL。如果启用了 hyperref 支持，该标识符会变成指向 Google Books 的可点击的链接。[39]

请注意 eprint 是一个抄录模式域，故而总是以未修改的形式给出标识符。例如没有必要将 _ 改成 _。对于如何为其它电子出版资源增加细致的支持，也可以参考 § 4.11.2 节。

3.12.8 外部摘要和注释

对于打印 abstract 和/或 annotation 域的样式，这里介绍另一种添加摘要或注释到参考文献的方法。即，除了包含在 bib 文件中，相关文本也可以保存在外部的 LaTeX 文件中。例如，除了在 bib 文件中写入如下内容之外，

```
@Article{key1,
  ...
  abstract        = {This is an abstract of entry ‘key1’.}
}
```

你也可以创建一个名为 bibabstract-key1.tex 的文件并将摘要放在该文件中：

```
This is an abstract of entry ‘key1’.
\endinput
```

外部文件名必须是条目键分别加上前缀 bibabstract- 或 bibannotation-。你可以通过重定义 \bibabstractprefix 和 \bibannotationprefix 来改变这些前缀。请注意，该特性需要通过显式地设置 § 3.1.2.1 中的宏包选项 loadfiles 来启用。缺省情况下出于性能原因该选项是关闭的。同样要注意的是，bib 文件中的任何 abstract 和 annotation 域都优先于外部文件。如果你的摘要或注释很长（这会显著增加内存需求），那么强烈推荐使用外部文件。此外，在专门的 LaTeX 文件中编辑文本也是很方便的。样式作者就更多信息应该参考 § 4.11.3 节。

[39] 请注意，Google Books ID 似乎是一个“内部”值。从这份手册开始，似乎没有办法在 Google Books 上搜索 ID。此时也许最好使用 url 域。

3.13 提示和注意事项

本节提供了其它一些使用提示，并介绍了一些常见问题和可能的错误概念。

3.13.1 与 KOMA-Script 文档类共用

当在 scrbook、scrreprt 或 scrartcl 文档类中使用 biblatex 时，§ 3.7.7 节中的标题 bibliography 和 shorthands 会与这些文档类的文献相关选项有关。[40] 可以通过使用 \printbibliography、\printbibheading 和 \printshorthands 的 heading 选项来覆盖缺省标题。详见 §§ 3.7.2、3.7.3、3.7.7 节。所有的缺省标题都在导入时调整以使得与这些文档类相称。如果 biblatex 探测到这些文档类中的某一个，它还会提供以下额外的测试，这对定制标题定义很有用：

\ifkomabibtotoc{⟨*true*⟩}{⟨*false*⟩}

如果该文档类将参考文献加入目录中则展开为 ⟨*true*⟩，否则为 ⟨*false*⟩。

\ifkomabibtotocnumbered{⟨*true*⟩}{⟨*false*⟩}

如果该文档类将参考文献加入目录作为带编号的一节，则展开为 ⟨*true*⟩，否则为 ⟨*false*⟩。如果该测试结果为 ⟨*true*⟩，那么 \ifkomabibtotoc 也总是为 ⟨*true*⟩，但反过来不一定。

3.13.2 与 Memoir 文档类共用

当在 memoir 文档类中使用 biblatex 时，大部分调整参考文献的文档类工具都没有效果。相反，请使用本宏包的相应工具（§§ 3.7.2、3.7.7、3.7.8 节）。使用 \printbibliography 和 \defbibheading（§§ 3.7.2 和 3.7.7 节）的 heading 选项，而不要重定义 memoir 的 \bibsection。使用 \printbibliography 和 \defbibnote（§§ 3.7.2 和 3.7.8 节）的 prenote 和 postnote 来取代 \prebibhook 和 \postbibhook。所有缺省标题都在导入时调整以与该文档类的缺省布局相称。缺省的标题 bibliography 和 shorthands （§ 3.7.7 节）也与 memoir 的 \bibintoc 和 \nobibintoc 开关相对应。长度计数器 \bibitemsep 在 biblatex 中的使用方法与在 memoir 类似（§ 3.10.4 节）。本节还解释额外一些对应于 memoir 中 \biblistextra 的长度计数器。最后，\setbiblabel 并不能对应于 biblatex 宏包的某一单独工具，因为参考文献中所有标签的样式由参考文献样式控制。详见本手册的 § 4.2.2。如果 biblatex 探测到 memoir 文档类的使用，它还会提供以下额外的测试，这对定制标题定义很有用：

\ifmemoirbibintoc{⟨*true*⟩}{⟨*false*⟩}

取决于 memoir 的 \bibintoc 和 \nobibintoc 开关，可以展开为 ⟨*true*⟩ 或 ⟨*false*⟩。这是对应于 memoir 的 \ifnobibintoc 测试的 LaTeX 前端。请注意该测试的逻辑可以反过来。

[40] 这既可以应用到传统的选项语法（bibtotoc 和 bibtotocnumbered），也可以应用到 KOMA-Script 3.x 引入的 ⟨*key*⟩=⟨*value*⟩ 语法，即 bibliography=nottotoc、bibliography=totoc 和 bibliography=totocnumbered。全局的 toc=bibliography 和 toc=bibliographynumbered 以及它们的别称也会检测到。在任何一种情况下，选项必须在 \documentclass 的可选项中全局地设置。

3.13.3 引用中的页码数

如果一个引用命令的 ⟨*postnote*⟩ 选项是页码数或页码范围，那么 biblatex 会自动给其增加缺省前缀“p.”或“pp.”。在通常情况下这很可靠，但有时也需要手动调整。这时理解该选项怎样处理的细节就很重要。首先 biblatex 检查后注是否是阿拉伯或罗马数字（大小写不敏感）。如果该测试成功，那么该后注会被认为是一个单独页码或者其它数字，这时会被加上前缀“p.”或其它取决于 pagination 域（见 § 2.3.10 节）的字符串。如果测试没有成功，那么会执行第二项测试来检测该后注是否是一个区间或者一列阿拉伯或罗马数字。如果该测试成功，那么该后注会被加上前缀“pp.”或其它复数形式的字符串。如果该测试也没有成功，该后注会依原样打印。请注意这两项测试都会展开 ⟨*postnote*⟩。因此所有在该选项中使用的命令都必须是鲁棒的或者用 \protect 加以保护。这里分别是一些 ⟨*postnote*⟩ 选项会被正确识别为单独数字、数字范围或者一列数字的一些例子：

```
\cite[25]{key}
\cite[vii]{key}
\cite[XIV]{key}
\cite[34--38]{key}
\cite[iv--x]{key}
\cite[185/86]{key}
\cite[XI \& XV]{key}
\cite[3, 5, 7]{key}
\cite[vii--x; 5, 7]{key}
```

然而在其它一些情况该测试会失败，此时需要考虑 § 3.8.8 节的一些辅助命令 \pno、\ppno 和 \nopp。例如，假设一部作品由一种包含数字和字母的特殊页码标记格式所引用。在这种格式中，字符串“27a”的意思是“page 27, part a”。因为对于 biblatex 来说该字符串并不像数字或者数字范围，因此你需要手动强制加上单独页码的前缀：

```
\cite[\pno~27a]{key}
```

同样地，\ppno 命令会强制为范围前缀，而 \nopp 命令会取消所有的前缀：

```
\cite[\ppno~27a--28c]{key}
\cite[\nopp 25]{key}
```

这些命令可以用于 ⟨*postnote*⟩ 选项的任何地方，也可以被多次使用。例如，当以卷数和页码数引用时，你或许希望在后注的开始取消前缀，而在字符串的中间加上：

```
\cite[VII, \pno~5]{key}
\cite[VII, \pno~3, \ppno~40--45]{key}
\cite[see][\ppno~37--46, in particular \pno~40]{key}
```

还有两个用于形如“the following page(s)”的后缀的辅助命令。使用辅助命令 \psq 和 \psqq 来代替用文本插入这样的后缀（这要求 \ppno 来强加一个前缀）：

```
\cite[\ppno~27~sq.]{key}
\cite[\ppno~55~sqq.]{key}
```

请注意数字和命令之前没有空格。该空格会自动插入并可以通过重定义 \sqspace 宏来修改。

```
\cite[27\psq]{key}
\cite[55\psqq]{key}
```

由于当后注包括任何非阿拉伯或罗马数字时将会以不带任何前缀的方式打印，也可以手动输入前缀：

```
\cite[p.~5]{key}
```

可以通过设置条目的 pagination 域为“none”来基于每一条目取消前缀，详见 § 2.3.10 节。如果你不需要任何前缀或者更想要手动输入，也可以在导言区或者配置文件中整个地关闭该机制，如下所示：

```
\DeclareFieldFormat{postnote}{#1}
```

⟨*postnote*⟩ 选项会像条目域一样处理，该域的格式由域格式指令来控制，而该指令可以自由地重定义。以上的定义会简单地将后注依原样打印。更多细节可以参见作者指南部分的 §§ 4.3.2 和 4.4.2 节。

3.13.4 姓名组成部分及其间距

biblatex 宏包可以使用户和样式作者对姓名空格和换行进行精细的控制。下面讨论的命令在本文档的 §§ 3.10.1 和 4.10.1 节。本节的目的在于大致介绍如何将这些命令结合起来。关于术语的注记：名部分是姓名中的基本成分，例如 first name 或 last name。姓名的每一部分可以是单个的名或者包含多个名。例如，名部分 ‘first name’ 可以包含 first name 和 middle name。后者可以认为是本节所说的名元素。我们首先考虑一个简单的名字 “John Edward Doe”，它包含了如下部分：

First	John Edward
Prefix	—
Last	Doe
Suffix	—

姓名中的空格、标点和断行行为由六个宏所控制：

a=\bibnamedelima	由后端程序插入在每一名部分的第一个元素后（如果该元素少于三个字符长度），在每一名部分的最后元素之前。
b=\bibnamedelimb	由后端程序插入在名部分的元素之间且 \bibnamedelima 没有使用之处。
c=\bibnamedelimc	当 useprefix=true 时，由格式指令插入在 name prefix 和 last name 之间。如果 useprefix=false，将使用 \bibnamedelimd。
d=\bibnamedelimd	由格式指令插入在名部分之间且 \bibnamedelimc 没有使用之处。
i=\bibnamedelimi	在首字符缩写之后代替 \bibnamedelima/b 的命令。
p=\revsdnamepunct	当名部分顺序反过来时，由格式指令插入在 last name 之后。

以下展示了如何使用这些分隔符：

John$_{a}$ Edward$_{d}$ Doe

Doe$_{p}$,$_{d}$ John$_{a}$ Edward

bib 文件中的首字符缩写会有一个特别的分隔符：

J.$_{i}$ Edward$_{d}$ Doe

考虑一个更复杂的名字：“Charles-Jean Étienne Gustave Nicolas de La Vallée Poussin”。它包含了如下几部分：

First	Charles-Jean Étienne Gustave Nicolas
Prefix	de
Last	La Vallée Poussin
Suffix	—

分隔符为：

Charles-Jean$_{b}$ Étienne$_{b}$ Gustave$_{a}$ Nicolas$_{d}$ de$_{c}$ La$_{a}$ Vallée$_{a}$ Poussin

请注意 \bibnamedelima/b/i 由后端程序插入。后端程序处理名部分并考虑组成名部分的元素之间的分隔符，从而分别处理每一部分。与此相反，全名的名部分之间的分隔符（\bibnamedelimc/d）由名称格式指令在处理过程的之后时间点添加。首字符缩写的空格和标点同样由后端程序处理，并且可以通过重定义以下三个宏来定制：

a=\bibinitperiod	由后端程序插入在首字母缩写之后。
b=\bibinitdelim	由后端程序插入在多个首字母缩写之间。
c=\bibinithyphendelim	由后端程序插入在带有连字符的名部分中首字母缩写之间，用以代替 \bibinitperiod 和 \bibinitdelim。

以下是使用方式：

J.$_{a}$.$_{b}$ E.$_{a}$ Doe

K.$_{c}$-H.$_{a}$ Mustermann

3.13.5 文献 filter 和引用标签

本宏包生成的引用标签在被文献 filter 分开之前就被分配给整个文献列表。因此能够确保在整个文档（或者一个 refsection 环境中）是唯一的，无论使用多少文献 filter。然而，当使用数值型引用格式时，这很可能会导致在各个分片参考文献中的编号不是连续的。使用 defernumber 宏包选项可以避免这一问题。如果启用该选项，数值标签会在任一文献条目中第一次打印时才被分配。

3.13.6 参考文献标题中的活动字符

babel、polyglossia、csquotes 和 underscore 等使用活动字符的宏包通常直到正文开始才将这些字符激活，这样可以避免与其它宏包的冲突。一个典型的活动字符例子是 Ascii 引号 "，它用于 babel/polyglossia 宏包的不同语言模块。如果在 \defbibheading 的选项中使用 "< 和 "a 等速记方式，并且标题定义在导言区中，那么标题定义中保存的是字符的非激活形式。因此当标题打印出来时，它们不会像命令一样起作用而仅仅依照原文打印。最直接的解决方法是将 \defbibheading 放在 \begin{document} 之后。此外，你也可以使用 babel 的 \shorthandon 和 \shorthandoff 命令来临时在导言区中激活这些简记方式。这同样应用于文献注记和 \defbibnote 命令。

3.13.7 参考文献分节和分段中的编组

所有在 \begin 和 \end 中的 LaTeX 环境形成了一个分组。如果该环境包含一些没有在组内使用的东西，那么可能会引起一些不良反应。这个问题并不仅仅针对 refsection 和 refsegment 环境，但显然也包括它们。由于这些环境通常比典型的 itemize 或其它环境包含更多文本，因此它们自然更有可能引起涉及到分组的问题。如果你在添加 refsection 环境之后观察到任何不正常的现象（例如，如果环境内有任何“受限”情况），请尝试使用以下语句来代替：

```
\chapter{...}
\refsection
...
\endrefsection
```

这不会形成一个分组，但是像正常一样工作。就 biblatex 而言，它并不影响你使用哪种语句。refsegment 环境也支持这种语句。请注意，命令 \newrefsection 和 \newrefsegment 不会形成分组。详见 §§ 3.7.4 和 3.7.5 节。

3.14 使用备选的 BibTeX 后端

biblatex 必须使用 biber 程序作为后端才能使用这里描述的所有特性。实际上，本文档正是默认这一假定。不过，如果只是使用受限制的一部分功能，也可以使用历史悠久的 BibTeX 程序（7-bit 的 bibtex 或者 8-bit 的 bibtex8）作为后端程序。

使用 BibTeX 作为后端程序需要在载入时开启选项 backend=bibtex 或 backend=bibtex8. biblatex 宏包随后会将合适的数据写入辅助文件以及特殊的数据文件中以供 BibTeX 使用（自动包含那些被 BibTeX 读取的文件）。然后 bibtex(8) 程序应当运行在每一辅助文件上：biblatex 会在日志文件中列出所有所需的文件。

BibTeX 后端的主要局限有：

- 排序是全局的，而且只限于按照 Ascii 顺序。
- 不可以重编码，因此数据库中的条目必须按照 LaTeX 内部字符表示[41]的形式，才可以确保程序可靠。
- 数据模型是固定的。
- 交叉引用有限制，条目集必须写入 .bib 文件。
- 内存容量有限。在 bibtex8 中，强烈建议使用 --wolfgang 选项以尽量减少这一问题的可能性。

4 样式作者指南

本节是样式作者指南，主要介绍biblatex 包的接口。该指南囊括了设计参考文献著录和标注样式或者本地化模型所需了解的所有知识。在阅读本节内容前最好先阅读上一节的用户手册。

4.1 概述

在讨论biblatex 提供的命令和工具之前，我们首先介绍一些基本概念。biblatex 包以一种特殊方式使用辅助文件。最值得注意的是当使用 BibTeX 后端程序时，bbl 文件的使用方式存在差别，即只有一个bst 文件可用来实现结构化的数据接口，该文件并非用来输出可打印数据。

使用 LaTeX 的标准参考文献工具，一个文档通常包含任意数量的文献引用命令，以及常放在文档末尾的\bibliographystyle 和\bibliography 命令。文献引用命令在文档中的位置是任意的，而\bibliographystyle 和\bibliography 命令则标记了打印参考文献表的位置，比如:

```
\documentclass{...}
\begin{document}
\cite{...}
...
\bibliographystyle{...}
\bibliography{...}
\end{document}
```

处理这些文件遵循一定的流程，其过程如下:

[41] LaTeX Internal Character Representation, LICR——译注

1. 运行latex: 第一次运行latex，在aux 文件中写入\bibstyle 和 \bibdata 命令，以及所有标注的\citation 命令。这时，各引文标注[42] 是未定义的，因为 LaTeX 等待 BibTeX 提供需要的数据，当然参考文献表也未生成。

2. 运行bibtex: BibTeX 在bbl 文件中写入一个thebibliography 环境，用以为aux 文件中\citation 命令提供所需的所有条目，这些条目的数据来自bib 文件。

3. 运行latex: 第二次运行latex，thebibliography 环境中的\bibitem 命令为各参考文献条目在aux 文件中写入\bibcite 命令。这些\bibcite 命令定义的标签将用于\cite 命令。然而，各引文标注仍然未定义，因为这些标签在最后一次运行latex 前仍是未知的。

4. 运行latex: 第三次运行，随着导言区最后读入了aux 文件，引文标注的标签定义完成。至此所有的标注可以正确打印。

注意到所有的参考文献数据都以最终格式 (指最后打印出的格式) 写入bbl 文件。该文件的读取和处理如同任何文档中的可打印章节。例如，考虑在一个bib 文件中有如下条目:

```
@Book{companion,
  author    = {Michel Goossens and Frank Mittelbach and Alexander Samarin
    ↪ },
  title     = {The LaTeX Companion},
  publisher = {Addison-Wesley},
  address   = {Reading, Mass.},
  year      = {1994},
}
```

根据plain.bst 样式，BibTeX 在bbl 文件中输出该条目如下:

```
\bibitem{companion}
Michel Goossens, Frank Mittelbach, and Alexander Samarin.
\newblock {\em The LaTeX Companion}.
\newblock Addison-Wesley, Reading, Mass., 1994.
```

默认情况下，LaTeX 生成顺序编码制标注标签，因此\bibitem 命令在aux 文件中写入的行如下所示:

```
\bibcite{companion}{1}
```

要实现一个不同的标注标签样式，意味着需要通过aux 文件传递更多的数据。比如，当使用natbib 包时，aux 文件包含的标注 (或引用) 信息行，如下所示:

[42]译者: 这里的 references 译为引文标注，指引用命令在正文中产生的标注，这个标注由标签 label 构成。

```
\bibcite{companion}{{1}{1994}{{Goossens et~al.}}{{Goossens, Mittelbach,
and Samarin}}}
```

biblatex 包支持任何格式的标注标签，因此标注命令需要访问完整的参考文献数据。观察同样需要在标注中提供完整参考文献数据的jurabib 包的输出，我们会更理解这一需求对上述处理过程意味着什么。

```
\bibcite{companion}{{Goossens\jbbfsasep Mittelbach\jbbstasep Samarin}%
  {}{{0}{}{book}{1994}{}{}{}{}{Reading, Mass.\bpubaddr{}Addison-Wesley%
  \bibbdsep{} 1994}}{{The LaTeX Companion}{}{}{2}{}{}{}{}{}}{\bibnf
  {Goossens}{Michel}{M.}{}{}\Bibbfsasep\bibnf{Mittelbach}{Frank}{F.}%
  {}{}\Bibbstasep\bibnf{Samarin}{Alexander}{A.}{}{}}{\bibtfont{The
  LaTeX Companion}.\ \apyformat{Reading, Mass.\bpubaddr{}
  Addison-Wesley\bibbdsep{} 1994}}}
```

在这种情况下，整个thebibliography 环境的内容是通过aux 文件进行有效传递的。数据首先从bbl 文件中读取出来，写入到aux 中，然后再从aux 读出保存到内存中。参考文献表本身的生成也需要先读入bbl 文件。这也使得biblatex 包被迫通过aux 文件回收所有数据。这意味着上述过程处理过度且多余，因为不管怎样数据都需要保存到内存中。

这种传统的处理过程都基于一个假设，即条目的完整数据只是参考文献表需要而所有的标注都使用短标签。这对于有内存限制的情况是非常高效的，但也意味着它很难扩展。这就是biblatex 采取另一种方式的原因。采用新的方式，首先文档结构略有变化。取消在文档内使用\bibliography 命令，数据库文件由导言区的\addbibresource 命令指定，完全忽略\bibliographystyle 命令 (所有的功能都将由包选项控制)，参考文献表则使用\printbibliography 命令打印:

```
\documentclass{...}
\usepackage[...]{biblatex}
\addbibresource{...}
\begin{document}
\cite{...}
...
\printbibliography
\end{document}
```

为简化整个过程，biblatex 基本上以应用aux 文件的方式应用bbl 文件，并舍弃了\bibcite 命令。于是，我们得到如下流程:

1. 运行latex: 第一步类似于上述的传统方式: 将\bibstyle 和 \bibdata 以及所有引用的\citation 命令写入到aux 文件中 (以 BibTeX 为后端程序) 或者写到bcf 文件中 (以biber 为后端程序)。然后等待后端程序提供需要的

数据。当以 BibTeX 为后端程序时，biblatex 使用一个特殊的bst 文件，该文件用于实现 BibTeX 后端程序的数据接口，因此\bibstyle 命令必须是\bibstyle{biblatex}。

2. 运行biber 或 bibtex: 后端程序为辅助文件中所有\citation 命令提供所需的条目，这些条目的数据来自bib 文件。然而，它并不在bbl 文件中写出一个可打印的参考文献表，而是一个结构化表达的参考文献数据。类似于aux 文件，读入该bbl 文件时不打印任何东西，仅是将数据存入内存中。

3. 运行latex: 第二次运行，bbl 文件在文档正文开始的时候处理，类似于aux 文件。从这开始，所有参考文献数据都已在内存中，所以所有的引用标注都可以正确打印。[43] 引用命令不仅可以访问预定义的标签，还可以访问完整的参考文献数据。参考文献表由内存中的相同数据生成，可以根据需要进行筛选和划分。

我们再次观察上面给出的条目示例:

```
@Book{companion,
  author    = {Michel Goossens and Frank Mittelbach and Alexander Samarin
    ↪ },
  title     = {The LaTeX Companion},
  publisher = {Addison-Wesley},
  address   = {Reading, Mass.},
  year      = {1994},
}
```

使用biblatex 及biber 后端程序，这一条目实际上以如下格式输出:

```
\entry{companion}{book}{}
  \labelname{author}{3}{}{%
    {{uniquename=0,hash=...}{Goossens}{G.}{Michel}{M.}{}{}{}{}}%
    {{uniquename=0,hash=...}{Mittelbach}{M.}{Frank}{F.}{}{}{}{}}%
    {{uniquename=0,hash=...}{Samarin}{S.}{Alexander}{A.}{}{}{}{}}%
  }
  \name{author}{3}{}{%
    {{uniquename=0,hash=...}{Goossens}{G.}{Michel}{M.}{}{}{}{}}%
    {{uniquename=0,hash=...}{Mittelbach}{M.}{Frank}{F.}{}{}{}{}}%
    {{uniquename=0,hash=...}{Samarin}{S.}{Alexander}{A.}{}{}{}{}}%
  }
  \list{publisher}{1}{%
    {Addison-Wesley}%
  }
```

[43] 如果defernumbers 包选项打开，biblatex 以类似于传统过程的一种算法来生成顺序制标签。这种情况下，这些数字将在参考文献表打印时指定，且需从后端辅助文件中回收。因此需要额外再运行一次 LaTeX 以在标注中应用。

```
  \list{location}{1}{%
    {Reading, Mass.}%
  }
  \field{title}{The LaTeX Companion}
  \field{year}{1994}
\endentry
```

由该示例可见，某种程度上，结构化的数据构成了bbl 文件内容[44]。此时关于参考文献条目最终格式的任何决定都未作出。而参考文献表和引用标注的格式化最终由 LaTeX 宏控制，这些宏则定义在参考文献著录和标注样式文件中。

4.2 参考文献著录样式

参考文献著录样式 是用于控制打印参考文献表条目的宏的集合，定义在扩展名为bbx 的文件中。biblatex 包在其末尾加载所选择的参考文献著录样式文件。需要注意: 多个标准样式共享的一些常用宏定义在biblatex.def 文件中。该文件同样在宏包末尾加载，但先于所选的著录样式文件。

4.2.1 参考文献著录样式文件

在我们讨论参考文献著录样式的各部分内容之前，先观察一个典型bbx 文件的总体结构:

```
\ProvidesFile{example.bbx}[2006/03/15 v1.0 biblatex bibliography style]

\defbibenvironment{bibliography}
  {...}
  {...}
  {...}
\defbibenvironment{shorthand}
  {...}
  {...}
  {...}
\InitializeBibliographyStyle{...}
\DeclareBibliographyDriver{article}{...}
\DeclareBibliographyDriver{book}{...}
\DeclareBibliographyDriver{inbook}{...}
...
\DeclareBibliographyDriver{shorthand}{...}
\endinput
```

参考文献著录样式文件的主要结构包含如下命令:

[44]译者: 这里应该是 bbl 文件而不是原文的 bib 文件

`\RequireBibliographyStyle{`⟨*style*⟩`}`

该命令是可选的，用于引入一些建立在更一般的参考文献样式上的特殊样式。该命令加载了样式文件`style.bbx`。

`\InitializeBibliographyStyle{`⟨*code*⟩`}`

该命令在参考文献表开始之前插入任意给定的⟨*code*⟩，但在参考文献表所形成的组内。该命令是可选的。它可以用于共享不同参考文献驱动需要的一些相同定义，但不能用于参考文献组外。记住，文档中可以有多个参考文献表，如果在某个文献表中对参考文献驱动进行了一些全局设置，最好在下一个参考文献表开始前进行重设。

`\DeclareBibliographyDriver{`⟨*entrytype*⟩`}{`⟨*code*⟩`}`

定义一个参考文献驱动。一个驱动 'driver' 是一个宏，用于控制某一具体的参考文献条目 (当打印参考文献表时) 或者某一具体命名了的参考文献表 (当打印多个参考文献表时)。⟨*entrytype*⟩ 与`bib` 文件中使用的条目类型对应，以小写字母给出 (见 § 2.1)。⟨*entrytype*⟩ 变量可以是一个星号。这种情况下，该驱动将作为未定义具体驱动的条目类型的驱动。⟨*code*⟩ 是用于打印各自⟨*entrytype*⟩ 的参考文献条目的任意代码。该命令是必须的。每个参考文献著录样式都应提供所用到的每类条目的驱动。

`\DeclareBibliographyAlias{`⟨*alias*⟩`}{`⟨*entrytype*⟩`}`

如果一个参考文献驱动用于处理多个条目类型，该命令可以用来定义以⟨*entrytype*⟩命名的驱动的别名。⟨*alias*⟩ 选项可以是一个星号，这种情况下，该驱动将作为未定义具体驱动的条目类型的驱动。

`\DeclareBibliographyOption[`⟨*datatype*⟩`]{`⟨*key*⟩`}[`⟨*value*⟩`]{`⟨*code*⟩`}`

该命令以⟨*key*⟩`=`⟨*value*⟩ 格式定义额外的导言区选项。⟨*key*⟩ 是选项键。⟨*code*⟩ 是当使用该选项时执行的任意 TeX 代码。键值作为`#1` 传递给⟨*code*⟩。可选的⟨*value*⟩ 是当该选项仅有键名而无键值给出时的默认键值。这对于布尔选项非常有用。⟨*datatype*⟩是选项的数据类型 (datatype)，如果缺省，那么默认为 'boolean'(布尔类型)。一个定义示例如下:

```
\DeclareBibliographyOption[boolean]{somekey}[true]{...}
```

给出'`somekey`' 而没有键值等价于'`somekey=true`'。有效的⟨*datatype*⟩ 在默认的`biber` 数据模型中定义，包括:

```
\DeclareDatamodelConstant[type=list]{optiondatatypes}{boolean,integer,
    ↪ string,xml}
```

\DeclareEntryOption[⟨*datatype*⟩]{⟨*key*⟩}[⟨*value*⟩]{⟨*code*⟩}

类似于\DeclareBibliographyOption，但用于定义具体条目类型的options 域 (见 § 2.2.3 节) 中的可设选项。⟨*code*⟩ 在biblatex 为标注命令和参考文献驱动准备数据时执行。

4.2.2 参考文献表环境

除了定义参考文献驱动，参考文献著录样式也要定义参考文献表环境用于控制参考文献表的输出。这些环境由\defbibenvironment 命令定义。默认情况下，\printbibliography 使用bibliography 环境。下面是一个适用于不打印标签的参考文献表的环境定义:

```
\defbibenvironment{bibliography}
  {\list
     {}
     {\setlength{\leftmargin}{\bibhang}%
      \setlength{\itemindent}{-\leftmargin}%
      \setlength{\itemsep}{\bibitemsep}%
      \setlength{\parsep}{\bibparsep}}}
  {\endlist}
  {\item}
```

该定义使用biblatex 提供的\bibhang尺寸/长度，应用了一个带悬挂缩进的list 环境。它允许使用\bibitemsep 和 \bibparsep 来进行一定程度的布局调整，biblatex 提供的这两个尺寸的目的也在于此 (见 § 4.10.3)。作者年制 (authoryear) 和作者题名制 (authortitle) 的参考文献样式使用类似于该例的定义。

```
\defbibenvironment{bibliography}
  {\list
     {\printfield[labelnumberwidth]{labelnumber}}
     {\setlength{\labelwidth}{\labelnumberwidth}%
      \setlength{\leftmargin}{\labelwidth}%
      \setlength{\labelsep}{\biblabelsep}%
      \addtolength{\leftmargin}{\labelsep}%
      \setlength{\itemsep}{\bibitemsep}%
      \setlength{\parsep}{\bibparsep}}%
      \renewcommand*{\makelabel}[1]{\hss##1}}
  {\endlist}
  {\item}
```

一些参考文献样式在参考文献表中打印标签。比如，设计一个顺序编码的参考文献样式需要在参考文献表中每个条目前面打印顺序编码数字，这样参考文献表看起来就像一个顺序列表。在第一个示例中，\list 命令的第一个参数是空的。在

上面这个示例中，我们需要在其中插入数字，这些数字由biblatex 的labelnumber 域提供。我们也应用biblatex 提供的几个尺寸和工具，详见 §§ 4.10.4 和 4.10.5。顺序编码制 (numeric) 参考文献样式使用如上的定义。顺序字母制 (alphabetic) 的样式也是类似的，只是用 labelalpha 和 labelalphawidth 代替了 labelnumber 和 labelnumberwidth。

参考文献列表以类似方式处理。\printbiblist 命令默认使用以 bibliography list 命名的环境。一个典型示例如下，其中的尺寸和工具定义详见第 §§ 4.10.4 和 4.10.5 节。

```
\defbibenvironment{shorthand}
  {\list
     {\printfield[shorthandwidth]{shorthand}}
     {\setlength{\labelwidth}{\shorthandwidth}%
      \setlength{\leftmargin}{\labelwidth}%
      \setlength{\labelsep}{\biblabelsep}%
      \addtolength{\leftmargin}{\labelsep}%
      \setlength{\itemsep}{\bibitemsep}%
      \setlength{\parsep}{\bibparsep}%
      \renewcommand*{\makelabel}[1]{##1\hss}}}
  {\endlist}
  {\item}
```

4.2.3 参考文献驱动

在讨论biblatex 包的数据接口命令前，了解一下参考文献驱动的结构是有益的。注意，虽然下面给出的示例是大为简化的，但仍具有说明价值。为可读性考虑，我们忽略了一些可能是@book 条目的域，并且简化处理没有忽略的域。主要是为了说明驱动的结构。关于 BibTeX 文件格式域与biblatex 宏包数据类型的映射信息，见 § 2.2。

```
\DeclareBibliographyDriver{book}{%
  \printnames{author}%
  \newunit\newblock
  \printfield{title}%
  \newunit\newblock
  \printlist{publisher}%
  \newunit
  \printlist{location}%
  \newunit
  \printfield{year}%
  \finentry}
```

标准的参考文献样式应用两个参考文献宏 begentry 和 finentry:

```
\DeclareBibliographyDriver{entrytype}{%
  \usebibmacro{begentry}
  ...
  \usebibmacro{finentry}}
```

默认定义为:

```
\newbibmacro*{begentry}{}
\newbibmacro*{finentry}{\finentry}
```

为方便在驱动开始或结束时使用钩子，推荐使用这两个宏。

上述给出的 book 条目类型的驱动中存在有一些缺省: 即\printnames, \printlist, 和 \printfield 命令所使用的格式化指令。为了说明一个格式化指令是什么，这里给出上述驱动示例中所使用的指令。域格式是很直接的，域的值直接作为参数传递给能实现期望格式的指令。下面的指令简单地将输入参数用一个\emph 命令包裹:

```
\DeclareFieldFormat{title}{\emph{#1}}
```

列表格式则稍微要复杂一些。在将列表划分为独立的项后，biblatex 将对列表中的每一项执行格式化指令。各项作为参数传递给格式化指令。列表中各项间的分隔符由相应的指令控制，因此我们在插入分隔符前必须要检查当前位置是位于列表中还是位于列表末尾。

```
\DeclareListFormat{location}{%
  #1%
  \ifthenelse{\value{listcount}<\value{liststop}}
    {\addcomma\space}
    {}}
```

姓名 (name) 的格式化指令类似于抄录列表。

依赖于数据模型常量‘nameparts’ 的姓名有如下默认定义:

```
\DeclareDatamodelConstant[type=list]{nameparts}
                                    {prefix,family,suffix,given}
```

可以对其进行设置，比如添加更多姓名成分来处理类似父系姓的问题 (见文件93-nameparts.tex)。自然的，数据源需要一个扩展的姓名格式。biblatexml (§ D) 可以处理该问题，其中有一个扩展的姓名格式，当使用biber 后端时 (见biber 文档)，可以处理自定义的姓名成分。

在姓名格式中，姓名成分常量声明将为数据模型中定义的每个姓名成分提供两个宏:

```
\namepart<namepart>
\namepart<namepart>i
```

姓名的格式化执令对姓名列表中的每一个姓名进行处理，示例如下:

```
\DeclareNameFormat{author}{%
  \ifthenelse{\value{listcount}=1}
    {\namepartfamily%
     \ifblank{\namepartgiven}{}{\addcomma\space\namepartgiven}}
    {\ifblank{\namepartgiven}{}{\namepartgiven\space}%
     \namepartfamily}%
  \ifthenelse{\value{listcount}<\value{liststop}}
    {\addcomma\space}
    {}}
```

上述格式化指令调换了第一个作者的姓名前后顺序 (“Last, First”)，而其余姓名则是常规顺序 (“First Last”)。注意: 当仅有一个姓名成分时，必须确保要它是姓 (last name)，因此我们需要检查实际数据中姓名的哪些成分存在。如果姓名的一些成分不存在，则相应的宏就为空。如同抄录列表的指令，在各独立项之间插入的分隔符也由格式化指令控制，因为我们也要检查当前位置是处于列表中还是在其末尾，这也是第二个`\ifthenelse` 命令做的事情。

4.2.4 特殊域

下面的列表和域用于`biblatex` 给参考文献驱动和标注命令传递数据。它们由宏包自动定义，并不在`bib` 文件中使用。但从参考文献著录和标注样式的角度看，它们与`bib` 文件中的域并没有什么不同。

4.2.4.1 一般域

<datetype>dateunspecified 域 (string)

如果 <datetype>date 具有一个 EDTF 5.2.2 ‘unspecified’，该域将被设置为`yearindecade`, `yearincentury`, `monthinyear`, `dayinmonth` 或`dayinyear` 之一，这些字符串定义了 unspecified 信息的粒度 (即表示日期的不确定度，yearincentury 表示一个世纪范围内不确定，yearindecade 表示 10 年范围内不确定)。这些字符串可用于日期范围的判断，日期范围自动根据‘unspecified’ 日期创建，一个样式可以选择一种特殊方式来格式化日期。参见 § 2.3.8。例如: 一个条目的日期为:

```
@book{key,
  date     = {19uu},
  origdate = {199u}
}
```

将在.bbl 产生如下信息:

```
@book{key,
  date     = {1900/1999},
  origdate = {1990/1999}
}
```

但也会额外地将dateunspecified 域设置为‘yearincentury’，将origdateunspecified 设置为‘yearindecade’。这一信息可以用来给date 提供可能的信息‘20th century’，给origdate 提供‘The 1990s’，这一信息无法单独从日期范围推算。当‘unspecified’元信息给出时，这一自动生成的范围即已知，因此使用该范围值进行特殊的格式化相对容易。但标准样式不做此处理，96-dates.tex 给出了一些示例。

entrykey 域 (string)

bib 文件中某一项的条目关键词 (entry key)。这是一个字符串，用于biblatex 及其后端程序确定bib 文件中的某一条目。

childentrykey 域 (string)

当引用一个条目集中的子条目时，biblatex 给标注命令提供@set 父条目集的数据。这意味着entrykey 将保存父条目集的关键词。而子条目的关键词在childentrykey 域中提供。该域仅在引用条目集的某一子条目时使用。

labelnamesource 域 (literal)

保存给labelname 提供信息的域的域名，由\DeclareLabelname 确定。

labeltitlesource 域 (literal)

保存给labeltitle 提供信息的域的域名，由\DeclareLabeltitle 确定。

labeldatesource 域 (literal)

保存如下内容之一:

- 由\DeclareLabeldate 选择的日期域域名中‘date’ 前的前缀内容。
- 一个域的域名。
- 一个抄录或本地化字符串。[45]

一般情况下保存由\DeclareLabeldate 选择的日期域域名中‘date’ 前的前缀内容。例如，如果 labeldate 域是eventdate，那么labeldatesource 就是‘event’。如果\DeclareLabeldate 命令选择了date 域，labeldatesource 将会定义为一个空字符串作为‘date’ 的前缀，因为date 域名中‘date’ 前内容为空。这就是说labeldatesource 的内容可以用于构建\DeclareLabeldate 选择的域的指针。因为\DeclareLabeldate 也可以选择抄录字符串作为备选，labeldatesource 可以指向一个域或者不进行定义。记住:\DeclareLabeldate 命令可以用于选择非日期域作为备选，所以labeldatesource 也可能包含一个域名。所以，总结起来，规则如下:

[45]译者: literal 译为抄录

```
\iffieldundef{labeldatesource}
  {}% labeldate package option is not set
  {\iffieldundef{\thefield{labeldatesource}year}
    % \DeclareLabeldate resolved to either a literal/localisation
    % string or a non-date field since
    % if a date is defined by a date field, there is
    % at least a year
    {\iffieldundef{\thefield{labeldatesource}}
       {}% \DeclareLabeldate resolved to a literal/localisation string
       {}% \DeclareLabeldate resolved to a non-date field
    }
    {} % \DeclareLabeldate resolved a date field name prefix like "" or
    ↪ "orig"
  }
```

entrytype 域 (string)

条目类型 (`@book`, `@inbook` 等)，以小写字母给出。

childentrytype 域 (string)

当引用一个条目集的子条目时，`biblatex` 为标注命令提供父集条目的数据。这意味着`entrytype` 保存父条目的类型。子条目的类型则由`childentrytype` 域提供。该域仅在引用一个条目集的子条目时使用。

entrysetcount 域 (integer)

该域保存的整数用于指明一个条目集中某个集成员的位置 (起始值是 `1`)。该域仅对一个条目集的子条目有用。

hash 域 (string)

该域非常特殊，仅在姓名格式化命令中使用。它保存一个 hash 字符串，用于唯一地确定姓名列表中的各个姓名。姓名列表中的所有姓名都具有该信息。另可参见`namehash` 和`fullhash`。

namehash 域 (string)

一个 hash 字符串用于唯一确定`labelname` 列表。用于再现检查。比如，一个将再次出现的作者和编者用一个类似'idem' 的字符串代替的标注样式，可以用`\savefield` 命令保存`namehash` 域，并用于后面的`\iffieldequals` 比较中 (见 §§ 4.6.1 和 4.6.2)。`namehash` 域由截短的`labelname` 列表确定，即与`maxnames` 和`minnames` 选项相关。另可参见`hash` 和`fullhash`。

<namelist>namehash 域 (string)

类似于`namehash`，但用于 'namelist' 姓名列表。

fullhash 域 (string)

一个 hash 字符串用于唯一确定labelname 列表。该域与namehash 有两点不同:1. 产生 hash 时忽略shortauthor 和shorteditor 列表。2. 该 hash 指的是完整的列表，忽略maxnames 和minnames 选项。另可参见hash 和namehash。

<namelist>fullhash 域 (string)

类似于fullhash，但用于 'namelist' 姓名列表。

pageref 列表 (literal)

如果backref 包选项打开，该域保存条目被引用所在各页的页码。如果文档中有refsection 环境，反向引用仅包含当前参考文献节 (refsection) 内的页码信息。

sortinit 域 (literal)

该域保存排序所使用信息的首字符。

sortinithash 域 (string)

该域保存排序字符串的第一个扩展字素集 (基本上是第一个字符) 的 Unicode 排序规则算法主权重的 hash 值。可用于按照字母表顺序划分参考文献列表，该域在\bibinitsep 命令内部使用 (见 § 3.10.4)。

clonesourcekey 域 (string)

该域保存复制条目的来源条目的关键词。例如，关联条目处理中related 域所涉及的条目往往需要进行复制。

4.2.4.2 标注 (引用) 标签中使用的域

labelalpha 域 (literal)

一个与传统 BibTeX 的alpha.bst 样式产生标签类似的标签。这一标签默认由labelname 列表抽取的首字母加上出版年的最后两个数字构成。label 域可用来重设它的非数字部分 (non-numeric portion)。如果定义了label 域，biblatex 将使用它的值加上出版年的后两个数字生成labelalpha。shorthand 域也可用于重设整个标签。如果定义了该域，labelalpha 就是shorthand 域，而不是一个自动生成的标签。用户可以定义一个模板用于构建字母顺序标签 (见 § 4.5.5)，而默认的模板与上面 bibtex 标签的格式相同。一个完整的字母顺序 ('alphabetic') 标签由labelalpha 加extraalpha 域构成。注意: 使用labelalpha 和extraalpha 域需要打开labelalpha 包选项 (§ 3.1.2.3)。另可参见 § 3.10.1 节的extraalpha 和\labelalphaothers。

extraalpha 域 (integer)

当参考文献中包含同一作者同年出版的多个引文时，'alphabetic' 标注样式的标签常需要一个额外的字母加以区分。这种情况下extraalpha 域保存一个整数，该整数可用命令\mknumalph 转换成字母或以其他方式格式化。该域类似于在作者年 (author-year) 样式中extrayear 的作用。完整的 'alphabetic' 的标签

由labelalpha 加 extraalpha 构成。注意: 使用labelalpha 和 extraalpha 域需要打开包选项 labelalpha(详见 § 3.1.2.3)。另可参见 § 3.10.1 节的labelalpha 和 cmdlabelalphaothers。表7 总结了用于消除歧义的不同extra*计数器 及其记录的信息。

labelname 列表 (name)

标注标签中打印的姓名。该列表可以是shortauthor, author, shorteditor, editor, 或translator 域的复制值，正常情况以该顺序检测。如果没有作者 (authors) 和编者 (authors)，该列表是未定义的。注意: 该列表也与use<name> 相关，见 § 3.1.3。标注样式打印引用标签中的姓名时使用这一列表。提供该列表仅为方便起见，没有附加的意义。该域可以定制，详见 § 4.5.10。

labelnumber 域 (literal)

参考文献条目的序号，用于顺序编码类的样式。如果定义了shorthand 域，biblatex 不再给各条目赋予一个数值。这种情况下，labelnumber 就是 shorthand 而不是一个数字。顺序编码类的样式必须使用该域的值而不是一个计数器值。注意: 使用该域需要打开包选项labelnumber，详见 § 3.1.2.3。另可参见 § 3.1.2.1 节的defernumbers 选项。

labelprefix 域 (literal)

如果要在一个 subbibliography 文献表的所有条目前都添加一个固定的字符串，设置了\newrefcontext 命令的labelprefix 选项，那么所有受影响的labelprefix 域将提供该字符串。如果未设置前缀，相应条目的labelprefix 域是未定义的。详见 § 3.7.10 节\newrefcontext 命令的labelprefix 选项。如果定义了shorthand 域，biblatex 不会给相应条目的labelprefix 域设置前缀。这种情况下labelprefix 是未定义的。

labeltitle 域 (literal)

一篇文献可打印题名 (标题)。在一些环境中，一个样式可能需要在一些可能的标题域中选择一个标题。例如，标注样式打印短标题可能需要打印shorttitle 域 (如果它存在的话)，否则将打印title 域。构建labeltitle 时考虑的域的列表可以自定义。详见 § 4.5.10。注意: 使用extratitle 域要求打开labeltitle 包选项，详见 § 3.1.2.3。另可参见extratitle。也要注意使用extratitleyear 域也需要打开labeltitleyear 包选项，另可参见extratitleyear。

extratitle 域 (integer)

该命令有时很有用，比如在 author-title 标注样式中，用于区分标题相同的文献。当有文献具有相同的labelname 和labeltitle 时，extratitle 域保存一个整数，可以利用\mknumalph 转换为一个字母或者以其它方式格式化 (或者可以仅仅作为一个标志，用于表示将一些其它域比如日期与labeltitle 域合并)。当文献表中具有相同labeltitle 和labelname 的文献只有一篇时，该域不定义。注意: 使用extratitle 域需要打开labeltitle 包选项，详见 § 3.1.2.3。另可参见labeltitle。7 总结了各种extra* 计数器及其作用。

extratitleyear 域 (integer)

该命令有时很有用，比如在 author-title 标注样式中，用于区分标题相同年份相同但没有责任者的文献。当有文献具有相同的labeltitle 和labelyear，extratitleyear 域保存一个整数，可以利用\mknumalph 转换为一个字母或者以其它方式格式化 (或者可以仅仅作为一个标志，用于表示将一些其它域比如出版者与labelyear 域合并)。当文献表中具有相同labeltitle 和labelyear 的文献只有一篇时，该域不定义。注意: 使用extratitle 域需要打开labeltitleyear 包选项，详见 § 3.1.2.3。另可参见labeltitleyear。7 总结了各种extra* 计数器及其作用。

labelyear 域 (literal)

\DeclareLabeldate(§ 4.5.10) 命令选择的日期域的年，或者year 域，用于作者年制的标签。一个完整的作者年制的标签由labelyear 加extrayear 域构成。注意使用labelyear 和extrayear 域需要打开labeldateparts 包选项，详见 § 3.1.2.3。另可参见extrayear。

labelendyear 域 (literal)

\DeclareLabeldate (§ 4.5.10) 命令选择的日期域的终止年，如果选择的日期是一个范围。

labelmonth 域 (datepart)

\DeclareLabeldate(§ 4.5.10) 命令选择的日期域的月，或者month 域，用于作者年制的标签。注意使用labelmonth 域需要打开 labeldateparts 包选项，详见 § 3.1.2.3。

labelendmonth 域 (datepart)

\DeclareLabeldate (§ 4.5.10) 命令选择的日期域的终止月，如果选择的日期是一个范围。

labelday 域 (datepart)

\DeclareLabeldate(§ 4.5.10) 命令选择的日期域的日，或者month 域，用于作者年制的标签。注意使用 labelday 域要求打开 labeldateparts 包选项，详见 § 3.1.2.3。

labelendday 域 (datepart)

\DeclareLabeldate (§ 4.5.10) 命令选择的日期域的终止日，如果选择的日期是一个范围。

extrayear 域 (integer)

当参考文献表中包含两个或更多的具有相同作者的文献且出版年份也相同时，author-year 标注样式常需要在年后面附加一个字母以示区分。这种情况下，extrayear 域保存一个整数可以利用\mknumalph 转换为一个字母或者以其它方式格式化。当文献表中某作者的文献只有一篇或者所有该作者的文献的出版年不同时，该域不定义。完整的作者年标签由labelyear 加extrayear 域构成。注意使用labelyear 和extrayear 域需要打开labeldateparts 包选项，详见 § 3.1.2.3。另可参见labelyear。7 总结了各种extra* 计数器及其作用。

4.2.4.3 Date 的成分域

注意，可以在数据模型中定义新的日期域，这些新定义的日期域的使用方式与本节将介绍的默认数据模型类似。

bib 文件中的日期域与样式接口提供的日期域如何关联详见表10。对样式中像origdate 这样的域做判断时，使用如下代码:

```
\iffieldundef{origyear}{...}{...}
```

它将告诉你相应的日期是否已定义。下面的判断:

```
\iffieldundef{origendyear}{...}{...}
```

将告诉你相应的日期和一个 (完全确定的) 范围是否已定义。下面的判断

```
\iffieldequalstr{origendyear}{}{...}{...}
```

将告诉你相应的日期和一个无期限的 (open-ended) 范围已经定义。无期限 (Open-ended，无终止日期的) 范围由一个空的 endyear 成分表示 (而不是一个未定义的 endyear 成分)。更多示例详见 § 2.3.8 节和36 页的表3。

bib File		Data Interface	
Field	**Value (Example)**	**Field**	**Value (Example)**
date	1988	day	undefined
		month	undefined
		year	1988
		season	undefined
		endday	undefined
		endmonth	undefined
		endyear	undefined
		endseason	undefined
		hour	undefined
		minute	undefined
		second	undefined
		timezone	undefined
		endhour	undefined
		endminute	undefined
		endsecond	undefined
		endtimezone	undefined
date	1997/	day	undefined
		month	undefined
		year	1997
		season	undefined
		endday	undefined
		endmonth	undefined
		endyear	empty
		endseason	undefined
		hour	undefined
		minute	undefined
		second	undefined
		timezone	undefined
		endhour	undefined
		endminute	undefined
		endsecond	undefined
		endtimezone	undefined

urldate	2009-01-31	urlday	31
		urlmonth	01
		urlyear	2009
		urlseason	undefined
		urlendday	undefined
		urlendmonth	undefined
		urlendyear	undefined
		urlendseason	undefined
		urlhour	undefined
		urlminute	undefined
		urlsecond	undefined
		urltimezone	undefined
		urlendhour	undefined
		urlendminute	undefined
		urlendsecond	undefined
		urlendtimezone	undefined
urldate	2009-01-31T15:34:04Z	urlday	31
		urlmonth	01
		urlyear	2009
		urlseason	undefined
		urlendday	undefined
		urlendmonth	undefined
		urlendyear	undefined
		urlendseason	undefined
		urlhour	15
		urlminute	34
		urlsecond	04
		urltimezone	Z
		urlendhour	undefined
		urlendminute	undefined
		urlendsecond	undefined
		urlendtimezone	undefined
urldate	2009-01-31T15:34:04+05:00	urlday	31
		urlmonth	01
		urlyear	2009
		urlseason	undefined
		urlendday	undefined
		urlendmonth	undefined
		urlendyear	undefined
		urlendseason	undefined
		urlhour	15
		urlminute	34
		urlsecond	04
		urltimezone	+0500
		urlendhour	undefined
		urlendminute	undefined
		urlendsecond	undefined
		urlendtimezone	undefined
urldate	2009-01-31T15:34:04/	urlday	31
	2009-01-31T16:04:34		
		urlmonth	1
		urlyear	2009
		urlseason	undefined
		urlendday	31
		urlendmonth	1
		urlendyear	2009
		urlendseason	undefined
		urlhour	15
		urlminute	34

		urlsecond	4
		urltimezone	floating
		urlendhour	16
		urlendminute	4
		urlendsecond	34
		urlendtimezone	floating
origdate	2002-21/2002-23	origday	undefined
		origmonth	01
		origyear	2002
		origseason	spring
		origendday	undefined
		origendmonth	02
		origendyear	2002
		origendseason	autumn
		orighour	undefined
		origminute	undefined
		origsecond	undefined
		origtimezone	undefined
		origendhour	undefined
		origendminute	undefined
		origendsecond	undefined
		origendtimezone	undefined
eventdate	1995-01-31/1995-02-05	eventday	31
		eventmonth	01
		eventyear	1995
		eventseason	undefined
		eventendday	05
		eventendmonth	02
		eventendyear	1995
		eventendseason	undefined
		eventhour	undefined
		eventminute	undefined
		eventsecond	undefined
		eventtimezone	undefined
		eventendhour	undefined
		eventendminute	undefined
		eventendsecond	undefined
		eventendtimezone	undefined

表 10: 日期接口 (译者注:biblatex3.7 版提供的四个可解析日期接口，分别是 date，origdate，eventdate，urldate，在多数场合已经够用)

hour 域 (datepart)

该域保存`date`域的小时 (hour) 成分，当日期是一个范围时，它保存开始日期的小时成分。

minute 域 (datepart)

该域保存`date`域的分钟成分，当日期是一个范围时，它保存开始日期的分钟成分。

second 域 (datepart)

该域保存`date`域的秒钟成分，当日期是一个范围时，它保存开始日期的秒钟成分。

timezone 域 (datepart)

该域保存`date`域的时区成分，当日期是一个范围时，它保存开始日期的时区成分。

day 域 (datepart)

该域保存date 域的日成分，当日期是一个范围时，它保存开始日期的日成分。

month 域 (datepart)

该域保存数据源文件中的month 域或者date 域的月成分，当日期是一个范围时，它保存开始日期的月成分。

year 域 (datepart)

该域保存数据源文件中的year 域或者date 域的年成分，当日期是一个范围时，它保存开始日期的年成分。

season 域 (datepart)

该域保存由EDTF 5.2.5(见 § 2.3.8) 规定的date 域的季节成分，它包含一个季节本地化字符串。当日期是一个范围时，它保存开始日期的季节成分。

endhour 域 (datepart)

如果date 域中给出的日期是一个范围，该域保存结束日期的小时成分。

endminute 域 (datepart)

如果date 域中给出的日期是一个范围，该域保存结束日期的分钟成分。

endsecond 域 (datepart)

如果date 域中给出的日期是一个范围，该域保存结束日期的秒钟成分。

endtimezone 域 (datepart)

如果date 域中给出的日期是一个范围，该域保存结束日期的时区成分。

endday 域 (datepart)

如果date 域中给出的日期是一个范围，该域保存结束日期的日成分。

endmonth 域 (datepart)

如果date 域中给出的日期是一个范围，该域保存结束日期的月成分。

endyear 域 (datepart)

如果date 域中给出的日期是一个范围，该域保存结束日期的年成分。空的 (但已定义) 的endyear 成分表示无期限的日期范围。

endseason 域 (datepart)

如果date 域中给出的日期是一个范围，该域保存EDTF 5.2.5 (§ 2.3.8) 规定的结束日期的季节成分。它包含一个季节本地化字符串 (见 § 4.9.2.21)，空的 (但已定义) 的endseason 成分表示无期限的日期范围。

orighour 域 (datepart)

该域保存`origdate` 域的小时 (hour) 成分，当日期是一个范围时，它保存开始日期的小时成分。

origminute 域 (datepart)

该域保存`origdate` 域的分钟成分，当日期是一个范围时，它保存开始日期的分钟成分。

origsecond 域 (datepart)

该域保存`origdate` 域的秒钟成分，当日期是一个范围时，它保存开始日期的秒钟成分。

origtimezone 域 (datepart)

该域保存`origdate` 域的时区成分，当日期是一个范围时，它保存开始日期的时区成分。

origday 域 (datepart)

该域保存`origdate` 域的日成分，当日期是一个范围时，它保存开始日期的日成分。

origmonth 域 (datepart)

该域保存`origdate` 域的月成分，当日期是一个范围时，它保存开始日期的月成分。

origyear 域 (datepart)

该域保存`origdate` 域的年成分，当日期是一个范围时，它保存开始日期的年成分。

origseason 域 (datepart)

该域保存由EDTF 5.2.5(见 § 2.3.8) 规定的`origdate` 域的季节成分，它包含一个季节本地化字符串。当日期是一个范围时，它保存开始日期的季节成分。

origendhour 域 (datepart)

如果`origdate` 域中给出的日期是一个范围，该域保存结束日期的小时成分。

origendminute 域 (datepart)

如果`origdate` 域中给出的日期是一个范围，该域保存结束日期的分钟成分。

origendsecond 域 (datepart)

如果`origdate` 域中给出的日期是一个范围，该域保存结束日期的秒钟成分。

origendtimezone 域 (datepart)

如果`origdate` 域中给出的日期是一个范围，该域保存结束日期的时区成分。

origendday 域 (datepart)

如果`origdate` 域中给出的日期是一个范围，该域保存结束日期的日成分。

origendmonth 域 (datepart)

如果`origdate` 域中给出的日期是一个范围，该域保存结束日期的月成分。

origendyear 域 (datepart)

如果`origdate` 域中给出的日期是一个范围，该域保存结束日期的年成分。空的 (但已定义) 的`origendyear` 成分表示无期限的日期范围。

origendseason 域 (datepart)

如果`origdate` 域中给出的日期是一个范围，该域保存EDTF 5.2.5 (§ 2.3.8) 规定的结束日期的季节成分。它包含一个季节本地化字符串 (见 § 4.9.2.21)，空的 (但已定义) 的`origendseason` 成分表示无期限的`origdate` 范围。

eventhour 域 (datepart)

该域保存`eventdate` 域的小时 (hour) 成分，当日期是一个范围时，它保存开始日期的小时成分。

eventminute 域 (datepart)

该域保存`eventdate` 域的分钟成分，当日期是一个范围时，它保存开始日期的分钟成分。

eventsecond 域 (datepart)

该域保存`eventdate` 域的秒钟成分，当日期是一个范围时，它保存开始日期的秒钟成分。

eventtimezone 域 (datepart)

该域保存`eventdate` 域的时区成分，当日期是一个范围时，它保存开始日期的时区成分。

eventday 域 (datepart)

该域保存`eventdate` 域的日成分，当日期是一个范围时，它保存开始日期的日成分。

eventmonth 域 (datepart)

该域保存`eventdate` 域的月成分，当日期是一个范围时，它保存开始日期的月成分。

eventyear 域 (datepart)

该域保存`eventdate` 域的年成分，当日期是一个范围时，它保存开始日期的年成分

eventseason 域 (datepart)

该域保存由EDTF 5.2.5(见 § 2.3.8) 规定的eventdate 域的季节成分，它包含一个季节本地化字符串。当日期是一个范围时，它保存开始日期的季节成分。

eventendhour 域 (datepart)

如果eventdate 域中给出的日期是一个范围，该域保存结束日期的小时成分。

eventendminute 域 (datepart)

如果eventdate 域中给出的日期是一个范围，该域保存结束日期的分钟成分。

eventendsecond 域 (datepart)

如果eventdate 域中给出的日期是一个范围，该域保存结束日期的秒钟成分。

eventendtimezone 域 (datepart)

如果eventdate 域中给出的日期是一个范围，该域保存结束日期的时区成分。

eventendday 域 (datepart)

如果eventdate 域中给出的日期是一个范围，该域保存结束日期的日成分。

eventendmonth 域 (datepart)

如果eventdate 域中给出的日期是一个范围，该域保存结束日期的月成分。

eventendyear 域 (datepart)

如果eventdate 域中给出的日期是一个范围，该域保存结束日期的年成分。空的 (但已定义) 的eventendyear 成分表示无期限的日期范围。

eventendseason 域 (datepart)

如果eventdate 域中给出的日期是一个范围，该域保存EDTF 5.2.5 (§ 2.3.8) 规定的结束日期的季节成分。它包含一个季节本地化字符串 (见 § 4.9.2.21)，空的 (但已定义) 的eventendseason 成分表示无期限的eventdate 范围。

urlhour 域 (datepart)

该域保存urldate 域的小时 (hour) 成分，当日期是一个范围时，它保存开始日期的小时成分。

urlminute 域 (datepart)

该域保存urldate 域的分钟成分，当日期是一个范围时，它保存开始日期的分钟成分。

urlsecond 域 (datepart)

该域保存urldate 域的秒钟成分，当日期是一个范围时，它保存开始日期的秒钟成分。

timezone 域 (urldatepart)

该域保存urldate 域的时区成分，当日期是一个范围时，它保存开始日期的时区成分。

urlday 域 (datepart)

该域保存urldate 域的日成分。

urlmonth 域 (datepart)

该域保存urldate 域的月成分。

urlyear 域 (datepart)

该域保存urldate 域的年成分。

urlseason 域 (datepart)

该域保存由EDTF 5.2.5(见 § 2.3.8) 规定的 urldate 域的季节成分，它包含一个季节本地化字符串。当日期是一个范围时，它保存开始日期的季节成分。

urlendhour 域 (datepart)

如果urldate 域中给出的日期是一个范围，该域保存结束日期的小时成分

urlendminute 域 (datepart)

如果urldate 域中给出的日期是一个范围，该域保存结束日期的分钟成分

urlendsecond 域 (datepart)

如果urldate 域中给出的日期是一个范围，该域保存结束日期的秒钟成分

urlendtimezone 域 (datepart)

如果urldate 域中给出的日期是一个范围，该域保存结束日期的时区成分

urlendday 域 (datepart)

如果urldate 域中给出的日期是一个范围，该域保存结束日期的日成分

urlendmonth 域 (datepart)

如果urldate 域中给出的日期是一个范围，该域保存结束日期的月成分

urlendyear 域 (datepart)

如果urldate 域中给出的日期是一个范围，该域保存结束日期的年成分。空的 (但已定义) 的urlendyear 成分表示无期限的日期范围。

urlendseason 域 (datepart)

如果urldate 域中给出的日期是一个范围，该域保存EDTF 5.2.5 (§ 2.3.8) 规定的结束日期的季节成分。它包含一个季节本地化字符串 (见 § 4.9.2.21)，空的 (但已定义) 的urlendseason 成分表示无期限的eventdate 范围。

4.3 标注样式

参考文献标注样式 是诸如\cite 等用于打印不同类型标注的命令集。这些样式定义在后缀为cbx 的文件中。biblatex 在包末尾加载它们。注意: 一些标准标注样式的常用共享宏集放在biblatex.def 文件中。这一文件也在包末尾加载，先于选择的标注样式。它也包含有来自 § 3.8.5 节的命令的定义。

4.3.1 标注样式文件

在讨论标注样式文件提供的各个命令前，观察如下一个典型cbx 文件的总体结构:

```
\ProvidesFile{example.cbx}[2006/03/15 v1.0 biblatex citation style]

\DeclareCiteCommand{\cite}{...}{...}{...}{...}
\DeclareCiteCommand{\parencite}[\mkbibparens]{...}{...}{...}{...}
\DeclareCiteCommand{\footcite}[\mkbibfootnote]{...}{...}{...}{...}
\DeclareCiteCommand{\textcite}{...}{...}{...}{...}
\endinput
```

\RequireCitationStyle{⟨*style*⟩}

这个命令是可选的，用于加载在一些更一般样式基础上构建特殊的标注样式。它加载标注样式style.cbx。

\InitializeCitationStyle{⟨*code*⟩}

指定初始化或重设标注样式需要的任意⟨*code*⟩。这个钩子将在包加载的时候执行一次，在使用 § 3.8.8 节的\citereset 命令时则每次都执行。\citereset 命令也将重设本宏包的内部标注追踪器。它会影响 § 4.6.2 节中列出的\ifciteseen, \ifentryseen, \ifciteibid 和\ifciteidem 等判断。当使用refsection 环境时，标注追踪器重设的是当前的refsection 局部环境。

\OnManualCitation{⟨*code*⟩}指定重设部分标注样式需要的任意⟨*code*⟩。这一钩子将在使用 § 3.8.8 中的\mancite 命令时调用。它有时特别有用，可以代替像 'ibidem' 或'op. cit.' 等缩写表示的重复标注，因为当自动生成和人工产生的标注混合使用的时候这些缩写可能会有歧义。\mancite 命令也会重设宏包的内部'ibidem' 和'idem' 追踪器，进而影响 § 4.6.2 节讨论的\ifciteibid 和\ifciteidem 判断。

\DeclareCiteCommand{⟨*command*⟩}[⟨*wrapper*⟩]{⟨*precode*⟩}{⟨*loopcode*⟩}{⟨*sepcode*⟩}{⟨*postcode*⟩}

\DeclareCiteCommand*{⟨*command*⟩}[⟨*wrapper*⟩]{⟨*precode*⟩}{⟨*loopcode*⟩}{⟨*sepcode*⟩}{⟨*postcode*⟩}

这是用于定义所有标注 (引用) 命令的核心命令。它有 1 个可选参数和 5 个必选参数。⟨*command*⟩ 是要定义的命令，比如\cite。如果给出可选的⟨*wrapper*⟩ 参数，整个标注将会作为一个参数传递给⟨*wrapper*⟩，即包围 (wrapper) 命令必须要取得一个必选参数。[46] ⟨*precode*⟩ 是在标注开始时执行的任意代码。典型地，它将处理

[46] 典型的包围命令是\mkbibparens 和\mkbibfootnote。

由prenote 域提供的⟨*prenote*⟩ 参数。它可以可用来对⟨*loopcode*⟩ 所需的宏进行初始化。⟨*loopcode*⟩ 是每个条目关键词传递给⟨*command*⟩ 命令时执行的任意代码。它是打印标注标签或其它任意数据的核心代码。⟨*sepcode*⟩ 是每次执行⟨*loopcode*⟩ 完成后执行的代码。它仅在条目关键词列表传递给⟨*command*⟩ 时起作用。⟨*sepcode*⟩ 常用于插入一些分隔符，比如逗号或分号等。⟨*postcode*⟩ 是在标注结束时执行的代码。典型地，它将处理由postnote 域提供的⟨*postnote*⟩ 参数。[47] 带星号的\DeclareCiteCommand 命令定义了一个带星号的⟨*command*⟩。例如\DeclareCiteCommand*{cite} 命令将定义\cite*。[48]

\DeclareMultiCiteCommand{⟨*command*⟩}[⟨*wrapper*⟩]{⟨*cite*⟩}{⟨*delimiter*⟩}

该命令定义‘multicite’ 类命令 (见 § 3.8.3)。⟨*command*⟩ 是要定义的 multicite 命令，比如\cites。它自动在由\DeclareCiteCommand 定义的后端命令基础上构建鲁棒的命令，其中⟨*cite*⟩ 参数用于指定使用的后端命令名。注意后端命令的包围命令(封套)(即传递给\DeclareCiteCommand 命令的⟨*wrapper*⟩ 参数) 自动忽略。使用可选的⟨*wrapper*⟩ 参数作为替换。⟨*delimiter*⟩ 是列表中单个标注之间的分隔字符串。下面给出的示例是典型的\multicitedelim 命令，取自biblatex.def 中的真实定义:

```
\DeclareMultiCiteCommand{\cites}%
        {\cite}{\multicitedelim}
\DeclareMultiCiteCommand{\parencites}[\mkbibparens]%
        {\parencite}{\multicitedelim}
\DeclareMultiCiteCommand{\footcites}[\mkbibfootnote]%
        {\footcite}{\multicitedelim}
```

\DeclareAutoCiteCommand{⟨*name*⟩}[⟨*position*⟩]{⟨*cite*⟩}{⟨*multicite*⟩}

该命令为\autocite 和\autocites 类命令提供定义 (见 § 3.8.4)。要使定义生效需要打开 § 3.1.2.1 节的autocite 包选项。⟨*name*⟩ 是一个标识向包选项传递一个值。autocite 类命令是在\parencite 和\parencites 等后端命令基础上构建的。⟨*cite*⟩ 和⟨*multicite*⟩ 参数指定了使用的后端命令。⟨*cite*⟩ 参数用于\autocite，而⟨*multicite*⟩ 用于\autocites。⟨*position*⟩ 参数控制标注后的任何标点符号的处理。可能的值是 l, r, f。r 表示标点置于标注的右侧，即它不会移动。l 表示将标点移动到标注的左侧。f 在脚注中的作用类似于 r，在其它情况下则类似于 l。该参数是可选的，默认是 r。另可参见 § 4.7.5 节的\DeclareAutoPunctuation 命令和 § 3.1.2.1 节的autopunct 包选项。下面的示例取自biblatex.def 中的真实定义:

```
\DeclareAutoCiteCommand{plain}{\cite}{\cites}
\DeclareAutoCiteCommand{inline}{\parencite}{\parencites}
```

[47] 能给⟨*loopcode*⟩ 提供的参考文献数据是正在处理的条目数据。此外，第一个 (‘First’) 条目的数据可以用于⟨*precode*⟩，最后一个 (‘last’) 条目的数据可以用于⟨*postcode*⟩。‘First’ and ‘last’ 指的是标注的打印顺序。如果sortcites 包选项打开，这是经过排序处理后的顺序。注意: 任何参考文献数据无法用于⟨*sepcode*⟩。

[48] 注意: 无星号的\DeclareCiteCommand 命令也将定义隐式的定义一个带星号的标注命令，除非该标注命令前面已经定义。这只是用于提供备选。这种隐式方式定义的命令将等同于不带星号的命令。

```
\DeclareAutoCiteCommand{footnote}[l]{\footcite}{\footcites}
\DeclareAutoCiteCommand{footnote}[f]{\smartcite}{\smartcites}
```

文档导言区提供的定义可以利用如下方式随后采用 (见 § 3.2.2):

```
\ExecuteBibliographyOptions{autocite=name}
```

4.3.2 特殊域

下面的域用于向标注命令传递数据。它们不bib 文件中使用而由宏包自动定义。从标注样式的角度看，它们与bib 中的域并无区别。另可参见 § 4.2.4。

prenote 域 (literal)

作为⟨*prenote*⟩ 参数向标注命令传递。该域仅用于标注而不能用在参考文献表中。如果⟨*prenote*⟩ 参数缺省或为空，该域不定义。

postnote 域 (literal)

作为⟨*postnote*⟩ 参数向标注命令传递。该域仅用于标注而不能用在参考文献表中。如果⟨*postnote*⟩ 参数缺省或为空，该域不定义。

multiprenote 域 (literal)

作为⟨*multiprenote*⟩ 参数向 multicite 类标注命令传递。该域仅用于标注而不能用在参考文献表中。如果⟨*multiprenote*⟩ 参数缺省或为空，该域不定义。

multipostnote 域 (literal)

作为⟨*multipostnote*⟩ 参数向 ulticite 类标注命令传递。该域仅用于标注而不能用在参考文献表中。如果⟨*multipostnote*⟩ 参数缺省或为空，该域不定义。

postpunct 域 (punctuation command)

作为拖尾的标点参数隐式地向标注命令传递。该域仅用于标注而不能用在参考文献表中。如果一个标注命令后面跟着的字符不在\DeclareAutoPunctuation (§ 4.7.5) 命令的定义中，该域不定义。

4.4 数据接口

数据接口是用于格式化和打印全部参考文献数据的工具。这些工具在著录和标注样式中均可使用。

4.4.1 数据命令

本节介绍biblatex 包的主要数据接口。这些命令处理了绝大部分工作，即实际上由它们来对列表和域提供的数据进行打印。

\DeprecateField{⟨*field*⟩}{⟨*message*⟩}
\DeprecateList{⟨*list*⟩}{⟨*message*⟩}
\DeprecateName{⟨*name*⟩}{⟨*message*⟩}

用于在打印⟨*field*⟩, ⟨*list*⟩, ⟨*name*⟩ 时给出表示不允许的警告信息⟨*message*⟩。它能为那些需要在样式中修改域名的样式作者提供帮助。注意: 不允许的项只能是未在当前工作的数据模型中定义的项，⟨*field*⟩, ⟨*list*⟩ 或⟨*name*⟩ 不能出现在\DeclareDatamodelFields 命令的参数中。

\printfield[⟨*format*⟩]{⟨*field*⟩}

该命令使用由\DeclareFieldFormat 定义的格式化指令⟨*format*⟩ 打印⟨*field*⟩。如果声明了具体条目类型 (type-specific) 的格式指令，那么它将优先于设置的通用格式化指令。如果⟨*field*⟩ 未定义则不作打印。如果⟨*format*⟩ 缺省，\printfield 将尝试使用以域名为指令名的格式化指令。例如: 要打印title 域，且⟨*format*⟩ 未指定，它将尝试使用域格式化指令 title。[49] 这种情况下，任何具体类型的格式化指令将优先于通用指令。如果所有的这些格式都未定义，它将回退到 default 作为最后的选择。注意: \printfield 为格式化指令提供当前正在\currentfield 中处理的域名。

\printlist[⟨*format*⟩][⟨*start*⟩–⟨*stop*⟩]{⟨*literal list*⟩}

该命令对所有在⟨*literal list*⟩ 中的项进行循环处理，从项数⟨*start*⟩ 开始，到项数⟨*stop*⟩ 结束，包括⟨*start*⟩ 和⟨*stop*⟩(所有的列表中项以 1 开始计数)。每一项都用由\DeclareListFormat 定义的格式化指令⟨*format*⟩ 打印。如果声明了具体条目类型的格式指令，其将优先于设置的通用格式指令。如果⟨*literal list*⟩ 未定义则不作打印。如果⟨*format*⟩ 缺省，\printlist 将尝试使用以列表名作为化指令名的格式化指令。这种情况下，任何具体类型的格式化指令将优先于通用指令采用。如果所有的这些格式都未定义，它将回退到 default 作为最后的选择。⟨*start*⟩ 参数默认是 1，⟨*stop*⟩ 默认是列表中项的总数。如果项的总数大于⟨*maxitems*⟩，⟨*stop*⟩ 默认为⟨*minitems*⟩(见 § 3.1.2.1)。更多细节参见\printnames。注意: \printlist 为格式化指令提供当前正在\currentlist 中处理的域名。

\printnames[⟨*format*⟩][⟨*start*⟩–⟨*stop*⟩]{⟨*name list*⟩}该命令对所有在⟨*name list*⟩ 中的项进行循环处理，从项数⟨*start*⟩ 开始，到项数⟨*stop*⟩ 结束，包括⟨*start*⟩ 和⟨*stop*⟩(所有的列表中项以 1 开始计数)。每一项都用由\DeclareNameFormat 定义的格式化指令⟨*format*⟩ 打印。如果声明了具体条目类型的格式指令，其将优先于设置的通用格式指令。如果⟨*name list*⟩ 未定义则不作打印。如果⟨*format*⟩ 缺省，\printnames 将尝试使用以列表名为指令名的格式化指令。这种情况下，任何具体类型的格式化指令将优先于通用指令采用。如果所有的这些格式都未定义，它将回退到 default 作为最

[49] 换句话说，\printfield{title} 等价于\printfield[title]{title}。

后的选择。⟨*start*⟩ 参数默认是 1，⟨*stop*⟩ 默认是列表中项的总数。如果项的总数大于⟨*maxnames*⟩，⟨*stop*⟩ 默认为⟨*minnames*⟩(见 § 3.1.2.1)。如果你要自己制定一个范围而又要使用默认的列表格式，第一个可选参数必须给出但要留空:

```
\printnames[][1-3]{...}
```

⟨*start*⟩ 和⟨*stop*⟩ 之一可以缺省，因此下面的参数都是有效的:

```
\printnames[...][-1]{...}
\printnames[...][2-]{...}
\printnames[...][1-3]{...}
```

如果你要重设⟨*maxnames*⟩ 和⟨*minnames*⟩ 并打印整个列表，你可以在第二个可选参数中以如下方式设置`listtotal` 计数器。

```
\printnames[...][-\value{listtotal}]{...}
```

当`\printnames` 和`\printlist` 处理一个列表时，当前状态的信息可以通过 4 个计数器获知: `listtotal` 计数器保存当前列表中项的总数，`listcount` 保存当前正在处理的项的序号，`liststart` 是传递给`\printnames` 或`\printlist` 命令的 ⟨*start*⟩ 参数，`liststop` 则是 ⟨*stop*⟩ 参数。这些计数器用于列表的格式化指令。`listtotal` 也可以在`\printnames` 和`\printlist` 命令第二个可选参数中使用。注意，这些计数器仅在列表格式化指令中有意义在其它任何地方都无效。对于每类列表，都有一个具有相同名字的计数器保存该类列表的项的总数。例如，`author` 计数器保存`author` 列表中的项的总数。这些计数器类似于`listtotal`，但可用于列表格式化指令之外。还有`maxnames` ，`minnames`，`maxitems` 和`minitems` 计数器，用于保存相应的包选项的值。这些内部计数器的完整列表详见 § 4.10.5。注意: `\printnames` 为格式化指令提供当前正在`\currentname` 中处理的域名。

`\printtext[`⟨*format*⟩`]{`⟨*text*⟩`}`

该命令用于打印⟨*text*⟩，可以是可打印的文本或者产生可打印文本的任意代码。它清除插入⟨*text*⟩ 之前的标点缓存并且通知`biblatex` 打印文本已经插入。这保证了所有之前和之后的`\newblock` 和`\newunit` 命令能产生预期的作用。`\printfield`、`\printnames` 、`\bibstring` 及其相关命令都这般自动处理 (见 § 4.8)。如果一个参考文献样式需要插入抄录文本（包括来自 §§ 4.7.3 和 4.7.4 的命令），需要使用该命令来确保 block 和 unit 标点在 § 4.7.1 节中所述功能正常运转。可选参数⟨*format*⟩ 指定一个域格式指令用于格式化⟨*text*⟩。当需要把若干个域打印为某一格式的集合块，这就会很有用，比如把集合块用括号或引号包围起来。如果声明了具体条目类型的格式化指令，其将优先于设置的通用格式化指令。如果⟨*format*⟩ 缺省，那么⟨*text*⟩ 如实输出 (原样打印)。更多实用细节见第 § 4.11.7 节。

\printfile[⟨*format*⟩]{⟨*file*⟩}

该命令类似于\printtext，差别在于第二个参数是一个文件名而不是抄录文本。⟨*file*⟩ 参数必须是一个能在 TeX 搜索路径找到的有效的 LaTeX 文件。\printfile 将使用\input 来加载该⟨*file*⟩。如果指定文件不存在，\printfile 不做任何操作。可选的⟨*format*⟩ 参数指定了一个域格式化指令应用于该⟨*file*⟩。如果声明了 type"=specific 的格式指令，其将优先于设置的通用格式指令。如果⟨*format*⟩ 缺省，那么⟨*file*⟩ 如实输出 (原样打印)。注意该功能需要显式的打开 § 3.1.2.1 节的包选项loadfiles。默认情况下，\printfile 不加载任何文件。

\printdate 该命令打印条目定义在date 或month/year 域中的日期。日期格式由 § 3.1.2.1 节中的date 包选项控制。另外也可以通过调整域格式 date (见 § 4.10.4) 来进一步格式化 (比如设置字体等)。注意: 该命令与标点追踪器自动交互，不必使用\printtext 命令将其包围起来。

\printdateextra 类似于\printdate，但指定的日期域是extrayear 域。用于设计作者年制的参考文献样式。

\printlabeldate 类似于\printdate，但打印的是日期域由\DeclareLabeldate 决定。日期格式由 § 3.1.2.1 节中的labeldate 包选项控制。另外也可以通过调整域格式 labeldate (见 § 4.10.4) 来进一步格式化。

\printlabeldateextra 类似于\printlabeldate，但指定的日期域是extrayear 域，用于设计作者年制的参考文献样式。

\print<datetype>date 类似于\printdate，但打印的是日期域是条目的<datetype>date 域。日期格式由 § 3.1.2.1 节中的<datetype>date 包选项控制。另外也可以通过调整域格式 <datetype>date (见 § 4.10.4) 来进一步格式化。<datetype> 在默认数据模型中有:'' (用于date 域), 'orig', 'event' 和'url'。

\printtime 该命令打印条目定义在date 域 (见 § 2.3.8) 中的时间范围，时间格式由 § 3.1.2.1 节中的time 包选项控制。另外也可以通过调整域格式 time (见 § 4.10.4) 来进一步格式化 (比如设置字体等)。时间格式化相关内容还包括timezeros 选项，\bibtimesep 和\bibtimezonesep 宏 (§ 3.10.3)。注意: 该命令与标点追踪器自动交互，不必使用\printtext 命令将其包围起来。注意该命令打印的是独立于日期成分 (元素) 的时间范围。当<datepart>dateusetime 选项打开时，也可以与日期范围一起打印，而不各自分开打印。

\print<datetype>time 类似于\printtime，但打印的是条目的<datetype>time 域。时间格式由 § 3.1.2.1 节中的<datetype>time 包选项控制。另外也可以通过调整域格式 <datetype>time(见 § 4.10.4) 来进一步格式化。<datetype> 在默认数据模型中有:'' (用于date 域), 'orig', 'event' 和'url'。

\indexfield[⟨*format*⟩]{⟨*field*⟩}

该命令类似于\printfield，差别在于不是打印⟨*field*⟩ 而是将其添加到索引中，其格式化指令⟨*format*⟩ 由\DeclareIndexFieldFormat 命令定义。如果声明了具体条目

类型的格式指令，其将优先于设置的通用格式指令。如果⟨*field*⟩域未定义，该命令不做任何操作。如果⟨*format*⟩缺省，那么\indexfield 将采用与域名相同的格式名。这种情况下任何具体条目类型的格式指令都将优先于通用的格式指令。若所有的这些格式都未定义，那么将采用 default 格式作为最后的选择。

\indexlist[⟨*format*⟩][⟨*start*⟩–⟨*stop*⟩]{⟨*literal list*⟩}

该命令类似于\printlist，差别在于不是打印列表的项而是将其添加到索引中，其格式化指令⟨*format*⟩由\DeclareIndexListFormat 命令定义。如果声明了具体条目类型的格式指令，其将优先于设置的通用格式指令。如果⟨*literal list*⟩未定义，该命令不做任何操作。如果⟨*format*⟩缺省，那么\indexlist 将采用与列表名相同的格式名。这种情况下任何具体条目类型的格式指令都将优先于通用的格式指令。若所有的这些格式都未定义，那么将采用 default 格式作为最后的选择。

\indexnames[⟨*format*⟩][⟨*start*⟩–⟨*stop*⟩]{⟨*name list*⟩}

该命令类似于\printnames，差别在于不是打印姓名列表的项而是将其添加到索引中，其格式化指令⟨*format*⟩由\DeclareIndexNameFormat 命令定义。如果声明了具体条目类型的格式指令，其将优先于设置的通用格式指令。如果⟨*name list*⟩未定义，该命令不做任何操作。如果⟨*format*⟩缺省，那么\indexnames 将采用与列表名相同的格式名。这种情况下任何具体条目类型的格式指令都将优先于通用的格式指令。若所有的这些格式都未定义，那么将采用 default 格式作为最后的选择。

\entrydata{⟨*key*⟩}{⟨*code*⟩}
\entrydata*{⟨*key*⟩}{⟨*code*⟩}

类似\printfield 的数据命令，正常情况下应用当前正在处理的条目数据。可以使用\entrydata 在局部环境中转换应用数据。⟨*key*⟩是要局部使用的条目的关键词。⟨*code*⟩是在当前局部环境执行的任意代码。这一代码将在一个编组中执行。示例见 § 4.11.6 节。注意: 该命令自动转换语言，如果autolang 包选项打开的话。带星号的命令\entrydata* 将复制封装条目 (the enclosing entry) 的所有域，并使用域、计数器和其它以字符串‘saved’为前缀命名的资源。这在比较两个数据集时很有用。例如，在⟨*code*⟩的参数中，author 域保存了条目⟨*key*⟩的作者，而封装条目的作者保存在savedauthor 域中。author 计数器保存了⟨*key*⟩条目的author 域的姓名数量，而封装条目的作者数量由savedauthor 计数器保存。

\entryset{⟨*precode*⟩}{⟨*postcode*⟩}

该命令用在处理@set 条目集的参考文献驱动中。它将对由entryset 域指出的集的所有成员进行循环处理，对集的各个成员执行相应的驱动。这相当于对每个集成员执行 § 4.6.4 节的\usedriver 命令。⟨*precode*⟩是在集的每项处理之前执行的任意代码。⟨*postcode*⟩是在集的每项处理之后执行的任意代码。两个参数语法上必须的，但可以留空。用法示例见 § 4.11.1 节。

\DeclareFieldInputHandler{⟨*field*⟩}{⟨*code*⟩}

该命令用于定义从.bbl 读取数据所采用的域的数据输入处理器。在⟨*code*⟩内，宏\NewValue 包含了域的值。比如，要忽略出现的volumes 域，可以作:

```
\DeclareFieldInputHandler{volumes}{\def\NewValue{}}
```

一般情况下，要删除和修改域需要使用\DeclareSourcemap(见 § 4.5.3 节)，而这一替代方法在一些情形下会很有用，例如当强调的是数据的外观而不是数据本身时，因为⟨*code*⟩ 可以是任意的 TeX 代码。

\DeclareListInputHandler{⟨*list*⟩}{⟨*code*⟩}

类似于\DeclareFieldInputHandler，但用于列表。在⟨*code*⟩ 内，宏\NewValue 包含了列表的值，而\NewCount 保存列表中项的序号。

\DeclareNameInputHandler{⟨*name*⟩}{⟨*code*⟩}

类似于\DeclareFieldInputHandler，但用于姓名列表。在⟨*code*⟩ 内，宏\NewValue 包含了姓名列表的值，而\NewCount 保存列表中姓名的序号，\NewOption 保存了.bbl 文件传递的各个姓名相关的任意选项。

4.4.2 格式化指令

本节介绍 § 4.4.1 节的数据命令所需的格式化指令的定义命令。注意: 所有标注的格式定义在biblatex_.def 文件中。

\DeclareFieldFormat[⟨*entrytype, ...*⟩]{⟨*format*⟩}{⟨*code*⟩}
\DeclareFieldFormat*{⟨*format*⟩}{⟨*code*⟩}

定义域格式⟨*format*⟩。该格式化指令是由\printfield 命令执行的任意⟨*code*⟩。域的值作为仅有的第一个参数传递给⟨*code*⟩。正在处理的域名在⟨*code*⟩ 中以\currentfield 表示。如果指定一种条目类型 (⟨*entrytype*⟩)，那么格式是该类型专属的。⟨*entrytype*⟩ 可以是一个逗号分隔 (comma"=separated) 的值列表。带星的命令类似于不带星的命令，区别在于它还将清除所有对具体条目类型做的格式定义。

\DeclareListFormat[⟨*entrytype, ...*⟩]{⟨*format*⟩}{⟨*code*⟩}
\DeclareListFormat*{⟨*format*⟩}{⟨*code*⟩}

定义抄录文本列表[50] 的格式⟨*format*⟩。格式化指令是\printlist 命令处理列表中每一项时执行的任意⟨*code*⟩。当前正在处理的项作为唯一的第一参数传递给⟨*code*⟩。正在处理的文本列表名在⟨*code*⟩ 中以\currentlist 表示。如果指定了⟨*entrytype*⟩，那么格式是该类型专属的。⟨*entrytype*⟩ 参数可以是一个逗号分隔 (comma"=separated) 的值列表。注意格式化指令也会处理在列表各项间插入的标点。需要对当前项处于列表中间或者末尾进行检测，即检查listcount 是否小于或等于liststop。带星的命令类似于不带星的命令，区别在于它还将清除所有对具体条目类型做的格式定义。

[50] 译者: literal 译为抄录文本

\DeclareNameFormat[⟨*entrytype, ...*⟩]{⟨*format*⟩}{⟨*code*⟩}
\DeclareNameFormat*{⟨*format*⟩}{⟨*code*⟩}

定义姓名列表的格式⟨*format*⟩。格式化指令是\printnames 命令处理列表中每一项时执行的任意⟨*code*⟩。如果指定了⟨*entrytype*⟩，那么格式是该类型专属的。⟨*entrytype*⟩ 参数可以是一个逗号分隔 (comma"=separated) 的值列表。单个姓名的各个成分 (组成部分) 由自动创建的宏表示 (见下)。默认数据模型定义了四个成分对应于标准的 BibTeX 姓名成分参数。

`family` 姓，BibTeX 中为'last' name 成分。当一个姓名只有一个成分时 (比如 'Aristotle')，这一成分将被处理为姓。

`given` 名。注意名在 BibTeX 中为'first' name 成分。

`prefix` 尊称 (前缀)，比如 von, van, of, da, de, del, della 等。注意尊称在 BibTeX 格式文件中为'von' 成分。

`suffix` 后缀，比如 Jr, Sr 等。注意后缀在 BibTeX 格式文件中为'Jr' 成分。

数据模型'nameparts' 常量的值 (见 § 4.2.3) 在姓名的数据模型中为每个姓名成分创建了两个宏。比如，在默认数据模型中，姓名格式由如下宏定义:

```
\namepartprefix %表示尊称(前缀)部分
\namepartprefixi %表示尊称首字母
\namepartfamily %表示姓
\namepartfamilyi %表示姓首字母
\namepartsuffix %表示后缀
\namepartsuffixi %表示后缀首字母
\namepartgiven %表示名
\namepartgiveni %表示名首字母
```

如果一个姓名的某些成分没有给出，相应的宏将为空，因此可以使用，比如etoolbox 中\ifdefvoid 这类的判断来检查姓名的各个成分。正在处理的姓名列表名在⟨*code*⟩ 中以\currentname 表示。注意格式化指令也会处理在列表各项间插入的标点。需要对当前项是在列表中间或者末尾进行检测，即检查listcount 是否小于或等于liststop(见 § 3.13.4 节)。带星的命令类似于不带星的命令，区别在于它还将清除所有对具体条目类型做的格式定义。

\DeclareIndexFieldFormat[⟨*entrytype, ...*⟩]{⟨*format*⟩}{⟨*code*⟩}
\DeclareIndexFieldFormat*{⟨*format*⟩}{⟨*code*⟩}

定义域格式⟨*format*⟩。该格式化指令是由\indexfield 命令执行的任意⟨*code*⟩。域的值作为仅有的第一个参数传递给⟨*code*⟩。正在处理的域名在⟨*code*⟩ 中以\currentfield 表示。如果指定一种条目类型 (⟨*entrytype*⟩)，那么格式是该类型专属的。⟨*entrytype*⟩ 可以是一个逗号分隔 (comma"=separated) 的值列表。该命令类似于\DeclareFieldFormat，差别在于⟨*code*⟩ 处理的数据不是用于打印而是用于索引。注意\indexfield 将执行⟨*code*⟩ 本身，即⟨*code*⟩ 必须包含\index 或类似命令。

带星的命令类似于不带星的命令，区别在于它还将清除所有对具体条目类型做的格式定义。

\DeclareIndexListFormat[⟨*entrytype, ...*⟩]{⟨*format*⟩}{⟨*code*⟩}
\DeclareIndexListFormat*{⟨*format*⟩}{⟨*code*⟩}

定义抄录文本列表格式⟨*format*⟩。该格式化指令是由\indexlist 命令执行的任意⟨*code*⟩。列表中当前值作为唯一参数传递给⟨*code*⟩。正在处理的列表名在⟨*code*⟩中以\currentlist 表示。如果指定一种条目类型 (⟨*entrytype*⟩)，那么格式是该类型专属的。⟨*entrytype*⟩ 可以是一个逗号分隔 (comma"=separated) 的值列表。该命令类似于\DeclareListFormat，差别在于⟨*code*⟩ 处理的数据不是用于打印而是用于索引。注意\indexlist 将执行⟨*code*⟩ 本身，即⟨*code*⟩ 必须包含\index 或类似命令。带星的命令类似于不带星的命令，区别在于它还将清除所有对具体条目类型做的格式定义。

\DeclareIndexNameFormat[⟨*entrytype, ...*⟩]{⟨*format*⟩}{⟨*code*⟩}
\DeclareIndexNameFormat*{⟨*format*⟩}{⟨*code*⟩}

定义姓名列表格式⟨*format*⟩。该格式化指令是由\indexnames 命令执行的任意⟨*code*⟩。列表中当前值作为唯一参数传递给⟨*code*⟩。正在处理的列表名在⟨*code*⟩中以\currentname 表示。如果指定一种条目类型 (⟨*entrytype*⟩)，那么格式是该类型专属的。⟨*entrytype*⟩ 可以是一个逗号分隔 (comma"=separated) 的值列表。该命令类似于\DeclareNameFormat，差别在于⟨*code*⟩ 处理的数据不是用于打印而是用于索引。注意\indexnames 将执行⟨*code*⟩ 本身，即⟨*code*⟩ 必须包含\index 或类似命令。带星的命令类似于不带星的命令，区别在于它还将清除所有对具体条目类型做的格式定义。

\DeclareFieldAlias[⟨*entry type*⟩]{⟨*alias*⟩}[⟨*format entry type*⟩]{⟨*format*⟩}

声明⟨*alias*⟩ 作为域格式⟨*format*⟩ 的别名。如果指定一种条目类型 (⟨*entrytype*⟩)，别名是该类型专属的。⟨*format entry type*⟩ 是后端格式的条目类型。这仅在声明某一具体条目类型的格式化指令的别名时需要。

\DeclareListAlias[⟨*entry type*⟩]{⟨*alias*⟩}[⟨*format entry type*⟩]{⟨*format*⟩}

声明⟨*alias*⟩ 作为抄录文本列表格式⟨*format*⟩ 的别名。如果指定一种条目类型 (⟨*entrytype*⟩)，别名是该类型专属的。⟨*format entry type*⟩ 是后端格式的条目类型。这仅在声明某一具体条目类型的格式化指令的别名时需要。

\DeclareNameAlias[⟨*entry type*⟩]{⟨*alias*⟩}[⟨*format entry type*⟩]{⟨*format*⟩}

声明⟨*alias*⟩ 作为姓名列表格式⟨*format*⟩ 的别名。如果指定一种条目类型 (⟨*entrytype*⟩)，别名是该类型专属的。⟨*format entry type*⟩ 是后端格式的条目类型。这仅在声明某一具体条目类型的格式化指令的别名时需要。

`\DeclareIndexFieldAlias[`⟨*entry type*⟩`]{`⟨*alias*⟩`}[`⟨*format entry type*⟩`]{`⟨*format*⟩`}`

声明⟨*alias*⟩ 作为域格式⟨*format*⟩ 的别名。如果指定一种条目类型 (⟨*entrytype*⟩)，别名是该类型专属的。⟨*format entry type*⟩ 是后端格式的条目类型。这仅在声明某一具体条目类型的格式化指令的别名时需要。

`\DeclareIndexListAlias[`⟨*entry type*⟩`]{`⟨*alias*⟩`}[`⟨*format entry type*⟩`]{`⟨*format*⟩`}`

声明⟨*alias*⟩ 作为抄录文本列表格式⟨*format*⟩ 的别名。如果指定一种条目类型 (⟨*entrytype*⟩)，别名是该类型专属的。⟨*format entry type*⟩ 是后端格式的条目类型。这仅在声明某一具体条目类型的格式化指令的别名时需要。

`\DeclareIndexNameAlias[`⟨*entry type*⟩`]{`⟨*alias*⟩`}[`⟨*format entry type*⟩`]{`⟨*format*⟩`}`

声明⟨*alias*⟩ 作为姓名列表格式⟨*format*⟩ 的别名。如果指定一种条目类型 (⟨*entrytype*⟩)，别名是该类型专属的。⟨*format entry type*⟩ 是后端格式的条目类型。这仅在声明某一具体条目类型的格式化指令的别名时需要。

4.5 定制

4.5.1 关联条目

关联条目相关功能由如下部分构成:

- 条目中的特殊域用于建立和描述关系
- 本地化字符串作为关联数据的前缀 (可选)
- 抽取和打印关联数据的宏
- 用于本地化字符串和关联数据格式化的格式

特殊域是`related`, `relatedtype`, `relatedstring` 和 `relatedoptions`:

related 与当前条目存在某种程度关联性的条目关键词列表[51]。注意: 条目关键词[52] 的顺序很重要。来自多个关联条目的数据是按该域中关键词的顺序打印的。

relatedtype 关联性类型。主要用于三个目的: 第一，如果该域的值解析为一个本地化字符串的标识，那么得到的本地化字符串将在来自关联条目的数据之前打印。第二，如果存在名为 `related:`⟨*relatedtype*⟩ 的宏，它将用于格式化来自关联条目的数据，如果宏不存在，则使用 `related:default` 宏。最后，如果存在名为 `related:`⟨*relatedtype*⟩ 的格式，它将用来格式化本地化字符串和关联条目数据。如果没有具体类型的格式，那么使用 `related` 格式。

relatedstring 如果一个条目包含该域，如果该域的值解析为一个本地化字符串的标识，那么本地化字符串的键值将在来自关联条目的数据之前打印。如果该域没有指定一个本地化键，则原样打印该域的值。如果`relatedtype` 和 `relatedstring` 都出现在条目中，`relatedstring` 用于数据之前的字符串 (但`relatedtype` 仍然用于确定打印数据时的格式和宏)。

[51] 译者: 这里 separated list 译为分离列表，列表

[52] 译者: 这里的 key 关键词就是是条目关键词，引用关键词，即 bibtex 键

relatedoptions 设置在关联条目上的各条目的选项列表 (实际上，是对关联条目的副本的设置，关联条目的副本作为数据源，而关联条目本身不做任何的改变，因为它自身有可能被引用)。

关联条目功能由 § 3.1.2.1 节的related 包选项默认启用。来自关联条目的相关信息数据通过一个调用\usebibmacro{related} 包含进来。标准样式调用该宏直到每个驱动结束。样式作者应该确保 (留意) 作为relatedtype 域值的本地化字符串的存在，比如 translationof 或者可能的 translatedas。本地化键 (关键词) 的⟨*relatedtype*⟩s 复数形式可以识别。在related 中给出的超过 1 个的键的对应字符串都会打印。用于打印由⟨*relatedtype*⟩ 关联的条目的参考文献宏和格式化指令应以 related:⟨*relatedtype*⟩ 为名进行定义。biblatex.def 包含了通用的关联类型的宏和格式，可以作为模板。特别的，\entrydata* 命令在这些宏中是必须的，因为要获取关联条目的数据。应用了该功能的条目数据可以在biblatex 发布的示例文件biblatex-examples.bib 中找到。针对该功能的一些用于控制关联条目间的分隔符的具体格式化宏见 § 4.10.1。

4.5.2 数据源集

在循环等操作中，能够定义数据源的域名的集是有用的。另外，biber 可以利用这种集来对某些特定的数据源域集应用选项或者执行操作。下面的宏允许用来定义任意的数据源域集，这些数据源域以etoolbox 列表的形式提供给biblatex，并通过.bcf 文件中传递给biber 。

\DeclareDatafieldSet{⟨*name*⟩}{⟨*specification*⟩}

声明一个数据源域的集，其名为⟨*name*⟩。

name=⟨*set name*⟩

集的名。

⟨*specification*⟩ 是 1 个或更多的\member(成员) 项:

\member

fieldtype=⟨*fieldtype*⟩

datatype=⟨*datatype*⟩

field=⟨*fieldname*⟩

一个\member 说明将域添加到集中。域可以由数据模型⟨*fieldtype*⟩ 和/或 ⟨*datatype*⟩ 指定 (见 § 4.5.4)。或者，域也可以通过使用⟨*field*⟩ 选项显式地以名字添加。一旦完成定义，集就以etoolbox 列表的形式存在，命名为\datafieldset‘setname’ 并通过.bcf 文件传递给biber 。

下面的示例就是biblatex 为姓名域和标题域定义的默认集:

```
\DeclareDatafieldSet{setnames}{
  \member[datatype=name, fieldtype=list]
}

\DeclareDatafieldSet{settitles}{
  \member[field=title]
  \member[field=booktitle]
  \member[field=eventtitle]
  \member[field=issuetitle]
  \member[field=journaltitle]
  \member[field=maintitle]
  \member[field=origtitle]
}
```

这将以etoolbox 列表形式定义\datafieldsetsetnames 和\datafieldsetsettitles 宏用来包含指定的数据源域成员的名称。[53]

4.5.3 数据动态修改

对自动生成或者无法控制的参考文献数据源进行修改在某种程度上会是一个问题。因此，biber 提供了对它所读取的数据进行修改的能力，这样你可以对源数据流进行修改而不必实际改变它。这种改变可以在biber 的配置文件 (见biber 文档) 中定义，或者通过biblatex 宏进行定义，通过宏定义的方法你可以在样式中或者以全局定义的方式，将修改应用在具体的文档中。

源映射发生在数据解析过程中，因此也在诸如继承和排序等任何其它操作之前。

源映射可以在不同的层 (“levels”) 进行定义，各层以某一定义的顺序进行处理。见biblatex 手册，考虑如下这些宏:

\DeclareSourcemap 命令定义的用户层 (user-level) 映射 →
在biber 配置文件定义的用户层 (user-level) 映射 (见 biber 文档)→
\DeclareStyleSourcemap 定义的样式层 (style-level) 映射 →
\DeclareDriverSourcemap 定义的驱动层 (driver-level) 映射

\DeclareSourcemap{⟨*specification*⟩}

定义源数据修改 (映射) 规则，可以用于执行如下任务或其任意组合:

- 将数据源条目类型映射为其它类型
- 将数据源域映射为其它域

[53] 译者: 应用时要注意集的定义名称和使用的名称是不同的。

- 给条目添加新域
- 从条目移除域
- 用标准的 Perl 正则表达式匹配和替换，修改域的内容。
- 将上述操作限制在来自特定数据源的条目，这些特定数据源在`\addresource`宏中定义。
- 将上述操作限制在某些条目类型。
- 将上述操作限制在某一特定的参考文献节。

⟨*specification*⟩ 是一个不限数量的`\maps` 指令的列表，这些指令说明了应用于某一特定数据源类型的映射规则的容器 (§ 3.7.1)。为实现良好的代码显示效果，可以自由使用空格、制表符、行末符号来整理⟨*specification*⟩ 中的代码。但空行是不允许的。这一命令仅能用于导言区并且只能使用一次—后面的命令将覆盖前面的定义。

`\maps[`⟨*options*⟩`]{`⟨*elements*⟩`}`

包含`\map` 元素的有序集，每个`\map` 都是应用于数据源的映射步的逻辑相关集。⟨*options*⟩ 包括:

`datatype=bibtex, biblatexml` default: `bibtex`

包含的`\map` 应用的数据源的类型 (见 § 3.7.1)

`overwrite=true, false` default: `false`

具体说明一个映射规则是否允许覆盖条目中已经存在数据。如果该选项未指定，默认是 `false`。简易形式`overwrite` 等价于 `overwrite=true`。

`\map[`⟨*options*⟩`]{`⟨*restrictions,steps*⟩`}`

一个包含有序的映射操作`\steps` 的容器，可以限制在特定的条目类型或者数据源上。这是一个编组元素允许一组映射操作应用到具体的条目类型或数据源上。映射操作必须包含在`\map` 元素内。⟨*options*⟩ 包括:

`overwrite=true, false`

与父元素`\maps` 的相同的选项。该选项允许 map 层级的覆盖操作。如果该选项未指定，默认是父元素`\maps` 的选项值。简易形式`overwrite` 等价于 `overwrite=true`。

`foreach=`⟨*loopval*⟩

遍历`\map` 内的`\steps`，将包含在⟨*loopval*⟩ 中的逗号分隔的各个值设置到特殊变量`$MAPLOOP` 中。⟨*loopval*⟩ 可以是一个显式的逗号分隔的值列表，也可以是由`\DeclareDatafieldSet` (见 § 4.5.2) 定义的任意数据域集的集名，从中获取数据源域并解析为一个逗号分隔的值列表。⟨*loopval*⟩ 以列表中的顺序处理。这使得一组`\steps` 操作可以遍历⟨*loopval*⟩ 的每个值。使用 regexp 映射，可以创建一个 CSV 域来配合该功能使用。特殊变量`$MAPUNIQ` 也可以在`\steps` 中用来随机生成一个唯一的字符串。这可以用于创建新条目的关键词。例如:

```
\DeclareSourcemap{
```

```
  \maps[datatype=bibtex]{
    \map[overwrite, foreach={author,editor, translator}]{
      \step[fieldsource=\regexp{$MAPLOOP}, match={Smith}, replace={Jones
    ↪ }]
    }%数据源域将遍历foreach定义的这些域，而这些域由宏\regexp{$MAPLOOP}
    ↪  表示
  }
}
```

refsection=⟨*integer*⟩

将包含的\step 命令应用于序号等于⟨*refsection*⟩ 的参考文献节的条目上。

\perdatasource{⟨*datasource*⟩}

将\map 元素内所有的\steps 限制在来自名为⟨*datasource*⟩ 的条目上。⟨*datasource*⟩ 名应是在\addresource 宏给出的文档的参考文献数据源文件名。一个\map 元素内允许出现多个\perdatasource 约束。

\pertype{⟨*entrytype*⟩}

将\map 元素内所有的\steps 限制在来自类型为⟨*entrytype*⟩ 的条目上。一个\map 元素内允许出现多个\pertype 约束。

\pernottype{⟨*entrytype*⟩}

将\map 元素内所有的\steps 限制在来自类型不是⟨*entrytype*⟩ 的条目上。一个\map 元素内允许出现多个\pernottype 约束。

\step[⟨*options*⟩]

一个映射步。每一步都顺序地应用于每个相关条目上，其中“相关”意为这些条目满足前述指定的数据源类型，条目类型，数据源文件限制。对条目应用每一映射步都在前面映射步完成之后。映射步执行的映射操作由如下选项 (⟨*option*⟩s) 确定:

typesource=⟨*entrytype*⟩

typetarget=⟨*entrytype*⟩

fieldsource=⟨*entryfield*⟩

notfield=⟨*entryfield*⟩

fieldtarget=⟨*entryfield*⟩

match=⟨*regexp*⟩

notmatch=⟨*regexp*⟩

replace=⟨*regexp*⟩

fieldset=⟨*entryfield*⟩

`fieldvalue`=⟨*string*⟩

`entryclone`=⟨*clonekey*⟩

`entrynew`=⟨*entrynewkey*⟩

`entrynewtype`=⟨*string*⟩

`entrytarget`=⟨*string*⟩

`entrynull`=`true`, `false` default: `false`

`append`=`true`, `false` default: `false`

`final`=`true`, `false` default: `false`

`null`=`true`, `false` default: `false`

`origfield`=`true`, `false` default: `false`

`origfieldval`=`true`, `false` default: `false`

`origentrytype`=`true`, `false` default: `false`

对于所有的布尔映射选项，简易形式`option` 等价于 `option=true`。一个映射步的应用规则如下:

- 如果设置 `entrynew`，将创建一个条目关键词为 `entrynewkey` 的新条目，条目的类型在选项 `entrynewtype` 中给出。这一条目仅在当前条目的处理过程中有效，能以 `entrytarget` 进行引用。在 `entrynewkey` 中，可以使用标准的 Perl 正则表达式来引用从之前的 `match` 步获取的匹配字符串。
- 当一个 `fieldset` 步设置的 `entrytarget` 为一个由 `entrynew` 创建的条目的关键词，域设置的目标将是 `entrytarget` 条目而不是当前正在处理的条目。这使得用户可以创建新的条目并且设置它的域。
- 如果设置 `entrynull`，`\map` 过程立即终止，当前条目不创建。如同该条目不存在于数据源。显然，需要利用前面的映射步来选择需要应用该操作的条目。
- 如果设置 `entryclone`，将创建一个名为 `clonekey` 的副本条目。显然这会影响作者年等样式中的标签生成，使用需谨慎。副本仅在当前条目的处理过程中有效，将其关键词作为 `entrytarget` 的值可对其进行修改。在 `clonekey` 中，可以使用标准的 Perl 正则表达式来引用从之前的 `match` 步获取的匹配字符串。
- 将 `typesource` ⟨*entrytype*⟩ 修改到 `typetarget` ⟨*entrytype*⟩，如果定义存在的话。如果 `final` 为 `true`，且条目的⟨*entrytype*⟩ 不是 `typesource`，父元素`\map` 将立即终止。
- 将 `fieldsource` ⟨*entryfield*⟩ 修改到 `fieldtarget`，如果定义存在的话。如果 `final` 为 `true`，且条目中不存在 `fieldsource` ⟨*entryfield*⟩，父元素`\map` 将立即终止。
- 如果使用 `notfield`，仅当⟨*entryfield*⟩ 不存在时才会应用该映射步。
- 如果定义了 `match` 但没有定义 `replace`，仅当 `fieldsource` ⟨*entryfield*⟩ 匹配 `match` 正则表达式时 (如果使用 `notmatch` 则是不匹配) 应用该映射步[54]。可以

[54] 正则表达式是 Perl 5.16 全集正则表达式。这意味着需要处理特殊字符，见后面的示例

使用圆括号获取匹配内容，并在后面 fieldvalue 设置中引用 ($1...$9)。这使得可以将一些域的部分内容提取出来放入其它一些域中。

- 如果定义了 match 和 replace，对 fieldsource ⟨*entryfield*⟩ 的值执行正则表达式匹配和替换操作。
- 如果定义了 fieldset，它的值⟨*entryfield*⟩ 将由进一步给出的选项来指定。如果 overwrite 是 false 且该域值已经存在，则该映射步将忽略。如果该映射步的 final 也是 true，则父元素 map 在此处终止。如果 append 是 true，则设置的值将添加到当前⟨*entryfield*⟩ 的值中。值仅由如下之一的必选参数设置:
  - fieldvalue — fieldset ⟨*entryfield*⟩ 设置为 fieldvalue ⟨*string*⟩。
  - null — The fieldset ⟨*entryfield*⟩ 忽略，如果它不存在于数据源中。
  - origentrytype — The fieldset ⟨*entryfield*⟩ 设置为前面最近的 typesource ⟨*entrytype*⟩ 名称。
  - origfield — The fieldset ⟨*entryfield*⟩ 设置为前面最近的 fieldsource ⟨*entryfield*⟩ 域名。
  - origfieldval — The fieldset ⟨*entryfield*⟩ 设置为前面最近的 fieldsource ⟨*entryfield*⟩ 域值。

使用 BibTeX 数据源，可能设置虚拟域entrykey 作为 fieldsource，它就是条目的引用关键词 (即 bibtex 键)。使用biblatexml 数据源的话，entrykey 是正常属性可以像任何其它属性一样引用。自然，该域'field' 是不能 (用修改操作 fieldset, fieldtarget，replace) 改变的。

\DeclareStyleSourcemap{⟨*specification*⟩}

该命令设置由样式使用的数据源映射。这种映射概念上是与由\DeclareSourcemap 定义的用户层映射分离的，并且在用户层映射之后应用。其语法与\DeclareSourcemap 完全相同。该命令使得样式作者可以定义样式专属的映射而不会与用户层和驱动层的映射相冲突。这一命令可以在样式文件中多次使用，并根据定义的顺序进行映射操作。

\DeclareDriverSourcemap[⟨*datatype=driver*⟩]{⟨*specification*⟩}

该命令为具体的⟨*driver*⟩ 设置驱动默认的数据源映射。这种映射概念上是与由\DeclareSourcemap 定义的用户层映射和\DeclareStyleSourcemap 定义的样式层映射分离的。它们由一些映射组成，也作为驱动组成部分。用户一般不需要修改它。驱动默认映射在用户层和样式层映射之后应用。默认的设置在附录 § A 介绍。⟨*specification*⟩ 与\DeclareSourcemap 的相同，但没有\maps 元素: ⟨*specification*⟩ 是一些\map 元素的列表，因为每个\DeclareDriverSourcemap 仅应用于某一具体的条目类型的驱动中。默认的定义见附录 § A 中的示例。

下面给出一些数据源映射的示例:

```
\DeclareSourcemap{
```

```
  \maps[datatype=bibtex]{
    \map{
      \perdatasource{example1.bib}
      \perdatasource{example2.bib}
      \step[fieldset=keywords, fieldvalue={keyw1, keyw2}]
      \step[fieldsource=entrykey]
      \step[fieldset=note, origfieldval]
    }
  }
}
```

这一示例是对能在 examples1.bib 和 examples2.bib 文件中找到的条目进行处理，增加一个keywords 域，域值为‘keyw1, keyw2’，并且设置note 域值为条目关键词。

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map{
      \step[fieldsource=title]
      \step[fieldset=note, origfieldval]
    }
  }
}
```

这一示例将title 域复制给note 域，除非note 域已经存在。

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map{
      \step[typesource=chat, typetarget=customa, final]
      \step[fieldset=type, origentrytype]
    }
  }
}
```

任何chat 类型的条目将变成为customa 条目类型，并且自动设置type 域为‘chat’，除非条目中type 域已经存在 (overwrite 默认是 false)。这一映射仅对@chat 类型的条目进行操作，因为第一步中设置了 final，所以当 typesource 不匹配时，\map 过程立即终止。

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map{
      \perdatasource{examples.bib}
```

```
      \pertype{article}
      \pertype{book}
       \step[fieldset=abstract, null]
       \step[fieldset=note, fieldvalue={Auto-created this field}]
    }
  }
}
```

任何来自 examples.bib 数据源，类型为@article 或@book 的条目的abstract 域将删除，增加一个note 域，域值为‘Auto-created this field’。

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map{
       \step[fieldset=abstract, null]
       \step[fieldsource=conductor, fieldtarget=namea]
       \step[fieldsource=gps, fieldtarget=usera]
    }
  }
}
```

这将删除所有条目的abstract 域，将conductor 域修改到namea 域，将gps 修改到usera 域中。

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map{
       \step[fieldsource=pubmedid, fieldtarget=eprint, final]
       \step[fieldset=eprinttype, origfield]
       \step[fieldset=userd, fieldvalue={Some string of things}]
    }
  }
}
```

仅对有pubmedid 域的条目做映射，将pubmedid 域映射到eprint 域中，设置eprinttype 值为pubmedid 域的值，并且设置userd 的值为字符串‘Some string of things’。

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map{
       \step[fieldsource=series,
             match=\regexp{\A\d*(.+)},
```

```
            replace=\regexp{\L$1}]
    }
  }
}
```

这里的正则表示式对series 域值进行处理，match 中 '\A' 确定匹配的起始位置为字符串开始处，\d 匹配一个数字，* 表示做\d 匹配 0 次或多次，即把所有从字符串起始位置的数字匹配出来，(标记一个子表达式开始，) 标记一个表达式开始结束，. 匹配除换行符外的任意字符，+ 表示做. 匹配 1 次或任意多次，即将域值数字结束后的所有字符标记为一个子表达式。replace 中\L 表示将后面所有的字符都转换为大写或小写，$1表示引用第一组括号内表达式匹配的字符即 match 中 () 内匹配的字符，即将原来series 域值中起始数字之外的字符全部转换为小写或大写[55]。因为正则表达式常包含各种特殊字符，最好将其用宏\regexp 包含起来，这样就能将其正确的传递给biber 。

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map{
       \step[fieldsource=maintitle,
             match=\regexp{Collected\s+Works.+Freud},
             final]
       \step[fieldset=keywords, fieldvalue=freud]
    }
  }
}
```

如果一个条目中maintitle 域能匹配一个特殊的正则表达式，则将设置一个特殊的 keyword，利用这一 keyword 可以将一些特定项构成一个参考文献节。(其中'\s' 匹配任何空白字符，包括空格、制表符、换页符。)

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map{
       \step[fieldsource=lista, match=\regexp{regexp}, final]
       \step[fieldset=lista, null]
    }
  }
}
```

如果一个条目的lista 匹配正则表达式'regexp'，则删除该条目。

[55]译者: 正则表达式参考: 正则表达式系统教程和精通正则表达式，\A，\L，$1见精通正则表达式 287,290,298 页

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map[overwrite=false]{
       \step[fieldsource=author]
       \step[fieldset=editor, origfieldval, final]
       \step[fieldsource=editor, match=\regexp{\A(.+?)\s+and.*}, replace
    ↪ ={$1}]
    }
  }
}
```

对于任何有author 域的条目，将editor 设置为与author 相同，但如果editor 已经存在则不设置并终止映射。如果完成设置，对editor 进一步匹配和替换。(正则表达式的作用是截取姓名列表中第一个姓名。其 match 规则为从字符串开始就标记子表达式开始，在遇到空白字符和 and 之前标记子表达式结束，子表达式匹配任意字符，利用问号结束贪婪过程，即在第一个空白字符和 and 之前结束匹配。replace 则表示提取第一个子表达式的匹配结果。)

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map{
       \step[fieldsource=author,
             match={Smith, Bill},
             replace={Smith, William}]
       \step[fieldsource=author,
             match={Jones, Baz},
             replace={Jones, Barry}]
    }
  }
}
```

这里，对同一个域使用多个匹配/替换来调整一些姓名。记住，一个map 元素中\step 步是按顺序处理的，后面的\step 步是在前面各步处理结果基础上再处理。注意，当匹配字符中没有一些特殊的字符，则没有必要用\regexp 来保护正则表达式。需要注意，因为在 LaTeX 中保护正则表达式的难度，不要在\regexp 中出现空格，而要使用空格对应的代码。比如:

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map{
       \step[fieldsource=author,
             match=\regexp{Smith,\s+Bill},
             replace=\regexp{Smith,\x20William}]
```

```
      \step[fieldsource=author,
            match=\regexp{Jones,\s+Baz},
            replace=\regexp{Jones,\x20Barry}]
    }
  }
}
```

其中，使用了十六进制符号命令‘\x20’来充当替换字符串中的空格。

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map[overwrite]{
      \step[fieldsource=author, match={Doe,}, final]
      \step[fieldset=shortauthor, origfieldval]
      \step[fieldset=sortname, origfieldval]
      \step[fieldsource=shortauthor,
            match=\regexp{Doe,\s*(?:\.|ohn)(?:[-]*)(?:P\.|Paul)*},
            replace={Doe, John Paul}]
      \step[fieldsource=sortname,
            match=\regexp{Doe,\s*(?:\.|ohn)(?:[-]*)(?:P\.|Paul)*},
            replace={Doe, John Paul}]
    }
  }
}
```

仅对author 匹配有‘Doe,’的条目做处理。首先将author 域复制给shortauthor 和sortname 域，当这两个域已存在仍然进行覆盖。然后对这两个新域做匹配和替换操作，用以规范化一个特定的姓名，因为该名字在数据源中可能有一些不同的形式。其中‘(?:pattern)’表示匹配 pattern 但不获取，第一个括号中只要是. 或 ohn 结尾即可匹配，第二个括号中只要有-字符即可匹配，第三个括号中以 P. 或 Paul 结尾即可匹配。

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map[overwrite]{
      \step[fieldsource=verba, final]
      \step[fieldset=verbb, fieldvalue=/, append]
      \step[fieldset=verbb, origfieldval, append]
      \step[fieldsource=verbb, final]
      \step[fieldset=verbc, fieldvalue=/, append]
      \step[fieldset=verbc, origfieldval, append]
    }
  }
```

```
}
```

该例验证了 step 步的顺序特性和append 选项。如果一个条目有verba 域，首先在verbb 域中增加一个斜杠，然后verba 域值添加到verbb 中。然后在verbc 中添加一个斜杠和verbb 域的值。

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map[overwrite]{
      \step[fieldset=autourl, fieldvalue={http://scholar.google.com/
    ↪ scholar?q="}]
      \step[fieldsource=title]
      \step[fieldset=autourl, origfieldval, append]
      \step[fieldset=autourl, fieldvalue={"+author:}, append]
      \step[fieldsource=author, match=\regexp{\A([^,]+)\s*,}]
      \step[fieldset=autourl, fieldvalue={$1}, append]
      \step[fieldset=autourl, fieldvalue={&as_ylo=}, append]
      \step[fieldsource=year]
      \step[fieldset=autourl, origfieldval, append]
      \step[fieldset=autourl, fieldvalue={&as_yhi=}, append]
      \step[fieldset=autourl, origfieldval, append]
    }
  }
}
```

该例假设使用 § 4.5.4 中的数据模型创建了一个名为autourl 的域用来保存比如由条目内容自动创建的 Google 学术搜索地址。逐步从条目中抽取信息并构建 URL。它验证了可以在后面所有的 fieldvalue 设置中引用前面最近匹配得到的结果，如果author 域格式是'family, given'，可以从中抽取姓 (family name)。结果域可以用作文献表中文献的标题的一个超链接。(其中 '[^,]+' 匹配不包含, 的任意字符并获取为\$1。)

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map{
      \step[fieldsource=title, match={A Title}, final]
      \step[entrynull]
    }
  }
}
```

当条目的title 匹配'A Title' 时，忽略该条目 (它将不出现在参考文献数据中)。

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map{
      \pernottype{book}
      \pernottype{article}
      \step[entrynull]
    }
  }
}
```

当条目不是@book 或@article 类型将被忽略。

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map{
      \perdatasource{biblatex-examples.bib}
      \step[entryclone={rel-}]
    }
  }
}
```

数据源中的条目将被复制。复制条目的关键词为原条目关键词加上 entryclone 参数给出的前缀。复制的条目在文档中以其新的关键词进行引用。这种类型的映射需要谨慎使用，因为标签生成可能产生问题，比如在作者年制使用extrayear 时。一种应用情况是在顺序编码制中多个文献表包含相同的条目时需要为这一相同同条目生成不同的序号标签。当需要对一个条目生成不同的标签时，这是很麻烦的，而创建具有不同关键词的副本条目能解决这一问题。

biblatexml 数据源比 BibTeX 更具结构性，因为它们是 XML。源的映射也可以对其处理，但源和目标的设置需要支持 XPath 1.0 路径以便能够处理结构化数据。域的设置可以利用上述的示例，必要时可以转换为 XPath 1.0 的内部访问方式。比如:

```
\DeclareSourcemap{
  \maps[datatype=biblatexml]{
    \map{
   \step[fieldsource=\regexp{./bltx:names[@type='author']/bltx:name[2]/
    ↪ bltx:namepart[@type='family']},
      match=\regexp{\ASmith},
      replace={Jones}]
    }
    \map{
      \step[fieldsource=editor, fieldtarget=translator]
```

```
    }
    \map{
      \step[fieldsource=\regexp{./bltx:names[@type='editor']},
            fieldtarget=\regexp{./bltx:names[@type='translator']}]
    }
    \map{
      \step[fieldset=\regexp{./bltx:names[@type='author']/bltx:name[2]/
    ↪ @useprefix},
            fieldvalue={false}]
    }
  }
}
```

这些映射，分别为:

- 将author 域的第二个姓名的姓'Smith' 替换为'Jones'。
- 将editor 域值设置到translator。
- 将editor 域值设置到translator，但使用显式的 XPaths。这和第二个映射操作结果是一样的。
- 将author 域的姓名列表useprefix 选项设置为'false'。

4.5.4 数据模型规范

biblatex 使用的数据模型包括 4 个主要元素:

- 字符串和字符串列表常量规范
- 有效条目类型规范
- 域及其类型、数据类型和其它特殊标识的规范
- 在什么条目中什么域有效的规范
- 可用于根据数据模型验证数据的约束规范

默认数据模型在核心biblatex 文件blx-dm.def 中定义，使用本节给出的宏。默认数据模型详见 § 2 节。数据模型由biblatex 及后端在内部使用。在实际中，改变数据模型意味着为数据源定义条目类型和域，并可根据数据模型进行验证。自然，这通常只有在样式能支持新的条目类型时有用，并且也可能引起样式间的可移植性问题。(尽管这一问题可以使用 § 4.5.3 节介绍的动态数据修改来解决。)

根据数据模型验证意味着，将数据源映射到数据模型后，biber (使用--validate_datamodel 选项) 可以检查:

- 所有条目类型是否都有效
- 所有的域对于其所属的条目类型是否有效

- 域是否都满足指定的不同格式约束

重定义数据模型可以在多个地方进行。样式作者可以创建一个.dbx 文件包含需要的数据模型宏，该文件将在biblatex 宏包根据style 选项加载.cbx 和.bbx 样式文件后搜索.dbx 文件来加载[56]。如果不使用style 选项，而使用citestyle 和bibstyle 选项，宏包将会搜索文件<citestyle>.dbx 和<bibstyle>.dbx 并加载。另一种方式，数据模型文件名可以不同于样式选项名，但扩展名必须是.dbx，该文件由datamodel 包选择加载。在加载完样式的数据模型文件之后，biblatex 寻找并加载biblatex-dm.cfg 配置文件，类似于加载biblatex.cfg 的方式。总之，最终数据模型由从这些地方加入的数据模型确定，顺序是:

```
blx-dm.def→
  <datamodel option>.dbx →
    <style option>.dbx →
      <citestyle option>.dbx and <bibstyle option>.dbx →
        biblatex-dm.cfg
```

在导言区使用宏来加载数据模型是不行的，因为导言内容是在biblatex 根据数据模型定义关键的内部宏之后读取。在正文中定义的任何数据模型宏都将被忽略并给出一个警告。数据模型由如下宏来定义:

\DeclareDatamodelConstant[⟨*options*⟩]{⟨*name*⟩}{⟨*constantdef*⟩}

声明⟨*name*⟩ 作为一个数据模型常量，其定义为⟨*constantdef*⟩。这种常量通常由biber 在内部使用。

type=string, list　　default: string

一个常量可以是一个简单的字符串 (默认情况下，如果⟨*type*⟩ 选项忽略的话) 或者一个逗号分隔的字符串列表。

\DeclareDatamodelEntrytypes[⟨*options*⟩]{⟨*entrytypes*⟩}

声明⟨*entrytypes*⟩ 的一个逗号分隔列表，表示数据模型中的有效条目类型。依旧如同在 TeX 的 csv 列表中，请确保每个元素都紧跟着一个逗号或者结束的括号，即不需要额外的空格。

skipout=true, false　　default: false

该条目类型不输出到.bbl 中。通常用于后端处理和使用的特殊条目类型比如@xdata 等。

\DeclareDatamodelFields[⟨*options*⟩]{⟨*fields*⟩}

声明⟨*fields*⟩ 的一个逗号分隔列表，表示数据模型中与逗号分隔⟨*options*⟩ 列表相关的有效域。⟨*type*⟩ 和⟨*datatype*⟩ 选项是必须的，全部有效选项如下:

[56]译者: 可用来定义标准和报纸文章两类条目，但实际上利用数据动态修改可能更为方便

type=⟨*field type*⟩

设置域的类型，典型如‘field’ 或‘list’，详见 § 2.2.1 节。

format=⟨*field format*⟩

该域的任意特殊格式。正常情况下不给出但获取‘xsv’ 值，该值告诉biber 该域具有多值格式。准确的分隔符由biber 选项xsvsep 控制，默认是由可选的空格包围的逗号。

datatype=⟨*field datatype*⟩

设置域的数据类型，典型如 ‘name’ 或‘literal’，详见 § 2.2.1 节。

nullok=true, false　　default: false

该域可以定义为空。

skipout=true, false　　default: false

该域不输出到.bbl 中因此不会出现在biblatex 样式处理中。依旧如同在 TeX 的 csv 列表中，请确保每个元素都紧跟着一个逗号或者结束的括号，即不需要额外的空格。

label=true, false　　default: false

该域可以用作文献表的标签。设置该选项会使得biblatex 为该域创建多个辅助宏，所以存在一些已经定义的内部长度和标题等。

\DeclareDatamodelEntryfields[⟨*entrytypes*⟩]{⟨*fields*⟩}

声明⟨*fields*⟩ 列表对于⟨*entrytypes*⟩ 列表是有效的。如果⟨*entrytypes*⟩ 不给出，则列表中的这些域对所有条目类型都有效。依旧如同在 TeX 的 csv 列表中，请确保每个元素都紧跟着一个逗号或者结束的括号，即不需要额外的空格。

\DeclareDatamodelConstraints[⟨*entrytypes*⟩]{⟨*specification*⟩}

如果给出⟨*entrytypes*⟩ 的逗号分隔列表，约束仅应用于这些条目。⟨*specification*⟩是\constraint 指令的不分隔列表。为实现良好的代码显示效果，可以自由使用空格、制表符、行末符号来整理⟨*specification*⟩ 中的代码，但空行不能使用。

\constraint[⟨*type=constrainttype*⟩]{⟨*elements*⟩}

指定⟨*constrainttype*⟩ 类型的约束，有效的约束类型有:

type=data, mandatory, conditional

‘data’ 类型的约束，应用于域的值。‘mandatory’ 类型的约束，指定一个条目类型应具有的域或域的组合。‘conditional’ 类型的约束允许更复杂的条件和量化的域约束。

datatype=integer, isbn, issn, ismn, date, pattern

用于⟨*data*⟩ 类型约束，限制域的值为给定的数据类型。

rangemin=⟨*num*⟩

用于⟨*data*⟩ 类型和‘integer’ 数据类型的约束，限制域的值最小为⟨*num*⟩。

`rangemax=`⟨*num*⟩

用于⟨*data*⟩类型和'integer'数据类型的约束，限制域的值最大为⟨*num*⟩。

`pattern=`⟨*patt*⟩

用于⟨*data*⟩类型和'pattern'数据类型的约束，限制域的值匹配正则表达式⟨*patt*⟩。最好利用`\regexp`宏将正则表达式包围起来，见 § 4.5.3 节。

一个`\constraint`宏可以包括如下任意内容:

`\constraintfieldsor{`⟨*fields*⟩`}`

用于'mandatory'类型的约束，指定一个条目必须包含`\constraintfields`的一个布尔或 (OR)。

`\constraintfieldsxor{`⟨*fields*⟩`}`

用于'mandatory'类型的约束，指定一个条目必须包含`\constraintfields`的一个布尔异或 (XOR)。

`\antecedent[`⟨*quantifier=quantspec*⟩`]{`⟨*fields*⟩`}`

用于'conditional'类型的约束，指定一个在约束的`\consequent`检查之前必须满足`\constraintfields`的量化集。⟨*quantspec*⟩应包含如下值:

`quantifier=all, one, none`

指定要满足条件约束前提，必须给出的`\antecedent`内的`\constrainfield`数量。

`\consequent[`⟨*quantifier=quantspec*⟩`]{`⟨*fields*⟩`}`

用于'conditional'类型的约束，指定如果满足前面约束的`\antecedent`时必须满足的`\constraintfields`量化集。⟨*quantspec*⟩应包含如下值:

`quantifier=all, one, none`

指定要满足的条件约束后件，必须给出的`\consequent`内的`\constrainfield`数量。

`\constraintfield{`⟨*field*⟩`}`

对于'data'类型的约束，约束应用于该⟨*field*⟩。对于'mandatory'类型的约束，条目必须包含⟨*field*⟩。

数据模型声明宏可以应用多次，用于在前面定义基础上增加定义。如果要替换，修改或移除已有的定义 (比如`biblatex`加载的默认模型)，需要利用下面的宏重设 (清除) 当前的定义然后再重设。通常，这些宏在任何`biblatex-dm.cfg`文件中都是首先给出的:

`\ResetDatamodelEntrytypes`

清除所有的数据模型 entrytype 信息

`\ResetDatamodelFields`

清除所有的数据模型 field 信息

\ResetDatamodelEntryfields

清除所有的数据模型 entrytypes 的 fields 信息

\ResetDatamodelConstraints

清除所有的数据模型 fields 约束信息

下面是一个简单的数据模型示例。默认数据模型设置参见biblatex 核心文件blx-dm.def。

```
\ResetDatamodelEntrytypes
\ResetDatamodelFields
\ResetDatamodelEntryfields
\ResetDatamodelConstraints

\DeclareDatamodelEntrytypes{entrytype1, entrytype2}

\DeclareDatamodelFields[type=field, datatype=literal]{field1,field2,
    ↪ field3,field4}

\DeclareDatamodelEntryfields{field1}
\DeclareDatamodelEntryfields[entrytype1]{field2,field3}
\DeclareDatamodelEntryfields[entrytype2]{field2,field3,field4}

\DeclareDatamodelConstraints[entrytype1]{
  \constraint[type=data, datatype=integer, rangemin=3, rangemax=10]{
    \constraintfield{field1}
  }
  \constraint[type=mandatory]{
    \constraintfield{field1}
    \constraintfieldsxor{
      \constraintfield{field2}
      \constraintfield{field3}
    }
  }
}
\DeclareDatamodelConstraints{
  \constraint[type=conditional]{
    \antecedent[quantifier=none]{
      \constraintfield{field2}
    }
    \consequent[quantifier=all]{
      \constraintfield{field3}
```

```
      \constraintfield{field4}
    }
  }
}
```

该模型设置了:

- 清除所有默认的数据模型
- 设置两种有效的条目类型`@entrytype1` and `@entrytype2`
- 设置四个有效的 literal 域。
- 设置`field1` 对所有条目类型都有效。
- 设置`field2` 和`field3` 对`entrytype1` 有效。
- 设置`field2`, `field3` 和`field4` 对`entrytype2` 有效。
- 设置`@entrytype1` 的约束:
  - `field1` 必须是一个 3 和 10 之间的整数
  - `field1` 必须给出
  - `field2` 或`field3` 其中之一必须唯一给出。
- 对于所有类型，`field2` 如果不给出，`field3` 和`field4` 必须要给出。

4.5.5 标签

字母顺序制样式使用一个标签来区分参考文献条目。这个标签使用一个描述怎么构建标签的模板由条目的内容构建。该模板可以全局自定义或者根据具体条目类型定义。如何抽取姓名域的成分作为标签则使用一个独立的模板，因为姓名域是相当复杂的域。

`\DeclareLabelalphaTemplate`[⟨*entrytype, ...*⟩]{⟨*specification*⟩}

为指定的条目类型定义字母顺序制标签模板。如果第一个参数中不指定具体的条目类型，则定义的是通用标签模板。⟨*specification*⟩ 是`\labelelement` 指令的一个无分隔列表，用来指定构建标签的元素。为实现良好的代码显示效果，可以自由使用空格、制表符、行末符号来整理⟨*specification*⟩ 中的代码，但空行不能使用。该命令仅可在导言区使用。

`\labelelement`{⟨*elements*⟩}

指定用于构建标签的元素。⟨*elements*⟩ 是一个`\field` 或`\literal` 命令构成的无分隔列表，这些命令以它们给出的顺序进行处理。从第一个展开为非空字符串的`\field` 或`\literal` 作为`\labelelement` 展开内容开始，后面的`\labelelement` 如果存在的话，接着进行处理。

`\field[`⟨*options*⟩`]{`⟨*field*⟩`}`

如果⟨*field*⟩非空，并作为当前标签的`\labelelement`，那么受如下选项约束。⟨*field*⟩的有用参数通常是姓名列表类型的域、日期域和标题域。可以使用虚域 'citekey' 来将引用关键词作为标签的一部分。姓名列表域需要特殊处理，当给出一个姓名列表域时，用`\DeclareLabelalphaNameTemplate` 命令定义的模板用来从姓名中抽取某些部分，并返回字符串给`\field` 选项使用。

`final=true, false` default: `false`

该选项标记一个`\field` 指令作为⟨*specification*⟩中的最后一个。如果⟨*field*⟩非空，该域用于标签中，⟨*specification*⟩中后面的剩余内容将被忽略。简写形式`final` 等价于`final=true`。

`lowercase=true, false` default: `false`

将从域中得到标签内容转换为小写。默认情况下，直接从域的源数据中取出的内容大小写形式不做改变。

`strwidth=`⟨*integer*⟩ default: `1`

使用的⟨*field*⟩的固定字符数。当从一个姓名中抽取字符时，该设置可能会被一个独立的姓名成分覆盖，见下面的`\DeclareLabelalphaNameTemplate`。

`strside=left, right` default: `left`

取 `strwidth` 数量字符开始的方向。当从一个姓名中抽取字符时，该设置可能会被一个独立的姓名成分覆盖，见下面的`\DeclareLabelalphaNameTemplate`。

`padside=left, right` default: `right`

当使用 `padchar` 选项时，向标签部分添加衬垫字符的方向。仅用于固定宽度标签字符串 (`strwidth`)。

`padchar=`⟨*character*⟩

如果存在，则将在标签部分的 `padside` 侧添加衬垫字符直到字符串长度为`strwidth`，仅用于固定宽度标签字符串 (`strwidth`)。

`uppercase=true, false` default: `false`

将从域中得到标签内容转换为大写。默认情况下，直接从域的源数据中取出的内容大小写形式不做改变。

`varwidth=true, false` default: `false`

使用一个宽度变量，并从⟨*field*⟩返回的字符串的左侧开始抽取字符。字符串的长度由要区分标签相同位置上字符串所需的最小长度来确定。对于姓名列表域，这意味着每个姓名的子字符串要与姓名列表中相同位置的所以其他姓名的字符串相区别 (见下面示例)。该选项会覆盖 `strwidth`，如果两者同时用。简写形式`varwidth` 等价于 `varwidth=true`。对于姓名列表域，设置`pre` 选项的`\nameparts` 将添加在非歧义化过程返回的字符串前面。

varwidthnorm=true, false default: false

类似 varwidth，但使得⟨*field*⟩ 的可区分子字符串长度为最长的子字符串。这可以用于调整标签的格式，如果需要的话。该选项覆盖 strwidth，如果两者同时用的话。简写形式varwidthnorm 等价于 varwidthnorm=true。

varwidthlist=true, false default: false

当域作为一个整体能与在标签相同位置的所有其它域区分时，标签自动非歧义化的替代方法。对于非姓名列表域，这等价于 varwidth。对于姓名列表域，姓名列表中某一相同位置的姓名无法区分，但整个列表可以区分。该选项覆盖 strwidth，如果两者同时用。简写形式varwidthlist 等价于 varwidthlist=true。对于姓名列表域，设置pre 选项的\nameparts 将添加在从非歧义化过程返回的字符串前面。

strwidthmax=⟨*integer*⟩

当使用 varwidth，该选项在可变宽度子字符串上设置一个限制 (在字符数上)。

strfixedcount=⟨*integer*⟩ default: 1

当使用 varwidthnorm，至少存在 strfixedcount 个可区分子字符串具有相同的最大长度来促使所有的可区分字符串具有该最大长度。

ifnames=⟨*range*⟩

当一个姓名列表域具有的姓名数在 ifnames 范围内，仅使用这一\field 设置。这允许\labelelement 可以有姓名长度条件 (见下面的示例)。范围可以设置如下:

```
ifnames=3     -> Only apply to name lists containing exactly 3 names
ifnames={2-4} -> Only apply to name lists containing minimum 2 and
    ↪ maximum 4 names
ifnames={-3}  -> Only apply to name lists containing at most 3 names
ifnames={2-}  -> Only apply to name lists containing at least 2 names
```

names=⟨*range*⟩

默认情况下，对于姓名列表域，使用姓名从第一个姓名开始到maxalphanames/minalphanames 截止。该选项可以覆盖该设置为一个显式的姓名范围。加号'+' 是一个特殊范围终止标记代表 max/minalphanames 的截断点。范围分隔符可以是任意的数字字符和统一码 (unicode) 破折号。例如:

```
name=3     -> Use first 3 names in the name list
name={2-3} -> Use second and thirds names only
name={-3}  -> Same as 1-3
name={2-}  -> Use all names starting with the second name (ignoring max/
    ↪ minalphanames truncation)
name={2-+} -> Use all names starting with the second name (respecting max
    ↪ /minalphanames truncation)
```

namessep=⟨*string*⟩ default: empty

在姓名列表的姓名之间插入任意的分隔字符串。

noalphaothers=true, false default: false

默认情况下，当姓名列表中的姓名数大于标签中显示的姓名数时，\labelalphaothers 附加在由姓名列表得到的标签部分后面。该选项可以关闭这一默认方式。

\literal{⟨*characters*⟩}

在标签的当前位置插入文本⟨*characters*⟩。

当一个姓名列表域已经给出，从中抽取字符串的方法由一个独立模板给出，该模板由如下命令设置:

\DeclareLabelalphaNameTemplate[⟨*entrytype, ...*⟩]{⟨*specification*⟩}

当\DeclareLabelalphaTemplate 中的一个\field 设置包含有一个姓名列表时，由该命令指出从姓名列表中抽取标签字符串的模板。该模板可以根据具体的条目类型定义。

\namepart[⟨*options*⟩]{⟨*namepart*⟩}

⟨*namepart*⟩ 是由\DeclareDatamodelConstant 命令定义的数据模型的姓名成分之一(见 § 4.2.3 节)。选项有:

use=true, false default: false

当存在一个相应的选项use'namepart'(比如useprefix) 并且值为 true，仅使用⟨*namepart*⟩ 用于构建标签信息。

pre=true, false default: false

当从姓名构建标签字符串时，无pre 选项的\namepart 将用于构建标签字符串，通过非歧义化处理过程，实施由\DeclareLabelalpaTemplate 中的\field 选项指定的子字符串操作。然后有pre 选项的\namepart 将会添加在结果前面 (当存在多个这种\namepart 时，按给出的顺序处理)。这就允许在姓名标签字符串前面无条件的添加一定的姓名成分信息，比如尊称。注意: \DeclareLabelalphaTemplate 中\field 的uppercase 和lowercase 选项应用于从\DeclareLabelalphaTemplate 返回的整个标签，无论它是不是带pre 选项的\namepart。

compound=true, false default: false

对于\DeclareLabelalphaTemplate 中静态 (非 varwidth) 区分方法，由空格或连字符 (混合姓名) 分隔的姓名成分在标签生成中将作为独立姓名成分。这意味着从如'Ballam Forsyth' 这样的姓中形成标签时，当用 1 个字符，左侧开始抽取子字符串，当 compound=true 时，该姓名会给出'BF'，而 compound=false 时会给出'B'。简写形式compound 等价于 compound=true。

strwidth=⟨*integer*⟩ default: 1

使用的⟨*namepart*⟩ 的字符数。

strside=left, right default: left

获取 strwidth 数量字符开始的方向。

注意标签模板可以根据具体类型分别定义，当使用自动标签非歧义化功能时应注意这一点。非歧义化是不分类型的，因此不同类型的不同标签格式在各自非歧义化过程中可能产生歧义。一般情况，需要针对不同类型使用非常不同的标签格式，使得标签的类型可以明确。

下面是一些示例。biblatex 字母顺序标签默认模板全局定义如下。首先shorthand 具有 final=true，因此当存在shorthand 域时，它将作为标签，模板的其它内容不再考虑。接着，label 作为第一个标签元素，如果它存在的话。另外，如果labelname 列表中仅有一个姓名 (ifnames=1)，那么labelname 中从姓的左侧取 3 个字符作为第一个标签元素。如果labelname 具有超过 1 个的姓名，从每个名字的姓的左侧取 1 个字符作为第一个标签元素。第二个标签元素包含从year 域的右侧取的两个字符。

示例中也给出了姓名构建标签的默认模板。该模板将任何尊称 (如果useprefix 选项为 true) 左侧的第一个字符添加到由姓 (根据调用来自\DeclareLabelalphaTemplate 的\field 的选项实现) 中抽取的标签之前，并支持混合 (compound) 的姓处理。

```
\DeclareLabelalphaTemplate{
  \labelelement{
    \field[final]{shorthand}
    \field{label}
    \field[strwidth=3,strside=left,ifnames=1]{labelname}
    \field[strwidth=1,strside=left]{labelname}
  }
  \labelelement{
    \field[strwidth=2,strside=right]{year}
  }
}

\DeclareLabelalphaNameTemplate{
  \namepart[use=true, pre=true, strwidth=1, compound=true]{prefix}
  \namepart{family}
}
```

要了解标签自动非歧义化 (消除歧义)(非模糊化) 处理的工作方式，观察如下的作者列表:

```
Agassi, Chang, Laver   (2000)
Agassi, Connors, Lendl (2001)
Agassi, Courier, Laver (2002)
Borg, Connors, Edberg  (2003)
Borg, Connors, Emerson (2004)
```

假设一个模板定义如下:

```
\DeclareLabelalphaTemplate{
  \labelelement{
    \field[varwidth]{labelname}
  }
}
```

那么标签会是:

```
Agassi, Chang, Laver    [AChLa]
Agassi, Connors, Lendl  [AConLe]
Agassi, Courier, Laver  [ACouLa]
Borg, Connors, Edberg   [BConEd]
Borg, Connors, Emerson  [BConEm]
```

使用规范化可变宽度标签定义:

```
\DeclareLabelalphaTemplate{
  \labelelement{
    \field[varwidthnorm]{labelname}
  }
}
```

会得到如下结果，其中每个位置的姓名子字符串拓展为相同位置上最长子字符串的长度:

```
Agassi, Chang, Laver    [AChaLa]
Agassi, Connors, Lendl  [AConLe]
Agassi, Courier, Laver  [ACouLa]
Borg, Connors, Edberg   [BConEd]
Borg, Connors, Emerson  [BConEm]
```

对标签元素的姓名成分有 2 个字符的限制，定义如下:

```
\DeclareLabelalphaTemplate{
  \labelelement{
    \field[varwidthnorm,strwidthmax=2]{labelname}
  }
}
```

得到结果如下 (注意每个姓构成的标签部分不再具有非歧义性):

```
Agassi, Chang, Laver    [AChLa]
Agassi, Connors, Lendl  [ACoLe]
```

```
Agassi, Courier, Laver  [ACoLa]
Borg, Connors, Edberg   [BCoEd]
Borg, Connors, Emerson  [BCoEm]
```

或者，可以将姓名列表作为一个整体进行消除歧义处理的:

```
\DeclareLabelalphaTemplate{
  \labelelement{
    \field[varwidthlist]{labelname}
  }
}
```

将得到:

```
Agassi, Chang, Laver    [AChL]
Agassi, Connors, Lendl  [ACoL]
Agassi, Courier, Laver  [ACL]
Borg, Connors, Edberg   [BCEd]
Borg, Connors, Emerson  [BCE]
```

或者仅需要考虑最多两个姓名用于标签生成，但又在姓名列表层次进行非歧义化处理:

```
\DeclareLabelalphaTemplate{
  \labelelement{
    \field[varwidthlist,names=2]{labelname}
  }
}
```

将得到:

```
Agassi, Chang, Laver    [ACh+]
Agassi, Connors, Lendl  [ACo+]
Agassi, Courier, Laver  [AC+]
Borg, Connors, Edberg   [BC+a]
Borg, Connors, Emerson  [BC+b]
```

在这最后一个示例中，可以看到已经添加了`\labelalphaothers` 以显示存在更多姓名。最后两个标签需要利用`\extraalpha` 来实现非歧义性，因为仅依靠列表中前两个姓名无法消除歧义。

最后是一个使用多个标签元素的示例:

```
\DeclareLabelalphaTemplate{
```

```
  \labelelement{
    \field[varwidthlist]{labelname}
  }
  \labelelement{
    \literal{-}
  }
  \labelelement{
    \field[strwidth=3,strside=right]{labelyear}
  }
}
```

将得到:

```
Agassi, Chang, Laver    [AChL-000]
Agassi, Connors, Lendl  [AConL-001]
Agassi, Courier, Laver  [ACouL-002]
Borg, Connors, Edberg   [BCEd-003]
Borg, Connors, Emerson  [BCEm-004]
```

下面是精心设计的另一个示例，展示当指定衬垫字符或文本时，不需要特别的引用 LaTeX 特殊字符 (显然除了‘%’ 之外):

```
\DeclareLabelalphaTemplate{
  \labelelement{
    \literal{>}
  }
  \labelelement{
    \literal{\%}
  }
  \labelelement{
    \field[namessep={/}, strwidth=4, padchar=_]{labelname}
  }
  \labelelement{
    \field[strwidth=3, padchar=&, padside=left]{title}
  }
  \labelelement{
    \field[strwidth=2,strside=right]{year}
  }
}
```

当有:

```
@Book{test,
```

```
  author    = {XXX YY and WWW ZZ},
  title     = {T},
  year      = {2007},
}
```

将得到[57]:

```
[>%YY/ZZ__&&T07]
```

当域中包含变音符 (diacritic)，连字符 (hyphen)，空格 (space) 等时，从域中生成标签可能存在一些困难。通常，在生成标签时需要忽略分隔符或者空格之类的字符。一个选项可以用来定制正则表达式使得在域传递给标签生成系统前能将其内容取出。

`\DeclareNolabel{`⟨*specification*⟩`}`

定义正则表达式实现在生成域的标签部分前从任意域取出其内容。⟨*specification*⟩是一个`\nolabel`指令构成的无分隔列表，该列表指定了正则表达式用于将其匹配内容从多个域中去除。可以自由使用空格，制表符，行末符号来整理其中的代码，达到满意的代码显示效果，但空行则不允许。该命令只能用于导言中。

`\nolabel{`⟨*regexp*⟩`}`

可以给出任意数量的`\nolabel`命令，用来指定去除副本域的⟨*regexp*⟩，该副本域能被标签生成系统看见。

如果不存在`\DeclareNolabel`设置，`biber`将默认设置:

```
\DeclareNolabel{
  % strip punctuation, symbols, separator and control characters
  \nolabel{\regexp{[\p{P}\p{S}\p{C}]+}}
}
```

这一`biber`默认设置将在向标签生成系统传递域字符串之前去除标点，符号，分隔符和控制符。(译者: 注意这里正则表达式中的`\p`，用于匹配有命名属性的字符。)

`\DeclareNolabelwidthcount{`⟨*specification*⟩`}`

当在固定宽度子字符串中统计字符数时，定义任意域中需要忽略的⟨*regexp*⟩(即正则表达式匹配的内容)。⟨*specification*⟩是一个`\nolabelwidthcount`指令构成的无分隔列表，该列表指定了正则表达式用于在固定宽度子字符串中统计字符时将其匹配内容忽略。可以自由使用空格，制表符，行末符号来整理其中的代码，达到满意的代码显示效果，但空行则不允许。该命令只能用于导言中。

[57] 译者: 注意这里没有说明姓名模板使用的是哪个？所以对于该结果是有疑惑的，strwidth 是应用到一个姓名还是多个姓名？

\nolabelwidthcount{⟨*regexp*⟩}

可以给出任意数量的\nolabelwidthcount 命令，用来指定在标签生成过程中当生成固定宽度子字符串时忽略的⟨*regexp*⟩。

没有默认的\DeclareNolabelwidthcount 设置。注意该设置只有在标签部分生成过程中使用固定宽度子字符串时考虑。见 § 4.5.5 节。

4.5.6 排序

除了应用 § 3.5 节讨论的预定义的排序格式外，还可以定义新的排序方式或者修改默认的格式。还可以通过根据具体的条目类型排除某些域和自动添加presort 域来进一步定义。

\DeclareSortingScheme[⟨*options*⟩]{⟨*name*⟩}{⟨*specification*⟩}

定义排序格式⟨*name*⟩。当选择了指定排序格式时，⟨*name*⟩ 是传递给sorting 选项 (见 § 3.1.2.1) 的标识。\DeclareSortingScheme 命令支持如下可选参数:

locale=⟨*locale*⟩

排序格式中的 locale 用于覆盖sortlocale 选项 (在 § 3.1.2.1 节讨论) 给出的全局排序 locale。

⟨*specification*⟩ 是\sort 指令的无分隔列表，这些\sort 指令用于指定需要在排序过程中考虑的元素。可以自由使用空格，制表符，行末符来调整代码呈现格式以达到满意视觉效果，空行不允许。该命令只能用于导言中。

\sort{⟨*elements*⟩}

指定在排序过程中考虑的元素。⟨*elements*⟩ 是\field，\literal，和 \citeorder 命令的无分隔列表，这些命令以给定的顺序考虑。注意: 一旦定义了一个元素，它将被添加到排序关键词中，然后排序过程跳到下一条\sort 指令。如果该元素未定义，则考虑下一个元素。因为文本字符串总是存在，\literal 命令应该单独或者最后出现在\sort 指令中。所有的⟨*elements*⟩ 应该与 § 2.2.2 节中描述数据类型一致，因为它们将会与其它条目中的⟨*elements*⟩ 进行比较。\sort 命令支持如下可选参数:

locale=⟨*locale*⟩

在排序元素的一个特定集层级覆盖用于排序的 locale。如果给出，locale 将覆盖在\DeclareSortingScheme 层级和全局设置的 locale 集。另可参见 § 3.1.2.1 节关于全局排序 locale 选项sortlocale 的讨论。

direction=ascending, descending　　default: ascending

排序方向，可以是 ascending 或 descending。默认是升序 (ascending)。

final=true, false　　default: false

该选项标记一个\sort 指令作为⟨*specification*⟩ 中的最后一个。如果⟨*elements*⟩ 中的元素存在，⟨*specification*⟩ 中剩下的内容将被忽略。简写形式final 等价于 final=true。

`sortcase=true, false`

设置排序是否大小写敏感 (case"=sensitively)。默认的设置取决于全局选项`sortcase`。

`sortupper=true, false`

设置排序是否大写字母开头的在小写字母开头的条目之前 (即'uppercase before lowercase'(`true`)or 'lowercase before uppercase' order (`false`))。默认设置取决于全局选项`sortupper`。

`\field[`⟨*key=value, ...*⟩`]{`⟨*field*⟩`}`

`\field` 元素向排序设置中添加⟨*field*⟩。如果⟨*field*⟩ 未定义，则跳过该元素。`\field` 支持如下可选参数:

`padside=left, right` default: `left`

使用`padchar` 在一个域的 `left` 或 `right` 侧添加衬垫字符，使其宽度为`padwidth`。如果 padding 选项没有设置，则不做任何衬垫。否则将执行衬垫，选项缺省将使用默认值。如果 padding 和子字符串匹配选项同时存在，那么子字符串匹配首先处理。

`padwidth=`⟨*integer*⟩ default: `4`

所需衬垫字符目标宽度

`padchar=`⟨*character*⟩ default: `0`

用于衬垫的字符。

`strside=left, right` default: `left`

从域的 `left` 或 `right` 侧执行子字符串匹配。匹配的字符数由相应的 `strwidth` 选择指定。如果没有设置 substring 选项，则不做匹配。如果存在 substring 选项，将执行匹配，选项缺省将使用默认值。如果 padding 和子字符串匹配选项同时存在，那么子字符串匹配首先处理。

`strwidth=`⟨*integer*⟩ default: `4`

用于匹配的字符数。

`\literal{`⟨*string*⟩`}`

`\literal` 元素添加一个文本的⟨*string*⟩ 到排序配置中。这在一些域不存在时作为一个备选使用。

`\citeorder` `\citeorder` 元素有特殊意义。它要求排序基于实际标注的词汇表顺序。对于在一个标注命令中的多个条目，如:

```
\cite{one,two}
```

词汇表顺序和语义顺序是不同的。这里的"one"和"two"有相同的语义顺序，但有不同的词汇表顺序。语义顺序只关心相同语义顺序下是否指定后续的排序设置来区分条目。比如，一个排序格式`none` 的定义为:

```
\DeclareSortingScheme{none}{
  \sort{\citeorder}
}
```

这将单纯地根据标注命令中的关键词顺序来排序文献。在上一个示例中，“one”排在“two”前面。如果假设“one”和“two”具有相同的语义顺序因为它们同时被引用，但需要根据 year 进一步排序，假设“two”条目的年份早于“one”:

```
\DeclareSortingScheme{noneyear}{
  \sort{\citeorder}
  \sort{year}
}
```

这次将“two”排在“one”，尽管从词汇表上看，“one”应排在“two”前面。这是可能的，因为语义顺序可以根据年份进一步排序区分。使用标准的none 排序格式，词汇表顺序和语义顺序是相同的，因为没有进一步的排序设置来区分。这意味着可以像使用其它元素一样来使用\citeorder，选择怎么来对同时引用 (在一个标注命令中) 的条目进行进一步排序。

\DeclareSortingNamekeyScheme[⟨*schemename*⟩]{⟨*specification*⟩}

定义姓名的排序关键词如何构建。这可以用来改变姓名的排序，因为在构建排序要比较的字符串时可以自由选择姓名成分来组合。如此定义的排序关键词的构建格式称为⟨*schemename*⟩，默认是全局的 (“global”)，如果该可选参数缺省的话。当为姓名构建排序关键词时，构建每个姓名成分的排序关键词，并利用一个专门的内部分隔符构成一个顺序的关键词列表。这一选项的目的是兼容语言或者需要自定义姓名排序的情况 (例如，冰岛人的姓名有时是用名排序而不是姓)。该宏可以多次使用为不同姓名的定义格式，以便后面使用。姓名排序关键词格式具有如下作用范围，以优先级增加的顺序包括:

- 无可选姓名参数定义的默认格式。
- 为一个参考文献环境 (见 § 3.7.10) 给出sortingnamekeyscheme 选项。
- 在参数文献数据源条目中给出具体条目的sortnamekeyscheme 选项。
- 为姓名列表给出的sortnamekeyscheme 选项。
- 为一个姓名给出的sortnamekeyscheme 选项。

默认情况下，仅有一个全局格式，⟨*specification*⟩ 定义如下:

```
\DeclareSortingNamekeyScheme{
  \keypart{
    \namepart[use=true]{prefix}
  }
```

```
  \keypart{
    \namepart{family}
  }
  \keypart{
    \namepart{given}
  }
  \keypart{
    \namepart{suffix}
  }
  \keypart{
    \namepart[use=false]{prefix}
  }
}
```

这意味着关键词由如下姓名成分按给出顺序联合构建: 前缀 (尊称)(仅当useprefix 选项为 true)，姓，名，后缀和前缀 (仅当useprefix 选项为 false)

\keypart{⟨*part*⟩}

⟨*part*⟩ 是一个\namepart 和\literal 设置构成的有序列表，这些设置联接在一起构建姓名排序关键词的一部分。

\literal{⟨*string*⟩}

插入姓名排序关键词中的文本字符串。

\namepart{⟨*name*⟩}

指定姓名成分的⟨*name*⟩，用于构建姓名排序关键词。

use=true, false　　default: true

表示仅当姓名成分⟨*name*⟩ 对应的use'name' 选项设置为指定的布尔值时使用该⟨*name*⟩ 用于构建排序关键词。

inits=true, false　　default: true

表示仅使用姓名成分⟨*name*⟩ 的首字母来构建排序关键词。

举个例子，假设用于排序的姓名成分中名要优先于姓，可以定义姓名排序关键词如下:

```
\DeclareSortingNamekeyScheme[givenfirst]{
  \keypart{
    \namepart{given}
  }
  \keypart{
    \namepart[use=true]{prefix}
```

```
  }
  \keypart{
    \namepart{family}
  }
  \keypart{
    \namepart[use=false]{prefix}
  }
}
```

然后可以在适当的范围内使用`givenfirst` 来使得`biber` 根据该格式构建排序关键词。比如，可以用在一个参考文献环境中使用:

```
\begin{refcontext}[sortnamekeyscheme=givenfirst]
\printbibliography
\end{refcontext}
```

或者在某一个特定条目中使用:

```
@BOOK{key,
  OPTIONS = {sortnamekeyscheme=givenfirst},
  AUTHOR = {Arnar Vigfusson}
}
```

或者在一个姓名列表中使用，该选项作为一个假名会自动忽略。

```
@BOOK{key,
  AUTHOR = {sortnamekeyscheme=givenfirst and Arnar Vigfusson}
}
```

或者在一个姓名中使用，通过将其作为`biber` 支持的姓名信息格式的一个扩展成分传递该选项 (见`biber` 文档):

```
@BOOK{key,
  AUTHOR = {given=Arnar, family=Vigfusson, sortnamekeyscheme=givenfirst}
}
```

下面我们给出一些排序格式的示例。在第一个示例中，我们定义了一个简单的 name/title/year 格式。姓名元素可以是`author`，`editor`，或`translator`。根据这一设置，排序过程将首先使用存在的第一个元素，然后继续处理`title`。注意`use<name>` 类选项在排序过程中自动处理:

```
\DeclareSortingScheme{sample}{
  \sort{
    \field{author}
```

```
    \field{editor}
    \field{translator}
  }
  \sort{
    \field{title}
  }
  \sort{
    \field{year}
  }
}
```

下一个示例中，我们以更详尽的方式定义了一个格式，考虑了presort, sortkey, sortname 等特殊域。因为sortkey 域指定了排序的主关键词，它需要覆盖所有除presort 之外的元素。这通过final 选项体现。如果sortkey 域存在，排序过程将停在此处。如果不存在，排序过程将继续下一个\sort 指令。这一设置就是 nty 格式的默认定义。

```
\DeclareSortingScheme{nty}{
  \sort{
    \field{presort}
  }
  \sort[final]{
    \field{sortkey}
  }
  \sort{
    \field{sortname}
    \field{author}
    \field{editor}
    \field{translator}
    \field{sorttitle}
    \field{title}
  }
  \sort{
    \field{sorttitle}
    \field{title}
  }
  \sort{
    \field{sortyear}
    \field{year}
  }
}
```

最后，给出一个覆盖全局的排序 locale 并在根据`origtitle` 域排序时再次覆盖的示例。注意: 在格式层覆盖 babel/polyglossia 语言名的是一个真实的 locale 标识。`biber` 会将其映射到一个合适的真实 locale 标识上 (本例中是 `sv_SE`)

```
\DeclareSortingScheme[locale=swedish]{custom}{
  \sort{
    \field{sortname}
    \field{author}
    \field{editor}
    \field{translator}
    \field{sorttitle}
    \field{title}
  }
  \sort[locale=de_DE_phonebook]{
    \field{origtitle}
  }
}
```

`\DeclareSortExclusion{`⟨*entrytype, ...*⟩`}{`⟨*field, ...*⟩`}`

指定排序具体类型的条目需要排除的域。⟨*entrytype*⟩ 和⟨*field*⟩ 参数可以是逗号分隔的列表。空的⟨*field*⟩ 将会清除该⟨*entrytype*⟩ 类型的所有排除设置。⟨*entrytype*⟩ 参数如果设置为'*' ，则排除的⟨*field,...*⟩ 针对所有类型。这等价于从排序设置中简单的删除一些域，且一般仅与`\DeclareSortInclusion` 连用，特别是当希望为显式给出的条目类型排除某个域时候使用。示例见下面的`\DeclareSortInclusion`。该命令只能用于导言区。

`\DeclareSortInclusion{`⟨*entrytype, ...*⟩`}{`⟨*field, ...*⟩`}`

仅与`\DeclareSortExclusion` 一同使用。指定具体条目类型排序需要包含的域。这使得用户可以为所有类型排序排除一个域，然后为某些特定条目类型改变这一设置。这有时比`\DeclareSortExclusion` 更便于为一些条目列出排除项。⟨*entrytype*⟩ 和⟨*field*⟩ 参数可以是逗号分隔的列表。该命令只能用于导言区。例如，可以设置`title` 在排序中仅用于`@article` 类型:

```
\DeclareSortExclusion{*}{title}
\DeclareSortInclusion{article}{title}
```

`\DeclarePresort[`⟨*entrytype, ...*⟩`]{`⟨*string*⟩`}`

定义一个字符串，用于当条目中不存在`presort` 域时，复制该字符串到`presort` 域。`presort` 可以全局定义或者根据具体条目定义。如果给出可选参数⟨*entrytype*⟩，⟨*string*⟩ 用于指定的条目类型。否则作为全局的默认值。给具体条目⟨*entrytype*⟩ 指定一个空的⟨*string*⟩，将会清除原来的条目设置。而⟨*entrytype*⟩ 参数可以是一个逗号分隔的列表。该命令只能用于导言中。

From	To	Description
iast	devanagari	Sanskrit IAST[58] to Devanāgarī

表 11: 可互相转换的文字

`\DeclareSortTranslit`[⟨*entrytype*⟩]{⟨*specification*⟩}

能用不同文字或字母书写的语言对于一种文字常常仅有 CLDR 排序方式，为了进行排序往往需要将其转换为支持的文字。一个常见示例是梵文 (Sanskrit)，它常用罗马化的婆罗米系文字书写正式文献，但需要在'sa' locale 中排序，而这需要 Devanāgarī 文字。这意味着需要在内部转换排序文字。`\DeclareSortTranslit` 声明条目的那个部分需要为排序目的而转化。如果没有⟨*entrytype*⟩ 参数，⟨*specification*⟩ 用于所有条目。⟨*specification*⟩ 包含一个或多个`\translit` 命令。

`\translit`{⟨*field or fieldset*⟩}{⟨*from*⟩}{⟨*to*⟩}

指定数据域`field` 或由`\DeclareDatafieldSet`(见 § 4.5.2) 声明的⟨*fieldset*⟩ 集中的所有域为排序目的应从⟨*from*⟩ 文字转换为⟨*to*⟩ 文字。域和集参数可以是 '*' 使其应用于所有域。有效的⟨*from*⟩ 和⟨*to*⟩ 在表11 中给出。注意`biblatex` 无意于支持通用的转换，而仅做排序目的。如果需要更多的转换，可以在 GitHub 上的`biblatex` 查找或询问。

下面给出一个标题转换的示例，从而可以正确的排序梵文:

```
\DeclareDatafieldSet{settitles}{
  \member[field=title]
  \member[field=booktitle]
  \member[field=eventtitle]
  \member[field=issuetitle]
  \member[field=journaltitle]
  \member[field=maintitle]
  \member[field=origtitle]
}

\DeclareSortTranslit{
  \translit[settitles]{iast}{devanagari}
}
```

4.5.7 参考文献表过滤器

当使用定制的参考文献表时 (见 § 3.7.3)，常需要在`.bbl` 中仅写入具有特定域的条目用于在文献表中总结。比如，当打印一个缩略词的常规列表，需要让`biber` 仅往`.bbl` 中写入一个仅包含具有`shorthand` 域的条目的列表。这可以通过利用`\DeclareBiblistFilter` 命令定义一个参考文献表过滤器来实现。这

与\defbibfilter(见 § 3.7.9) 定义的过滤器不同，因为\defbibfilter 定义的过滤器在biblatex 内运行且在.bbl 被生成之后。而且，.bbl 中的 Bibliography List 不包含条目数据，仅是条目的引用关键词，所有\defbibfilter 定义的过滤器不能用于 bibliography list。[59]

\DeclareBiblistFilter{⟨*name*⟩}{⟨*specification*⟩}

定义一个参考文献列表过滤器⟨*name*⟩。⟨*specification*⟩ 由一个或多个的\filter 或\filteror 宏构成，对于传递给过滤器的条目所有这些宏都需要满足。

\filter[⟨*filterspec*⟩]{⟨*filter*⟩}

根据⟨*filterspec*⟩ 和⟨*filter*⟩ 过滤条目。⟨*filterspec*⟩ 可以是:

type/nottype 条目是/不是entrytype ⟨*filter*⟩

subtype/notsubtype 条目是/不是subtype ⟨*filter*⟩

keyword/notkeyword 条目具有/不具有keyword ⟨*filter*⟩

field/notfield 条目具有/不具有⟨*filter*⟩ 域

\filteror{⟨*type*⟩}{⟨*filters*⟩}

在一个或多个\filter 命令外的封套，用于指定这些\filter 构成了一个并集，即其中任意一个⟨*filters*⟩ 都要满足。

数据模型中标记为'Label fields'(见 § 4.5.4) 的域自动拥有为其定义的过滤器，过滤器名同域名，能将不包含该域的条目过滤掉。比如，biblatex 自动为shorthand 域生成一个过滤器:

```
\DeclareBiblistFilter{shorthand}{
  \filter[type=field,filter=shorthand]
}
```

4.5.8 姓名首字母生成控制

当姓名中存在前缀，变音符，连字符等时，要从一个给定姓名中生成各姓名成分的首字母存在一些困难。当生成首字母时，我们经常需要忽略像前缀之类的东西，比如"al-Hasan"的首字母就是"H"而不是"a-H"，但当你需要将姓名"Ho-Pun"生成首字母为"H-P"时这就变得相当棘手。

\DeclareNoinit{⟨*specification*⟩}

定义用于在生成首字母前从姓名剥离的匹配正则表达式。⟨*specification*⟩ 是\noinit 指令的不分隔列表，\noinit 指令指定从姓名中移除匹配内容的正则表达式。可以自由使用空格，制表符，行末符来调整代码呈现格式以达到满意代码显示效果，空行不允许。该命令只能用于导言中。

[59] 译者: 这里 Bibliography List 不是简单参考文献表，而是写入到 bbl 文件中某个列表，仅包含列表中各条目的关键词

\noinit{⟨*regexp*⟩}

可以给出任意数量的\noinit 命令，每一个\noinit 都用来指定从首字母生成系统能看到的姓名副本中去除⟨*regexp*⟩。因为正则表达式常包含特殊字符，最好将他们用\regexp 宏包围起来—使得能将表达式正确地传递给biber 。

如果没有\DeclareNoinit 设置，biber 采取默认方式为:

```
\DeclareNoinit{
  % strip lowercase prefixes like 'al-' when generating initials from
    ↪ names
  \noinit{\regexp{\b\p{Ll}{2}\p{Pd}}}
  % strip some common diacritics when generating initials from names
  \noinit{\regexp{[\x{2bf}\x{2018}]}}
}
```

biber 利用这一代码默认在生成首字母前从姓名中剥离小写的前缀和一对变音符。

4.5.9 排序微调

对排序进行微调是有用的，它可以忽略一些特殊域的某些部分。

\DeclareNosort{⟨*specification*⟩}

定义正则表达式用来在排序时对特定的域或者特定类型的域的内容做剥离处理。⟨*specification*⟩ 是\nosort 指令的无分隔列表，\nosort 指令用于定义正则表达式来移除特定域或特定类型的域中的某些内容。可以自由使用空格，制表符，行末符来调整代码呈现格式以达到满意视觉效果，空行不允许。该命令只能用于导言中。

\nosort{⟨*field or field type*⟩}{⟨*regexp*⟩}

可以给出任意数量的\nosort 命令，这些命令将⟨*field*⟩ 或⟨*field type*⟩ 中的⟨*regexp*⟩ 移除。⟨*field type*⟩ 简单方便的构建了一个语义类似的域的集，在这些域中希望移除regexp。表12 给出了可用的域类型。因为正则表达式常包含特殊字符，最好将他们用\regexp 宏包围起来—使得能将表达式正确地传递给biber 。

默认是:

```
\DeclareNosort{
  % strip prefixes like 'al-' when sorting names
  \nosort{type_names}{\regexp{\A\p{L}{2}\p{Pd}}}
  % strip some diacritics when sorting names
  \nosort{type_names}{\regexp{[\x{2bf}\x{2018}]}}
}
```

排序时，biber 默认从姓名中去除一对变音符以及前缀。如果排序时需要忽略掉title 域开头的“The”，那么可以设置如下:

```
\DeclareNosort{
  \nosort{title}{\regexp{\AThe\s+}}
}
```

或者如果想要忽略任意标题域开头的“The”:

```
\DeclareNosort{
  \nosort{type_title}{\regexp{\AThe\s+}}
}
```

Field Type	Fields
type_name	author
	afterword
	annotator
	bookauthor
	commentator
	editor
	editora
	editorb
	editorc
	foreword
	holder
	introduction
	namea
	nameb
	namec
	shortauthor
	shorteditor
	translator
type_title	booktitle
	eventtitle
	issuetitle
	journaltitle
	maintitle
	origtitle
	title

表 12: \nosort 中使用的域类型

4.5.10 特殊域

§ 4.2.4.2 节的一些自动生成的域可以重新定制:

\DeclareLabelname[⟨*entrytype, ...*⟩]{⟨*specification*⟩}

当生成labelname 域时 (见 § 4.2.4.2)，定义要考虑的域。⟨*specification*⟩ 是\field 命令的一个有序列表。这些域以列出的顺序进行检查，第一个有效的域作为labelname，默认的定义为:

```
\DeclareLabelname{%
  \field{shortauthor}
  \field{author}
  \field{shorteditor}
  \field{editor}
  \field{translator}
}
```

labelname 域可以全局的或者根据具体条目类型定制。如果给出了可选参数⟨*entrytype*⟩，设置应用于相应的条目类型。如果没有给出，则是全局的。⟨*entrytype*⟩参数可以是逗号分隔的列表。该命令只能用于导言区。

\DeclareLabeldate[⟨*entrytype, ...*⟩]{⟨*specification*⟩}

当生成labelyear, labelmonth, labelday, labelendyear, labelendmonth和labelendday 域 (见 § 4.2.4.2) 时，定义要考虑的日期成分。⟨*specification*⟩是\field 或 \literal 命令的有序列表。其中各项以列出的顺序检查，可用的第一项将复制为前述生成的域。注意: \field 项不一定必须是数据模型中的'date' 日期类型，所以可以创建假年标签，例如利用pubstate 域的内容 (如果存在该域)，适当地作为定义\DeclareLabeldate 的年标签。注意: 当\literal 命令被发现时，将总被使用，所以它应该放到列表的最后。如果\literal 命令的值是一个有效的本地化字符串，那么它将会解析为当前语言的字符串，否则将作为文本字符串照抄。默认定义如下:

```
\DeclareLabeldate{%
  \field{date}
  \field{year}
  \field{eventdate}
  \field{origdate}
  \field{urldate}
  \literal{nodate}
}
```

注意: date 域由后端自动分割成为year, month[60]，它们在默认的数据模型中也是有效域。为了支持传统的数据，比如直接设置year 和/或 month，\DeclareLabeldate 中的'date' 设置也将匹配year 和month，如果它们存在。label* 域可以全局的或者根据具体条目类型定制。如果给出了可选参数⟨*entrytype*⟩，设置应用于相应的条目类型。如果没有给出，则是全局的。⟨*entrytype*⟩ 参数可以是逗号分隔的列表。该命令只能用于导言区。另见 § 4.2.4.3 节。

[60]译者: 这是需要注意的，在设计样式时判断日期域存在要用 year 而不是 date，因为 date 已经自动解析分割了。

\DeclareLabeltitle[⟨*entrytype, ...*⟩]{⟨*specification*⟩}

定义生成labeltitle 域 (见 § 4.2.4.2) 时要考虑的域。⟨*specification*⟩ 是\field 命令的一个有序列表。这些域以给出的顺序检查，第一个有效的域作为labeltitle，默认的定义是:

```
\DeclareLabeltitle{%
  \field{shorttitle}
  \field{title}
}
```

labeltitle 可以全局的或者根据具体条目类型定义。如果给出了可选参数⟨*entrytype*⟩，设置应用于相应的条目类型。如果没有给出，则是全局的。⟨*entrytype*⟩ 参数可以是逗号分隔的列表。该命令只能用于导言区。

4.5.11 数据继承 (crossref)

biber 提供了高度可定制的交叉引用机制和灵活的数据继承规则。本节介绍这些配置接口。默认配置见附录 B 节。关于术语: *child* 或 *target* 是具有crossref 域的条目，*parent* 或 *source* 是crossref 域指向的条目。子条目从父条目继承数据:

\DefaultInheritance[⟨*exceptions*⟩]{⟨*options*⟩}

配置默认的继承行为。该命令只能用于导言区。默认的继承行为可以通过下面的⟨*options*⟩ 定制:

all=true, false　　default: true

是否默认从父条目继承所有的域

all=true 意为子条目从父条目继承所有域，除非有\DeclareDataInheritance 设置的更具体的继承规则。如果对于一个域定义了继承规则，则数据继承由该规则控制。

all = false 意为默认不从父条目继承数据，每个域都需要根据\DeclareDataInheritance 设定的明确继承规则来执行继承。包默认是 all = true。

override=true, false　　default: false

是否用源域覆盖目标域，当两者都存在时。该选项能作用于自动继承和显式继承两种规则。包默认是 override=false，即子条目存在的域都不覆盖。

ignore=⟨*csv list of uniqueness options*⟩

该选项的取值是一个逗号分隔的列表，由一个或多个'singletitle', 'uniquetitle', 'uniquebaretitle' 和/或 'uniquework' 构成。该选项的目的是，当会触发这些信息追踪的域 (表6) 被继承时，忽略这三个选项的追踪信息。一个例子是，假设有多个@book 条目都引用一个@mvbook 条目中的author 域。你可能需要一个\ifsingletitle 判断来返回'true'，这一作者的唯一著作 ('work') 是@mvbook 条目。类似的情况也会出

现在应用\ifuniquetitle, \ifuniquebaretitle，\ifuniquework 判断时。ignore 选项列出需要忽略追踪信息的判断，当继承的域会引发它们产生追踪信息时。思路是一个继承的域不参与决定参考文献数据中姓名/标题组合的唯一性。例如，下面修改的默认设置将会忽略singletitle 和uniquetitle 追踪:

```
\DefaultInheritance{ignore={singletitle,uniquetitle}, all=true, override
    ↪ =false}
```

当然，当继承的域不参与信息追踪，追踪忽略不做任何处理。只有表6 列出的域与这一选项相关。

可选的⟨*exceptions*⟩ 是\except 指令的一个不分隔列表。可以自由使用空格，制表符，行末符来调整代码呈现格式以达到满意视觉效果，空行不允许。

\except{⟨*source*⟩}{⟨*target*⟩}{⟨*options*⟩}

定义默认继承规则的例外规则。

\DeclareDataInheritance 为一个具体的⟨*source*⟩ 和⟨*target*⟩ 组合设置继承选项⟨*options*⟩。⟨*source*⟩ 和⟨*target*⟩ 参数指定了父条目和子条目。星号匹配所有的类型，可用于任一参数中。

\DeclareDataInheritance[⟨*options*⟩]{⟨*source, ...*⟩}{⟨*target, ...*⟩}{⟨*rules*⟩}

定义继承规则，⟨*source*⟩ 和⟨*target*⟩ 参数指定了父条目和子条目。每个参数可以是单个条目，或者一个逗号分隔的类型列表或者星号。星号匹配所有的类型。⟨*rules*⟩ 是\inherit 和/或\noinherit 指令的无分隔列表。可以自由使用空格，制表符，行末符来调整代码呈现格式以达到满意视觉效果，空行不允许。

ignore=⟨*csv list of uniqueness options*⟩

类似于上述\DefaultInheritance 的ignore 选项。当给出设置，它将高于由\DefaultInheritance 设置的全局选项。下面示例中，当一个@book 条目从@mvbook 条目继承数据时，将忽略singletitle 和uniquetitle 追踪。

```
\DeclareDataInheritance[ignore={singletitle,uniquetitle}]{mvbook}{book}{
    ↪ ...}
```

\inherit[⟨*option*⟩]{⟨*source*⟩}{⟨*target*⟩}

定义一个继承规则，通过从⟨*source*⟩ 域向⟨*target*⟩ 域映射实现，⟨*option*⟩ 可以是:

override=true, false　　default: false

类似于上述\DefaultInheritance 中的override 选项，当给出设置，它将高于\DefaultInheritance 设置的全局选项。

`\noinherit{⟨source⟩}`

无条件阻止从⟨*source*⟩ 域的继承。

`\ResetDataInheritance` 清除由`\DeclareDataInheritance` 定义所有继承规则。该命令只能用于导言中。

下面是一些实际示例:

```
\DefaultInheritance{all=true,override=false}
```

该示例给出了怎么设置默认的继承行为。该设置是包的默认设置。

```
\DefaultInheritance[
  \except{*}{online}{all=false}
]{all=true,override=false}
```

该示例类似于上一示例，差别在于增加了一个例外规则:`@online` 类型的条目默认将不从任何父条目继承数据。

```
\DeclareDataInheritance{collection}{incollection}{
  \inherit{title}{booktitle}
  \inherit{subtitle}{booksubtitle}
  \inherit{titleaddon}{booktitleaddon}
}
```

到目前为止，我们已经看到了标准的继承设置。例如 `all=true` 意味着一个源条目的`publisher` 域将被复制到目标条目的`publisher` 域中。然而，在一些情况下，需要非对称的映射。它们通过`\DeclareDataInheritance` 来定义。上面的示例为`@incollection` 条目引用`@collection` 信息设置了 3 条典型规则。将源条目的`title` 及其相关域映射到目标条目对应的`booktitle` 相关域中。

```
\DeclareDataInheritance{mvbook,book}{inbook,bookinbook}{
  \inherit{author}{author}
  \inherit{author}{bookauthor}
}
```

这一规则是一个一对多映射的规则: 为了能运行压缩`inbook/bookinbook` 条目，它将源条目的`author` 域映射到目标条目的`author` 和`bookauthor` 域中。源可以是一个`@mvbook` 或`@book` 条目，目标可以是一个`@inbook` 或`@bookinbook` 条目。

```
\DeclareDataInheritance{*}{inbook,incollection}{
  \noinherit{introduction}
}
```

这一规则阻止对introduction 域的继承。应用的目标条目是@inbook 或@incollection，源条目则是任意的。

```
\DeclareDataInheritance{*}{*}{
  \noinherit{abstract}
}
```

该规则应用于所有条目类型，阻止abstract 域的继承。

```
\DefaultInheritance{all=true,override=false}
\ResetDataInheritance
```

该例展示怎么模拟实现传统的 BibTeX 的交叉引用机制。它默认打开继承功能，禁止覆盖，并清除所有的其它规则和映射。

在一个参考文献条目中，当值是由\DeclareDatafieldSet(§ 4.5.2) 定义的数据域集，可以给出一个‘noinherit’ 选项。这会阻止具体条目对在该集中的域的继承。例如:

```
\DeclareDatafieldSet{nobtitle}{
  \member[field=booktitle]
}
```

```
@INBOOK{s1,
  OPTIONS  = {noinherit=nobtitle},
  TITLE    = {Subtitle},
  CROSSREF = {s2}
}

@BOOK{s2,
  TITLE = {Title}
}
```

这里s1 不会将s2 的TITLE 继承为BOOKTITLE，因为这已经被数据域集所阻止，该集以noinherit 选项的值的方式给出。

需要重点注意的是，如果他们已经具有某一类型日期的某一成分，子条目不会从父条目继承该类型日期的任何成分域。例如:

```
@INBOOK{b1,
  DATE     = {2004-03-03},
  ORIGDATE = {2004-03},
  CROSSREF = {b2}
}
```

```
@BOOK{b2,
  DATE      = {2004-03-03/2005-08-09},
  ORIGDATE  = {2004-03/2005-08},
  EVENTDATE = {2004-03/2005-08},
}
```

这里，b1 条目将不会继承任何的endyear, endmonth, endday, origendyear or origendmonth，因为这可能导致与自身日期的混乱。考虑默认继承规则，它将继承所有的event* 日期成分。

4.6 辅助命令

本节的工具用来分析和保存参考文献数据而不是对其进行格式化或者打印。

4.6.1 数据命令

本节的命令允许对未格式化的参考文献数据进行底层访问。这些命令不是用来输出，而是用来将数据保存到临时宏中，可以用于下一步的比较。

\thefield{⟨*field*⟩}

展开为未格式化的⟨*field*⟩。如果⟨*field*⟩ 未定义那么展开为一个空字符串。

\strfield{⟨*field*⟩}

类似于\thefield 命令，但其值经自动处理 (sanitized)，以便安全地用于构成控制序列名。

\csfield{⟨*field*⟩}

类似于\thefield 命令，但禁止展开

\usefield{⟨*command*⟩}{⟨*field*⟩}

执行⟨*command*⟩ 命令使用未格式化的⟨*field*⟩ 作为其参数

\thelist{⟨*literal list*⟩}

展开为未格式化的⟨*literal list*⟩。如果 list 未定义那么展开为一个空字符串。注意该命令中将⟨*literal list*⟩ 转存为本宏包使用的内部格式。这一格式不适合打印。

\strlist{⟨*literal list*⟩}

类似于\thelist，差别在于该命令能自动处理列表的内部表示，因此列表的值可以安全地用于控制序列名的构建。

\thename{⟨*name list*⟩}

展开为未格式化的⟨*name list*⟩。如果 list 未定义那么展开为一个空字符串。注意该命令中将⟨*name list*⟩ 转存为本宏包使用的内部格式。这一格式不适合打印。

\strname{⟨*name list*⟩}

类似于\thename，差别在于该命令能自动处理列表的内部表示，因此列表的值可以安全地用于控制序列名的构建。

\savefield{⟨*field*⟩}{⟨*macro*⟩}
\savefield*{⟨*field*⟩}{⟨*macro*⟩}

将未格式化的⟨*field*⟩ 拷贝到一个⟨*macro*⟩ 中。不带星的命令全局的定义⟨*macro*⟩，而带星的命令是局部定义。

\savelist{⟨*literal list*⟩}{⟨*macro*⟩}
\savelist*{⟨*literal list*⟩}{⟨*macro*⟩}

将未格式化的⟨*literal list*⟩ 拷贝到一个⟨*macro*⟩ 中。不带星的命令全局的定义⟨*macro*⟩，而带星的命令是局部定义。

\savename{⟨*name list*⟩}{⟨*macro*⟩}
\savename*{⟨*name list*⟩}{⟨*macro*⟩}

将未格式化的⟨*name list*⟩ 拷贝到一个⟨*macro*⟩ 中。不带星的命令全局的定义⟨*macro*⟩，而带星的命令是局部定义。

\savefieldcs{⟨*field*⟩}{⟨*csname*⟩}
\savefieldcs*{⟨*field*⟩}{⟨*csname*⟩}

类似于\savefield 命令，但将控制序列名⟨*csname*⟩(即没有斜杠) 作为参数，而不是宏。

\savelistcs{⟨*literal list*⟩}{⟨*csname*⟩}
\savelistcs*{⟨*literal list*⟩}{⟨*csname*⟩}

类似于\savelist 命令，但将控制序列名⟨*csname*⟩(即没有斜杠) 作为参数，而不是宏。

\savenamecs{⟨*name list*⟩}{⟨*csname*⟩}
\savenamecs*{⟨*name list*⟩}{⟨*csname*⟩}

类似于\savename 命令，但将控制序列名⟨*csname*⟩(即没有斜杠) 作为参数，而不是宏。

\restorefield{⟨*field*⟩}{⟨*macro*⟩}

从之前用\savefield 命令定义的⟨*macro*⟩ 中将⟨*field*⟩ 恢复回来。该域是在局部范围内恢复。

\restorelist{⟨*literal list*⟩}{⟨*macro*⟩}

从之前用\savelist 命令定义的⟨*macro*⟩ 中将⟨*literal list*⟩ 恢复回来。该 list 是在局部范围内恢复。

`\restorename{`⟨*name list*⟩`}{`⟨*macro*⟩`}`

从之前用`\savename` 命令定义的⟨*macro*⟩ 中将⟨*name list*⟩ 恢复回来。该 list 是在局部范围内恢复。

`\clearfield{`⟨*field*⟩`}`

在局部范围内清除⟨*field*⟩。以这种方式清除的域对于后续的数据命令来说相当于没有定义。

`\clearlist{`⟨*literal list*⟩`}`

在局部范围内清除⟨*literal list*⟩。以这种方式清除的 list 对于后续的数据命令来说相当于没有定义。

`\clearname{`⟨*name list*⟩`}`

在局部范围内清除⟨*name list*⟩。以这种方式清除的 list 对于后续的数据命令来说相当于没有定义。

4.6.2 独立判断命令

本节的命令是不同类型的独立 (stand-alone) 判断命令，用于参考文献著录和标注样式中。

`\if<datetype>julian{`⟨*true*⟩`}{`⟨*false*⟩`}`

当日期'datetype'date 因为`julian` 和`gregorianstart` 选项的设置转换为儒略历 (Julian Calendar) 时，展开为⟨*true*⟩。

`\ifdatejulian{`⟨*true*⟩`}{`⟨*false*⟩`}`

类似于`\if<datetype>julian` 但用于`\mkbibdate*` 格式化命令中 (§ 4.10.2)，在这些格式化命令中恰当使用`\if<datetype>julian` 命令等价于应用该命令。

`\if<datetype>dateera{`⟨*era*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

当日期'datetype'date(`date`, `urldate`, `eventdate` 等) 指定了一个纪元等于⟨*era*⟩，则展开为⟨*true*⟩，否则展开为⟨*false*⟩。`biber` 确认并在`.bbl` 文件中传递的可用⟨*era*⟩ 字符串是:

bceBCE/BC era

ceCE/AD era

该命令用于确定是否打印 § 4.9.2.21 节的本地化字符串[61]。

`\ifdateera{`⟨*era*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

类似于`\if<datetype>dateera`，但用于`\mkbibdate*` 格式化命令 (§ 4.10.2)，在这些格式化命令中恰当使用`\if<datetype>dateera` 命令等价于应用该命令。

[61]译者: 这里的 location strings 应是笔误，而应是 local strings，这从 § 4.9.2.21 节内容可以看出

\if<datetype>datecirca{⟨*true*⟩}{⟨*false*⟩}

当日期'datetype'date(date, urldate, eventdate 等) 在数据源中具有一个'circa' 标记时，则展开为⟨*true*⟩，否则展开为⟨*false*⟩。参见 § 2.3.8。该命令用于确定是否打印 § 4.9.2.21 节中的本地化字符串。

\ifdatecirca{⟨*true*⟩}{⟨*false*⟩}

类似于\if<datetype>datecirca，但用于\mkbibdate* 格式化命令 (§ 4.10.2)，在这些格式化命令中恰当使用的\if<datetype>datecirca 命令等价于该命令。

\if<datetype>dateuncertain{⟨*true*⟩}{⟨*false*⟩}

当日期'datetype'date(date, urldate, eventdate 等) 在数据源中具有一个不确定标记时，则展开为⟨*true*⟩，否则展开为⟨*false*⟩。参见 § 2.3.8。该命令用于确定是否打印例如年份后的一个问号。

\ifdateuncertain{⟨*true*⟩}{⟨*false*⟩}

类似于\if<datetype>dateuncertain，但用于\mkbibdate* 格式化命令 (§ 4.10.2)，在这些格式化命令中恰当使用\if<datetype>dateuncertain 命令等价于应用该命令。

\ifenddateuncertain{⟨*true*⟩}{⟨*false*⟩}

类似于\ifend<datetype>dateuncertain，但用于\mkbibdate* 格式化命令 (§ 4.10.2)，在这些格式化命令中恰当使用\ifend<datetype>dateuncertain 命令等价于应用该命令。

\ifcaselang[⟨*language*⟩]{⟨*true*⟩}{⟨*false*⟩}

如果可选的⟨*language*⟩ 是\DeclareCaseLangs(见 § 4.6.4) 声明的语言之一，展开为⟨*true*⟩，否则展开为⟨*false*⟩。当可选参数不给出时，对\currentlang 值进行判断。

\ifsortnamekeyscheme{⟨*string*⟩}{⟨*true*⟩}{⟨*false*⟩}

如果⟨*string*⟩ 与当前作用范围内姓名排序关键词格式名 (见4.5.6) 相同则展开为⟨*true*⟩，否则展开为⟨*false*⟩。

\iffieldundef{⟨*field*⟩}{⟨*true*⟩}{⟨*false*⟩}

展开为⟨*true*⟩，如果⟨*field*⟩ 未定义，否则展开为⟨*false*⟩

\iflistundef{⟨*literal list*⟩}{⟨*true*⟩}{⟨*false*⟩}

展开为⟨*true*⟩，如果⟨*literal list*⟩ 未定义，否则展开为⟨*false*⟩

\ifnameundef{⟨*name list*⟩}{⟨*true*⟩}{⟨*false*⟩}

展开为⟨*true*⟩，如果⟨*name list*⟩ 未定义，否则展开为⟨*false*⟩

`\ifieldsequal{`⟨*field 1*⟩`}{`⟨*field 2*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

展开为⟨*true*⟩，如果⟨*field 1*⟩ 和⟨*field 2*⟩ 相等，否则展开为⟨*false*⟩

`\iflistsequal{`⟨*literal list 1*⟩`}{`⟨*literal list 2*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

展开为⟨*true*⟩，如果⟨*literal list 1*⟩ 和⟨*literal list 2*⟩ 相等，否则展开为⟨*false*⟩

`\ifnamesequal{`⟨*name list 1*⟩`}{`⟨*name list 2*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

展开为⟨*true*⟩，如果⟨*name list 1*⟩ 和⟨*name list 2*⟩ 相等，否则展开为⟨*false*⟩

`\ifieldequals{`⟨*field*⟩`}{`⟨*macro*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

展开为⟨*true*⟩，如果⟨*field*⟩ 的值和⟨*macro*⟩ 的定义相等，否则展开为⟨*false*⟩。[62]

`\iflistequals{`⟨*literal list*⟩`}{`⟨*macro*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

展开为⟨*true*⟩，如果⟨*literal list*⟩ 的值和⟨*macro*⟩ 的定义相等，否则展开为⟨*false*⟩。

`\ifnameequals{`⟨*name list*⟩`}{`⟨*macro*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

展开为⟨*true*⟩，如果⟨*name list*⟩ 的值和⟨*macro*⟩ 的定义相等，否则展开为⟨*false*⟩。

`\ifieldequalcs{`⟨*field*⟩`}{`⟨*csname*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

类似于`\ifieldequals`，但将控制序列名⟨*csname*⟩(不带斜杠) 作为参数，而不是一个宏名。

`\iflistequalcs{`⟨*literal list*⟩`}{`⟨*csname*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

类似于`\iflistequals`，但将控制序列名⟨*csname*⟩(不带斜杠) 作为参数，而不是一个宏名。

`\ifnameequalcs{`⟨*name list*⟩`}{`⟨*csname*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

类似于`\ifnameequals`，但将控制序列名⟨*csname*⟩(不带斜杠) 作为参数，而不是一个宏名。

`\ifieldequalstr{`⟨*field*⟩`}{`⟨*string*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

展开为⟨*true*⟩，如果⟨*field*⟩ 的值和字符串⟨*string*⟩ 的定义相等，否则展开为⟨*false*⟩。该命令是鲁棒的。

`\iffieldxref{`⟨*field*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

如果一个条目定义了`crossref/xref`，该命令检测⟨*field*⟩ 是否与 cross"=referenced 父条目相关联。如果子条目的⟨*field*⟩ 与父条目对应的⟨*field*⟩ 相等，那么执行⟨*true*⟩，否则执行⟨*false*⟩。如果`crossref/xref` 未定义，总是执行⟨*false*⟩。该命令是鲁棒的。`crossref` 和 `xref` 域的描述见 § 2.2.3，更多关于 cross"=referencing 的信息见 § 2.4.1。

[62] 译者: 比如应用于 gb7714-2015 中的新闻和标准条目类型的判断

\iflistxref{⟨*literal list*⟩}{⟨*true*⟩}{⟨*false*⟩}

类似于\iffieldxref 命令，但检测⟨*literal list*⟩ 是否与 cross"=referenced 父条目相关联。crossref 和 xref 域的描述见 § 2.2.3，更多关于 cross"=referencing 的信息见 § 2.4.1。

\ifnamexref{⟨*name list*⟩}{⟨*true*⟩}{⟨*false*⟩}

类似于\iffieldxref 命令，但检测⟨*name list*⟩ 是否与 cross"=referenced 父条目相关联。crossref 和 xref 域的描述见 § 2.2.3，更多关于 cross"=referencing 的信息见 § 2.4.1。

\ifcurrentfield{⟨*field*⟩}{⟨*true*⟩}{⟨*false*⟩}

执行⟨*true*⟩，如果当前域为⟨*field*⟩，否则执行⟨*false*⟩。该命令是鲁棒的。它主要用于域格式指令中，如果在其它环境中总是执行⟨*false*⟩。

\ifcurrentlist{⟨*literal list*⟩}{⟨*true*⟩}{⟨*false*⟩}

执行⟨*true*⟩，如果当前 list 为⟨*literal list*⟩，否则执行⟨*false*⟩。该命令是鲁棒的。它主要用于域格式指令中，如果在其它环境中总是执行⟨*false*⟩。

\ifcurrentname{⟨*name list*⟩}{⟨*true*⟩}{⟨*false*⟩}

执行⟨*true*⟩，如果当前 list 为⟨*name list*⟩，否则执行⟨*false*⟩。该命令是鲁棒的。它主要用于域格式指令中，如果在其它环境中总是执行⟨*false*⟩。

\ifuseprefix{⟨*true*⟩}{⟨*false*⟩}

执行⟨*true*⟩，如果useprefix 选项打开 (无论是全局的还是针对当前条目)，否则执行⟨*false*⟩。该选项的细节见 § 3.1.3。

\ifuseauthor{⟨*true*⟩}{⟨*false*⟩}

这只是下面的\ifuse<name> 宏的一个特例，因为author 是默认数据模型的一部分所以放到这里来说。如果useauthor 选项打开 (无论是全局的还是针对当前条目)，执行⟨*true*⟩，否则执行⟨*false*⟩。该选项的细节见 § 3.1.3。

\ifuseeditor{⟨*true*⟩}{⟨*false*⟩}

这只是下面的\ifuse<name> 宏的一个特例，因为editor 是默认数据模型的一部分所以放到这里来说。如果useeditor 选项打开 (无论是全局的还是针对当前条目)，执行⟨*true*⟩，否则执行⟨*false*⟩。该选项的细节见 § 3.1.3。

\ifusetranslator{⟨*true*⟩}{⟨*false*⟩}

这只是下面的\ifuse<name> 宏的一个特例，因为translator 是默认数据模型的一部分所以放到这里来说。如果usetranslator 选项打开 (无论是全局的还是针对当前条目)，执行⟨*true*⟩，否则执行⟨*false*⟩。该选项的细节见 § 3.1.3。

\ifuse<name>{⟨*true*⟩}{⟨*false*⟩}

展开为⟨*true*⟩，如果选项use<name> 打开 (无论全局还是当前条目的选项)，否则展开为⟨*false*⟩。这一选项的细节详见第 § 3.1.3 节。

\ifcrossrefsource{⟨*true*⟩}{⟨*false*⟩}

展开为⟨*true*⟩，如果包含在.bbl 中的条目的间接引用 (referenced，译者: 是交叉引用) 次数大于mincrossrefs，否则展开为⟨*false*⟩。见 § 3.1.2.1。如果条目被直接引用则展开为⟨*false*⟩。

\ifxrefsource{⟨*true*⟩}{⟨*false*⟩}

展开为⟨*true*⟩，如果包含在.bbl 中的条目的间接引用 (referenced) 次数大于minxrefs，否则展开为⟨*false*⟩。见 § 3.1.2.1。如果条目被直接引用则展开为⟨*false*⟩。类似于 ifcrossrefsource，但针对 xref 域。

\ifsingletitle{⟨*true*⟩}{⟨*false*⟩}

展开为⟨*true*⟩，如果文献表中只有一篇文献具有labelname，否则展开为⟨*false*⟩。如果条目的labelname 未设置，总是展开为⟨*false*⟩。注意: 使用该功能需要显式打开宏包选项singletitle。

\ifuniquetitle{⟨*true*⟩}{⟨*false*⟩}

展开为⟨*true*⟩，如果只有一篇文献的题名是labeltitle，否则展开为⟨*false*⟩。如果条目的labeltitle 未设置，总是展开为⟨*false*⟩。注意: 要使用这一功能需要显式地打开包选项uniquetitle。

\ifuniquebaretitle{⟨*true*⟩}{⟨*false*⟩}

展开为⟨*true*⟩，如果labelname 域为空且只有一篇文献的题名是labeltitle，否则展开为⟨*false*⟩。如果条目的labeltitle 未设置，总是展开为⟨*false*⟩。注意: 要使用这一功能需要显式地打开包选项uniquebaretitle。

\ifuniquework{⟨*true*⟩}{⟨*false*⟩}

展开为⟨*true*⟩，如果文献表中只有一篇文献的标签名是labelname 且题名是labeltitle，否则展开为⟨*false*⟩。如果条目的labelname 和labeltitle 均未设置，总是展开为⟨*false*⟩。注意: 要使用这一功能需要显式地打开包选项uniquework。如果同一条目的singletitle 和uniquetitle 都是 false，可能是因为有其他条目具有相同的labelname，还有其他条目具有相同的labeltitle。uniquework 可以让我们知道是否有条目同时具有相同的labelname 和labeltitle。这在多人合作时很有用，当多人同时维护一个参考文献数据源时，可能存在添加内容相同但条目关键词不同的文献的风险。这一判断能帮助找到这种存在副本情况。

\ifuniqueprimaryauthor{⟨*true*⟩}{⟨*false*⟩}

展开为⟨*true*⟩，如果一篇文献的对于其labelname 的第一作者的姓是唯一的，否则展开为⟨*false*⟩。如果条目的labelname 未设置，将展开为⟨*false*⟩。注意: 使用该功能需要显式打开包选项uniqueprimaryauthor。

`\ifandothers{`⟨*list*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

展开为⟨*true*⟩，如果⟨*list*⟩ 已定义并且在bib 文件中以关键词‘and others’截短了，否则展开为⟨*false*⟩。⟨*list*⟩ 可以是 literal 或 name 列表。

`\ifmorenames{`⟨*true*⟩`}{`⟨*false*⟩`}`

展开为⟨*true*⟩，如果当前姓名列表已经截短或将截短，否则展开为⟨*false*⟩。该命令用于姓名列表的格式化指令中，在其它地方使用将展开为⟨*false*⟩。该命令对当前列表执行与\ifandothers 判断一样的操作。如果判断结果为否，它还将检测listtotal 是否大于liststop。该命令用于格式化命令中用以决定是否需要在列表默认打印“and others” or “et al.”这样的标注。注意: 还需要检测是否处于列表中间或者末尾时，即listcount 是否小于或等于liststop，详见第 § 4.4.1 节。

`\ifmoreitems{`⟨*true*⟩`}{`⟨*false*⟩`}`

类似于\ifmorenames，但检测 literal 列表。用于 literal 列表的格式化指令，在其它地方使用总是展开为⟨*false*⟩。

`\if<namepart>inits{`⟨*true*⟩`}{`⟨*false*⟩`}`

根据firstinits 包选项的状态，展开为⟨*true*⟩ 或⟨*false*⟩(见第 § 3.1.2.3 节)。该命令用于姓名列表的格式化指令。

`\ifterseinits{`⟨*true*⟩`}{`⟨*false*⟩`}`

根据terseinits 包选项的状态，展开为⟨*true*⟩ 或⟨*false*⟩(见第 § 3.1.2.3 节)。该命令用于姓名列表的格式化指令。

`\ifentrytype{`⟨*type*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

如果当前处理条目类型是⟨*type*⟩，则展开为⟨*true*⟩，否则展开为⟨*false*⟩。

`\ifkeyword{`⟨*keyword*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

如果⟨*keyword*⟩ 能在当前处理条目的keywords 域中找到，展开为⟨*true*⟩，否则展开为⟨*false*⟩。

`\ifentrykeyword{`⟨*entrykey*⟩`}{`⟨*keyword*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

将条目关键词作为第一个参数的\ifkeyword 命令的变化形式，用于判断当前处理条目是否是某一条目。

`\ifcategory{`⟨*category*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

执行⟨*true*⟩，如果当前正在处理条目被指派到由\addtocategory 命令定义的⟨*category*⟩ 类中，否则执行⟨*false*⟩。

`\ifentrycategory{`⟨*entrykey*⟩`}{`⟨*category*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

将条目关键词作为第一个参数的\ifcategory 命令的变化形式，用于判断当前处理条目是否是某一条目。

\ifciteseen{⟨*true*⟩}{⟨*false*⟩}

展开为⟨*true*⟩，如果当前条目之前已经被引用过，否则展开为⟨*false*⟩。该命令是鲁棒的，用于标注样式中。如果文档中有refsection 环境，引用追踪是基于这些环境的。注意: 引用追踪器需要显式打开包选项citetracker，如果追踪器未打开，该命令总是展开为⟨*false*⟩。另可参见第 § 4.6.4 节的\citetrackertrue 和\citetrackerfalse 开关。

\ifentryseen{⟨*entrykey*⟩}{⟨*true*⟩}{⟨*false*⟩}

将条目关键词作为第一个参数的 \ifciteseen 命令的变化形式。因为⟨*entrykey*⟩ 先于判断展开，它也可以用来测试在xref 等域中的条目关键词。

```
\ifentryseen{\thefield{xref}}{true}{false}
```

除了一个额外参数，\ifentryseen 的操作类似于\ifciteseen。

\ifentryinbib{⟨*entrykey*⟩}{⟨*true*⟩}{⟨*false*⟩}

如果⟨*entrykey*⟩ 出现当前文献表中，执行⟨*true*⟩，否则执行⟨*false*⟩。该命令用于参考文献著录样式。

\iffirstcitekey{⟨*true*⟩}{⟨*false*⟩}

如果当前处理条目是引用列表中的第一个条目，执行⟨*true*⟩，否则执行⟨*false*⟩。该命令依赖于citecount, citetotal, multicitecount 和 multicitetotal 计数器 (见 § 4.10.5)，因此只能用于\DeclareCiteCommand 命令定义的标注命令的循环执行代码⟨*loopcode*⟩ 中。

\iflastcitekey{⟨*true*⟩}{⟨*false*⟩}

类似于\iffirstcitekey，但判断的是是否为引用列表中的最后一个条目。

\ifciteibid{⟨*true*⟩}{⟨*false*⟩}

如果当前处理条目与前一条相同，展开为⟨*true*⟩，否则展开为⟨*false*⟩。该命令用于标注样式。如果有refsection 环境，追踪器是基于这些环境的。注意: 'ibidem' 追踪器需要由ibidtracker 包选项显式打开。该判断命令的运行方式与追踪器运行的模式相关，详见 § 3.1.2.3。如果追踪器未打开，总是展开为⟨*false*⟩。另可参见 § 4.6.4 节的\citetrackertrue 和\citetrackerfalse 开关。

\ifciteidem{⟨*true*⟩}{⟨*false*⟩}

如果当前处理条目的责任者 (即作者或编者) 与前一条目的相同，展开为⟨*true*⟩，否则展开为⟨*false*⟩。该命令用于标注样式。如果有refsection 环境，追踪器是基于这些环境的。注意: 'idem' 追踪器需要由idemtracker 包选项显式打开。该判断命令的运行方式与追踪器运行的模式相关，详见 § 3.1.2.3。如果追踪器未打开，总是展开为⟨*false*⟩。另可参见 § 4.6.4 节的\citetrackertrue 和\citetrackerfalse 开关。

\ifopcit{⟨*true*⟩}{⟨*false*⟩}

该命令类似于\ifciteibid，但只要当前处理等条目的作者或编者 与前一条目相同，则展开为⟨*true*⟩。注意: 'opcit' 追踪器需要由opcittracker 包选项显式的打开。该判断命令的运行方式与追踪器运行的模式相关，详见 § 3.1.2.3。如果追踪器未打开，总是展开为⟨*false*⟩。另可参见 § 4.6.4 节的\citetrackertrue 和\citetrackerfalse 开关。

\ifloccit{⟨*true*⟩}{⟨*false*⟩}

该命令类似于\ifopcit，但还要比较⟨*postnote*⟩ 的参数，如果他们相同且是数值 (§ 4.6.2 节的\ifnumerals 命令判断)，则展开为⟨*true*⟩。即: 如果引文的页码与前一文献相同则展开为 true。注意: 'loccit' 追踪器需要由loccittracker 包选项显式打开。该判断命令的运行方式与追踪器运行的模式相关，详见 § 3.1.2.3。如果追踪器未打开，总是展开为⟨*false*⟩。另可参见 § 4.6.4 节的\citetrackertrue 和\citetrackerfalse 开关。

\iffirstonpage{⟨*true*⟩}{⟨*false*⟩}

该命令的运行与pagetracker 包选项相关，如果选项设置成 page，当前项是页中的第一项，展开为⟨*true*⟩，否则展开为⟨*false*⟩。如果选项设置成 spread，当前项是合页 (double-page spread) 中的第一项，展开为⟨*true*⟩，否则展开为⟨*false*⟩。如果选项未打开，总是展开为⟨*false*⟩。根据所处环境不同，'item' 可以是一个标注，或者参考文献表中的条目。注意该命令区分正文文本和脚注，例如，当在某页的第一个脚注中使用，即便是文中有一个标注且先于该脚注，它也展开为⟨*true*⟩。另可参见 § 4.6.4 节的\pagetrackertrue 和\pagetrackerfalse 开关。

\ifsamepage{⟨*instance 1*⟩}{⟨*instance 2*⟩}{⟨*true*⟩}{⟨*false*⟩}

如果两个引用实例位于同于页或者同一合页中，展开为⟨*true*⟩，否则为⟨*false*⟩。一个引用实例可以是一个标注也可以是文献表中的条目。这些实例用instcount 计数区分，见 § 4.10.5。该命令的运行与pagetracker 包选项相关，如果选项设置成 spread，其本质是'if same spread'(是否同一合页) 的判断。如果选项未打开，总是展开为⟨*false*⟩。参数⟨*instance 1*⟩ 和⟨*instance 2*⟩ 以e-TeX's \numexpr 方式当成整数表达式处理。这意味着可以在参数中计算。比如:

```
\ifsamepage{\value{instcount}}{\value{instcount}-1}{true}{false}
```

注意: \value 命令不是以\the 为前缀，在第二个参数中做了减法运算。如果⟨*instance 1*⟩ 或 ⟨*instance 2*⟩ 是无效数字 (比如一个负值)，总是展开为⟨*false*⟩。也要注意该命令不区分正文和脚注。另可参见 § 4.6.4 节的\pagetrackertrue 和\pagetrackerfalse 开关。

\ifinteger{⟨*string*⟩}{⟨*true*⟩}{⟨*false*⟩}

如果⟨*string*⟩ 是一个正整数，展开为⟨*true*⟩，否则为⟨*false*⟩，该命令鲁棒。

\ifnumeral{⟨string⟩}{⟨true⟩}{⟨false⟩}

如果⟨string⟩是一个阿拉伯或者罗马数字，展开为⟨true⟩，否则为⟨false⟩，该命令鲁棒。另可参见§ 4.6.4 节的\DeclareNumChars 和\NumCheckSetup 命令。

\ifnumerals{⟨string⟩}{⟨true⟩}{⟨false⟩}

如果⟨string⟩是一个阿拉伯或者罗马数字的范围或列表，展开为⟨true⟩，否则为⟨false⟩，该命令鲁棒。相比于\ifnumeral 命令，当参数像“52–58”,“14/15”,“1, 3, 5”等时，该命令会执行⟨true⟩。另可参见§ 4.6.4 节的\DeclareNumChars，\NumCheckSetup，\DeclareRangeCommands 和 \NumCheckSetup 命令。

\ifpages{⟨string⟩}{⟨true⟩}{⟨false⟩}

类似于\ifnumerals，但也考虑§ 4.6.4 节的\DeclarePageCommands 命令。

\iffieldint{⟨field⟩}{⟨true⟩}{⟨false⟩}

类似于\ifinteger 命令，但使用⟨field⟩的值而不是一个字符串，如果域未定义，执行⟨false⟩。

\iffieldnum{⟨field⟩}{⟨true⟩}{⟨false⟩}

类似于\ifnumeral 命令，但使用⟨field⟩的值而不是一个字符串，如果域未定义，执行⟨false⟩。

\iffieldnums{⟨field⟩}{⟨true⟩}{⟨false⟩}

类似于\ifnumerals 命令，但使用⟨field⟩的值而不是一个字符串，如果域未定义，执行⟨false⟩。

\iffieldpages{⟨field⟩}{⟨true⟩}{⟨false⟩}

类似于\ifpages 命令，但使用⟨field⟩的值而不是一个字符串，如果域未定义，执行⟨false⟩。

\ifbibstring{⟨string⟩}{⟨true⟩}{⟨false⟩}

如果⟨string⟩是已知的本地化关键词，展开为⟨true⟩，否则⟨false⟩。默认定义的本地化字符串见§ 4.9.2。新的字符串可以用命令\NewBibliographyString 定义。

\ifbibxstring{⟨string⟩}{⟨true⟩}{⟨false⟩}

类似于\ifbibstring，但⟨string⟩是展开的。

\iffieldbibstring{⟨field⟩}{⟨true⟩}{⟨false⟩}

类似于\ifbibstring，但使用⟨field⟩域的值而不是一个字符串，如果域未定义，执行⟨false⟩。

\ifdriver{⟨entrytype⟩}{⟨true⟩}{⟨false⟩}

展开为⟨true⟩如果⟨entrytype⟩的驱动存在，否则为⟨false⟩。

\ifcapital{⟨*true*⟩}{⟨*false*⟩}

如果biblatex 的标点追踪器将当前位置的本地化字符串大写，则执行⟨*true*⟩，否则执行⟨*false*⟩。该命令鲁棒，用于格式化指令中对姓名的某些成分做有条件的大写。

\ifcitation{⟨*true*⟩}{⟨*false*⟩}

当处于标注中则展开为⟨*true*⟩，否则为⟨*false*⟩。注意这一命令与其所在的最外层环境有关。比如，当由\DeclareCiteCommand 命令定义的标注命令执行一个由\DeclareBibliographyDriver 定义的驱动，则任何在该驱动中的\ifcitation 都会展开为⟨*true*⟩。一个例子见 § 4.11.6 节。

\ifbibliography{⟨*true*⟩}{⟨*false*⟩}

当处于文献表中则展开为⟨*true*⟩，否则为⟨*false*⟩。注意这一命令与其所在的最外层环境有关。比如，当由\DeclareBibliographyDriver 命令定义的驱动执行一个由\DeclareCiteCommand 定义的标注，则任何在该标注中的\ifbibliography 都会展开为⟨*true*⟩。一个例子见 § 4.11.6 节。

\ifnatbibmode{⟨*true*⟩}{⟨*false*⟩}

根据 § 3.1.1 节的natbib 选项展开为⟨*true*⟩ 或⟨*false*⟩。

\ifciteindex{⟨*true*⟩}{⟨*false*⟩}

根据 § 3.1.2.1 节的indexing 选项展开为⟨*true*⟩ 或⟨*false*⟩。

\ifbibindex{⟨*true*⟩}{⟨*false*⟩}

根据 § 3.1.2.1 节的indexing 选项展开为⟨*true*⟩ 或⟨*false*⟩。

\iffootnote{⟨*true*⟩}{⟨*false*⟩}

当处于脚注中时，展开为⟨*true*⟩，否则为⟨*false*⟩。注意: 在minipage 中的脚注被认为正文的一部分。当处于页面底部的脚注中或者由endnotes 提供的 endnotes 中时，只会展开为⟨*true*⟩。

citecounter 这一计数器表示当前处理条目在当前 reference section 中的引用次数。注意: 该功能需要显式打开包选项citecounter。如果选项设置为 context，正文和脚注中的引用分别计数。这种情况下，citecounter 记录其所在环境中的值。

uniquename 这一计数器用于labelname 列表。它以每个名字为基础进行设置。如果姓不同，它的值设置为 0，当增加姓名的其它成分的首字母使得姓名能区分时，则设置为 1，如果需要完整的姓名才能区分，则设置为 2。作者年制和作者标题制的标注格式需要这一信息来增加姓名的其它成分以对同姓的不同作者进行引用。比如: (考虑默认的\DeclareUniquenameTemplate 定义) 当引用列表中有一个‘John Doe’ 和一个‘Edward Doe’，该计数器将设置为 1。如果有一个‘John Doe’ 和一个‘Jane Doe’，该计数器将设置为 2。如果选项设置成 init/allinit/mininit，那么计数器将限制值最大为 1。这对于标注样式不打印全名而使用首字母来区分姓名很有用。如果添加首字母还无法区分姓名，uniquename 也将设置为 0。该功能需要显式打开包选

项uniquename。注意uniquename 是\printnames 局部使用的，仅根据labelname 列表或其来源姓名列表 (典型如author 或editor) 设置。它的值在其它环境中都是 0，即它仅在处理姓名的格式化指令中计算，更多细节和实例见 § 4.11.4。

uniquelist 该计数器用于labelname 列表。它以每个域为基础进行设置。它的值表示当使用maxnames/minnames 自动将姓名列表截短后导致标注歧义时，消除歧义需要的最小姓名数。比如，有一篇作者是'Doe/Smith/Johnson' 的文献和另一篇作者是'Doe/Edwards/Williams' 的文献，设置 maxnames=1 将导致两篇的作者都是'Doe et al.'。这种情况下，两个条目的labelname 列表的uniquelist 将设置成 2，因为至少需要两个名字来区分。注意uniquelist 是\printnames 命令局部使用的，仅根据labelname 列表或其来源姓名列表 (典型如author 或editor) 设置。它的值在其它环境中都是 0，即它仅在处理姓名的格式化指令中计算，如果该值存在，则\printnames 命令在处理姓名列表时将自动应用，即自动覆盖maxnames/minnames。该功能需要显式打开选项uniquelist。更多细节和实例见 § 4.11.4。

parenlevel 圆括号和/或方括号的嵌套层级。该信息仅在 § 3.1.2.3 的parentracker 选项打开的情况下提供。

4.6.3 使用\ifboolexpr 和\ifthenelse 的判断

第 § 4.6.2 节介绍的判断可以与etoolbox 宏包提供的\ifboolexpr 命令和ifthen 宏包提供的\ifthenelse 命令一同使用。这种情况下，其语法略有差异，判断命令的⟨*true*⟩ 和⟨*false*⟩ 参数自动省略，而直接传递给\ifboolexpr 或 \ifthenelse。注意，使用这些命令需要一些计算代价。如果不需要任何布尔运算，使用 § 4.6.2 节的独立判断命令更高效。

\ifboolexpr{⟨*expression*⟩}{⟨*true*⟩}{⟨*false*⟩}

该etoolbox 包命令允许进行包括布尔运算和编组的复杂判断。

```
\ifboolexpr{ (
               test {\ifnameundef{editor}}
               and
               not test {\iflistundef{location}}
             )
             or test {\iffieldundef{year}}
  }
  {...}
  {...}
```

\ifthenelse{⟨*tests*⟩}{⟨*true*⟩}{⟨*false*⟩}

该ifthen 包命令允许进行包括布尔运算和编组的复杂判断。

```
\ifthenelse{ \(
               \ifnameundef{editor}
```

```
                \and
                \not \iflistundef{location}
            \)
            \or \iffieldundef{year}
}
{...}
{...}
```

biblatex 提供的附加判断命令，仅在标注命令和文献表中使用\ifboolexpr或\ifthenelse时可用。

4.6.4 综合命令

本节介绍参考文献著录和标注样式中使用的一些综合命令和小助手。

\newbibmacro{⟨*name*⟩}[⟨*arguments*⟩][⟨*optional*⟩]{⟨*definition*⟩}
\newbibmacro*{⟨*name*⟩}[⟨*arguments*⟩][⟨*optional*⟩]{⟨*definition*⟩}

定义一个用于后面\usebibmacro 调用的宏。该命令的语法类似于\newcommand，除了⟨*name*⟩ 可以包含一些数字或标点，但不以斜杠开头。可选参数⟨*arguments*⟩是一个整数用于指定宏需要处理的参数数量。如果⟨*optional*⟩ 给出，它指定了该宏的第一个参数的默认值，这第一个参数自动变成为可选参数。相比于\newcommand，当宏已经定义时，\newbibmacro 命令会给出一个警告信息，并自动转换为\renewbibmacro 命令。类似于\newcommand，该命令的常规形式在定义中使用\long 前缀，而带星的命令则没有。如果一个宏声明为 long，它的参数可以包含\par 记号。提供\newbibmacro 和\renewbibmacro 命令是为了方便使用，样式作者也可以使用\newcommand 或\def。然而，需要注意，共享文件 biblatex.def 中的绝大多数定义都是用\newbibmacro 定义的，因此，要使用和修改它们要用相应的方式处理。

\renewbibmacro{⟨*name*⟩}[⟨*arguments*⟩][⟨*optional*⟩]{⟨*definition*⟩}
\renewbibmacro*{⟨*name*⟩}[⟨*arguments*⟩][⟨*optional*⟩]{⟨*definition*⟩}

类似于\newbibmacro，但用于重定义⟨*name*⟩。相比于\newcommand，当宏未定义时，\renewbibmacro 命令给出一个警告信息，并自动转换为\newbibmacro 命令。

\providebibmacro{⟨*name*⟩}[⟨*arguments*⟩][⟨*optional*⟩]{⟨*definition*⟩}
\providebibmacro*{⟨*name*⟩}[⟨*arguments*⟩][⟨*optional*⟩]{⟨*definition*⟩}

类似于\newbibmacro，但仅在⟨*name*⟩ 未定义时定义宏。该命令概念上类似于\providecommand。

\usebibmacro{⟨*name*⟩}
\usebibmacro*{⟨*name*⟩}

该命令执行由\newbibmacro 定义的宏⟨*name*⟩。如果宏带参数，只要简单的跟在⟨*name*⟩ 后面即可。该命令的常规形式会处理 (净化，改造，sanitize)⟨*name*⟩，而带星的命令不会。

\savecommand{⟨*command*⟩}
\restorecommand{⟨*command*⟩}

这两个命令用来保存和恢复⟨*command*⟩，其中⟨*command*⟩ 必须是以斜杠开头的命令。两个命令都在局部范围内起作用。它们主要用于本地化文件中。

\savebibmacro{⟨*name*⟩}
\restorebibmacro{⟨*name*⟩}

这两个命令用来保存和恢复宏⟨*name*⟩，其中⟨*name*⟩ 由\newbibmacro 定义的宏的标识。两个命令都在局部范围内起作用。它们主要用于本地化文件中。

\savefieldformat[⟨*entry type*⟩]{⟨*format*⟩}
\restorefieldformat[⟨*entry type*⟩]{⟨*format*⟩}

这两个命令用来保存和恢复格式化指令⟨*format*⟩，其中⟨*format*⟩ 由\DeclareFieldFormat 定义。两个命令都在局部范围内起作用。它们主要用于本地化文件中。

\savelistformat[⟨*entry type*⟩]{⟨*format*⟩}
\restorelistformat[⟨*entry type*⟩]{⟨*format*⟩}

这两个命令用来保存和恢复格式化指令⟨*format*⟩，其中⟨*format*⟩ 由\DeclareListFormat 定义。两个命令都在局部范围内起作用。它们主要用于本地化文件中。

\savenameformat[⟨*entry type*⟩]{⟨*format*⟩}
\restorenameformat[⟨*entry type*⟩]{⟨*format*⟩}

这两个命令用来保存和恢复格式化指令⟨*format*⟩，其中⟨*format*⟩ 由\DeclareNameFormat 定义。两个命令都在局部范围内起作用。它们主要用于本地化文件中。

\ifbibmacroundef{⟨*name*⟩}{⟨*true*⟩}{⟨*false*⟩}

如果参考文献宏⟨*name*⟩ 未定义，展开为⟨*true*⟩ 否则为⟨*false*⟩。

\iffieldformatundef[⟨*entry type*⟩]{⟨*name*⟩}{⟨*true*⟩}{⟨*false*⟩}
\iflistformatundef[⟨*entry type*⟩]{⟨*name*⟩}{⟨*true*⟩}{⟨*false*⟩}
\ifnameformatundef[⟨*entry type*⟩]{⟨*name*⟩}{⟨*true*⟩}{⟨*false*⟩}

如果参考文献格式化指令⟨*format*⟩ 未定义，展开为⟨*true*⟩ 否则为⟨*false*⟩。

\usedriver{⟨*code*⟩}{⟨*entrytype*⟩}

执行⟨*entrytype*⟩ 类条目的参考文献驱动。在由\DeclareCiteCommand 定义的标注命令的⟨*loopcode*⟩ 中调用该命令是打印类似于一个参考文献条目的完整标注的简单方法。诸如\newblock 等命令无法用于标注，自动省略。附加的初始化命令可以通过⟨*code*⟩ 参数传递。该参数在一个编组内执行，这一编组用于运行相应驱动。注意: 该参数语法上是必须的，但可以留空。也要注意如果autolang 包选项打开的话，该命令会自动切换语言。

\bibhypertarget{⟨*name*⟩}{⟨*text*⟩}

hyperref 的\hypertarget 命令的封套 (包围命令)。⟨*name*⟩ 是超链接锚的名字，⟨*text*⟩ 的内容作为超链接锚，可以是任意可打印文字或代码。如果文档中存在refsection 环境，⟨*name*⟩ 是基于当前 refsection 环境。如果hyperref 包选项未打开或者hyperref 包未加载，该命令简单的传递⟨*text*⟩ 变量。另可参见 § 4.10.4 节的格式化指令 bibhypertarget。

\bibhyperlink{⟨*name*⟩}{⟨*text*⟩}

hyperref 的\hyperlink 命令的封套。⟨*name*⟩ 是由\bibhypertarget 定义的超链接锚的名字，⟨*text*⟩ 的内容将转变成超链接，可以是任意可打印文字或代码。如果文档中存在refsection 环境，⟨*name*⟩ 是基于当前 refsection 环境。如果hyperref 包选项未打开或者hyperref 包未加载，该命令简单的传递⟨*text*⟩ 变量。另可参见 § 4.10.4 节的格式化指令 bibhyperlink。

\bibhyperref[⟨*entrykey*⟩]{⟨*text*⟩}

将⟨*text*⟩ 转变为指向参考文献表中的⟨*entrykey*⟩(即某一条目) 的内部链接。如果⟨*entrykey*⟩ 省略，该命令使用当前正在处理的条目的引用关键词。该命令用于将标注转换为可点击的超链接，可以链接到参考文献表中的相应条目。链接目标由biblatex 自动标记。如果文档中有多个文献表，链接目标将是所有文献表中第一个出现的⟨*entrykey*⟩ 条目。如果文档中存在refsection 环境，则超链接基于当前 refsection 环境。另可参见 § 4.10.4 节的格式化指令 bibhyperref。

\ifhyperref{⟨*true*⟩}{⟨*false*⟩}

展开为⟨*true*⟩，如果hyperref 包选项已打开 (意味着hyperref 包已加载)，否则展开为⟨*false*⟩。

\docsvfield{⟨*field*⟩}

类似于etoolbox 包的\docsvlist 命令，差别在于它的参数是一个域名。域的值将以一个英文逗号分隔 (comma-separated) 的列表进行解析。如果⟨*field*⟩ 未定义，该命令展开为空字符串。

\forcsvfield{⟨*handler*⟩}{⟨*field*⟩}

类似于etoolbox 包的\forcsvlist 命令，差别在于它的参数是一个域名。域的值将以一个英文逗号分隔 (comma-separated) 的列表进行解析。如果⟨*field*⟩ 未定义，该命令展开为空字符串。

\MakeCapital{⟨*text*⟩}

类似于\MakeUppercase，但仅将⟨*text*⟩ 的第一个可打印字符转换为大写。注意: \MakeUppercase 命令的限制也适用于这一命令。即: ⟨*text*⟩ 中的所有命令必须是鲁棒的或者以\protect 为前缀，因为在大写操作中⟨*text*⟩ 需要展开。除了 Ascii 字符和标准重音命令外，该命令也处理inputenc 包的活动字符和babel 包的缩略词。如果⟨*text*⟩ 以一个控制序列开头，不做任何大写操作。该命令是鲁棒的。

\MakeSentenceCase{⟨*text*⟩}
\MakeSentenceCase*{⟨*text*⟩}

将⟨*text*⟩ 参数转换为 sentence case(句子模式)，即字符串中的第一个单词首字母大写而剩下其他部分转换为小写。该命令是鲁棒的。带星号的命令与常规命令 (不带星号) 的差别在于它能考虑条目的语言，根据langid 域指定。只有当langid 未定义或者值为由\DeclareCaseLangs 命令 (见后面) 声明的某种语言时，它才将⟨*text*⟩ 转换为句子模式。[63] 否则⟨*text*⟩ 不做任何改变。推荐使用\MakeSentenceCase* 而不是常规命令。两个命令都支持bib 文件的传统 BibTeX 规范，即: 遇到任何以花括号包围的内容大小写都不作变化，例如:

```
\MakeSentenceCase{an Introduction to LaTeX}
\MakeSentenceCase{an Introduction to {LaTeX}}
```

将得到:

```
An introduction to latex
An introduction to LaTeX
```

在以传统 BibTeX 方式设计的bib 文件中，为阻止字母的大小写变化 (case-changing)，将单个字母用花括号包围是一种相当常见的方法。

```
title = {An Introduction to {L}a{T}e{X}}
```

这种方式存在一个问题是括号会压缩被包围字母两侧的字距。最好的方式是如第一个示例所示的那样，将整个单词都包围起来。

\mkpageprefix[⟨*pagination*⟩][⟨*postpro*⟩]{⟨*text*⟩}

该命令用于域格式化指令中，对标注命令的⟨*postnote*⟩ 参数和文献条目的pages 域进行格式化。默认情况下，它将会解析⟨*text*⟩ 参数，并且以'p.' or 'pp.' 做为前缀。可选参数⟨*pagination*⟩ 保存指示 pagination 类型的域名，可以是pagination 或bookpagination，默认是pagination。前缀与⟨*text*⟩ 之间的间距可以通过重定义\ppspace 命令来调整。默认是一个不可断行的词内空格。详见 §§ 2.3.10 和 3.13.3。另可参见\DeclareNumChars, \DeclareRangeChars, \DeclareRangeCommands,

[63] 默认情况下，如下语言支持转换: american, british, canadian, english, australian, newzealand as well as the aliases USenglish and UKenglish. 要扩展或修改该列表请使用\DeclareCaseLangs 命令。

Input	Output		
	mincomprange=10	mincomprange=100	mincomprange=1000
11–15	11–5	11–15	11–15
111–115	111–5	111–5	111–115
1111–1115	1111–5	1111–5	1111–5
	maxcomprange=1000	maxcomprange=100	maxcomprange=10
1111–1115	1111–5	1111–5	1111–5
1111–1155	1111–55	1111–55	1111–1155
1111–1555	1111–555	1111–1555	1111–1555
	mincompwidth=1	mincompwidth=10	mincompwidth=100
1111–1115	1111–5	1111–15	1111–115
1111–1155	1111–55	1111–55	1111–155
1111–1555	1111–555	1111–555	1111–555

表 13: \mkcomprange 设置

和\NumCheckSetup。可选参数⟨*postpro*⟩ 指定了用于对⟨*text*⟩ 后处理的宏。如果只给出一个可选参数，将作为⟨*pagination*⟩，下面是两个典型示例:

```
\DeclareFieldFormat{postnote}{\mkpageprefix[pagination]{#1}}
\DeclareFieldFormat{pages}{\mkpageprefix[bookpagination]{#1}}
```

第一个示例中的可选参数pagination 可以省略。

\mkpagetotal[⟨*pagination*⟩][⟨*postpro*⟩]{⟨*text*⟩}

该命令类似于\mkpageprefix，差别在于它用于条目的pagetotal 域，即它将打印“123 pages”而不是“page 123”。可选参数⟨*pagination*⟩ 默认是bookpagination。在⟨*text*⟩ 和后缀之间的间距可由对\ppspace 重定义进行调整。可选参数⟨*postpro*⟩ 指定了用于对⟨*text*⟩ 后处理的宏。如果只给出一个可选参数，将作为⟨*pagination*⟩，下面是一个典型示例:

```
\DeclareFieldFormat{pagetotal}{\mkpagetotal[bookpagination]{#1}}
```

在本例中可选参数bookpagination 可省略。

\mkcomprange[⟨*postpro*⟩]{⟨*text*⟩}
\mkcomprange*[⟨*postpro*⟩]{⟨*text*⟩}

该命令用于域格式化指令，将⟨*text*⟩ 参数解析为页码范围并且压缩这些范围。例如，“125–129”可能格式化为“125–9”。可以通过调整 LaTeX 计数器mincomprange, maxcomprange 和 mincompwidth 来设置\mkcomprange 的操作方式，如表13 所示。默认的设置分别是 10, 100000, and 1。这意味着该命令默认是尽可能的压缩。使用\setcounter 来调整参数。扫描程序将\bibrangedash 和 hyphens 作为范围间隔符。它通过将使用\bibrangedash 的任意数量连续连字符替换破折号 (dash) 实现

正规化。支持以\bibrangessep 分隔的多范围列表。后端会将逗号或冒号 (commas/semicolon) 的多范围分隔符转换为\bibrangessep。如果因为某些原因需要对 list/range 扫描程序隐藏一个字符，那么可以将该字符或者整个字符串用花括号包围起来。可选参数⟨*postpro*⟩ 指定了一个用于对⟨*text*⟩ 进行后处理的宏。这对于需要将\mkcomprange 和其它也要解析它们自身的⟨*text*⟩ 参数的格式化宏 (比如\mkpageprefix) 联合使用非常重要。简单的嵌套可能无法如期正常工作。使用⟨*postpro*⟩ 参数构建的处理链，如下:

```
\DeclareFieldFormat{postnote}{\mkcomprange[{\mkpageprefix[pagination]}
    ↪ ]{#1}}
```

注意:\mkcomprange 命令首先处理，\mkpageprefix 则作为后处理器。也要注意⟨*postpro*⟩ 被额外的一对花括号包围。这仅在特殊情况下需要，为阻止 LaTeX 的可选参数扫描器与嵌套的方括号混淆。带星的命令与不带星命令的差别是它的⟨*postpro*⟩ 参数应用于列表中的各项，例如:

```
\mkcomprange[\mkpageprefix]{5, 123-129, 423-439}
\mkcomprange*[\mkpageprefix]{5, 123-129, 423-439}
```

将输出:

```
pp. 5, 123-9, 423-39
p. 5, pp. 123-9, pp. 423-39
```

\mkfirstpage[⟨*postpro*⟩]{⟨*text*⟩}
\mkfirstpage*[⟨*postpro*⟩]{⟨*text*⟩}

该命令用于域格式化指令，将⟨*text*⟩ 参数解析为页码范围并且仅打印这些范围的起始页码。扫描程序将\bibrangedash 和 hyphens 作为范围间隔符。支持范围列表以\bibrangessep 分隔。如果因为某些原因需要对 list/range 扫描程序隐藏一个字符，那么可以将该字符或者整个字符串用花括号包围起来。可选参数⟨*postpro*⟩ 指定了一个用于对⟨*text*⟩ 进行后处理的宏。怎么使用该参数见\mkcomprange 命令。带星的命令的差别在于⟨*postpro*⟩ 参数应用于列表的各项。例如:

```
\mkfirstpage[\mkpageprefix]{5, 123-129, 423-439}
\mkfirstpage*[\mkpageprefix]{5, 123-129, 423-439}
```

将输出:

```
p. 5, 123, 423
p. 5, p. 123, p. 423
```

\rangelen{⟨*rangefield*⟩}该命令将其参数解析为一个范围，并返回范围的长度。计算由biber负责可以处理很多特殊情况。对于无终点的 (开口的) 范围将返回-1。特别的，\rangelen 可以:

- 计算同一个域中多个范围比如‘1-10, 20-30’ 的总数
- 处理隐含的范围比如‘22-4’ and ‘130-33’
- 处理大小写的罗马数字范围，处理 ASCII 码和统一码表示的罗马数字范围。

下面是一些示例:

pages = ‘10’	\rangelen{pages} returns ’1’
pages = ‘10-15’	\rangelen{pages} returns ’6’
pages = ‘10-15,47-53’	\rangelen{pages} returns ’13’
pages = ‘10-’	\rangelen{pages} returns ’-1’
pages = ‘-10’	\rangelen{pages} returns ’-1’
pages = ‘48-9’	\rangelen{pages} returns ’2’
pages = ‘172-77’	\rangelen{pages} returns ’6’
pages = ‘i-vi’	\rangelen{pages} returns ’6’
pages = ‘X-XX’	\rangelen{pages} returns ’11’
pages = ‘VII-xii’	\rangelen{pages} returns ’6’
pages = ‘VII-xii, 145-7, 135-39’	\rangelen{pages} returns ’14’

\rangelen 命令可以用于判断中:

```
\ifnumcomp{\rangelen{pages}}{=}{1}{add ’f’}{do nothing}
```

\DeclareNumChars{⟨*characters*⟩}
\DeclareNumChars*{⟨*characters*⟩}

该命令设置 § 4.6.2 节的\ifnumeral, \ifnumerals 和\ifpages 命令。该设置也将影响\iffieldnum, \iffieldnums, \iffieldpages, \mkpageprefix 和 \mkpagetotal 命令。⟨*characters*⟩ 参数是一个无分隔符的符号列表，将作为数字的一部分进行处理。不带星命令将替换当前设置，带星命令则将其参数附加到当前列表中。默认设置为:

```
\DeclareNumChars{.}
```

这意味着 (节或者其他) 数值比如 ‘3.4.5’ 将被认为是一个数字。注意，默认检测的是阿拉伯和罗马数字，没有必要对此做显式声明。

\DeclareRangeChars{⟨*characters*⟩}
\DeclareRangeChars*{⟨*characters*⟩}

该命令设置 § 4.6.2 的\ifnumerals 和\ifpages 命令。其设置还将影响\iffieldnums，\iffieldpages，\mkpageprefix 和\mkpagetotal。⟨*characters*⟩ 参数是一个无分隔符的符号列表，将作为范围指示符进行处理。不带星命令将替换当前设置，带星命令则将其参数附加到当前列表中。默认设置为:

```
\DeclareRangeChars{~,;-+/}
```

这意味着比如‘3–5’, ‘35+’, ‘8/9’ 等字符串会被\ifnumerals 和\ifpages 认为是一个范围。这些字符串中的非范围字符将被认为是数字。因此，类似于‘3a–5a’ 和‘35b+’之类的字符串默认情况下不被认为是范围。更多细节详见 §§ 2.3.10 和 3.13.3。

\DeclareRangeCommands{⟨*commands*⟩}

\DeclareRangeCommands*{⟨*commands*⟩}

该命令类似于\DeclareRangeChars，差别在于⟨*commands*⟩ 参数是一个无分隔符的命令列表，将被视为范围指示符。不带星命令将替换当前设置，带星命令则将其参数附加到当前列表中。默认列表相当长，应该覆盖所有一般情况。下面是一个简单示例:

```
\DeclareRangeCommands{\&\bibrangedash\textendash\textemdash\psq\psqq}
```

更多细节参见 §§ 2.3.10 和 3.13.3。

\DeclarePageCommands{⟨*commands*⟩}

\DeclarePageCommands*{⟨*commands*⟩}

该命令类似于\DeclareRangeCommands，差别在于它仅影响\ifpages 和\iffieldpages 判断，而不影响\ifnumerals 和\iffieldnums。默认设置为:

```
\DeclarePageCommands{\pno\ppno}
```

\NumCheckSetup{⟨*code*⟩}

该命令用于临时重定义一些将与 § 4.6.2 节的\ifnumeral, \ifnumerals, \ifpages 命令执行的判断产生冲突的命令。该设置也将影响\iffieldnum, \iffieldnums, \iffieldpages, \mkpageprefix 和\mkpagetotal。这些命令将在组内执行⟨*code*⟩。因为上述命令将展开为字符串用于分析，可以利用将冲突命令展开为空字符串 (将被判断命令忽略) 的方式来移除这些命令。更多细节参见 §§ 2.3.10 和 3.13.3。

\DeclareCaseLangs{⟨*languages*⟩}

\DeclareCaseLangs*{⟨*languages*⟩}

定义语言列表，该列表在\MakeSentenceCase* 命令将一个字符串转换成句子模式时考虑。⟨*languages*⟩ 参数是一个由babel/polyglossia 语言标识构成的逗号分隔的列表。不带星命令用于替换当前设置，而带星的命令用于附加当前列表。默认的设置为:

```
\DeclareCaseLangs{%
  american,british,canadian,english,australian,newzealand,USenglish,
    ↪ UKenglish}
```

语言标识的列表见babel/polyglossia 手册和表2。

\BibliographyWarning{⟨*message*⟩}

该命令类似于\PackageWarning，但打印内容除了输入行号外还有当前处理条目的引用关键词。如果⟨*message*⟩ 相当长，可以使用\MessageBreak 命令来断行。注意: 标准的\PackageWarning 命令在参考文献中使用时无法提供一个有意义的提示，因为其打印的输入行号只是\printbibliography 命令所在的行号。

\pagetrackertrue
\pagetrackerfalse

这些命令将打开或关闭局部引用追踪器 (这将影响来自 § 4.6.2 节的\iffirstonpage 和\ifsamepage 判断)。可在标注命令定义或者正文中的任意位置使用。要使标注命令完全排除页码追踪，可以在\DeclareCiteCommand 命令的⟨*precode*⟩ 参数中使用\pagetrackerfalse。详见 § 4.3.1。注意: 当全局页码追踪关闭时，这些命令无效。

\citetrackertrue
\citetrackerfalse

这些命令将打开或关闭所有的局部引用追踪器 (这将影响来自 § 4.6.2 节的 \ifciteseen, \ifentryseen, \ifciteibid, 和\ifciteidem 判断)。可在标注命令定义或者正文中的任意位置使用。要使标注命令完全排除页码追踪，可以在\DeclareCiteCommand 命令的⟨*precode*⟩ 参数中使用\citetrackerfalse。详见 § 4.3.1。注意: 当全局追踪关闭时，这些命令无效。

\backtrackertrue
\backtrackerfalse

这些命令将打开或关闭所有的局部 backref 追踪器。可在标注命令定义或者正文中的任意位置使用。要使标注命令完全排除反向链接追踪，可以在\DeclareCiteCommand 命令的⟨*precode*⟩ 参数中使用\backtrackerfalse。注意: 当 backref 选项未进行全局设置，这些命令无效。

4.7 标点和间距

biblatex 宏包提供了设计用来在文献著录表和标注中管理和追踪标点与空格的精致工具。这些工具在两个层面工作。§ 4.7.1 节讨论的高层 (high-level) 命令处理由著录样式在一个参考文献条目的不同部分插入的标点和空格。而 §§ 4.7.2、4.7.3、4.7.4 节中的命令在更低一层工作。它们以一种健壮高效的方式使用 TeX 的间距因子 并间距因子修改代码来追踪标点。采用这种方法，不仅可以检测在域间显式插入的标点符号，也可以检测在域内末尾的标点符号。相同的技术也用在本地化字符串的自动大写处理过程中，详见 § 4.7.5 节的\DeclareCapitalPunctuation 命令以及 § 4.8 节。注意: 这些工具仅在标注和文献著录表内局部使用。它们不会影响正文中其他任何部分。

4.7.1 块和单元的标点

参考文献条目的主要组成部分是块 (‘blocks’) 和单元 (‘units’)。其中块更大，单元相对较小，至多长度上与块相等。例如，诸如title 或note 等域的值常构成一个单元，并且用一个句号或逗号与后面的数据分隔开来。一个块可以由多个域构成，这些域被认为是独立的单元，例如publisher, location 和year。一个条目如何划分成块和单元完全是由著录样式决定的。条目内容的划分通过在合适位置插入\newblock 和\newunit 命令，在末尾插入\finentry 命令 (见 § 4.2.3 的示例) 来实现。一些实用提示见 § 4.11.7 节。

\newblock 标记一个块的终点。该命令不打印任何内容，仅标记块的结束。块的分隔符\newblockpunct 将由后面跟着的\printtext, \printfield, \printlist, \printnames, 或\bibstring 命令插入。可以在合适的位置使用 \newblock 而不需要担心产生多余的块。一个新的块仅在其后是\printfield(或类似) 命令且打印一些内容的情况下开始。详见 § 4.11.7。

\newunit 记录一个单元的终点，并将默认分隔符 \newunitpunct 放入标点缓存中。该命令不打印任何内容，仅标记单元的结束。标点缓存将由接下来的\printtext, \printfield, \printlist, \printnames, 或\bibstring 命令插入。可以在类似\printfield 的命令后面使用\newunit，而不需要担心带来多余的标点和空格。标点缓存只能由后面的\printfield 或者类似命令且打印的域非空时插入。这一方式也应用于\printtext, \printlist, \printnames 和\bibstring 命令。详见 § 4.11.7 节。

\finentry 插入 \finentrypunct。该命令应在每个参考文献条目的最末尾使用。

\setunit{⟨*punctuation*⟩}
\setunit*{⟨*punctuation*⟩}

\setunit 类似于\newunit，除了它使用⟨*punctuation*⟩ 代替\newunitpunct。带星号命令与不带星号命令的差别在于它检查前面最近的\printtext, \printfield, \printlist, \printnames 或 \bibstring 命令是否实际打印内容。如果没有打印，该命令不做任何处理。

\printunit{⟨*punctuation*⟩}
\printunit*{⟨*punctuation*⟩}

\printunit 命令类似于\setunit，但⟨*punctuation*⟩ 继续存留在缓存中。这能确保⟨*punctuation*⟩ 能在由\printtext, \printfield, \printlist, \printnames, or \bibstring 命令打印下一个非空域前插入—不管任何立即调用的\newunit 或\setunit 命令。(译者: 即该命令设置的标点只要后面还有非空域需要打印都会插入。)

\setpunctfont{⟨*command*⟩}

该命令用于域格式化指令中,提供了处理打印为另一种字体(比如斜体打印的标题)的域后面的单元标点的替换方式。标准的 LaTeX 处理方式是在后面添加一个小的空格，称为斜体修正。该命令允许将标点调整为之前的域采用的字体。⟨*command*⟩ 应是带一个参数的 text 字体命令，比如\emph 或\textbf。该命令仅影响由 § 4.7.3 节的命令插入的标点符号。字体调整仅应用于接下来的标点符号,处理之后将自动重设。如果希望在它生效之前手动重设,使用\resetpunctfont。如果punctfont 选项关闭,该命令不做任何处理。注意 § 4.10.4 节的\mkbibemph, \mkbibitalic 和\mkbibbold 封套命令自动包含该功能。

\resetpunctfont 该命令在其生效之前重设由\setpunctfont 定义的单元标点字体。如果punctfont 选项关闭，该命令不做任何处理。

4.7.2 标点判断

下面的命令可以用来在标注和文献表的任意位置判断前面的标点符号。

`\ifpunct{`⟨*true*⟩`}{`⟨*false*⟩`}`

如果前面的是除缩略点外的标点符号，执行⟨*true*⟩，否则执行⟨*false*⟩。

`\ifterm{`⟨*true*⟩`}{`⟨*false*⟩`}`

如果前面的是一个终结标点 (terminal punctuation mark)，执行⟨*true*⟩，否则执行⟨*false*⟩。终结标点是设置用来自动大写的任意标点符号，可以使用默认符号或者用`\DeclareCapitalPunctuation` 设定，详见 § 4.7.5。默认情况下，用句号，叹号和问号。

`\ifpunctmark{`⟨*character*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

如果前面的是一个⟨*character*⟩ 标点符，执行⟨*true*⟩，否则执行⟨*false*⟩。⟨*character*⟩ 可以是逗号，分号，冒号，句号，叹号，问号或星号。注意，一个句号代表一个结束句子的句号。使用星号用来判断缩略词后的点号。如果该命令用于姓名列表的格式化指令，即在`\DeclareNameFormat` 的参数中，⟨*character*⟩ 也可以是撇号。

`\ifprefchar{`⟨*true*⟩`}{`⟨*false*⟩`}`

如果前面的是一个由`\DeclarePrefChars` 声明的前缀符，执行⟨*true*⟩，否则执行⟨*false*⟩。

4.7.3 添加标点

下面的命令设计用来防止重复标点。参考文献著录和标注样式总需要使用这些命令来代替原样输出的标点符号。本节中所有的`\add...` 命令自动利用`\unspace` 命令移除前面的空白 (见 § 4.7.4)。注意: 下面讨论的所有的`\add...` 命令的作用是宏包默认的，无论`biblatex` 换哪种语言都会重新恢复。其作用可以通过`\DeclarePunctuationPairs` 命令进行调整，见 § 4.7.5 节。

`\adddot` 除非前面输出的是任何一种标点符号，否则添加一个句点 (period)。该命令的目的是在一个缩写后面插入点 (dot)。以这种方式插入的点被认为与其它标点命令插入的标点性质相同。该命令也用来将前面插入的文本句点 (原样输出的句点，literal period) 转换成一个缩写的点。

`\addcomma` 除非前面输出的是一个逗号 (comma)、分号 (semicolon)、冒号 (colon) 和句点 (period)，否则添加一个逗号。

`\addsemicolon` 除非前面输出的是一个逗号，另一个分号，冒号和句号，否则添加一个分号。

`\addcolon` 除非前面输出的是一个逗号，分号，另一个冒号，或句号，否则添加一个冒号。

`\addperiod` 除非前面输出的是一个缩写点或其它任意标点符号，否则添加一个句号。该命令也可以用来将前面插入的缩写点转换为句号，比如在句子的末尾 (译者: 这种转换应该是经过某种判断的，比如在末尾就转换，不再末尾的话缩写点则不转换)。

\addexclam 除非前面输出的是缩写点之外的任意标点，否则添加一个叹号。

\addquestion 除非前面输出的是缩写点之外的任意标点，否则添加一个问号。

\isdot 将前面输出的是句号转换为缩写点。与\adddot 相反，如果前面不是一个句号，则不添加任何符号。

\nopunct 添加一个内部标记，将导致接下来的标点命令不打印任何标点。

4.7.4 添加空格

下面的命令设计用来防止多余的空格。参考文献著录和标注样式总需要使用这些命令来代替原样输出的空格。与 §§ 4.7.2 和 4.7.3 节的命令不同，这些命令不仅限在标注和著录样式中使用，而是可以全局使用的。

\unspace 移除前面的空格，即消除来自当前水平列末尾的所有间距 (skip) 和阀值。下面的所有命令都隐含执行该命令。

\addspace 添加一个可断行的词内空格。

\addnbspace 添加一个不可断行 (non-breakable) 的词内空格。

\addthinspace 添加一个可断行的短空格 (thin space)。

\addnbthinspace 添加一个不可断行的短空格 (thin space)，类似于命令\, 和\thinspace。

\addlowpenspace 添加一个由lownamepenalty 计数器值作为阀值控制的空格，详见 §§ 3.10.4 和 4.10.3。

\addhighpenspace 添加一个由highnamepenalty 计数器值作为阀值控制的空格，详见 §§ 3.10.4 和 4.10.3。

\addlpthinspace 类似于\addlowpenspace，但添加的是可断行的短空格。

\addhpthinspace 类似于\addhighpenspace，但添加的是可断行的短空格。

\addabbrvspace 添加一个由abbrvpenalty 计数器值作为阀值控制的空格，详见 §§ 3.10.4 和 4.10.3。

\addabthinspace 类似于\addabbrvspace，但添加一个短空格。

\adddotspace 执行\adddot 并且添加一个由abbrvpenalty 计数器值作为阀值控制的空格，详见 §§ 3.10.4 和 4.10.3。

\addslash 添加一个可断行的斜杠，该命令与 latex 内核提供的\slash 命令不同，因为\slash 之后的断行完全没有阀值控制。

注意: 本节的命令隐式地执行\unspace 来去除多余的空格，因此它们可以互相覆盖。比如，可以使用\addnbspace 将前面插入的词内空格转换为一个不可断行的空格，而\addspace 可以将一个不可断行空格转换为可断行空格。[64]

[64]译者: 注意有的时候\unspace 看似能够起到作用，但其实并不能随意使用的。在 beamer 中 printtext 老是有些问题，可能是实现 printtext 命令的依赖命令，在 beamer 中重定义了，跟 aritcle 文档类中的情况差别很大。

4.7.5 配置标点和大写

下面的命令配置与标点和自动大写相关的各种功能。

\DeclarePrefChars{⟨*characters*⟩}

该命令声明，测试是否将在姓名前缀和姓之间插入\bibnamedelimc 时，需做特殊处理的字符。如果一个字符在⟨*characters*⟩ 中，那么\bibnamedelimc 不会插入。它用于一些前缀特殊的缩写名中比如'd'Argent'，其中撇号之后没有空格，默认设置为:

```
\DeclarePrefChars{'}
```

\DeclareAutoPunctuation{⟨*characters*⟩}

该命令定义当标注命令扫描标点时需要考虑的标点符号。注意，⟨*characters*⟩ 是一个不分隔字符列表。有效的⟨*characters*⟩ 包括句号、逗号、分号、冒号、叹号和问号。默认设置为:

```
\DeclareAutoPunctuation{.,;:!?}
```

当autopunct 包选项设置为 true 时，该定义将自动恢复。执行\DeclareAutoPunctuation{} 等价于设置 autopunct = false，即它会关闭该功能。

\DeclareCapitalPunctuation{⟨*characters*⟩}

当biblatex 插入本地化字符串时，即关键项如 'edition' 或'volume' 等，该命令将把终结标点后的字符自动大写。该命令定义的一些标点符号将导致一些本地化字符串大写，如果某个标点符号在一个字符串之前。注意⟨*characters*⟩ 是一个不分隔字符列表，有效的⟨*characters*⟩ 包括句号、逗号、分号、冒号、叹号和问号。默认设置为:

```
\DeclareCapitalPunctuation{.!?}
```

使用参数为空的\DeclareCapitalPunctuation 命令等价于关闭自动大写功能。因为该功能与语言相关，所以该命令必须用在\DefineBibliographyExtras 命令 (当用在导言区时) 或者\DeclareBibliographyExtras(当用在本地化模型中) 的参数中。详见 §§ 3.9 和 4.9。默认情况下，在句号、叹号和问号后面的本地化字符串将大写。所有段落开头的字符串一般都需要大写 (事实上无论 TeX 是否在垂直模式中)。

\DeclarePunctuationPairs{⟨*identifier*⟩}{⟨*characters*⟩}

使用该命令来声明有效的标点符号对。这将影响 § 4.7.3 节讨论的标点命令。例如，\addcomma 命令的解释是: 该命令添加一个逗号，除非前面的字符是另一个逗号，分号，冒号或句号。换句话说，在缩写点，叹号，问号之后加入逗号是允许的。这些标点有效对声明如下:

```
\DeclarePunctuationPairs{comma}{*!?}
```

⟨*identifier*⟩ 选择需要配置的命令。标识对应于 § 4.7.3 节标点命令的标点名称 (即去除\add 前缀后的命令名)，即有效的⟨*identifier*⟩ 字符串包括 dot, comma, semicolon, colon, period, exclam, question。⟨*characters*⟩ 是标点符号的不分隔列表。有效的⟨*characters*⟩ 是逗号，分号，冒号，句号，叹号，问号和星号。⟨*characters*⟩ 中的句号代表一个句末点号，星号代表缩写后的点号。下面是默认的设置，当biblatex 切换语言时总是自动回复默认设置，并且与 § 4.7.3 描述的行为对应:

```
\DeclarePunctuationPairs{dot}{}
\DeclarePunctuationPairs{comma}{*!?}
\DeclarePunctuationPairs{semicolon}{*!?}
\DeclarePunctuationPairs{colon}{*!?}
\DeclarePunctuationPairs{period}{}
\DeclarePunctuationPairs{exclam}{*}
\DeclarePunctuationPairs{question}{*}
```

因为该功能与语言相关，\DeclarePunctuationPairs 必须用在\DefineBibliographyExtras(当在导言区使用) 或\DeclareBibliographyExtras(当在本地化模块中使用) 的参数中。详见 §§ 3.9 和 4.9。注意: 一些本地化模块可能使用不同于宏包默认的设置。[65]

\DeclareQuotePunctuation{⟨*characters*⟩}

该命令控制'American-style' 标点。§ 4.10.4 节的\mkbibquote 封套可以与 §§ 4.7.1、4.7.3、4.7.4 节讨论的标点工具交互。\mkbibquote 之后的标点将移动到引号内，如果它们已在\DeclareQuotePunctuation 声明过。注意⟨*characters*⟩ 是一个字符的不分隔列表。有效的⟨*characters*⟩ 包括: 句号，逗号，分号，冒号，叹号和问号。下面是一个示例:

```
\DeclareQuotePunctuation{.,}
```

执行\DeclareQuotePunctuation{} 等价于关闭该功能。这是包的默认情形。因为该功能与语言相关，该命令必须用在\DefineBibliographyExtras(当在导言区使用) 或\DeclareBibliographyExtras(当在本地化模块中使用) 的参数中。详见 §§ 3.9 和 4.9。另可参见 § 3.11.1。

[65] 到本文档写作时，american 模块针对美语样式 ('American-style') 标点使用不同的设置。

\uspunctuation 使用底层命令\DeclareQuotePunctuation 和\DeclarePunctuationPairs 激活美语样式标点的缩略命令。详见 § 3.11.1。提供该缩略命令是为了使用方便。有效的设置由底层命令应用。

\stdpunctuation 取消由\uspunctuation 命令产生的设置，恢复到标准的标点样式。因为标准标点样式是默认设置，仅需要使用该命令来覆盖前面执行的\uspunctuation 命令即可。详见 § 3.11.1。

4.7.6 修正标点追踪

在一般情况下标点追踪和自动大写的工具是很可靠的，但总是存在一些少量情况可能需要手动干预。典型的问题包括当本地化字符串作为脚注 (就大写而言脚注常被认为是在一个段落的开始，但 TeX 在此时并不处于垂直模式中) 的第一个单词打印时，或者句号 (不是真正的句末点号，例如在一个像“[...]”的省略号之后诸如\addperiod 等命令将不会进行任何操作，因为圆括号和方括号对于标点追踪器来说是透明的) 之后的标点。[66] 在这些情况中，如果有需要，可在著录和标注样式中使用如下命令来标记句子的开头或者中间位置:

\bibsentence 该命令标记句子的开头。紧跟在其后面的一个本地化字符串将会大写并且标点追踪器也会重设，即该命令对标点追踪器隐藏前面所有的标点并且强制大写。

\midsentence 该命令标记句子的中间位置。紧跟在其后面的一个本地化字符串将不会大写并且标点追踪器也会重设，即该命令对标点追踪器隐藏前面所有的标点并且抑制大写。

\midsentence* \midsentence 命令的带星号版本，差别在于，对于该命令前面的缩写点不会隐藏，即\midsentence* 之后的任何代码都能看见前面的缩写点。而所有其他标点将对标点追踪器隐藏，大写也将被抑制。

4.8 本地化字符串

本地化字符串如‘edition’ 或‘volume’ 之类的关键词项将由biblatex 本地化模块自动转换。本地化字符串概述见 § 4.9，所有默认支持的字符串列表见 § 4.9.2 节。本节的命令用于打印本地化的项。

\bibstring[⟨*wrapper*⟩]{⟨*key*⟩}

打印本地化字符串⟨*key*⟩，其中⟨*key*⟩ 是一个以小写字母书写的标识 (见 § 4.9.2)。需要时字符串会大写，详见 § 4.7.5 节。根据 § 3.1.2.1 节的abbreviate 包选项，\bibstring 打印短字符串或长字符串。如果本地化字符串处于嵌套中，即\bibstring 用在另一个字符串中，它的作用类似于\bibxstring。如果给出⟨*wrapper*⟩ 参数，字符串将传递给⟨*wrapper*⟩ 用于格式化。这常用于字体命令比如\emph 等。

\biblstring[⟨*wrapper*⟩]{⟨*key*⟩}

类似于\bibstring，但总是打印长字符串，忽略abbreviate 选项。

[66] 译者: 重点注意括号对于标点追踪器是透明的，这个问题在样式设计时经常会碰到。

\bibsstring[⟨*wrapper*⟩]{⟨*key*⟩}

类似于\bibstring，但总是打印短字符串，忽略abbreviate 选项。

\bibcpstring[⟨*wrapper*⟩]{⟨*key*⟩}

类似于\bibstring，但字符串总是首字母大写

\bibcplstring[⟨*wrapper*⟩]{⟨*key*⟩}

类似于\biblstring，但字符串总是首字母大写

\bibcpsstring[⟨*wrapper*⟩]{⟨*key*⟩}

类似于\bibsstring，但字符串总是首字母大写

\bibucstring[⟨*wrapper*⟩]{⟨*key*⟩}

类似于\bibstring，但字符串总是大写

\bibuclstring[⟨*wrapper*⟩]{⟨*key*⟩}

类似于\biblstring，但字符串总是大写

\bibucsstring[⟨*wrapper*⟩]{⟨*key*⟩}

类似于\bibsstring，但字符串总是大写

\biblcstring[⟨*wrapper*⟩]{⟨*key*⟩}

类似于\bibstring，但字符串总是小写

\biblclstring[⟨*wrapper*⟩]{⟨*key*⟩}

类似于\biblstring，但字符串总是小写

\biblcsstring[⟨*wrapper*⟩]{⟨*key*⟩}

类似于\bibsstring，但字符串总是小写

\bibxstring{⟨*key*⟩}

\bibstring 命令一个简化的但可展开的版本。注意，这一命令不自动首字母大写，也不与标点追踪器交互。用于当字符串嵌套或者一个判断中需要一个可展开的本地化字符串等特殊情况。

\bibxlstring[⟨*wrapper*⟩]{⟨*key*⟩}

类似于\bibxstring，但总是打印长字符串，忽略abbreviate 选项。

\bibxsstring[⟨*wrapper*⟩]{⟨*key*⟩}

类似于\bibxstring，但总是打印短字符串，忽略abbreviate 选项。

`\mainlang`

从当前语言转换到主文档语言。可以用在上述本地化字符串命令的⟨*wrapper*⟩参数中。

4.9 本地化模块

本地化模块提供了诸如'edition'或'volume'等关键项，或者一些具体语言相关功能定义比如日期格式和序数等的转换 (翻译)。这些定义在后缀为`lbx`的文件中给出。文件的基本名称必须是`babel/polyglossia`包已定义的语言名称。`lbx`文件也用于将`babel/polyglossia`语言名映射为`biblatex`包的后端模块。所有的本地化模型根据需要在正文中加载。注意文件的内容在一个编组内处理，`@`字符的类别码临时设置为'letter'。

4.9.1 本地化命令

用户层 (user-level) 的本地化命令已经在 § 3.9 节介绍过了。然而，用在`lbx`文件中的本地化命令的语法与用在导言区和配置文件中的语法略有不同。当在本地化文件中使用，不需要指定⟨*language*⟩，因为字符串要映射的语言已经由`lbx`文件名给出。

`\DeclareBibliographyStrings{`⟨*definitions*⟩`}`

该命令仅在`lbx`文件中提供。用于定义本地化字符串。⟨*definitions*⟩由⟨*key*⟩=⟨*value*⟩对构成，用于给一个标识赋予一个表达式。默认支持的完整关键词 (键) 列表在 § 4.9.2 节给出。注意:`lbx`文件中值的语法是不同的。[67] 赋予的键值由两个短语构成，每个部分都包含在一个括号中。下面是一个示例:

```
\DeclareBibliographyStrings{%
  bibliography  = {{Bibliography}{Bibliography}},
  shorthands    = {{List of Abbreviations}{Abbreviations}},
  editor        = {{editor}{ed.}},
  editors       = {{editors}{eds.}},
}
```

第一个值是用于写出的长短语，第二个是缩写或简易形式。两个字符串都必须给出即便它们是相同的，比如当一个短语总是要 (或不要) 缩写时。根据`abbreviate`包选项设置 (见 § 3.1.2.1)，当加载`abbreviate`时`biblatex`选择一个短语。也存在一个名为 `inherit` 特殊键，从另一种语言中拷贝字符串。这用于仅有部分短语不同的语言间，比如德国和奥地利，或者美国和英国英语。例如，下面给出的奥地利语完整定义:

```
\DeclareBibliographyStrings{%
```

[67] 译者: 重点注意定义本地化字符串的语法在 lbx 文件和样式文件中是不同的，这在设计样式文件是会碰到。

```
  inherit       = {german},
  january       = {{J\"anner}{J\"an.}},
}
```

这里示例略微简化了。真实的本地化文件应该使用 §§ 4.7.3 和 3.10 节讨论的标点和格式化命令而不是文本标点 (literal punctuation)[68]。下面的内容摘录自一个真实的本地化文件:

```
  bibliography     = {{Bibliography}{Bibliography}},
  shorthands       = {{List of Abbreviations}{Abbreviations}},
  editor           = {{editor}{ed\adddot}},
  editors          = {{editors}{eds\adddot}},
  byeditor         = {{edited by}{ed\adddotspace by}},
  mathesis         = {{Master's thesis}{MA\addabbrvspace thesis}},
```

注意上例中缩略点的处理，缩略短语中的空格以及首字母大写等。所有的短语在一个句子中使用时，都是首字母大写的。biblatex 包将自动将句首的第一个单词的首字母大写，详见 § 4.7.5 节的\DeclareCapitalPunctuation。用作标题的短语是特殊的。它们常使用一种适合标题的方式并且不应使用缩略形式 (但它们可能具有简易形式)。

\InheritBibliographyStrings{⟨*language*⟩}

该命令仅在lbx 文件中提供。它将⟨*language*⟩ 语言的本地化字符串复制到lbx 文件名指出的当前语言中。

\DeclareBibliographyExtras{⟨*code*⟩}

该命令仅在lbx 文件中提供。用于调整语言相关的功能比如日期格式和序号。任意的 LaTeX 代码⟨*code*⟩，常由 § 4.10.2 节的格式化命令的重定义构成。

\UndeclareBibliographyExtras{⟨*code*⟩}

该命令仅在lbx 文件中提供。用于恢复由\DeclareBibliographyExtras 命令修改的格式化命令。如果一个重定义命令包含在 § 4.10.2 中，则没有必要恢复前一定义，因为这些命令总是会根据所有语言模块进行本地化。

\InheritBibliographyExtras{⟨*language*⟩}

该命令仅在lbx 文件中提供。它将⟨*language*⟩ 语言的参考文献附加规则复制到lbx 文件名指出的当前语言中。

\DeclareHyphenationExceptions{⟨*text*⟩}

该命令对应于 § 3.9 节的\DefineHyphenationExceptions。差别在于它仅在lbx 文件中提供，并且没有⟨*language*⟩ 参数。连字符例外规则将影响正在处理的lbx 文件的语言。

[68]译者: 在样式文件中使用文本标点时可能会遇到一些不可预料的问题，应尽可能的使用 biblatex 提供的标点命令

\DeclareRedundantLanguages{⟨*language, language, ...*⟩}{⟨*langid, langid, ...*⟩}

该命令提供了 § 3.1.2.1 节clearlang 语言选项要求的语言映射。⟨*language*⟩是language 域给出的字符串 (不需要可选的 lang 前缀)，⟨*langid*⟩是babel/polyglossia 的语言标识，在加载babel 包的\usepackage 命令可选参数或者使用polyglossia 包的\setdefaultlanguage 或\setotherlanguages 命令中给出。这一命令可以用于lbx 文件中或者文档导言区中。下面是一些示例:

```
\DeclareRedundantLanguages{french}{french}
\DeclareRedundantLanguages{german}{german,ngerman,austrian,naustrian,
       nswissgerman,swissgerman}
\DeclareRedundantLanguages{english,american}{english,american,british,
       canadian,australian,newzealand,USenglish,UKenglish}
```

注意这一功能需要全局地打开 § 3.1.2.1 节的clearlang 选项。如果关闭该选项，所有的映射将被忽略。如果⟨*langid*⟩参数为空，biblatex 将清除相应语言的映射，即仅关闭该⟨*language*⟩的功能。

\DeclareLanguageMapping{⟨*language*⟩}{⟨*file*⟩}

该命令将babel/polyglossia 语言标识映射到一个 lbx 文件中。⟨*language*⟩必须是一个babel/polyglossia 已经定义的语言，即表2 中列出的标识之一。⟨*file*⟩参数是选择的lbx 文件的出.lbx 后缀外的文件名。可以多次声明相同的映射。后面的声明将覆盖前面的声明。该命令只能用在导言中。详见 § 4.11.8。

\NewBibliographyString{⟨*key*⟩}

该命令可以用在导言区 (包括cbx 和bbx 文件) 或者lbx 文件中，用于声明新的本地化字符串，即它初始化一个关键词 (键) 可以用于 \DefineBibliographyStrings 或 \DeclareBibliographyStrings 命令的⟨*definitions*⟩中。⟨*key*⟩选项可以是一个逗号分隔的列表。当在lbx 中使用时，⟨*key*⟩只完成lbx 文件名指定的语言的初始化。默认的键在 § 4.9.2 节列出。

4.9.2 本地化关键词 (键)

本节中的本地化关键词是默认定义的，由biblatex 附带的本地化文件提供。注意这些字符串仅在标注、著录表和文献列表中使用。所有的短语当在句中使用时常需要首字母大写。而biblatex 会自动将句首的字符串首字母大写。唯一的例外规则是三个用于标题中的字符串。

4.9.2.1 标题 下面的字符串比较特殊是因为他们用在标题中，并通过宏可以全局使用。因此，通常他们在标题中使用时需要首字母大写，并且不能包含任何作为biblatex 作者接口的本地命令。

bibliography 词 (术语)'bibliography'，也作\bibname 使用。

references 词'references'，也作\refname 使用。

shorthands 词‘list of shorthands’ 或‘list of abbreviations’，也作`\biblistname` 使用。

4.9.2.2 角色，解释为职业

下面的关键词指出的角色，可以解释为职业 (‘editor’, ‘translator’) 而不是行动 (‘edited by’, ‘translated by’)。

editor The term ‘editor’, referring to the main editor. This is the most generic editorial role.

editors The plural form of `editor`.

compiler The term ‘compiler’, referring to an editor whose task is to compile a work.

compilers The plural form of `compiler`.

founder The term ‘founder’, referring to a founding editor.

founders The plural form of `founder`.

continuator An expression like ‘continuator’, ‘continuation’, or ‘continued’, referring to a past editor who continued the work of the founding editor but was subsequently replaced by the current editor.

continuators The plural form of `continuator`.

redactor The term ‘redactor’, referring to a secondary editor.

redactors The plural form of `redactor`.

reviser The term ‘reviser’, referring to a secondary editor.

revisers The plural form of `reviser`.

collaborator A term like ‘collaborator’, ‘collaboration’, ‘cooperator’, or ‘cooperation’, referring to a secondary editor.

collaborators The plural form of `collaborator`.

translator The term ‘translator’.

translators The plural form of `translator`.

commentator The term ‘commentator’, referring to the author of a commentary to a work.

commentators The plural form of `commentators`.

annotator The term ‘annotator’, referring to the author of annotations to a work.

annotators The plural form of `annotators`.

4.9.2.3 合并的编辑角色，解释为职业

下面的这些关键词类似于 `editor`, `translator` 等的作用。它们常用来说明编辑的附加角色，比如‘editor and translator’, ‘editor and foreword’。

editortr Used if `editor`/`translator` are identical.

editorstr The plural form of `editortr`.

editorco Used if `editor`/`commentator` are identical.

editorsco The plural form of `editorco`.

editoran Used if `editor/annotator` are identical.

editorsan The plural form of `editoran`.

editorin Used if `editor/introduction` are identical.

editorsin The plural form of `editorin`.

editorfo Used if `editor/foreword` are identical.

editorsfo The plural form of `editorfo`.

editoraf Used if `editor/aftword` are identical.

editorsaf The plural form of `editoraf`.

Keys for `editor/translator/`⟨*role*⟩ combinations:

editortrco Used if `editor/translator/commentator` are identical.

editorstrco The plural form of `editortrco`.

editortran Used if `editor/translator/annotator` are identical.

editorstran The plural form of `editortran`.

editortrin Used if `editor/translator/introduction` are identical.

editorstrin The plural form of `editortrin`.

editortrfo Used if `editor/translator/foreword` are identical.

editorstrfo The plural form of `editortrfo`.

editortraf Used if `editor/translator/aftword` are identical.

editorstraf The plural form of `editortraf`.

Keys for `editor/commentator/`⟨*role*⟩ combinations:

editorcoin Used if `editor/commentator/introduction` are identical.

editorscoin The plural form of `editorcoin`.

editorcofo Used if `editor/commentator/foreword` are identical.

editorscofo The plural form of `editorcofo`.

editorcoaf Used if `editor/commentator/aftword` are identical.

editorscoaf The plural form of `editorcoaf`.

Keys for `editor/annotator/`⟨*role*⟩ combinations:

editoranin Used if `editor/annotator/introduction` are identical.

editorsanin The plural form of `editoranin`.

editoranfo Used if `editor/annotator/foreword` are identical.

editorsanfo The plural form of `editoranfo`.

editoranaf Used if `editor/annotator/aftword` are identical.

editorsanaf The plural form of `editoranaf`.

Keys for `editor/translator/commentator/`⟨*role*⟩ combinations:

editortrcoin Used if `editor/translator/commentator/introduction` are identical.

editorstrcoin The plural form of `editortrcoin`.

editortrcofo Used if `editor/translator/commentator/foreword` are identical.

editorstrcofo The plural form of `editortrcofo`.

editortrcoaf Used if `editor/translator/commentator/aftword` are identical.

editorstrcoaf The plural form of `editortrcoaf`.

Keys for `editor/annotator/commentator/`⟨*role*⟩ combinations:

editortranin Used if `editor/annotator/commentator/introduction` are identical.

editorstranin The plural form of `editortranin`.

editortranfo Used if `editor/annotator/commentator/foreword` are identical.

editorstranfo The plural form of `editortranfo`.

editortranaf Used if `editor/annotator/commentator/aftword` are identical.

editorstranaf The plural form of `editortranaf`.

4.9.2.4 合并的译者角色，解释为职业

下面的这些关键词类似于 `translator` 的作用。它们常用来说明编辑的附加角色，比如‘translator and commentator’, ‘translator and introduction’。

translatorco Used if `translator/commentator` are identical.

translatorsco The plural form of `translatorco`.

translatoran Used if `translator/annotator` are identical.

translatorsan The plural form of `translatoran`.

translatorin Used if `translator/introduction` are identical.

translatorsin The plural form of `translatorin`.

translatorfo Used if `translator/foreword` are identical.

translatorsfo The plural form of `translatorfo`.

translatoraf Used if `translator/aftword` are identical.

translatorsaf The plural form of `translatoraf`.

Keys for `translator/commentator/`⟨*role*⟩ combinations:

translatorcoin Used if `translator/commentator/introduction` are identical.

translatorscoin The plural form of `translatorcoin`.

translatorcofo Used if `translator/commentator/foreword` are identical.

translatorscofo The plural form of `translatorcofo`.

translatorcoaf Used if `translator/commentator/aftword` are identical.

translatorscoaf The plural form of `translatorcoaf`.

Keys for `translator/annotator/`⟨*role*⟩ combinations:

translatoranin Used if `translator/annotator/introduction` are identical.

translatorsanin The plural form of `translatoranin`.

translatoranfo Used if `translator/annotator/foreword` are identical.

translatorsanfo The plural form of `translatoranfo`.

translatoranaf Used if `translator/annotator/aftword` are identical.

translatorsanaf The plural form of `translatoranaf`.

4.9.2.5 角色，解释为行为

下面的关键词指的角色解释为行为 (‘edited by’, ‘translated by’) 而不是职业 (‘editor’, ‘translator’)。

byauthor The expression ‘[created] by ⟨*name*⟩’.

byeditor The expression ‘edited by ⟨*name*⟩’.

bycompiler The expression ‘compiled by ⟨*name*⟩’.

byfounder The expression ‘founded by ⟨*name*⟩’.

bycontinuator The expression ‘continued by ⟨*name*⟩’.

byredactor The expression ‘redacted by ⟨*name*⟩’.

byreviser The expression ‘revised by ⟨*name*⟩’.

byreviewer The expression ‘reviewed by ⟨*name*⟩’.

bycollaborator An expression like ‘in collaboration with ⟨*name*⟩’ or ‘in cooperation with ⟨*name*⟩’.

bytranslator The expression ‘translated by ⟨*name*⟩’ or ‘translated from ⟨*language*⟩ by ⟨*name*⟩’.

bycommentator The expression ‘commented by ⟨*name*⟩’.

byannotator The expression ‘annotated by ⟨*name*⟩’.

4.9.2.6 合并的编者角色，解释为行为

下面的这些关键词类似于 `byeditor`, `bytranslator` 等的作用。它们常用来说明编辑的附加角色，比如‘edited and translated by’, ‘edited and furnished with an introduction by’, ‘edited, with a foreword, by’。

byeditortr Used if `editor/translator` are identical.

byeditorco Used if `editor/commentator` are identical.

byeditoran Used if `editor/annotator` are identical.

byeditorin Used if `editor/introduction` are identical.

byeditorfo Used if `editor/foreword` are identical.

byeditoraf Used if `editor/aftword` are identical.

Keys for `editor/translator/`⟨*role*⟩ combinations:

byeditortrco Used if `editor/translator/commentator` are identical.

byeditortran Used if `editor/translator/annotator` are identical.

byeditortrin Used if `editor/translator/introduction` are identical.

byeditortrfo Used if `editor/translator/foreword` are identical.

byeditortraf Used if `editor/translator/aftword` are identical.

Keys for `editor/commentator/`⟨*role*⟩ combinations:

byeditorcoin Used if `editor/commentator/introduction` are identical.

byeditorcofo Used if `editor/commentator/foreword` are identical.

byeditorcoaf Used if `editor/commentator/aftword` are identical.

Keys for `editor/annotator/`⟨*role*⟩ combinations:

byeditoranin Used if `editor/annotator/introduction` are identical.

byeditoranfo Used if `editor/annotator/foreword` are identical.

byeditoranaf Used if `editor/annotator/aftword` are identical.

Keys for `editor/translator/commentator/`⟨*role*⟩ combinations:

byeditortrcoin Used if `editor/translator/commentator/introduction` are identical.

byeditortrcofo Used if `editor/translator/commentator/foreword` are identical.

byeditortrcoaf Used if `editor/translator/commentator/aftword` are identical.

Keys for `editor/translator/annotator/`⟨*role*⟩ combinations:

byeditortranin Used if `editor/annotator/commentator/introduction` are identical.

byeditortranfo Used if `editor/annotator/commentator/foreword` are identical.

byeditortranaf Used if `editor/annotator/commentator/aftword` are identical.

4.9.2.7 合并的译者角色，解释为行为

下面的这些关键词类似于 `bytranslator` 的作用。它们常用来说明译者的附加角色，比如‘translated and commented by’, ‘translated and furnished with an introduction by’, ‘translated, with a foreword, by’。

bytranslatorco Used if `translator/commentator` are identical.

bytranslatoran Used if `translator/annotator` are identical.

bytranslatorin Used if `translator/introduction` are identical.

bytranslatorfo Used if `translator/foreword` are identical.

bytranslatoraf Used if `translator/aftword` are identical.

Keys for `translator/commentator/`⟨*role*⟩ combinations:

bytranslatorcoin Used if `translator/commentator/introduction` are identical.

bytranslatorcofo Used if `translator/commentator/foreword` are identical.

bytranslatorcoaf Used if `translator/commentator/aftword` are identical.

Keys for `translator/annotator/`⟨*role*⟩ combinations:

bytranslatoranin Used if `translator/annotator/introduction` are identical.

bytranslatoranfo Used if `translator/annotator/foreword` are identical.

bytranslatoranaf Used if `translator/annotator/aftword` are identical.

4.9.2.8 角色，解释为对象

与补充材料相关的角色可以解释为对象 ('with a commentary by') 而不是职业 ('commentator') 或行为 ('commented by')。

withcommentator The expression 'with a commentary by ⟨*name*⟩'.

withannotator The expression 'with annotations by ⟨*name*⟩'.

withintroduction The expression 'with an introduction by ⟨*name*⟩'.

withforeword The expression 'with a foreword by ⟨*name*⟩'.

withafterword The expression 'with an afterword by ⟨*name*⟩'.

4.9.2.9 补充材料

commentary The term 'commentary'.

annotations The term 'annotations'.

introduction The term 'introduction'.

foreword The term 'foreword'.

afterword The term 'afterword'.

4.9.2.10 出版信息细节

volume The term 'volume', referring to a book.

volumes The plural form of `volume`.

involumes The term 'in', as used in expressions like 'in ⟨*number of volumes*⟩ volumes'.

jourvol The term 'volume', referring to a journal.

jourser The term 'series', referring to a journal.

book The term 'book', referring to a document division.

part The term ‘part’, referring to a part of a book or a periodical.

issue The term ‘issue’, referring to a periodical.

newseries The expression ‘new series’, referring to a journal.

oldseries The expression ‘old series’, referring to a journal.

edition The term ‘edition’.

in The term ‘in’, referring to the title of a work published as part of another one, e. g., ‘⟨*title of article*⟩ in ⟨*title of journal*⟩’.

inseries The term ‘in’, as used in expressions like ‘volume ⟨*number*⟩ in ⟨*name of series*⟩’.

ofseries The term ‘of’, as used in expressions like ‘volume ⟨*number*⟩ of ⟨*name of series*⟩’.

number The term ‘number’, referring to an issue of a journal.

chapter The term ‘chapter’, referring to a chapter in a book.

version The term ‘version’, referring to a revision number.

reprint The term ‘reprint’.

reprintof The expression ‘reprint of ⟨*title*⟩’.

reprintas The expression ‘reprinted as ⟨*title*⟩’.

reprintfrom The expression ‘reprinted from ⟨*title*⟩’.

translationof The expression ‘translation of ⟨*title*⟩’.

translationas The expression ‘translated as ⟨*title*⟩’.

translationfrom The expression ‘translated from [the] ⟨*language*⟩’.

reviewof The expression ‘review of ⟨*title*⟩’.

origpubas The expression ‘originally published as ⟨*title*⟩’.

origpubin The expression ‘originally published in ⟨*year*⟩’.

astitle The term ‘as’, as used in expressions like ‘published by ⟨*publisher*⟩ as ⟨*title*⟩’.

bypublisher The term ‘by’, as used in expressions like ‘published by ⟨*publisher*⟩’.

4.9.2.11 出版状态

inpreparation The expression ‘in preparation’ (the manuscript is being prepared for publication).

submitted The expression ‘submitted’ (the manuscript has been submitted to a journal or conference).

forthcoming The expression ‘forthcoming’ (the manuscript has been accepted by a press or journal).

inpress The expression ‘in press’ (the manuscript is fully copyedited and out of the author’s hands; it is in the final stages of the production process).

prepublished The expression ‘pre-published’ (the manuscript is published in a preliminary form or location, such as online version in advance of print publication).

4.9.2.12 页码

page The term ‘page’.

pages The plural form of `page`.

column The term ‘column’, referring to a column on a page.

columns The plural form of `column`.

section The term ‘section’, referring to a document division (usually abbreviated as §).

sections The plural form of `section` (usually abbreviated as §§).

paragraph The term ‘paragraph’ (i. e., a block of text, not to be confused with `section`).

paragraphs The plural form of `paragraph`.

verse The term ‘verse’ as used when referring to a work which is cited by verse numbers.

verses The plural form of `verse`.

line The term ‘line’ as used when referring to a work which is cited by line numbers.

lines The plural form of `line`.

4.9.2.13 类型

下面的关键词常用于`@thesis`，`@report`，`@misc` 和其它一些条目的`type` 域中:

mathesis An expression equivalent to the term ‘Master’s thesis’.

phdthesis The term ‘PhD thesis’, ‘PhD dissertation’, ‘doctoral thesis’, etc.

candthesis An expression equivalent to the term ‘Candidate thesis’. Used for ‘Candidate’ degrees that have no clear equivalent to the Master’s or doctoral level.

techreport The term ‘technical report’.

resreport The term ‘research report’.

software The term ‘computer software’.

datacd The term ‘data CD’ or ‘CD-ROM’.

audiocd The term ‘audio CD’.

4.9.2.14 杂项

nodate The term to use in place of a date when there is no date for an entry e. g., ‘n.d.’

and The term ‘and’, as used in a list of authors or editors, for example.

andothers The expression ‘and others’ or ‘et alii’, used to mark the truncation of a name list.

andmore Like `andothers` but used to mark the truncation of a literal list.

4.9.2.15 标签

下面的字符串用于形成标签，比如‘Address: ⟨*url*⟩’ 或者‘Abstract: ⟨*abstract*⟩’ 等。

url The term ‘address’ in the sense of an internet address.

urlfrom An expression like ‘available from ⟨*url*⟩’ or ‘available at ⟨*url*⟩’.

urlseen An expression like ‘accessed on ⟨*date*⟩’, ‘retrieved on ⟨*date*⟩’, ‘visited on ⟨*date*⟩’, referring to the access date of an online resource.

file The term ‘file’.

library The term ‘library’.

abstract The term ‘abstract’.

annotation The term ‘annotations’.

4.9.2.16 标注

标注中使用的传统学术短语:

idem The term equivalent to the Latin ‘idem’ (‘the same [person]’).

idemsf The feminine singular form of `idem`.

idemsm The masculine singular form of `idem`.

idemsn The neuter singular form of `idem`.

idempf The feminine plural form of `idem`.

idempm The masculine plural form of `idem`.

idempn The neuter plural form of `idem`.

idempp The plural form of `idem` suitable for a mixed gender list of names.

ibidem The term equivalent to the Latin ‘ibidem’ (‘in the same place’).

opcit The term equivalent to the Latin term ‘opere citato’ (‘[in] the work [already] cited’).

loccit The term equivalent to the Latin term ‘loco citato’ (‘[at] the place [already] cited’).

confer The term equivalent to the Latin ‘confer’ (‘compare’).

sequens The term equivalent to the Latin ‘sequens’ (‘[and] the following [page]’), as used to indicate a range of two pages when only the starting page is provided (e. g., ‘25 sq.’ or ‘25 f.’ instead of ‘25–26’).

sequentes The term equivalent to the Latin ‘sequentes’ (‘[and] the following [pages]’), as used to indicate an open”=ended range of pages when only the starting page is provided (e. g., ‘25 sqq.’ or ‘25 ff.’).

passim The term equivalent to the Latin ‘passim’ (‘throughout’, ‘here and there’, ‘scatteredly’).

其他在标注中常用的短语:

see The term ‘see’.

seealso The expression ‘see also’.

seenote An expression like ‘see note ⟨*footnote*⟩’ or ‘as in ⟨*footnote*⟩’, used to refer to a previous footnote in a citation.

backrefpage An expression like ‘see page ⟨*page*⟩’ or ‘cited on page ⟨*page*⟩’, used to introduce back references in the bibliography.

backrefpages The plural form of `backrefpage`, e. g., ‘see pages ⟨*pages*⟩’ or ‘cited on pages ⟨*pages*⟩’.

quotedin An expression like ‘quoted in ⟨*citation*⟩’, used when quoting a passage which was already a quotation in the cited work.

citedas An expression like ‘henceforth cited as ⟨*shorthand*⟩’, used to introduce a shorthand in a citation.

thiscite The expression used in some verbose citation styles to differentiate between the page range of the cited item (typically an article in a journal, collection, or conference proceedings) and the page number the citation refers to. For example: “Author, Title, in: Book, pp. 45–61, `thiscite` p. 52.”

4.9.2.17 月份名

january The name ‘January’.

february The name ‘February’.

march The name ‘March’.

april The name ‘April’.

may The name ‘May’.

june The name ‘June’.

july The name ‘July’.

august The name ‘August’.

september The name ‘September’.

october The name ‘October’.

november The name ‘November’.

december The name ‘December’.

4.9.2.18 语言名

langamerican The language ‘American’ or ‘American English’.

langbrazilian The language ‘Brazilian’ or ‘Brazilian Portuguese’.

langcatalan The language ‘Catalan’.

langcroatian The language ‘Croatian’.

langczech The language ‘Czech’.

langdanish The language ‘Danish’.

langdutch The language ‘Dutch’.

langenglish The language ‘English’.

langestonian The language ‘Estonian’.

langfinnish The language ‘Finnish’.

langfrench The language ‘French’.

langgerman The language ‘German’.

langgreek The language ‘Greek’.

langitalian The language ‘Italian’.

langjapanese The language ‘Japanese’.

langlatin The language ‘Latin’.

langnorwegian The language ‘Norwegian’.

langpolish The language ‘Polish’.

langportuguese The language ‘Portuguese’.

langrussian The language ‘Russian’.

langslovak The language ‘Slovak’.

langslovene The language ‘Slovene’.

langspanish The language ‘Spanish’.

langswedish The language ‘Swedish’.

下面的字符串用于像‘translated from [the] English by ⟨*translator*⟩’之类的短语中:

fromamerican The expression ‘from [the] American’ or ‘from [the] American English’.

frombrazilian The expression ‘from [the] Brazilian’ or ‘from [the] Brazilian Portuguese’.

fromcatalan The expression ‘from [the] Catalan’.

fromcroatian The expression ‘from [the] Croatian’.

fromczech The expression ‘from [the] Czech’.

fromdanish The expression ‘from [the] Danish’.

fromdutch The expression ‘from [the] Dutch’.

fromenglish The expression ‘from [the] English’.

fromestonian The expression ‘from [the] Estonian’.

fromfinnish The expression ‘from [the] Finnish’.

fromfrench The expression ‘from [the] French’.

fromgerman The expression ‘from [the] German’.

fromgreek The expression ‘from [the] Greek’.

fromitalian The expression ‘from [the] Italian’.

fromjapanese The expression ‘from [the] Japanese’.

fromlatin The expression ‘from [the] Latin’.

fromnorwegian The expression ‘from [the] Norwegian’.

frompolish The expression ‘from [the] Polish’.

fromportuguese The expression ‘from [the] Portuguese’.

fromrussian The expression ‘from [the] Russian’.

fromslovak The expression ‘from [the] Slovak’.

fromslovene The expression ‘from [the] Slovene’.

fromspanish The expression ‘from [the] Spanish’.

fromswedish The expression ‘from [the] Swedish’.

4.9.2.19 国名

国名利用 `country` 加ISO-3166 国家代码实现本地化。不同语言的译文的简易版本应是ISO-3166 国家代码。注意: 仅默认定义了少量国名关键词，用来说明这一方法。这些关键词用在`@patent` 条目的`location` 域中，但也可以用于其他地方。

countryde 名称‘Germany’，缩写为 `DE`

countryeu 名称‘European Union’，缩写为 `EU`

countryep 类似于`countryeu`，但缩写为 `EP`。用于`patent` 条目。

countryfr 名称‘France’，缩写为 `FR`

countryuk 名称‘United Kingdom’，缩写为 `GB`(根据ISO-3166)

countryus 名称‘United States of America’，缩写为 `US`

4.9.2.20 专利和专利申请

与专利相关的字符串通过使用术语 `patent` 加ISO-3166 国家代码作为关键词实现本地化。注意: 仅默认定义了少量专利关键词，用来说明这一方法。这些关键词用在`@patent` 的`type` 域中。

patent 通用术语‘patent’

patentde 短语‘German patent’

patenteu 短语‘European patent’

patentfr 短语‘French patent’

patentuk 短语‘British patent’

patentus 短语‘U.S. patent’

专利申请以类似方式处理，使用字符串 `patreq` 作为关键词的基本名称:

patreq 通用术语‘patent request’

patreqde 短语‘German patent request’

patreqeu 短语‘European patent request’

patreqfr 短语‘French patent request’

patrequk 短语‘British patent request’

patrequs 短语‘U.S. patent request’

4.9.2.21 日期和时间

标准纪元的缩略词。支持世俗的或基督教两种版本。

commonera 纪元‘CE’(表示: 公元)

beforecommonera 纪元‘BCE’(表示: 公元前)

annodomini 纪元‘AD’(表示: 公元)

beforechrist 纪元‘BC’(表示: 公元前)

‘circa’ 日期的缩略词:

circa 词‘circa’(表示: 大约在)

从EDTF 日期解析的季节的缩略词:

spring 词‘spring’(表示: 春)

summer 词‘summer’(表示: 夏)

autumn 词‘autumn’(表示: 秋)

winter 词‘winter’(表示: 冬)

AM/PM 的缩略词:

am 词‘AM’(表示: 上午)

pm 词‘PM’(表示: 下午)

4.10 格式化命令

本节对应于用户指南部分的 § 3.10 节。著录和标注样式需要一些本节讨论的命令和工具来提供一定程度的高层可配置性。用户不需要非得写一个新的样式，如果仅要求修改文献表中的空格以及标注中的标点的话。

4.10.1 用户可定义的命令和钩子

本节对应用户指南部分的 § 3.10.1 节。这里讨论的命令和钩子可以由用户重定义，但著录和标注样式可能会提供一个不同于宏包定义的默认定义。这些命令在`biblatex.def` 中定义。注意所有的这些命令以`\mk…` 开头具有一个必选参数。

`\bibnamedelima` 这一分隔符控制构成姓名成分的元素间的间距。它由后端自动添加，位于第一个元素后面如果该元素少于三个字符长度和最后一个元素之前。默认的定义为`\addhighpenspace`，即一个由`highnamepenalty` 计数器 (§ 3.10.4 节) 值控制的空间，更多细节见 § 3.13.4。

\bibnamedelimb 这一分隔符控制构成姓名成分的元素间的间距。它由后端自动添加，位于所有元素之间，但存在\bibnamedelima 时不添加。默认的定义为\addlowpenspace，即一个由lownamepenalty 计数器 (§ 3.10.4 节) 值控制的空间，更多细节见 § 3.13.4。

\bibnamedelimc 这一分隔符控制构成姓名成分的元素间的间距。它由后端自动添加，位于前缀和姓之间，当 useprefix=true 时。默认的定义为\addhighpenspace，即一个由highnamepenalty 计数器 (§ 3.10.4 节) 值控制的空间，更多细节见 § 3.13.4。

\bibnamedelimd 这一分隔符控制构成姓名成分的元素间的间距。它由后端自动添加，位于所有元素之间，但存在\bibnamedelimc 时不添加。默认的定义为\addlowpenspace，即一个由lownamepenalty 计数器 (§ 3.10.4 节) 值控制的空间，更多细节见 § 3.13.4。

\bibnamedelimi 这一分隔符代替首字母后的\bibnamedelima/b。注意: 这仅应用于在bib 文件中给出的首字母后，而不是由biblatex 自动生成的首字母后，因为它使用自己的分隔符。

\bibinitperiod 当不应用\bibinithyphendelim 时由后端自动在所有缩写首字母后插入的标点。默认的定义是句点 (\adddot)。更多细节见 § 3.13.4。

\bibinitdelim 当不应用\bibinithyphendelim 时由后端自动在多个缩写首字母见插入空间。默认的定义是不可断行的词内空格。更多细节见 § 3.13.4。

\bibinithyphendelim 由后端自动在连字符连接的姓名成分的缩写首字母间插入的标点，代替\bibinitperiod 和\bibinitdelim。默认的定义时句点加一个不可断行连字符。更多细节见 § 3.13.4。

\bibindexnamedelima 用于在索引中取代\bibnamedelima。

\bibindexnamedelimb 用于在索引中取代\bibnamedelimb。

\bibindexnamedelimc 用于在索引中取代\bibnamedelimc。

\bibindexnamedelimd 用于在索引中取代\bibnamedelimd。

\bibindexnamedelimi 用于在索引中取代\bibnamedelimi。

\bibindexinitperiod 用于在索引中取代\bibinitperiod。

\bibindexinitdelim 用于在索引中取代\bibinitdelim。

\bibindexinithyphendelim 用于在索引中取代\bibinithyphendelim。

\revsdnamepunct 当姓和名顺序相反时两者之间插入的标点。默认是逗号。该命令应在姓名列表的格式化指令中使用。更多细节见 § 3.13.4。

\bibnamedash 用于代替参考文献表中接连再出现的责任者的破折号。默认是一个‘em’ 或‘en’ 破折号，根据文献表的缩进选取。

\labelnamepunct 该分隔符在文献表中用来按字母顺序排列的责任者 (author 或editor, 如果author 域未定义) 之后打印。使用该分隔符代替该位置的\newunitpunct。默认是\newunitpunct，即它与一般的单元标点并无不同，但允许方便地重设。

\subtitlepunct 该分隔符在title 和subtitle 域，booktitle 和booksubtitle，以及maintitle 和mainsubtitle 之间打印。替代该位置处的\newunitpunct。默认是\newunitpunct，即它与一般的单元标点并无不同，但允许方便地重设。

\intitlepunct 该分隔符在“in”与其后面的一些条目类型的标题之间打印。替代该位置处的\newunitpunct。默认是一个冒号加词内空格。

\bibpagespunct 该分隔符在pages 域前打印。替代该位置处的\newunitpunct。默认是一个逗号加词内空格。

\bibpagerefpunct 该分隔符在pageref 域前打印。替代该位置处的\newunitpunct。默认是一个词内空格。

\multinamedelim 该分隔符在author 或editor 之类的姓名列表的各项之间打印，如果列表中存在超过 2 个姓名的话。如果列表中仅有两个姓名，则使用\finalnamedelim。该命令应在姓名列表的所有格式化指令中使用。

\finalnamedelim 用于替代姓名列表中最后一个名字前的\multinamedelim。

\revsdnamedelim 在由两个姓名构成的姓名列表中第一个姓名之后打印的额外分隔符 (加在\finalnamedelim 后面)，如果第一个姓名的姓和名顺序相反的话。该命令应在姓名列表的所有格式化指令中使用。

\andothersdelim 一个author 或editor 之类的姓名列表被截短后在本地化字符串‘andothers’前打印的分隔符。该命令应在姓名列表的所有格式化指令中配合使用。

\multilistdelim 该分隔符在publisher 或location 之类的文本列表的各项之间打印，如果列表中存在超过 2 个文本项的话。如果列表中仅有两项，则使用\finallistdelim。该命令应在文本列表的所有格式化指令中使用。

\finallistdelim 用于替代文本列表中最后一个项前的\multilistdelim。

\andmoredelim 一个publisher 或location 之类的文本列表被截短后在本地化字符串‘andmore’前打印的分隔符。该命令应在文本列表的所有格式化指令中使用。

\multicitedelim 该分隔符在传递给当个标注命令的多个条目关键词之间打印。该命令应在标注命令定义中使用，例如在传递给\DeclareCiteCommand 的⟨*sepcode*⟩ 参数中。详见 § 4.3.1。

\supercitedelim 类似于\multicitedelim，但仅用于\supercite 命令中。

\compcitedelim 类似于\multicitedelim，但仅用于压缩 (‘compress’) 多个引用的标注样式中，即打印作者一次，如果后面接着的引用文献的作者相同的话。

\textcitedelim 类似于\multicitedelim，但仅用于\textcite 和相关命令 (§ 3.8.2) 中。

\nametitledelim 在责任者和标题之间打印的分隔符。该命令应在作者标题制和一些长标注样式的所有标注命令定义中使用。

\nameyeardelim 在责任者和年份之间打印的分隔符。该命令应在作者年制标注样式的所有标注命令定义中使用。

\namelabeldelim 在 name/title 和标签之间打印的分隔符。该命令应在字母顺序编码和数字顺序编码标注样式的所有标注命令定义中使用。

\nonameyeardelim 作者年制标注样式中 labelname 的替代者 (当 labelname 不存在时) 与年份之间打印的分隔符。仅用于无 labelname 的情况，因为当其存在时使用的是\nameyeardelim。

\volcitedelim 在\volcite 和相关命令的卷部分和页码/文本部分之间打印的分隔符 (见 § 3.8.6)。

\prenotedelim 在标注命令的⟨*prenote*⟩ 参数后面打印的分隔符。

\postnotedelim 在标注命令的⟨*postnote*⟩ 参数后面打印的分隔符。

\extpostnotedelim 当 postnote 出现在标注括号外时，标注命令中在标注和插入的⟨*postnote*⟩ 参数之间打印的分隔符。在标准样式中，这仅发生在标注使用条目的缩略域时。

\mkbibnamefamily{⟨*text*⟩}姓的格式化钩子，用于姓名列表的所有格式化指令中。

\mkbibnamegiven{⟨*text*⟩}类似于\mkbibnamefamily，当用于名。

\mkbibnameprefix{⟨*text*⟩}类似于\mkbibnamefamily，当用于姓名前缀。

\mkbibnamesuffix{⟨*text*⟩}类似于\mkbibnamefamily，当用于姓名后缀。

\relatedpunct 在相关类型参考文献本地化字符串和第一个关联条目数据之间的分隔符。

\relateddelim 在多个关联条目数据之间打印的分隔符。默认是断行。

\relateddelim<relatedtype> 在'relatedtype' 类型的关联条目中的多个关联条目数据之间打印的分隔符。没有默认设置，如果不指定具体类型的分隔符，则使用\relateddelim。

4.10.2 具体语言的命令

本节对应用户指南部分的 § 3.10.3 节。下面讨论的命令常在本地化模型中处理，但用户可能根据具体的语言重定义。注意，所有的命令以\mk… 开头，具有一个或更多的必选参数。

\bibrangedash 具体语言的范围破折号，默认是\textendash。

\bibrangessep 用于多个范围间的具体语言的分隔符。默认是逗号加一个空格。

\bibdatesep 简洁日期格式中各日期成分之间使用的具体语言的分隔符。默认是连字符 (\hyphen)。

\bibdaterangesep 用于日期范围之间的具体语言的分隔符。除了ymd 格式默认是\slash 外，其它所有日期格式中默认是\textendash。edtf 选项的日期格式是硬编码 (不轻易改变的) 为\slash，因为这是一种需符合标准的格式。

\mkbibdatelong 取三个域的名作为参数，对应三个日期成分 (以 year/month/day 的顺序)，并使用这些域的值以具体语言的长日期格式打印日期。

\mkbibdateshort 类似于\mkbibdatelong，但使用具体语言的短日期格式。

\mkbibtimezone 修改作为唯一参数传递进来的时区。默认情况下，修改‘Z’ 为\bibtimezone 的值。

\bibdateuncertain 当全局选项dateuncertain 打开时，在不确定日期后用的具体语言的标记。默认是一个空格加一个问号。

\bibdateeraprefix 当dateera 设为‘astronomical’ 时，在日期范围中作为起始 BCE/BC 日期前缀打印的具体语言标记。有定义的话，默认是\textminus，否则是\textendash。

\bibdateeraendprefix 当dateera 设为‘astronomical’ 时，在日期范围中作为终点 BCE/BC 日期前缀打印的具体语言标记。当\bibdaterangesep 设置为破折号 (dash) 时默认是短空格 (thin space)，否则是\bibdateeraprefix。这是一个独立宏，所以可以在一个负日期标记 (比如跟在一个破折号日期范围标记之后的) 前添加额外的空格，因为它看起来有点奇特。

\bibtimesep 分隔时间成分的具体语言标记，默认是分号。

\bibutctimezone UTC 时区的具体语言的打印字符串，默认是‘Z’。

\bibtimezonesep 分隔时间的可选时区成分的具体语言的标记，默认为空。

\bibdatetimesep 当时间和日期同时打印时分隔时间成分和日期成分的具体语言的分隔符。(见 § 3.1.2.1 节的<datetype>dateusetime 选项)。默认是一个空格对于 non-EDTF 输出格式，对于 EDTF 输出格式则是’T’。

\finalandcomma 在枚举中最后的‘and’ 前插入的逗号，如果可以用于具体的语言。

\finalandsemicolon 在枚举中最后的‘and’ 前插入的分号，如果可以用于具体的语言。

\mkbibordinal{⟨*integer*⟩}

取一个整数参数并打印成一般数字。

\mkbibmascord{⟨*integer*⟩}

类似于\mkbibordinal，但打印一个男性用的序号，如果可以用于具体的语言。

\mkbibfemord{⟨*integer*⟩}

类似于\mkbibordinal，但打印一个女性用的序号，如果可以用于具体的语言。

\mkbibneutord{⟨*integer*⟩}

类似于\mkbibordinal，但打印一个中性用的序号，如果可以用于具体的语言。

\mkbibordedition{⟨*integer*⟩}

类似于\mkbibordinal，但与术语‘edition’ 连用。

\mkbibordseries{⟨*integer*⟩}

类似于\mkbibordinal，但与术语'series' 连用。

4.10.3 用户可定义的长度和计数器

本节对应用户指南部分的 § 3.10.4 节。下面讨论的长度和计数器用户可以修改。著录和标注样式需要的时候应该使用它们，也可以提供不同于 biblatex 包提供的默认设置

\bibhang 如果用的话，是参考文献表的悬挂缩进。该长度在加载时初始化为\parindent。如果\parindent 因为某些原因设置为 0，\bibhang 将默认为 1em。

\biblabelsep 条目和对应标签之间的水平间距。使用list 环境并打印标签的参考文献样式应在环境定义中设置\labelsep 为\biblabelsep。

\bibitemsep 文献表中各条目间的垂直间距。使用list 环境的参考文献样式应在环境定义中设置\itemsep 为\bibitemsep。

\bibparsep 文献表中条目内段落间的垂直间距。使用list 环境的参考文献样式应在环境定义中设置\parsep 为\bibparsep。

abbrvpenalty 用于\addabbrvspace, \addabthinspace 和\adddotspace 的阀值，详见 § 4.7.4 节。

lownamepenalty 用于\addlowpenspace 和\addlpthinspace 的阀值，详见 § 4.7.4 节。

highnamepenalty 用于\addhighpenspace 和\addhpthinspace 的阀值，详见 § 4.7.4 节。

biburlnumpenalty 如果该计数器设置为大于 0 的值，biblatex 将允许在以url 包的\url 命令格式化的所有字符串中允许数字后面的断行。这将影响文献表中的URLs 和DOIs。断行点阀值将由该计数器的值确定。如果文献表中的URLs and/or DOIs 超出到页边中，尽可能设置该计数器值大于 0 但小于 10000(通常需要使用一个大值如 9000)。设置该计数器为 0 将关闭该功能。这是默认设置。[69]

biburlucpenalty 类似于biburlnumpenalty，差别在于它将会在所有大写字母后面添加断点。

biburllcpenalty 类似于biburlnumpenalty，差别在于它将会在所有小写字母后面添加断点。

4.10.4 辅助命令和钩子

本节的辅助命令和钩子具有特殊用途。从某种意义上说，其中一些用于biblatex 与著录和标注样式之间的通信。

\mkbibemph{⟨*text*⟩}

通用命令将其参数打印为强调的文本内容。这是一个包围标准\emph 命令的简单封套。除此之外，它使用 § 4.7.1 节的\setpunctfont 来调整紧接在设为斜体的文本后的标点符号的字体。如果punctfont 包选项未打开，该命令作用同\emph。

[69]译者: url、doi 超出页边时往往需要用到。

\mkbibitalic{⟨*text*⟩}

类似于\mkbibemph 的概念，但打印斜体文本。这是在一个标准\textit 命令的简单封套，其中包含\setpunctfont 命令。如果punctfont 包选项未打开，该命令作用同\textit。

\mkbibbold{⟨*text*⟩}

类似于\mkbibemph 的概念，但打印斜体文本。这是在一个标准\textbf 命令的简单封装，其中包含\setpunctfont 命令。如果punctfont 包选项未打开，该命令作用同\textbf。

\mkbibquote{⟨*text*⟩}

将其参数用引号包围起来的通用命令。如果加载了csquotes 包，该命令使用该包提供的具体语言的引号。\mkbibquote 也支持'American-style' 的标点，详见 § 4.7.5 节的\DeclareQuotePunctuation 命令。

\mkbibparens{⟨*text*⟩}

将其参数用圆括号包围起来的通用命令。该命令可以嵌套。当嵌套时，它将根据嵌套的层级交替使用圆括号和方括号。

\mkbibbrackets{⟨*text*⟩}

将其参数用方括号包围起来的通用命令。该命令可以嵌套。当嵌套时，它将根据嵌套的层级交替使用圆括号和方括号。

\bibopenparen⟨*text*⟩\bibcloseparen

\mkbibparens 命令的替代语法。这能跨编组使用。注意 \bibopenparen 和 \bibcloseparen 必须配套使用。

\bibopenbracket⟨*text*⟩\bibclosebracket

\mkbibbrackets 命令的替代语法。这能跨编组使用。注意 \bibopenbracket 和 \bibclosebracket 必须配套使用。

\mkbibfootnote{⟨*text*⟩}

将其参数作为脚注的通用命令。它是标准 LaTeX \footnote 命令的封套，并能消除脚注标记前的多余空格，阻止嵌套脚注。默认情况下，\mkbibfootnote 需要在脚注内容开始时大写并在结束时自动添加一个句号。可以重定义下面介绍的\bibfootnotewrapper 宏来修改其作用。

\mkbibfootnotetext{⟨*text*⟩}

类似于\mkbibfootnote，但使用\footnotetext 命令。

\mkbibendnote{⟨*text*⟩}

类似于\mkbibfootnote 的概念，但将其参数打印为尾注。\mkbibendnote 能消除尾注标记前的多余空格，并阻止嵌套。它支持由endnotes 包提供的\endnote 命令和pagenote 包和memoir 类提供的\pagenote 命令。如果两个命令都可用，\endnote 优先。如果没有可用的尾注命令，\mkbibendnote 将报错并回退为\footnote。默认情况下，\mkbibendnote 需要在尾注内容开始时大写并在结束时自动添加一个句号。可以重定义下面介绍的\bibfootnotewrapper 宏来修改其作用。

\mkbibendnotetext{⟨*text*⟩}

类似于\mkbibendnote，但使用\endnotetext 命令。请注意，对于这种写法，pagenote 包和memoir 都不提供相应的\pagenotetext 命令。这种情况下，\mkbibendnote 将报错并回退为\footnotetext。

\bibfootnotewrapper{⟨*text*⟩}

一个内部封套，将 \mkbibfootnote 和 \mkbibfootnotetext 命令的⟨*text*⟩ 参数包围起来。例如，\mkbibfootnote 最终归结为:

```
\footnote{\bibfootnotewrapper{text}}
```

该封套确保注文内容开始时大写并在结束时自动添加一个句号，默认定义为:

```
\newcommand{\bibfootnotewrapper}[1]{\bibsentence #1\addperiod}
```

如果不想大写首字母或者在注文尾部添加句号，不修改\mkbibfootnote，而要重定义\bibfootnotewrapper。

\bibendnotewrapper{⟨*text*⟩}

类似于\bibfootnotewrapper 的概念，但对应于\mkbibendnote 和\mkbibendnotetext 命令。

\mkbibsuperscript{⟨*text*⟩}

一个将参数转换成上标的通用命令。它是标准 LaTeX \textsuperscript 命令的封套，并能消除多余空格，允许前面的单词使用连字符。

\mkbibmonth{⟨*integer*⟩}

该命令根据其整数参数打印月份名。尽管该命令的输出与具体语言相关，但它的定义不是，因此在本地化模型中通常不重定义。

\mkbibseason{⟨*string*⟩}

该命令根据季节本地化字符串打印与包选项dateabbrev 对应版本的字符串。尽管该命令的输出与具体语言相关，但其定义并非如此，因此一般情况下不用在本地化模型中重定义。

\mkyearzeros{⟨*integer*⟩}

该命令根据datezeros 包选项 (§ 3.1.2.1) 设置移除或增添年份的前导零串。用于在日期格式化宏的定义中。

\mkmonthzeros{⟨*integer*⟩}

该命令根据datezeros 包选项 (§ 3.1.2.1) 设置移除或增添月份的前导零串。用于在日期格式化宏的定义中。

\mkdayzeros{⟨*integer*⟩}

该命令根据datezeros 包选项 (§ 3.1.2.1) 设置移除或增添日的前导零串。用于在日期格式化宏的定义中。

\mktimezeros{⟨*integer*⟩}

该命令根据timezeros 包选项 (§ 3.1.2.1) 设置移除或增添时间的前导零串。用于在日期格式化宏的定义中。

\forcezerosy{⟨*integer*⟩}

该命令将零串添加到年份中 (或者任何 4 位数的数字中)。用于日期格式化和序数中。

\forcezerosmdt{⟨*integer*⟩}

该命令将零串添加到月份、日或时间成分中 (或者任何 2 位数的数字中)。用于日期/时间格式化和序数中。

\stripzeros{⟨*integer*⟩}

该命令移除数字中的前导零串。用于日期格式化和序数中。

<labelfield>width 对于数据模型中任何标记为'Label field' 的域，根据上述的 shorthandwidth 自动创建一个格式化指令。因为默认的数据模型中shorthand 就是如此标记的，所以该功能是 shorthandwidth 功能的父集。

labelnumberwidth 类似于 shorthandwidth，但指的是labelnumber 域和长度\labelnumberwidth。顺序编码样式应该调整该指令以便与参考文献表中应用的格式一致。

labelalphawidth 类似于 shorthandwidth，但指的是labelalpha 域和长度\labelalphawidth。字母顺序样式应该调整该指令以便与参考文献表中应用的格式一致。[70]

bibhyperref 与\printfield 和\printtext 配合使用的一个特殊格式化指令。该指令将其参数包含在\bibhyperref 命令中，详见 § 4.6.4。

bibhyperlink 与\printfield 和\printtext 配合使用的一个特殊格式化指令。该指令将其参数包含在\bibhyperlink 命令中，详见 § 4.6.4。⟨*name*⟩ 传递给\bibhyperlink 命令的是entrykey 域的值。

[70]译者: 注意这个命令和上一个命令的差别，Alphabetic 和 Numeric 样式的差别

bibhypertarget 与\printfield 和\printtext 配合使用的一个特殊格式化指令。该指令将其参数包含在\bibhypertarget 命令中，详见 § 4.6.4。⟨*name*⟩ 传递给\bibhypertarget 命令的是entrykey 域的值。

volcitepages 控制类似\volcite 等标注命令参数中的页码或文本部分格式的一个特殊格式化指令。

volcitevolume 控制类似\volcite 等标注命令参数中的卷部分格式的一个特殊格式化指令。

date 控制\printdate 格式的一个特殊格式化指令 (§ 4.4.1)。注意，日期格式 (long/short 等) 由 § 3.1.2.1 节的包选项date 控制。该格式化指令仅控制如字体等额外的格式。

labeldate 类似于 date，当控制\printlabeldate 的格式。

<datetype>date 类似于 date，当控制\print<datetype>date 的格式。As date but controls the format of \print<datetype>date.

time 控制\printtime (§ 4.4.1) 格式的一个特殊格式化指令。注意: 时间格式 (24h/12h 等) 由 § 3.1.2.1 节的包选项time 控制。该格式化指令仅控制如字体等额外的格式。

labeltime 类似于 time，但控制\printlabeltime 的格式。

<datetype>time 类似于 time，但控制\print<datetype>time 的格式。As time but controls the format of \print<datetype>time.

4.10.5 辅助长度、计数器和其它功能

这里讨论的长度和计数器用于在biblatex 中项著录和标注样式传递信息。可以将它们认为是只读的 (read"=only)。注意: 所有的计数器都是 LaTeX 计数器。使用\value{counter} 来读取当前值。

\<labelfield>width 对于数据模型中任何标记为'label' 的域，根据上述的 shorthandwidth 自动创建一个长度。因为shorthand 在默认数据模型中就是如此标记的，所以该功能是 shorthandwidth 描述功能的父集。

\labelnumberwidth 表示最宽labelnumber 的长度。顺序编码著录样式应在参考文献表环境的定义中考虑该长度。

\labelalphawidth 表示最宽labelalpha 的长度。字母顺序编码著录样式应在参考文献表环境的定义中考虑该长度。

maxextraalpha 该计数器保存在extraalpha 域中能找到的最大数值。

maxextrayear 该计数器保存在extrayear 域中能找到的最大数值。

refsection 该计数器表示当前的refsection 环境。当在一个文献列表标题中请求时，该计数器返回传递给\printbibliography 命令的refsection 选项的值。

refsegment 该计数器表示当前的refsegment 环境。当在一个文献列表标题中请求时，该计数器返回传递给\printbibliography 命令的refsegment 选项的值。

maxnames 该计数器保存maxnames 包选项的设置。

minnames 该计数器保存minnames 包选项的设置。

maxitems 该计数器保存maxitems 包选项的设置。

minitems 该计数器保存minitems 包选项的设置。

instcount 该计数器由biblatex 根据每个出现的引用自动增加，在文献列表中则根据出现的条目自动增加。该计数器的值唯一的确定文档中一篇文献对象。[71]

citetotal 该计数器，仅在\DeclareCiteCommand 定义的标注命令的⟨*loopcode*⟩ 中提供，用于保存传递给标注命令的有效条目关键词总数。

citecount 该计数器，仅在\DeclareCiteCommand 定义的标注命令的⟨*loopcode*⟩ 中提供，用于保存⟨*loopcode*⟩ 正在处理的条目的序号。

multicitetotal 该命令类似于citetotal，但仅在 multicite 类命令中提供。它保存传递给 multicite 类命令的标注命令总数。注意，其包含的各个标注命令可能包含多于一个条目关键词。这一信息由citetotal 计数器提供。

multicitecount 该命令类似于citecount，但仅在 multicite 类命令中提供。它保存正在处理的标注命令序号。注意，其包含的各个标注命令可能包含多于一个条目关键词。这一信息由citetotal 和citecount 计数器提供。

listtotal 该计数器保存当前列表中项的总数。用于列表的格式化指令中，在其它任何地方使用时不保存一个有意义的数值。作为一个特例，它可能用在\printnames 和\printlist 命令的第二个参数中，详见 § 4.4.1。对于每个列表，都有一个与其名称相同的计数器用于保存相应列表的项的总数。例如，author 计数器保存了author 列表中的项的总数。无论姓名列表还是文本列表都是如此。这些计数器有点类似listtotal，差别在于他们可以用于列表格式化命令外独立使用。例如，一个参考文献著录样式可能会检查editor 计数器来决定是否在编者列表之后打印术语“editor”或者其复数形式“editors”

listcount 该计数器保存列表当前正在处理的项的序号。用于列表格式化指令，在其它任何地方使用无意义。

liststart 该计数器保存传递给\printnames 和\printlist 命令的⟨*start*⟩ 参数。用于列表格式化指令，在其它任何地方使用无意义。

liststop 该计数器保存传递给\printnames 和\printlist 命令的⟨*stop*⟩ 参数。用于列表格式化指令，在其它任何地方使用无意义。

\currentlang biblatex 中当前活动语言的名称。可以用于任何地方，默认为文档主体语言。它能在定义langid 的条目内部自动转换，如果autolang 和language 选项设置合适的话。注意，它不能追踪文档中所有的语言改变，仅用于当前的biblatex 设置。

[71] 译者:a single instance of a reference in the document

`\currentfield` `\printfield` 命令正在处理的域的名称。这一信息仅在局部的域格式化指令中提供。

`\currentlist` `\printlist` 命令正在处理的文本列表的名称。这一信息仅在局部的域格式化指令中提供。

`\currentname` `\printnames` 命令正在处理的姓名列表的名称。这一信息仅在局部的域格式化指令中提供。

4.10.6 多用途钩子

`\AtBeginBibliography{`⟨*code*⟩`}`

向在打印文献表开始时执行的内部钩子添加⟨*code*⟩。⟨*code*⟩ 在引文列表开始处执行，紧跟在`\defbibenvironment` 环境的⟨*begin code*⟩ 后面。该命令只能在导言区中使用。

`\AtBeginShorthands{`⟨*code*⟩`}`

向在缩略列表开始时执行的内部钩子添加⟨*code*⟩。⟨*code*⟩ 在缩略列表开始处执行，紧跟在`\defbibenvironment` 环境的⟨*begin code*⟩ 后面。该命令只能在导言区中使用。

该命令也等价于:

```
\AtBeginBiblist{shorthand}{code}
```

`\AtBeginBiblist{`⟨*biblistname*⟩`}{`⟨*code*⟩`}`

向在打印⟨*biblistname*⟩ 开始时执行的内部钩子添加⟨*code*⟩。⟨*code*⟩ 在引文列表开始处执行，紧跟在`\defbibenvironment` 环境的⟨*begin code*⟩ 后面。该命令只能在导言区中使用。

`\AtEveryBibitem{`⟨*code*⟩`}`

向在文献表中打印各条目开始时执行的内部钩子添加⟨*code*⟩。⟨*code*⟩ 紧跟在`\defbibenvironment` 环境的⟨*item code*⟩ 后面。各条目的数据此时已经提供。该命令只能在导言区中使用。

`\AtEveryLositem{`⟨*code*⟩`}`

向在缩略表中打印各项开始时执行的内部钩子添加⟨*code*⟩。⟨*code*⟩ 紧跟在`\defbibenvironment` 环境的⟨*item code*⟩ 后面。各条目的数据此时已经提供。该命令只能在导言区中使用。

该命令也等价于:

```
\AtEveryBiblistitem{shorthand}{code}
```

\AtEveryBiblistitem{⟨*biblistname*⟩}{⟨*code*⟩}

向在列表⟨*biblistname*⟩ 中打印各项开始时执行的内部钩子添加⟨*code*⟩。⟨*code*⟩ 紧跟在\defbibenvironment 环境的⟨*item code*⟩ 后面。各条目的数据此时已经提供。该命令只能在导言区中使用。

\AtNextBibliography{⟨*code*⟩}

类似于\AtBeginBibliography，但仅影响下一个\printbibliography。一旦执行该命令，原内部钩子定义将被清除。该命令可以用于正文中。

\AtEveryCite{⟨*code*⟩}

向在每个标注命令开始时执行的内部钩子添加⟨*code*⟩。⟨*code*⟩ 在标注命令的⟨*precode*⟩ 前执行 (见 § 4.3.1)。各条目的数据此时未提供。该命令只能在导言区中使用。

\AtEveryCitekey{⟨*code*⟩}

向在把各条目关键词传递给一个标注命令时执行一次的内部钩子添加⟨*code*⟩。⟨*code*⟩ 在标注命令的⟨*loopcode*⟩ 前执行 (见 § 4.3.1)。各条目的数据此时已经提供。该命令只能在导言区中使用。

\AtEveryMultiCite{⟨*code*⟩}

向在每个 multicite 命令开始时执行的内部钩子添加⟨*code*⟩。⟨*code*⟩ 在multiprenote 域 (见 § 4.3.2) 之前执行。各条目的数据此时未提供。该命令只能在导言区中使用。

\AtNextCite{⟨*code*⟩}

类似于\AtEveryCite，但仅影响下一个标注命令。一旦执行该命令，原内部钩子定义将被清除。该命令可以用于正文中。

\AtEachCitekey{⟨*code*⟩}

类似于\AtEveryCitekey，但仅影响当前标注命令。该命令可以用于正文中。当在一个标注中时，⟨*code*⟩ 添加到一个局部的内容部钩子中，可以用\ifcitation 判断是否是在一个标注中。

\AtNextCitekey{⟨*code*⟩}

类似于\AtEveryCitekey，但仅影响下一个条目关键词。一旦执行该命令，原内部钩子定义将被清除。该命令可以用于正文中。

\AtNextMultiCite{⟨*code*⟩}

类似于\AtEveryMultiCite，但仅影响下一个 multicite 命令。一旦执行该命令，原内部钩子定义将被清除。该命令可以用于正文中。

`\AtDataInput[`⟨*entrytype*⟩`]{`⟨*code*⟩`}`

向参考文献数据从`bbl` 文件导入后每个条目执行一次的内部钩子添加⟨*code*⟩。⟨*entrytype*⟩ 是⟨*code*⟩ 应用的条目类型。如果要应用于所有条目类型，则忽略该可选参数。⟨*code*⟩ 在条目导入后立即执行。该命令只能用于导言中。注意:⟨*code*⟩ 对于一个条目可能被执行多次，这发生在当一个相同条目在不同的`refsection` 环境中引用或者`sorting` 选项设置包含多余一个的排序格式时。当数据导入时，`refsection` 计数器保存各参考文献节的序号。

`\UseBibitemHook`

执行对应`\AtEveryBibitem` 的内部钩子。

`\UseEveryCiteHook`

执行对应`\AtEveryCite` 的内部钩子。

`\UseEveryCitekeyHook`

执行对应`\AtEveryCitekey` 的内部钩子。

`\UseEveryMultiCiteHook`

执行对应`\AtMultiEveryCite` 的内部钩子。

`\UseNextCiteHook`

执行对应`\AtNextCite` 的内部钩子。

`\UseNextCitekeyHook`

执行对应`\AtNextCitekey` 的内部钩子。

`\UseNextMultiCiteHook`

执行对应`\AtNextMultiCite` 的内部钩子。

`\DeferNextCitekeyHook`

局部地取消由`\AtNextCitekey` 设定的内部钩子。其本质是当钩子在`\DeclareCiteCommand` (见 § 4.3.1) 的⟨*precode*⟩ 参数中执行时，将该钩子延迟到标注列表中的下一个条目关键词。

4.11 提示与警告

本节提供了关于 biblatex 宏包接口的一些附加提示，也将论述一些普遍性的问题和容易误解的概念。

4.11.1 条目集

条目集已经在 § 3.12.5 节介绍过，本节主要讨论怎么在著录样式中处理条目集。从驱动的角度看，静态和动态的条目集并无差别。两者都以相同方式处理。只需要使用 § 4.4.1 的\entryset 命令遍历集的所有成员 (以在@set 条目的entryset 域中的出现的顺序，或者它们传递给\defbibentryset 命令的顺序进行遍历)，并在最后加上\finentry 命令即可。格式化则由集的成员各自条目类型的驱动控制。

```
\DeclareBibliographyDriver{set}{%
  \entryset{}{}%
  \finentry}
```

需要注意: 本宏包附带的 numeric 样式支持条目集细分，即集成员以一个字母或者其他记号来标记，标注命令可以引用整个集或者其中的某一具体成员。记号由样式文件以如下方式生成:

```
\DeclareBibliographyDriver{set}{%
  \entryset
    {\printfield{entrysetcount}%
     \setunit*{\addnbspace}}
    {}%
  \finentry}
```

entrysetcount 域保存了一个整数用于指示集成员在整个集中的位置。数字是转换为一个字母还是其他记号由域格式entrysetcount 控制。所有驱动需要做的是打印域和一些空格 (或者换行)。在标注中打印记号的方式类似。当顺序编码制样式给出\printfield{labelnumber} 时，可以简单地加上entrysetcount 域。

```
\printfield{labelnumber}\printfield{entrysetcount}
```

因为该域仅在处理标注指向一个集成员时定义，所以没有必要添加任何更多的判断。

4.11.2 电子出版信息

标准样式主要支持 arXiv 网站的文献[72]。其它资源的支持很容易增加。标准样式以如下方式处理eprint 域:

```
\iffieldundef{eprinttype}
  {\printfield{eprint}}
  {\printfield[eprint:\strfield{eprinttype}]{eprint}}
```

[72]译者:arXiv 原先是由物理学家保罗 • 金斯巴格在 1991 年建立的网站，本意在收集物理学的论文预印本，随后括及天文、数学等其它领域。金斯巴格因为这个网站获得了 2002 年的麦克阿瑟奖。arXiv 原先挂在洛斯阿拉莫斯国家实验室，是故早期被称为「LANL 预印本数据库」。目前的 arXiv 落脚于康乃尔大学，并在全球各地设有镜像站点。网站在 1999 年改名为 arXiv.org 。

如果eprinttype 域存在，上述代码将使用域格式 eprint:⟨*eprinttype*⟩。如果该格式未定义，\printfield 自动退回到使用域格式 eprint。有两种预定义的域格式，具体类型的域格式 eprint:arxiv 和通用域格式 eprint。

```
\DeclareFieldFormat{eprint}{...}
\DeclareFieldFormat{eprint:arxiv}{...}
```

换句话说，增加其他数据源的支持只需要定义一个名为 eprint:⟨*resource*⟩ 的与格式，其中⟨*resource*⟩ 是在eprinttype 域中使用的标识。

4.11.3 外部摘要和注释

外部摘要和注释已经在 § 3.12.8 节讨论过，本节为样式作者提供更多的背景知识。标准样式使用如下的宏 (来自 biblatex.def) 来处理摘要和注释:

```
\newbibmacro*{annotation}{%
  \iffieldundef{annotation}
    {\printfile[annotation]{\bibannotationprefix\thefield{entrykey}.tex
    ↪ }}%
    {\printfield{annotation}}}
\newcommand*{\bibannotationprefix}{bibannotation-}

\newbibmacro*{abstract}{%
  \iffieldundef{abstract}
    {\printfile[abstract]{\bibabstractprefix\thefield{entrykey}.tex}}%
    {\printfield{abstract}}}
\newcommand*{\bibabstractprefix}{bibabstract-}
```

如果abstract/annotation 域未定义，上述代码将从外部文件中加载摘要和注释。\printfile 将根据用户定义的前缀来搜索文件名。注意: 必须显式地设置 § 3.1.2.1 节的loadfiles 包选项来打开\printfile 功能。基于性能原因该功能默认是关闭的。

4.11.4 消除姓名歧义

在 § 3.1.2.3 节引入的uniquename 和uniquelist 选项支持多种操作模式。本节用举例方式介绍不同模式的差别。uniquename 选项消除labelname 列表中各姓名间的歧义，而uniquelist 则消除因maxnames/minnames 截短导致的labelname 列表歧义。两个选项可以单独使用也可以联合使用:

消除姓名歧义原理是根据由一个或多个姓名成分构成的'base'，来确定需要在其基础上添加什么 (如果存在的话)，使得姓名在当前参考文献节中是唯一的。消除姓名歧义由如下命令声明的 uniquename 模板控制:

\DeclareUniquenameTemplate{⟨*specification*⟩}

⟨*specification*⟩ 是\namepart 命令列表，定义了确定 uniquename 信息所使用的姓名成分。

\namepart[⟨*options*⟩]{⟨*namepart*⟩}

⟨*namepart*⟩ 是数据模型中的姓名成分，由\DeclareDatamodelConstant 命令定义 (见 § 4.2.3)。选项包括:

use=true, false　　default: false

在构建 uniquename 信息中仅使用⟨*namepart*⟩，如果存在相应的选项use'namepart' 并且值为 true。

base=true, false　　default: false

⟨*namepart*⟩ 是'base' 的部分，'base' 是用作唯一性区分的 namepart(s) 信息的主段。

默认的 uniquename 模板是:

```
\DeclareUniquenameTemplate{
  \namepart[use=true, base=true]{prefix}
  \namepart[base=true]{family}
  \namepart{given}
}
```

这意味着要区分的'base' 由姓 ('family') 和前缀 (如果useprefix 选项是 true) 构成。消除歧义主要通过增加模板中任何非'base' 姓名成分来实现，这里就是名 ('given') 成分。

4.11.4.1 单个姓名 (姓名间的区分)(uniquename)

下面从一些uniquename 示例开始，考虑如下数据:

```
John Doe   2008
Edward Doe 2008
John Smith 2008
Jane Smith 2008
```

假设我们使用作者年制且设置 uniquename=false，这种情况下，我们得到如下引用标注:

```
Doe 2008a
Doe 2008b
Smith 2008a
Smith 2008b
```

因为姓有歧义，且所有的年都相同，所以年后附加的字符用来区分并消除歧义。然而，很多样式指南强制要求附加字符只能用于相同作者的区分，而不能用于作者相同的姓的区分。为了消除作者姓的歧义，需要增加姓名的其它完整部分或者缩写来区分。这一需要由uniquename 选项处理，下面是使用了 uniquename=init 的引用标注:

```
J. Doe 2008
E. Doe 2008
Smith 2008a
Smith 2008b
```

uniquename=init 限制了用缩写来区分姓名。但因为'J. Smith' 仍然有歧义，所以没有增加。而使用 uniquename=full，标注如下:

```
J. Doe 2008
E. Doe 2008
John Smith 2008
Jane Smith 2008
```

为了说明 uniquename=init/full 和 allinit/allfull 的差别，我们下面介绍 'visible' 姓名的概念。'visible' 姓名是位于maxnames/minnames/uniquelist 截短点前的姓名，比如，给出数据:

```
William Jones/Edward Doe/Jane Smith
John Doe
John Smith
```

当 maxnames=1, minnames=1, uniquename=init/full 时，我们得到如下的引用标注:

```
Jones et al.
Doe
Smith
```

在消除歧义的时候，uniquename=init/full 仅考虑可见的姓名。因为本例中所有的可见姓名的姓都是不同的，所有没有姓名的其他部分附加进来。比较一下使用 uniquename=allinit 的输出:

```
Jones et al.
J. Doe
Smith
```

allinit 认为所有在labelname 列表中的姓名，包括列表截短后已经隐藏并且由'et al.' 代替的姓名。在本例中，'John Doe' 与 'Edward Doe' 存在歧义。因为两个 'Smiths' 无法通过添加缩写的方式区分，所以没有添加。现在来比较一下 uniquename = allfull 的输出:

```
Jones et al.
J. Doe
John Smith
```

`uniquename=mininit/minfull` 选项类似于 `init/full` 仅考虑可见姓名，但仅执行最小的歧义消除。即，仅对姓列表的歧义进行处理，考虑如下数据:

```
John Doe/William Jones
Edward Doe/William Jones
John Smith/William Edwards
Edward Smith/Allan Johnson
```

使用 `uniquename=init/full`，得到:

```
J. Doe and Jones
E. Doe and Jones
J. Smith and Edwards
E. Smith and Johnson
```

使用 `uniquename=mininit/minfull`，得到:

```
J. Doe and Jones
E. Doe and Jones
Smith and Edwards
Smith and Johnson
```

'Smiths' 并无歧义，因为姓名列表时没有歧义。`mininit/minfull` 选项仅对姓的列表相同情况进行处理。全局的看姓的列表，注意当未截短的列表的可见名相同的时候，截短的列表时也可能是不同的，比如下面的数据:

```
John Doe/William Jones
Edward Doe
```

使用 `maxnames=1`, `uniquename=init/full`:

```
J. Doe et al.
E. Doe
```

使用 `uniquename=mininit/minfull`:

```
Doe et al.
Doe
```

因为列表有 'et al.' 的不同，姓名列表就不歧义。

4.11.4.2 姓名列表 (列表间的区分) (uniquelist)

姓名列表也可能存在歧义问题。如果labelname 列表由maxnames/minnames 选项截短就可能产生歧义。这类问题由uniquelist 选处理，考虑如下数据:

```
Doe/Jones/Smith   2005
Smith/Johnson/Doe 2005
Smith/Doe/Edwards 2005
Smith/Doe/Jones   2005
```

很多作者年制样式需要在标注中截短，比如使用 maxnames=1 选项，得到:

```
Doe et al. 2005
Smith et al. 2005a
Smith et al. 2005b
Smith et al. 2005c
```

因为截短后作者存在歧义，所以添加额外字符确保引用标注的唯一性。同样的，一些样式强制要求额外字符只能用于所有作者都相同的情况。为了区分作者列表，必须增加更多的姓名，这样就会超出maxnames/minnames 选项设定的截短点。uniquelist 选项即描述这一需求，当 uniquelist=true，有:

```
Doe et al. 2005
Smith, Johnson et al. 2005
Smith, Doe and Edwards 2005
Smith, Doe and Jones 2005
```

uniquelist 选项以条目为限重设maxnames/minnames。大体上，标注的 'et al.' 部分扩展到无歧义的点–而且也基本到此为止。uniquelist 也可以与uniquename 联合使用，考虑如下数据:

```
John Doe/Allan Johnson/William Jones  2009
John Doe/Edward Johnson/William Jones 2009
John Doe/Jane Smith/William Jones     2009
John Doe/John Smith/William Jones     2009
John Doe/John Edwards/William Jones   2009
John Doe/John Edwards/Jack Johnson    2009
```

使用 maxnames=1，得到:

```
Doe et al. 2009a
Doe et al. 2009b
Doe et al. 2009c
Doe et al. 2009d
Doe et al. 2009e
Doe et al. 2009f
```

使用 maxnames=1, uniquename=full, uniquelist=true 则有:

```
Doe, A. Johnson et al. 2009
Doe, E. Johnson et al. 2009
Doe, Jane Smith et al. 2009
Doe, John Smith et al. 2009
Doe, Edwards and Jones 2009
Doe, Edwards and Johnson 2009
```

使用 uniquelist=minyear，消除列表歧义仅在可见列表和labelyear 相同的时候。这对于仅仅需要整个标注整体具有唯一性的作者年制样式是很有用的，但是不保证作者姓名的非歧义性。这一模式概念上域 uniquename=mininit/minfull 选项相关。考虑如下示例:

```
Smith/Jones   2000
Smith/Johnson 2001
```

使用 maxnames=1 和 uniquelist=true，得到:

```
Smith and Jones 2000
Smith and Johnson 2001
```

使用 uniquelist=minyear，则得到:

```
Smith et al. 2000
Smith et al. 2001
```

使用 uniquelist=minyear，两个文献的作者是否相同并不清楚，但标注的整体是非歧义的，因为年份的不同。与此相反，uniquelist=true 需要消除作者列表的歧义即便这一信息对于参考文献表的唯一引用是不必要的，看看如下示例:

```
Vogel/Beast/Garble/Rook  2000
Vogel/Beast/Tremble/Bite 2000
Vogel/Beast/Acid/Squeeze 2001
```

使用 maxnames=3, minnames=1, uniquelist=true，得到

```
Vogel, Beast, Garble et al. 2000
Vogel, Beast, Tremble et al. 2000
Vogel, Beast, Acid et al. 2001
```

使用 uniquelist=minyear 选项，则有:

```
Vogel, Beast, Garble et al. 2000
Vogel, Beast, Tremble et al. 2000
Vogel et al. 2001
```

在最后一个引用中，uniquelist=minyear 不重写maxnames/minnames，因为年份的不同，所以不需要消除与其它两个间的歧义。

4.11.5 浮动体和TOC/LOT/LOF 中的追踪器

当引用命令出现在浮动体 (比如图和表的题注) 中，因为浮动体无论是物理上还是逻辑上都在文本流之外，这会导致的学术反向引用 (比如'ibidem'，意为“出处同上”) 和基于页码追踪器的反向引用难以区分，因此这种引用的逻辑很难在其中应用。为避免这种问题，标注 (引用) 和页码追踪器在所有的浮动体中临时关闭。并且，这些追踪器加上反向引用追踪器 (backref) 在目录，图和表目录中也临时关闭。

4.11.6 混合编程接口

biblatex 宏包给样式作者提供了 2 个主要的编程接口即：bbx 文件中使用的\DeclareBibliographyDriver 命令用来定义各类参考文献条目的驱动 (即条目的格式处理器)，cbx 文件中使用的\DeclareCiteCommand 命令用来定义新的标注命令。然而有时候，混合使用这两个接口会很方便。比如\fullcite 命令就可以打印类似于完整参考文献条目的长串标注，该命令定义大体如下：

```
\DeclareCiteCommand{\fullcite}
  {...}
  {\usedriver{...}{\thefield{entrytype}}}
  {...}
  {...}
```

如上所见，打印标注的核心代码简单地为当前的条目类型执行了\DeclareBibliographyDriver 定义的驱动命令。当为长 (verbose，冗长的) 标注格式编写标注样式文件的时候，使用下面的结构是非常方便的：

```
\ProvidesFile{example.cbx}[2007/06/09 v1.0 biblatex citation style]

\DeclareCiteCommand{\cite}
  {...}
  {\usedriver{...}{cite:\thefield{entrytype}}}
  {...}
  {...}

\DeclareBibliographyDriver{cite:article}{...}
\DeclareBibliographyDriver{cite:book}{...}
\DeclareBibliographyDriver{cite:inbook}{...}
...
```

混合接口的另一个有用情况是在参考文献表中使用交叉引用 (cross”=references) 时。比如当打印@incollection 类型的条目，数据继承自@collection 父条目，可由一个指向对应父条目的简短指针来代替。

[1] Audrey Author: *Title of article.* In: [2], pp. 134–165.

[2] Edward Editor, ed.: *Title of collection*. Publisher: Location, 1995.

实现参考文献表内的这种交叉引用的一种方法是将它们当成标注，并使用`xref` 或 `crossref` 域的值作为条目关键词 (条目 bibtex 键)，示例如下:

```
\ProvidesFile{example.bbx}[2007/06/09 v1.0 biblatex bibliography style]

\DeclareCiteCommand{\bbx@xref}
  {}
  {...}% code for cross-references
  {}
  {}

\DeclareBibliographyDriver{incollection}{%
  ...
  \iffieldundef{xref}
    {...}% code if no cross-reference
    {\bbx@xref{\thefield{xref}}}%
  ...
}
```

当定义`\bbx@xref` 命令时，`\DeclareCiteCommand` 命令的⟨*precode*⟩, ⟨*postcode*⟩, 和⟨*sepcode*⟩ 参数留空，是因为上面示例中没有用到。交叉引用由`\bbx@xref` 命令的⟨*loopcode*⟩ 参数打印。更多的关于`xref` 域的细节见 § 2.2.3 节以及 § 2.4.1 节中的注意事项。在 § 4.6.2 节我们也看到了`\iffieldxref`, `\iflistxref`, 和`\ifnamexref` 测试命令。这些都也可以用 § 4.4.1 节的`\entrydata` 命令来实现。

```
\ProvidesFile{example.bbx}[2007/06/09 v1.0 biblatex bibliography style]

\DeclareBibliographyDriver{incollection}{%
  ...
  \iffieldundef{xref}
    {...}% code if no cross-reference
    {\entrydata{\thefield{xref}}{%
      % code for cross-references
      ...
    }}%
  ...
}
```

4.11.7 使用标点追踪

4.11.7.1 标点基础 样式作者设计参考文献驱动时需要记住一点原则: 块和单元的标点是异步处理的。用示例最容易解释这一点，看下面一段代码:

```
\printfield{title}%
\newunit
\printfield{edition}%
\newunit
\printfield{note}%
```

如果没有edition 域，那么这段代码的打印结果不会是:

```
Title. . Note
```

而会是

```
Title. Note
```

因为单元的标点追踪器是异步方式工作的。\newunit 命令将不会立即打印标点。它仅是记录一个单元的边界并且将\newunitpunct 命令放入标点缓存中。该缓存会有接下来的\printfield、\printlist 或类似命令进行处理，且仅当这些命令各自处理的域或列表已定义的时候才会处理。像\printfield 这样的命令在插入任何块和单元的标点之前将首先考虑 3 个因素:

- 是否有新的单元/块的输出请求？

  = 前面是否有\newunit 或者\newblock 命令？

- 前面的命令是否有打印输出？

  = 前面是否有\printfield 或者相似命令？

  = 该命令是否实际打印了任何东西？

- 现在是否要打印一些东西？

  要进行打印处理的域或列表是否已定义？

块和单元的标点只会在上述所有条件满足的时候才会输出。让我们再次考虑上面的示例:

```
\printfield{title}%
\newunit
\printfield{edition}%
\newunit
\printfield{note}%
```

如果edition 域没有定义会发生什么呢？第一个\printfield 命令打印了标题并设置一个内部的‘new text’ 标志。第一个\newunit 命令设置一个内部的‘new unit’ 标志。这使没有任何标点输出。第二个\printfield 命令不进行任何处理因为edition 域未定义。接下来的\newunit 命令再次设置‘new unit’ 标志，仍然没有标点输出。

第三个\printfield 命令检测note 域是否已定义，如果是，它会寻找‘new text’和‘new unit’ 标志。如果两个标志都存在，那么它会在打印 note 前插入标点缓存。然后它会清除‘new unit’ 标志然后再次设置‘new text’ 标志。

所有这些听起来似乎很复杂，但实际上，这意味着可以用顺序的方式写一个具有很多部件的参考文献驱动。这种方法的优势在不使用标点追踪而实现上述代码功能时会体现的很明显。如果不用标点追踪，那么会因为大量对所有可能存在域的判断产生一个复杂的\iffieldundef 判断命令集合。

```
\iffieldundef{title}%
  {\iffieldundef{edition}
     {\printfield{note}}
     {\printfield{edition}%
      \iffieldundef{note}%
        {}
        {. \printfield{note}}}}
  {\printfield{title}%
   \iffieldundef{edition}
     {}
     {. \printfield{edition}}%
   \iffieldundef{note}
     {}
     {. \printfield{note}}}%
```

4.11.7.2 常见错误 把单元的标点处理认为是同步处理的是一个相当常见的误解。这会导致当驱动中包含抄录文本时出现一些典型错误。考虑下面导致标点错位的错误代码段:

```
\printfield{title}%
\newunit
(\printfield{series} \printfield{number})%
```

这段代码将产生下面的结果:

```
Title (. Series Number)
```

这里发生了什么呢？第一个\printfield 命令打印了标题，然后\newunit 命令标记了一个新的单元边界但不打印任何内容。单元的标点由下一个\printfield 命令打印。这是前面提过的异步机制。然而因为左括号在下一个 \printfield 命令插入标点前立即打印，所以导致了错误的句点。当插入任何原样文本比如括号 (还包括由 \bibopenparen 和\mkbibparens 命令打印的括号) 时，总需要将这些文本用\printtext 命令包起来。要让标点追踪正常运转，需要让驱动知道所有插入的原样文本。这是\printtext 命令的作用所在。\printtext 命令联系标点追踪器确

保标点缓存在原样文本打印前插入。它也设定内部 'new text' 标志。注意本例中还有第三处原样文本即\printfield{series} 后面的空格。在改正的示例中，我们将使用标点追踪器来处理该空格。

```
\printfield{title}%
\newunit
\printtext{(}%
\printfield{series}%
\setunit*{\addspace}%
\printfield{number}%
\printtext{)}%
```

尽管上面的代码能够如常工作，但处理括号、引号和其它包围某个域的标点是，推荐的方式是定义一个域格式:

```
\DeclareFieldFormat{parens}{\mkbibparens{#1}}
```

域格式可以同时用于\printfield 和\printtext 命令，因此我们可以利用它对若干个域用一堆括号进行包裹。

```
\printtext[parens]{%
  \printfield{series}%
  \setunit*{\addspace}%
  \printfield{number}%
}%
```

这里我们还需要处理没有 series 信息时的情况，因此进一步改进代码如下:

```
\iffieldundef{series}
  {}
  {\printtext[parens]{%
     \printfield{series}%
     \setunit*{\addspace}%
     \printfield{number}}}%
```

最后的一点提示: 本地化字符串对于标点追踪器来说不是原样文本。因为\bibstring 和相似命令能联系标点追踪器，因此就不需要用\printtext 包裹起来。

4.11.7.3 高级用法

标点追踪器也可用来处理更复杂的情况。比如，考虑需要对location、publisher 和 year 根据数据是否提供以如下的格式打印：

```
...text. Location: Publisher, Year. Text...
```

```
...text. Location: Publisher. Text...
...text. Location: Year. Text...
...text. Publisher, Year. Text...
...text. Location. Text...
...text. Publisher. Text...
...text. Year. Text...
```

这个问题可以用一个相当复杂的`\iflistundef` 和`\iffieldundef` 判断集进行处理，通过这些判断可以确定所有可能的域的组合：

```
\iflistundef{location}
  {\iflistundef{publisher}
     {\printfield{year}}
     {\printlist{publisher}%
      \iffieldundef{year}
        {}
        {, \printfield{year}}}}
  {\printlist{location}%
   \iflistundef{publisher}%
     {\iffieldundef{year}
        {}
        {: \printfield{year}}}
     {: \printlist{publisher}%
      \iffieldundef{year}
        {}
        {, \printfield{year}}}}%
```

可以应用`\ifthenelse` 命令和 § 4.6.3 讨论的布尔运算可使上面的代码更具可读性。但本质上是一样的。然而，也可以按顺序写成如下方式:

```
\newunit
\printlist{location}%
\setunit*{\addcolon\space}%
\printlist{publisher}%
\setunit*{\addcomma\space}%
\printfield{year}%
\newunit
```

在实际使用中，你会经常使用标点追踪器执行一些显式或隐式的组合判断，比如，考虑如下格式 (注意当没有 publisher 时 location 后面的标点)

```
...text. Location: Publisher, Year. Text...
...text. Location: Publisher. Text...
```

```
...text. Location, Year. Text...
...text. Publisher, Year. Text...
...text. Location. Text...
...text. Publisher. Text...
...text. Year. Text...
```

这可以用如下代码进行处理:

```
\newunit
\printlist{location}%
\iflistundef{publisher}
  {\setunit*{\addcomma\space}}
  {\setunit*{\addcolon\space}}%
\printlist{publisher}%
\setunit*{\addcomma\space}%
\printfield{year}%
\newunit
```

因为当没有 publisher 时 location 后面的标点的特殊性，我们需要用一个\iflistundef 判断来确保正确性。剩下其它的则有标点追踪器处理。

4.11.8 本地化定制模型

参考文献格式要求可能包含某些规定，比如像'edition' 之类的字符串怎么缩写或者需要表示成一些固定的形式。例如MLA 格式要求作者在参考文献表的标题中使用'Works Cited' 而不是'Bibliography' 或'References'。本地化命令比如 § 3.9 节的\DefineBibliographyStrings 可以在cbx 和bbx 文件中处理这些情况。然而，用翻译内容重载样式文件是相当不便的。而这正是 § 4.9.1 节的\DeclareLanguageMapping 命令发挥作用的地方。这一命令将一个lbx 文件映射为某一babel/polyglossia 语言的替代翻译。例如，可以创建一个名为french-humanities.lbx 的文件提供适用于人文学科的法语翻译并在导言区或者配置文件中将其映射为babel/polyglossia 语言 french。

```
\DeclareLanguageMapping{french}{french-humanities}
```

如果正文的语言设置为 french，french-humanities.lbx 将替换french.lbx。回到前述的MLA 示例，一个MLA 样式可能带有一个american-mla.lbx 文件来提供符合MLA 格式规要求的字符串。它将在cbx 和/或 bbx 文件中声明如下映射:

```
\DeclareLanguageMapping{american}{american-mla}
```

因为替换的lbx 文件可以从标准的american.lbx 模型中继承字符串，所以american-mla.lbx 可以简化为:

```
\ProvidesFile{american-mla.lbx}[2008/10/01 v1.0 biblatex localization]
\InheritBibliographyExtras{american}
\DeclareBibliographyStrings{%
  inherit          = {american},
  bibliography     = {{Works Cited}{Works Cited}},
  references       = {{Works Cited}{Works Cited}},
}
\endinput
```

替换的lbx 文件必须保证本地化模型是完整的。这可以通过从相应的标准模型中继承来实现。如果语言 american 映射为american-mla.lbx，biblatex 将不会加载american.lbx 除非该模型被明确要求加载。在上述示例中，继承'strings' 和'extras' 将使得biblatex 在american-mla.lbx 中应用修改前加载american.lbx。

注意:\DeclareLanguageMapping 不用于处理语言的变体 (比如 American English 和 British English)) 或者babel/polyglossia 语言别名 (比如 USenglish vs. american)。例如，babel/polyglossia 提供了 USenglish 选项类似于 american。因此biblatex 附带一个USenglish.lbx 文件，该文件简单的从american.lbx 文件中继承所有数据 (而该文件又从english.lbx 文件中获取字符串)。换句话说，语言变体和babel/polyglossia 语言别名的映射发射在文件层，因此biblatex 的语言支持可以简单地增加额外的lbx 文件来实现拓展。没有必要进行集中的映射，如果需要支持比如 Portuguese (babel/polyglossia: portuges)，可以创建一个名为portuges.lbx 的文件。如果babel/polyglossia 提供一个名为 brasil 的别名，可以创建一个brasil.lbx 文件并可从portuges.lbx 中继承数据。相比之下，\DeclareLanguageMapping 主要用于处理文体上的变化，像'humanities 对 natural sciences' or 'MLA 对 APA' 等，这通常是建立在现有的lbx 文件基础上的。

4.11.9 编组

在标注和著录样式中，可能需要设置标志或保存一些值以便后面使用。这种情况下，理解宏包执行的基本编组结构很关键。一条经验法则是，无论诸如 § 4.6 节讨论的样式作者命令是否存在，所有的工作都是在一个大的编组内进行的，因为本宏包的作者接口都是局部的。当存在参考文献数据时，至少存在一个额外的编组，下面是一些基本规则:

- 由\printbibliography 或类似命令打印的整个文献表是在一个编组中处理。表中的每个条目都是在一个额外的编组中，这一编组包含了\defbibenvironment 的⟨*item code*⟩ 和所有的驱动代码。
- 由\printbiblist 命令打印的整个文献表是在一个编组中处理。表中的每个条目都是在一个额外的编组中，这一编组包含了\defbibenvironment 的⟨*item code*⟩ 和所有的驱动代码。
- 所有由\DeclareCiteCommand 命令定义的标注命令都是在一个编组中处理，该编组包含由⟨*precode*⟩, ⟨*sepcode*⟩, ⟨*loopcode*⟩, 和⟨*postcode*⟩ 等参数构成的完整

标注代码。每次执行的⟨*loopcode*⟩ 都包含在一个额外的编组中。如果指定了任何的⟨*wrapper*⟩，包含⟨*wrapper*⟩ 代码和标注代码的整个单元都在一个额外的编组中。

- 除了由`\DeclareCiteCommand` 定义的所有后端命令会产生编组外，所有的'autocite' 和 'multicite' 定义也都会产生一个额外的编组。
- `\printfile`, `\printtext`, `\printfield`, `\printlist`, 和`\printnames` 命令也形成编组。这意味着所有的格式化指令都是在它们自身的编组中处理。
- 所有的`lbx` 文件都是在一个编组中加载和处理。如果一个`lbx` 文件包含的一些代码不是`\DeclareBibliographyExtras` 的一部分，那么这些定义是全局的。

注意: 在标注和著录样式中使用`\aftergroup` 是不可靠的，因为在一定环境中应用的编组的精确层数在宏包的未来版本中可能发生变化。上述说明中如果说某些东西在一个编组中处理，这意味着至少存在一个编组，也可能存在多层嵌套的编组。

4.11.10 命名空间

为减小命名冲突的风险，LaTeX 宏包通常在其内部宏名前加上一个代表该宏包的短字符串。例如: 如果`foobar` 宏包需要一个内部使用的宏，它通常会命名为`\FB@macro` 或`\foo@macro` 而不是`\macro` or `\@macro`。下面是`biblatex` 使用或推荐使用的前缀字符串:

`blx` 所有的宏名像`\blx@name` 的宏严格作为内部使用。这也应用于计数器名，长度名，布尔开关等。这些宏可能会以非后向兼容的方式改变，它们可能会重命名设置删除掉而不会有更多的说明。这种改变也不会在版本修改历史和版本发布信息中出现。简而言之: 不要在任何样式中使用以 `blx` 字符串开头的宏。

`abx` 以 `abx` 为前缀的宏也是内部使用，但会相当稳定。但仍应优先使用正式的作者接口提供的工具，但有些情况下使用某一 `abx` 宏可能会比较方便。

`bbx` 建议在著录样式中内部使用的宏名的前缀

`cbx` 建议在标注样式中内部使用的宏名的前缀

`lbx` 建议在本地化模型中内部使用的宏名的前缀。本地化模型需要添加第二个前缀来指定语言，比如一个为西班牙语本地化模型定义的内部宏可以命名为`\lbx@es@macro`。

附录

A 驱动层的默认数据源映射

下面是驱动层的默认数据源映射。

A.1 bibtex

bibtex驱动当然是biblatex/biber 支持数据格式中最全面和成熟的。§ 2.1.2和§ 2.2.5节的别名通过下面这些默认数据源映射实现。

```
\DeclareDriverSourcemap[datatype=bibtex]{
  \map{
    \step[typesource=conference, typetarget=inproceedings]
    \step[typesource=electronic, typetarget=online]
    \step[typesource=www,        typetarget=online]
  }
  \map{
    \step[typesource=mastersthesis, typetarget=thesis, final]
    \step[fieldset=type,            fieldvalue=mathesis]
  }
  \map{
    \step[typesource=phdthesis, typetarget=thesis, final]
    \step[fieldset=type,        fieldvalue=phdthesis]
  }
  \map{
    \step[typesource=techreport, typetarget=report, final]
    \step[fieldset=type,         fieldvalue=techreport]
  }
  \map{
    \step[fieldsource=address,       fieldtarget=location]
    \step[fieldsource=school,        fieldtarget=institution]
    \step[fieldsource=annote,        fieldtarget=annotation]
    \step[fieldsource=archiveprefix, fieldtarget=eprinttype]
    \step[fieldsource=journal,       fieldtarget=journaltitle]
    \step[fieldsource=primaryclass,  fieldtarget=eprintclass]
    \step[fieldsource=key,           fieldtarget=sortkey]
    \step[fieldsource=pdf,           fieldtarget=file]
  }
}
```

B 默认继承设置

下表给出了默认的biber 交叉引用规则。更多解释见 §§ 2.4.1 和 4.5.11节。

Types		Fields	
Source	**Target**	**Source**	**Target**
*	*	ids	–
		crossref	–
		xref	–
		entryset	–
		entrysubtype	–
		execute	–
		label	–
		options	–
		presort	–
		related	–
		relatedoptions	–
		relatedstring	–
		relatedtype	–
		shorthand	–
		shorthandintro	
		sortkey	
mvbook, book	inbook, bookinbook, suppbook	author	author
		author	bookauthor
mvbook	book, inbook, bookinbook, suppbook	title	maintitle
		subtitle	mainsubtitle
		titleaddon	maintitleaddon
		shorttitle	–
		sorttitle	–
		indextitle	–
		indexsorttitle	–
mvcollection, mvreference	collection, reference, incollection, inreference, suppcollection	title	maintitle
		subtitle	mainsubtitle
		titleaddon	maintitleaddon
		shorttitle	–
		sorttitle	–
		indextitle	–
		indexsorttitle	–
mvproceedings	proceedings, inproceedings	title	maintitle
		subtitle	mainsubtitle
		titleaddon	maintitleaddon
		shorttitle	–
		sorttitle	–
		indextitle	–
		indexsorttitle	–
book	inbook, bookinbook, suppbook	title	booktitle
		subtitle	booksubtitle
		titleaddon	booktitleaddon
		shorttitle	–
		sorttitle	–
		indextitle	–
		indexsorttitle	–
collection, reference	incollection, inreference, suppcollection	title	booktitle
		subtitle	booksubtitle
		titleaddon	booktitleaddon
		shorttitle	–
		sorttitle	–
		indextitle	–
		indexsorttitle	–

Types		Fields	
Source	**Target**	**Source**	**Target**
proceedings	inproceedings	title	booktitle
		subtitle	booksubtitle
		titleaddon	booktitleaddon
		shorttitle	–
		sorttitle	–
		indextitle	–
		indexsorttitle	–
periodical	article, suppperiodical	title	journaltitle
		subtitle	journalsubtitle
		shorttitle	–
		sorttitle	–
		indextitle	–
		indexsorttitle	–

C 默认的排序格式

C.1 字母表顺序格式 1

下表给出了默认的标准字母表顺序排序格式。更多解释见 § 3.5节。

Option	Sorting scheme				
nty	presort	→sortname	→sorttitle	→sortyear	→volume
	↪mm	↪author	↪title	↪year	
		↪editor			
		↪translator			
		↪sorttitle			
		↪title			
nyt	presort	→sortname	→sortyear	→sorttitle	→volume
	↪mm	↪author	↪year	↪title	
		↪editor			
		↪translator			
		↪sorttitle			
		↪title			
nyvt	presort	→sortname	→sortyear	→volume	→sorttitle
	↪mm	↪author	↪year		↪title
		↪editor			
		↪translator			
		↪sorttitle			
		↪title			
all	presort	→sortkey			
	↪mm				

C.2 字母表顺序格式 2

下表是为 alphabetic 样式定义的默认字母表顺序排序格式。更多解释见 § 3.5节。

Option	Sorting scheme					
anyt	presort ↪ mm	→ labelalpha	→ sortname ↪ author ↪ editor ↪ translator ↪ sorttitle ↪ title	→ sortyear ↪ year	→ sorttitle ↪ title	→ volume
anyvt	presort ↪ mm	→ labelalpha	→ sortname ↪ author ↪ editor ↪ translator ↪ sorttitle ↪ title	→ sortyear ↪ year	→ volume	→ sorttitle ↪ title
all	presort ↪ mm	→ labelalpha	→ sortkey			

C.3 纪年格式

下表给出了默认定义的纪年排序格式。更多解释见 § 3.5节。

Option	Sorting scheme			
ynt	presort ↪ mm	→ sortyear ↪ year ↪ 9999	→ sortname ↪ author ↪ editor ↪ translator ↪ sorttitle ↪ title	→ sorttitle ↪ title
ydnt	presort ↪ mm	→ sortyear (desc.) ↪ year (desc.) ↪ 9999	→ sortname ↪ author ↪ editor ↪ translator ↪ sorttitle ↪ title	→ sorttitle ↪ title
all	presort ↪ mm	→ sortkey		

D biblatex XML 数据源格式 (biblatexml)

biblatexml XML 数据源格式是为biblatex 用户设计的一个可扩展的现代数据源格式。BibTeX 格式.bib文件存在一些局限，特别是涉及到支持 UTF-8 和姓名格式问题时。biber 通过采用修改的 btparse C 库等多种方式处理 UTF-8 方面的限制，但针对 BibTeX 的相当古老的姓名解析规则是根据简单的西方人名硬编码的。

biblatexml 是为参考文献数据提供的一种 XML 格式。当biber 读写biblatexml 格式数据源时，它自动为数据源输出一个 RelaXNG XML 纲要，这一概要由活动的biblatex 数据模型动态生成。biblatexml 数据源因为可用的域依赖于数据模型所以不存在静态的纲要。biblatexml 数据源格式具有相当高的自解析性—要理解其格式，通常只需要从已有的 BibTeX 格式数据源生成一个biblatexml 数据源，然后分析其内容就可以做到。biber 也允许用户根据数据模型生成纲要验证biblatexml 数据源。

因为biblatexml 格式是 XML 并且依赖于数据模型，且数据模型是用户可扩展的 (见 § 4.5.4)，biblatexml 格式可以处理 BibTeX 格式数据源无法处理的一些扩展内容比如新的姓名成分，子条目上的选项等。因为它是一种 XML 格式，所以使用标准的 XML 处理库和工具转换成其他 XML 格式或 HTML 也相当简单。

下面用示例阐述该格式。习惯上，biblatexml 文件扩展名为.bltxml，kpsewhich能识别这一格式。

D.1 文件头

biblatexml 文件一个标准 XML 文件头开始:

```
<?xml version="1.0" encoding="UTF-8"?>
```

纲要模型、类型和纲要类型空间为:

```
<?xml-model href="biblatexml.rng"
            type="application/xml"
            schematypens="http://relaxng.org/ns/structure/1.0"?>
```

当biber 生成biblatexml 数据源时，它自动添加添加这一行，并为自动生成的 RelaXNG XML 纲要指定纲要模型 (href) 属性以便验证。

D.2 正文

biblatexml 数据源正文类似于:

```
<bltx:entries
  xmlns:bltx="http://biblatex-biber.sourceforge.net/biblatexml">

  <bltx:entry id="" entrytype="">
  </bltx:entry>
       .
       .
       .
  <bltx:entry id="" entrytype="">
  </bltx:entry>

</bltx:entries>
```

正文是一个或多个entry元素存在于顶层的entries元素中，并且所有都处于bltx命名空间内。一个条目具有一个id属性对应于 BibTeX 条目关键词，具有一个entrytype属性对应于 BibTeX 条目类型。例如:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="biblatexml.rng"
```

```
            type="application/xml"
            schematypens="http://relaxng.org/ns/structure/1.0"?>
<bltx:entries
  xmlns:bltx="http://biblatex-biber.sourceforge.net/biblatexml">
  <bltx:entry id="key1" entrytype="book">
  </bltx:entry>
</bltx:entries>
```

对应的 bib 文件为:

```
@book{key1,
}
```

一般来说，biblatexml 格式数据源中的 XML 元素具有与数据模型中的域对应的名称，如同 BibTeX 格式数据源一般。所以，当 BibTeX 数据源是:

```
@book{key1,
  TITLE = {...},
  ISSUE = {...},
  NOTE = {...}
}
```

biblatexml 中将是:

```
  <bltx:entry id="key1" entrytype="book">
    <bltx:title>...</bltx:title>
    <bltx:issue>...</bltx:issue>
    <bltx:note>...</bltx:note>
  </bltx:entry>
```

这种简单映射下的如下一些例外需要注意。

D.2.1 关键词别名

标注关键词别名设置如下:

```
    <bltx:ids>
      <bltx:key>alias1</bltx:key>
      <bltx:key>alias2</bltx:key>
    </bltx:ids>
```

对应于 BibTeX 格式为:

```
@book{key1,
  IDS = {alias1,alias2}
}
```

D.2.2 姓名

为了规范biblatex 的姓名处理能力，biblatexml 中的姓名配置更为复杂。用户需要更了解姓名成分，从而可以在更大范围上处理不同类型的姓名和姓名成分。biblatexml 格式中的一个姓名类似于:

```
<bltx:names type="author" morenames="1" useprefix="true">
  <bltx:name gender="sm">
    <bltx:namepart type="given">
      <bltx:namepart initial="J">John</bltx:namepart>
      <bltx:namepart initial="A">Arthur</bltx:namepart>
    </bltx:namepart>
    <bltx:namepart type="family">Smith</bltx:namepart>
    <bltx:namepart type="prefix" initial="v">von</bltx:namepart>
  </bltx:name>
  <bltx:name useprefix="false">
    <bltx:namepart type="given">
      <bltx:namepart>Raymond</bltx:namepart>
    </bltx:namepart>
    <bltx:namepart type="family">Brown</bltx:namepart>
  </bltx:name>
</bltx:names>
```

姓名列表域包含在names元素中，该元素具有type属性给出姓名列表的名称。需要注意:

- morenames可选属性的作用类似于 BibTeX 数据源格式中姓名列表末尾的‘and others’ 字符串。
- 注意useprefix可选参数可以在姓名列表或者单个的姓名层级设置。这在 BibTeX 数据源中是不可能的。
- 单个姓名可以具有可选的gender属性，但必须是在数据模型‘gender’ 常量列表中定义的值。目前标准样式并不使用，但在需要的时候可以在biblatex 姓名格式中使用。
- 一个姓名列表由一个或多个name元素构成。
- 每个姓名都由具有数据模型‘nameparts’ 常量定义的type的姓名成分构成。
- 每个姓名成分可能具有可选的initial属性，用于显式的指定姓名成分的首字母。如果不给出该选项，biber 将尝试自动确定姓名成分的首字母。
- 姓名成分还可以包含下层姓名成分，所以可以处理混合姓名。

不考虑biblatexml 独有的功能，该数据对应的 BibTeX 的数据格式类似于:

```
AUTHOR = {von Smith, John Arthur and Brown, Raymond and others}
```

D.2.3 列表

数据源可以根据域内列表中元素是否超过一个，采用两种方式将域 (见 § 2.2.1) 列出。

```
<bltx:publisher>London</bltx:publisher>
<bltx:location>
  <bltx:item>London</bltx:item>
  <bltx:item>Moscow</bltx:item>
</bltx:location>
```

D.2.4 范围

数据源范围域 (见 § 2.2.1) 可以表示为:

```
<bltx:pages>
  <bltx:item>
    <bltx:start>1</bltx:start>
    <bltx:end>10</bltx:end>
  </bltx:item>
  <bltx:item>
    <bltx:start>30</bltx:start>
    <bltx:end>34</bltx:end>
  </bltx:item>
</bltx:pages>
```

一个范围域是各范围的列表，每个范围都有自身的`item`。一个范围项都有一个`start`元素和一个可选的`end`元素，因为范围可以无终点。

D.2.5 日期

数据源日期域 (见 § 2.2.1) 可以根据是否构成一个日期范围采用两种方式表示:

```
<bltx:date>1985-04-02</bltx:date>
<bltx:date type=”event”>
  <bltx:start>1990-05-16</bltx:start>
  <bltx:end>1990-05-17</bltx:end>
</bltx:date>
```

一个日期元素的`type`属性对应于数据模型中定义的日期的某一特定类型。

D.2.6 关联条目

关联条目定义如下:

```
    <bltx:related>
      <bltx:item type="reprint"
                 ids="rel1,rel2"
                 string="Somestring"
                 options="skipbiblist"/>
    </bltx:related>
```

这对应于 BibTeX 格式:

```
@book{key1,
  RELATED         = {rel2,rel2},
  RELATEDTYPE     = {reprint},
  RELATEDSTRING   = {Somestring},
  RELATEDOPTIONS  = {skipbiblist}
}
```

如 § 4.5.1所述，string和options和属性是可选的。

E 选项的作用范围

下表概括了各种包选项的作用范围 (全局或者具体类型或者具体条目):

Option	Scope			
	Load-time	Global	Per-type	Per-entry
abbreviate	•	•	–	–
alldates	•	•	–	–
alldatesusetime	•	•	–	–
alltimes	•	•	–	–
arxiv	•	•	–	–
autocite	•	•	–	–
autopunct	•	•	–	–
autolang	•	•	–	–
backend	•	–	–	–
backref	•	•	–	–
backrefsetstyle	•	•	–	–
backrefstyle	•	•	–	–
bibencoding	•	•	–	–
bibstyle	•	–	–	–
bibwarn	•	•	–	–
block	•	•	–	–
citecounter	•	•	–	–
citereset	•	•	–	–
citestyle	•	–	–	–
citetracker	•	•	–	–
clearlang	•	•	–	–
datamodel	•	–	–	–
dataonly	–	–	•	•
date	•	•	–	–
labeldate	•	•	–	–
<datetype>date	•	•	–	–

Option	Scope			
	Load-time	Global	Per-type	Per-entry
dateabbrev	•	•	–	–
datecirca	•	•	–	–
dateera	•	•	–	–
dateerauto	•	•	–	–
dateuncertain	•	•	–	–
datezeros	•	•	–	–
defernumbers	•	•	–	–
doi	•	•	–	–
eprint	•	•	–	–
<namepart>inits	•	•	–	–
gregorianstart	•	•	–	–
hyperref	•	•	–	–
ibidtracker	•	•	–	–
idemtracker	•	•	–	–
indexing	•	•	•	•
isbn	•	•	–	–
julian	•	•	–	–
labelalpha	•	•	•	–
labelnamefield	–	–	–	•
labelnumber	•	•	•	–
labeltitle	•	•	•	–
labeltitlefield	–	–	–	•
labeltitleyear	•	•	•	–
labeldateparts	•	•	•	–
labeltime	•	•	–	–
labeldateusetime	•	•	–	–
<datetype>time	•	•	–	–
<datetype>dateusetime	•	•	–	–
language	•	•	–	–
loadfiles	•	•	–	–
loccittracker	•	•	–	–
maxalphanames	•	•	•	•
maxbibnames	•	•	•	•
maxcitenames	•	•	•	•
maxitems	•	•	•	•
maxnames	•	•	•	•
maxparens	•	•	–	–
mcite	•	–	–	–
minalphanames	•	•	•	•
minbibnames	•	•	•	•
mincitenames	•	•	•	•
mincrossrefs	•	•	–	–
minxrefs	•	•	–	–
minitems	•	•	•	•
minnames	•	•	•	•
natbib	•	–	–	–
noinherit	–	–	–	•
notetype	•	•	–	–
opcittracker	•	•	–	–
openbib	•	•	–	–
pagetracker	•	•	–	–
parentracker	•	•	–	–
punctfont	•	•	–	–
refsection	•	•	–	–
refsegment	•	•	–	–
safeinputenc	•	•	–	–

Option	Scope			
	Load-time	Global	Per-type	Per-entry
seconds	•	•	–	–
singletitle	•	•	•	–
skipbib	–	–	•	•
skipbiblist	–	–	•	•
skiplab	–	–	•	•
sortcase	•	•	–	–
sortcites	•	•	–	–
sorting	•	•	–	–
sortnamekeyscheme	–	–	–	•
sortlocale	•	•	–	–
sortlos	•	•	–	–
sortupper	•	•	–	–
style	•	–	–	–
terseinits	•	•	–	–
texencoding	•	•	–	–
timezeros	•	•	–	–
timezones	•	•	–	–
uniquelist	•	•	•	•
uniquename	•	•	•	•
uniquetitle	•	•	•	–
uniquebaretitle	•	•	•	–
uniquework	•	•	•	–
uniqueprimaryauthor	•	•	–	–
url	•	•	–	–
useprefix	•	•	•	•
use<name>	•	•	•	•

F 更新历史

更新历史是与宏包用户相关的一些更新列表。不影响用户接口或者宏包行为的更技术性的更新并未在列。更多的技术细节可以参考CHANGES.org文件。右侧的数字表示本手册中的相关节号。

3.7 2016-12-08

修正了 \bibdateeraprefix的默认设置， 4.10.2

增加了 \DeclareSortInclusion 4.5.6

增加了 \relateddelim<relatedtype> 3.10.1

3.6 2016-09-15

修正了文档的一些内容，修复了 labeldate 本地化字符串相关的一个错误

3.5 2016-09-10

增加\ifuniquebaretitle判断 4.6.2

为\labelnamesource和\labeltitlesource增加说明 4.2.4.1

增加\bibdaterangesep 3.10.3

为\DeclareSourcemap增加refsection选项 4.5.3

为继承设置增加suppress选项， 4.5.11

增加\ifuniquework 4.6.2

修改\DeclareStyleSourcemap可多次使用 4.5.3

增加\forcezerosy 和 \forcezerosmdt 4.10.4

修改\mkdatezeros为\mkyearzeros, \mkmonthszeros 和\mkdayzeros 4.10.4

为所有姓名列表域增加namehash和fullhash 4.2.4.1

为所有姓名成分推广giveninits选项， 3.1.2.3

为\DeclareSortingNamekeyScheme增加inits选项 4.5.6

增加\DeclareLabelalphaNameTemplate 4.5.5

增加完整的EDTF的 0 层次和 1 层次的规则用于解析和打印时间 2.3.8

修改日期完全符合EDTF0 层次和 1 层次规则，相关判断和本地化字符串 . 2.3.8

增加timezeros 3.1.2.1

增加mktimezeros 4.10.4

修改iso8601为edtf 3.1.2.1

增加\DeclareUniquenameTemplate 4.11.4

移除尝试 RIS 支持的内容

在 BibTeX 数据源中，可以对具体姓名列表和具体姓名设置sortnamekeyscheme 和 useprefix。

增加\DeclareDelimcontextAlias 3.10.2

增加爱沙尼亚本地化字符串 (Benson Muite)

命名了参考文献环境 3.7.10

在源映射中增加notfield映射步 4.5.3

3.4 2016-05-10

增加\ifcrossrefsource和\ifxrefsource 4.6.2

增加数据注释功能 3.6

增加包选项minxrefs 3.1.2.1

增加\ifuniqueprimaryauthor和相关全局选项 4.6.2

增加\DeprecateField, \DeprecateList和\DeprecateName 4.4.1

增加\ifcaselang 4.6.2

增加\DeclareSortTranslit 4.5.6

增加uniquetitle test 4.6.2

增加\namelabeldelim 3.10.1

为\assignrefcontext*宏增加带星号的版本 3.7.10

新的上下文敏感的分隔符接口 3.10.2

将prefixnumbers选项移到\newrefcontext并重命名为labelprefix 3.7.10

增加\DeclareDatafieldSet 4.5.2

3.3 2016-03-01

增加参考文献环境自动分配的新宏 3.7.10

增加了biblatexml 格式的说明 . . . . . D

增加了biblatexml 格式的源映射说明和示例 4.5.3

修改姓名格式用于规范姓名成分 4.4.2

useprefix可以在biblatexml 数据源中根据具体姓名列表和姓名进行设置

增加数据源映射选项用于动态创建新条目并遍历映射步 4.5.3

增加noalphaothers并增强了\DeclareLabelalphaTemplate中的姓名范围选择 4.5.5

增加\DeclareDatamodelConstant 4.5.4

重命名了firstinits and sortfirstinits

增加\DeclareSortingNamekeyScheme 4.5.6

移除尝试为biber 提供对 endnote 和 zoterordf 支持的内容

增加\nonameyeardelim 3.10.1

增加\extpostnotedelim 3.10.1

3.2 2015-12-28

为\DeclareLabelalphaTemplate增加pstrwidth和pcompound 4.5.5

增加\AtEachCitekey 4.10.6

3.1 2015-09

增加\DeclareNolabel 4.5.5

增加\DeclareNolabelwidthcount 4.5.5

3.0 2015-04-20

完善丹麦语 (Jonas Nyrup) 和西班牙语 (ludenticus) 翻译

由biblatex 代替biber 来实现labelname and labeltitle，实现更大灵活性和可扩展性。

增加\entryclone映射动作用于在数据源映射过程中复制条目 4.5.3

增加\pernottype实现的排除具体条目类型的数据源映射 4.5.3

增加范围计算命令\frangelen 4.6.4

增加参考文献环境功能 3.7.10

数据模型中的姓名列表自动创建\ifuse<name>判断和布尔值 3.1.3.1 and 4.6.2

中英文术语对照表

计数器	counter	143
间距因子	space factor	228
参考文献标注样式	citation style	153
参考文献环境	refcontext	293
参考文献著录样式	Bibliography Style	134
域	field	5
尺寸/长度	length	136
条目	entry	131
条目类型	entry type	5
样式	style	5

G 后记

之所以摘译 biblatex，一是出于对 keep texing 的兴趣，二是考虑到中文资料里总体介绍 latex 的资料其实较多，反而是在一些专项部分中文资料较少，看宏包的英文说明当然没有问题，但有的文档内容长达几百页，一般用户真心没有精力去看，即便是找一些自己需要的功能也比较麻烦，所以考虑对参考文献的 biblatex 宏包文档进行摘译，是希望能在这一方面有所贡献。关于这一点 Wenbo 兄也很有同感，在几年前 biblatex 还是 2.x 版本的时候他就深入研究了 biblatex 并翻译了 1–4 节很多内容。我在 biblatex-gb7714-2015 样式宏包中提议翻译 biblatex 文档之后，我们决定合作来推进这个事情，尽管都只有部分业余时间可以利用，但我们认为只要有空就积累一点，那么终有完成的时候。

如同我在 biblatex-gb7714-2015 样式宏包说明文档中介绍的那样，biblatex 宏包具有很多强大功能比如参考文献表划分、文献集、样式定制、动态数据处理等等，在科技论文或书籍写作中特别有用 (尤其是在对参考文献著录和标注格式有特殊要求的情况下)。可以说，biblatex 是 latex 文档写作中参考文献问题的完整解决方案，也一定程度上代表了这一方面的未来趋势。总之，本项目的总体任务是完成 biblatex 宏包说明文档关键内容的摘译，希望能对使用 biblatex 和对参考文献样式有深度定制要求的用户有所帮助，当然也希望能使中文 latex 资料库更为全面和深入。需要说明的是，限于作者水平，其中难免存在一些错误和理解不到位的地方，欢迎批评指正，欢迎 @ 译者邮箱。最后感谢 CTEX 和 Latexstudio 论坛，感谢论坛上各位作者关于 biblatex 参考文献方面的工作分享和经验介绍。