

The `amsrefs` package

Michael Downes and David M. Jones
American Mathematical Society

Version 2.0, 2004/06/30

Contents

1	Introduction	3
2	Package options	3
3	More about the <code>\bib</code> command	4
3.1	Field names for the <code>\bib</code> command	4
3.2	Bibliography entry types	4
4	Customizing the bibliography style	5
5	Miscellaneous commands provided by the <code>amsrefs</code> package	6
6	Implementation	9
6.1	Overview	9
6.1.1	Normal \LaTeX processing of cites	9
6.2	How cites are processed by <code>amsrefs</code>	10
6.3	Data structures	11
6.4	Preliminaries	12
6.5	Utilities	13
6.6	Declaring package options	16
6.6.1	The <code>?</code> option	18
6.7	Loading auxiliary packages	18
6.7.1	The <code>lite</code> option	18
6.8	Key-value setup	18
6.8.1	Standard field names (the <code>bib</code> group)	19
6.8.2	Auxiliary properties (the <code>prop</code> group)	20
6.9	Bibliography type specifications	20
6.10	The standard bibliography types	23
6.11	The <code>biblist</code> environment	27
6.12	Processing bibliography entries	29
6.12.1	<code>\@bibdef</code> Implementations	30
6.12.2	<code>\bib@exec</code> Implementations	34
6.12.3	Resolving cross-references	35
6.12.4	Bib field preprocessing	39
6.12.5	Date setup	40
6.12.6	Language setup	41
6.12.7	Citation label setup	42

6.12.8	Printing the bibliography	45
6.13	Name, journal and publisher abbreviations	46
6.14	Processing <code>.ltb</code> files	46
6.15	Citation processing	49
6.15.1	The <code>\citesel</code> structure	49
6.15.2	The basic <code>\cite</code> command	50
6.15.3	Fancier <code>\cite</code> commands	55
6.15.4	The <code>\nocite</code> command	56
6.15.5	Citation sorting	57
6.15.6	Range compression	59
6.15.7	Munging the <code>.aux</code> file	62
6.15.8	Back references	63
6.15.9	<code>hyperref</code> and <code>showkeys</code> support	64
6.16	Lexical structure of names	64
6.16.1	Text accents	65
6.16.2	Text symbols	66
6.16.3	<code>\edef</code> -like macros for names	67
6.17	Name parsing	70
6.18	Extracting initials	74
6.18.1	The algorithm	75
6.18.2	The implementation	76
6.19	Generating alphabetic labels	82
6.19.1	The algorithm	82
6.19.2	The implementation	82
6.20	Generating short alphabetic labels	87
6.21	Formatting series	87
6.22	Formatting names and series of names	91
6.23	The <code>partial</code> field	98
6.24	Special formatting for other fields	98
6.25	BIB _T E _X support	103
6.26	Implementing package options	103
6.26.1	The <code>alphabetic</code> option	103
6.26.2	The <code>shortalphabetic</code> option	103
6.26.3	The <code>backrefs</code> option	104
6.26.4	The <code>citation-order</code> option	104
6.26.5	The <code>initials</code> option	104
6.26.6	The <code>jpa</code> option	105
6.26.7	The <code>logical-quotes</code> option	105
6.26.8	The <code>non-compressed-cites</code> option	105
6.26.9	The <code>non-sorted-cites</code> option	105
6.26.10	The <code>short-journals</code> option	105
6.26.11	The <code>short-publishers</code> option	105
6.26.12	The <code>short-months</code> option	106
6.26.13	The <code>y2k</code> option	106
6.26.14	The <code>bibtex-style</code> option	106
6.26.15	The <code>author-year</code> option	106

6.27	The <code>amsbst</code> package	113
------	---	-----

1 Introduction

The `amsrefs` package is a \LaTeX package for bibliographies that provides an archival data format similar to the format of `BibTeX` database files, but adapted to make direct processing by \LaTeX easier. The package can be used either in conjunction with `BibTeX` or as a replacement for `BibTeX`.

This document is written for anyone who wants to implement a new bibliography style for `amsrefs` or who is just curious about how the package is implemented. The reader should be familiar with the contents of the “User’s Guide to the `amsrefs` Package” [?Jones2004] (`amsrdoc.tex`).

For the publisher or implementor, the chief advantages of the `amsrefs` package are as follows:

Preservation of structure The internal structural information of the bibliography entries is not lost when they are imported from the database file into the \LaTeX document. This takes on its greatest significance when archiving documents in \LaTeX form or transmitting them to another user (such as a publisher).

Deferred formatting This means that the style of the bibliography can be readily changed without reimporting everything from the original database(s).

Setup requires only \LaTeX knowledge All bibliography setup can be done in \LaTeX ; learning another programming language (such as the one used in `BibTeX` `bst` files) is unnecessary.

2 Package options

In addition to the options documented in the user’s guide, there are a few additional options that were omitted either because they are obsolete or deprecated options included only for backwards compatibility or because they are still considered experimental and not yet ready for widespread use.

? Informational option. This causes `amsrefs` to display a pointer to the User’s Guide on the terminal and in the log file. (In previous versions, it displayed much more material, including a summary of package options.)

traditional-quotes, logical-quotes With the *traditional quotes* option (default), quotation marks produced by `\bibquotes` ($\$??$) fall outside of other punctuation, “like this,” whereas with the *logical quotes* option the order is reversed, “like this”.

beta, jpa Obsolete; these applied only to the beta version of the `amsrefs` package.

3 More about the `\bib` command

3.1 Field names for the `\bib` command

In addition to the fields discussed in the user's guide, the following fields are used internally:

fulljournal Used internally by `\DefineJournal`.

name Used internally by the `name` bibliography type and `\DefineName`.

transition A dummy field used inside `\BibSpecs` when we want to force an action unconditionally.

The following fields are included for backwards compatibility:

institution, school These are provided as aliases for `organization` for compatibility with `BIBTEX`.

place A synonym for `address`. In earlier versions of `amsrefs`, `place` was preferred and `address` was considered as an alias for `place`. However, this seemed like a gratuitous incompatibility with `BIBTEX` to me, so I have reinstated `address` as the primary field and `place` is now an undocumented alias.

The following fields are reserved for future use:

doi Digital Object Identifier

setup This is a special field that can be used to give arbitrary commands to be executed at the beginning of the current `\bib` entry, after all the fields have been read. The idea is that one can alter the formatting of an individual entry through this field, to handle special cases.

This is fully implemented, but I've been unable to think of any good examples of its use; so, I've decided to suppress it until such an example comes to light.

url Universal Resource Locator.

3.2 Bibliography entry types

The following additional entry types (or, really, pseudo-entry types) are used internally by `amsrefs`:

collection.article

proceedings.article

partial

conference

innerbook

name

nameLE

nameBE

nameinverted

publisher

The following are currently undocumented aliases for various of the standard types:

miscellaneous

periodical

4 Customizing the bibliography style

If you use the `amsrefs` package as is, the bibliography style you get is the kind of style customarily seen in AMS publications. The recommended way to get a different bibliography style is to write a \LaTeX package which loads the `amsrefs` package with `\RequirePackage` and then makes the desired changes by using suitable `\BibSpec` commands as explained below. Thus, the general form of the custom package will be

```
\ProvidesPackage{xyzbib}[2002/11/06 v1.28]
```

```
\RequirePackage{amsrefs}\relax
```

```
\BibSpec{article}{
  ...
}
```

```
\BibSpec{book}{
  ...
}
```

The interior formatting within entries is specified by `\BibSpec` commands, one for each entry type. To illustrate, let's look at an example style specification for entries of type `article`:

```
\BibSpec{article}{%
  +{}{\PrintAuthors} {author}
  +{,}{ \textit}      {title}
  +{,}{ }             {journal}
  +{}{ \textbf}      {volume}
  +{}{ \parenthesize} {date}
  +{,}{ }             {pages}
  +{,}{ }             {note}
  +{.}{ }             {transition}
  +{}{ }              {review}
}
```

It should be pretty obvious that each line specifies the formatting for a particular field. After reading the data for a particular `\bib` command, \LaTeX steps through the style specification and for each field listed, prints the field with the given formatting *if and only if the field has a nonempty value*. The `+` character at the beginning of each field specification must be followed by three arguments: the punctuation to be added if the field is nonempty; space and/or other material to be added after the punctuation; and the field name. It is permissible for the second part to end with a command that takes an argument, such as `\textbf`, in which case it will receive the field's value as its argument.

By defining a suitable command and using it here you can place material after the field contents as well as before; `\parenthesize` is an example of this.

The reason that the punctuation and the following space are specified separately is that between them there is a crucial boundary for line breaks. If you put a `\linebreak` command at the end of a field value, the break point will actually be carried onward to a suitable point after the next bit of punctuation (whose actual value may vary depending on which of the following fields is the first to turn up with a nonempty value).

The meaning of the `\parenthesize` command, supplied by `amsrefs`, should be obvious. The meaning of the `\PrintAuthors` command is a different story. But I don't think it is all that hard to understand. If we have two or more author names which were given separately, and we need to combine them into a conventional name list using commas and the word "and", then it would be nice if we had a command which could take a list of names and Do The Right Thing. And that is just what `\PrintAuthors` is.

The `rkeyval` package allows keys to be defined as additive: if the key occurs more than once, each successive value will be concatenated to the previous value, along with a prefix. The setup done by `amsrefs` for the `author` field is

```
\DefineAdditiveKey{bib}{author}{\name}
```

This means that if two names are given, as in

```
author={Bertram, A.},
author={Wentworth, R.},
```

then the final value of the `author` field seen when \LaTeX processes the style specification will be

```
\name{Bertram, A.}\name{Wentworth, R.}
```

The `transition` field in our `\BibSpec` example is a dummy field to be used when punctuation or other material must be added at a certain point in the bibliography without regard to the emptiness or non-emptiness of the fields after it. The `transition` field always tests as non-empty but has no printed content. So when you use it you always get the indicated punctuation and space at the indicated point in the list of fields. If it were the last thing in this `\BibSpec` example, it could serve just to put in the final period that is always wanted. But in AMS bibliographies, if a *Mathematical Reviews* reference is given, it is conventionally printed *after* the final period. Using the `transition` field as shown here ensures that the final period will be always printed, even when the `review` field is empty.

5 Miscellaneous commands provided by the `amsrefs` package

Most of the following commands are helper commands for use in `\BibSpec` statements. The others are intended for use in bibliography data.

`\parenthesize` This command adds parentheses around its argument. It is useful in `\BibSpec` statements because there is no special provision for adding material after the field value.

- `\bibquotes` This command is much like `\parenthesize` but it adds quotes around its argument and it has one other important difference: there are special arrangements to print the closing quote *after* a following comma or similar punctuation (unless the `amsrefs` package is invoked with the `logical-quotes` option, in which case `\bibquotes` puts the closing quote immediately after the quoted material).
- `\voltext` This is used to format volume numbers. By default, it precedes the volume number by “vol.”
- `\issuetext` This is used to format issue numbers. By default, it precedes the volume number by “no.”
- `\editiontext` This command produces “ed.” following an edition number. See `\PrintEdition` for more information.
- `\DashPages` This command is similar in spirit to `\voltext` but more complicated in its implementation. It takes one argument which is expected to contain one or more page numbers or a range of page numbers. The argument is printed with a prefix of “p.” if it seems to be a single page number, otherwise with a prefix of “pp.”
- `\tsup`, `\tsub`, `\tprime` These are for text subscripts and superscripts, with `\tprime` producing a superscript prime symbol. Unlike the standard `\textsuperscript` and `\textsubscript` functions provided by L^AT_EX, these do not use math mode at all.¹
- `\nopunct` This command causes following punctuation to be omitted if it is added with the internal function `\@addpunct`.
- `\PrintPrimary` This is a relatively complicated function that determines the “primary” contributors for an entry and formats them, or replaces them by `\sameauthors` if appropriate. It should be used when an entry type might have editors or translators instead of authors. It prefers authors over editors and editors over translators and generates a warning if there are no primary contributors.
- `\PrintAuthors` This is used to format the list of authors as the primary contributors for an entry type.
- `\PrintEditorsA` This is similar to `\PrintAuthors` but adds (ed.) or (eds.) following the editors.
- `\PrintEditorsB` This is similar to `\PrintEditorsA` but puts parentheses around the entire list of editors. It’s used by, for example, the `article` type to print the editors of a `proceedings` or `collection`.
- `\PrintEditorsC` Similar to `\PrintEditorsA` but precedes the editors by `Edited by`. It’s used when the editors should be treated as subsidiary contributors, rather than the primary contributor.
- `\PrintTranslatorsA` This is similar to `\PrintEditorsA` but adds (trans.) following the translators.

¹There is one drawback: If you don’t want to get the prime symbol for `\tprime` from the `cmsy` font, you will need to redefine `\tprime` in some suitable way.

- `\PrintTranslatorsB` This is similar to `\PrintEditorsB`. It's not currently used, but is provided for symmetry.
- `\PrintTranslatorsC` Similar to `\PrintEditorsC` but precedes the translators by `Translated by`.
- `\sameauthors` This is a function of one argument. If you use the default set of `\BibSpecs` from the `amsrefs`, `\sameauthors` is applied to the author name for a given `\bib` command if it matches exactly the author name of the preceding `\bib` command. Change the definition of `\sameauthors` if you don't want to get a bysame dash.
- `\bysame` This is a horizontal rule of length 3 em. The default definition of `\sameauthors` prints `\bysame` instead of the author names.
- `\Plural`, `\SingularPlural` These are helper functions that allow you to conditionally print singular or plural forms such as `(ed.)` or `(eds.)` depending on the number of names in the current name list. The definition of `\PrintEditorsA` reads, in part,
- ```
... (ed\Plural{s}.) ...
```
- `\PrintReviews` This is similar to `\AuthorList` but is used for printing (possibly multiple) MR numbers given in the `review` field.
- `\BibField` This is for more complicated programming tasks such as may be necessary for some `\BibSpecs`. It takes one argument, a field name, and yields the contents of that field for the current `\bib` entry.
- `\IfEmptyBibField` If one writes
- ```
\IfEmptyBibField{isbn}{A}{B}
```
- then the commands in A will be executed if the `isbn` field is empty, otherwise the commands in B.
- `\PrintEdition` If a bibliography entry has
- ```
edition={2}
```
- and the `\BibSpec` used `\PrintEdition` to handle this field, then the edition information will be printed as "2nd ed."—that is, the number is converted to cardinal form and "ed." is added (taken from `\editiontext`).
- `\CardinalNumeric` This provides the conversion to cardinal number form used by `\PrintEdition`.
- `\PrintDate`, `\PrintYear` These functions convert a date in canonical form (ISO 8601) to the form required by the current bibliography style. You can get your preferred date form by redefining these functions or by changing your `\BibSpec` statements to use another function of your own devising. The original definition of `\PrintDate` adds parentheses (as for the year of a journal article in normal AMS style), whereas the `\PrintYear` function simply prints the year without any additional material (as for a book's year of publication in normal AMS style).
- `\mdash`, `\ndash` These are short forms for `\textemdash` and `\textendash`, recommended instead of the more usual `---` and `--` notation. From the `textcmds` package.

et cetera ... [mjd,2002-01-03] See the .dtx files for further possibilities that I have not managed to get properly documented yet!

## 6 Implementation

### 6.1 Overview

It will be a while yet before we get to any actual code. First we need to understand what the code needs to accomplish in order to provide the user interface described above in a way that is as compatible as possible with existing L<sup>A</sup>T<sub>E</sub>X mechanisms.

#### 6.1.1 Normal L<sup>A</sup>T<sub>E</sub>X processing of cites

**First L<sup>A</sup>T<sub>E</sub>X pass** Various commands are written to the .aux file that are mostly used by BibT<sub>E</sub>X.

1. A `\cite{moo}` command writes one line to the .aux file: `\citation{moo}`. This indicates to BibT<sub>E</sub>X that it should include ‘moo’ in the list of cited items to be searched for. The `\cite` command also checks to see if `\b@{moo}` contains the corresponding citation label, but since this is the first pass, the label won’t be known yet, so L<sup>A</sup>T<sub>E</sub>X emits an ‘Undefined citation’ warning and prints a placeholder (i.e., ???) instead of the citation label.
2. A `\bibliographystyle{har}` command writes one line to the .aux file: `\bibstyle{har}`. This indicates to BibT<sub>E</sub>X that it should use `har.bst` to determine the style for sorting and formatting the bibliography items.
3. A `\bibliography{hij,klm,...}` command writes one line to the .aux file: `\bibdata{hij,klm,...}`. This indicates to BibT<sub>E</sub>X that it should look in `hij.bib, klm.bib, ...` for bibliographic data. The `\bibliography` also tries to input the .bbl file, but on the first pass it won’t exist yet.

On the first pass all `\cite`’s normally are reported as undefined because the .bbl file has not yet been created.

**BibT<sub>E</sub>X pass** For a document named `xyz.tex`, the command `bibtex xyz` is used to invoke BibT<sub>E</sub>X. It looks in `xyz.aux` to find the citation information written there by L<sup>A</sup>T<sub>E</sub>X. For each `\citation` line, BibT<sub>E</sub>X searches for a corresponding entry in the specified .bib files and formats it. The entire list is then sorted in whatever way dictated by the bibliography style, and written out to the file `xyz.bbl`. This normally produces entries that look something like:

```
\bibitem{BGL} P. Busch, M. Grabowski and P. J. Lahti:
{\it Operational Quantum Physics.}
Springer Verlag, New York (1995).
```

**Second L<sup>A</sup>T<sub>E</sub>X pass** Now the .bbl file exists and contains some `\bibitem` commands. At `\begin{document}`, L<sup>A</sup>T<sub>E</sub>X reads the .aux file, hoping to find some `\bibcite` commands, but it will not find them until the next time around. `\citation`, `\bibstyle`, and `\bibdata` commands in the .aux file are simply ignored by L<sup>A</sup>T<sub>E</sub>X. Then L<sup>A</sup>T<sub>E</sub>X proceeds to typeset the body of the document.

1. Instances of `\cite` still print question marks.
2. The `\bibliography` command causes  $\LaTeX$  to input `xyz.bbl` and typeset its contents.
3. A `\bibitem{moo}` command writes one line to the `.aux` file: `\bibcite{moo}{9}`, where 9 is the current item number.
4. A `\bibitem[Moody]{moo}` command writes one line to the `.aux` file: `\bibcite{moo}{Moody}`, using the supplied label instead of a number.

**Third  $\LaTeX$  pass** Now the `.aux` file contains some `\bibcite` commands. Once again,  $\LaTeX$  reads the `.aux` file when it reaches `\begin{document}`.

1. A `\bibcite{moo}{Moody}` causes  $\LaTeX$  to define `\b@moo` with ‘Moody’ as the replacement text.
2. If two `\bibcite` commands have the same citation key,  $\LaTeX$  gives a warning message. This happens at `\begin{document}`, during the reading of the `.aux` file.
3. Instances of `\cite` in the body of the document will print the appropriate labels obtained from the `.aux` file.
4. If there are any `\cite` commands for which the `.aux` file did not have a `\bibcite` command,  $\LaTeX$  will give an ‘Undefined citation’ warning. This often happens if the `.aux` file is incomplete due to a  $\TeX$  error on the preceding pass.

## 6.2 How cites are processed by `amsrefs`

In order to support its additional features (e.g., author-year citations and the `backrefs` option), the `amsrefs` package stores additional information for each cite in the macro `\b@whatever`. Instead of simply using the defined or undefined status of this macro to trigger the standard warnings, we add some boolean flags to allow us to discriminate more finely what the current situation is.

- Each time an item is cited in the body of the document, a `backref` entry is added to the info of that item. The `backref` info is the current page and section location. Section location is a bit hard to get right without better support from the document class. So we provide a hook to allow it to work better when the support is there.
- When a cite occurs, if the info is undefined then a warning is issued and the info structure is created. A `\citation` command and a `\citedest` command (providing `backref` info) are written to the `.aux` file. Because the `backref` info includes page number, it has to be a non-immediate write. An undefined info structure would normally happen only on a first pass when no `.aux` file exists, or when a new cite is added. I.e., when the corresponding `\citation` command is not yet present in the `.aux` file.
- When a `\citation` command occurs in the `.aux` file, it initializes the info structure if necessary, setting the “bib-info-present” flag to 0.
- When a `\citedest` command occurs in the `.aux` file, it initializes the info structure if necessary—but this shouldn’t happen: if the corresponding

`\citation` command did not already get processed, then something is wrong. So normally, the `\citedest` command merely needs to add its backref info to the existing info structure.

- When a `\babcite` command occurs in the `.aux` file, it will normally find that `\b@whatever` is already defined, if the bibliography occurs after all the `\cite` commands. What it must do is fill in the appropriate blank slots in the info structure set up by a previous `\citation` command.
- The `.aux` file is actually processed two times, once at the beginning of the document and once at the end. In the latter case, `\babcite` should give a warning if the backref-list is empty, since that means there were no `\cite` commands for the given key.
- When processing the bibliography: The `\bib` command needs to check if it is using a key that is already used by another `\bib` command.

We therefore have

```
\b@xyz -> \citesel 00{label}{year}{backref-list}
```

where the first 0 is replaced by 1 if there has already been another citation for the same key earlier in the document (some citation styles use abbreviated forms for all instances after the first), and the second 0 is replaced by 1 if the same key was already used by an earlier `\bib` command.

Because the backref-list often includes page number information, it cannot be built on the fly as we go along; instead we have to write the information to the `.aux` file and read it in at the beginning of the next run.

If there was no `\babcite` in the `.aux` file for a given key, then the info is

```
\b@xyz -> \citesel 00{}{}{backref-list}
```

If there was neither `\citation` nor `\babcite` in the `.aux` file for a given key, then the `\cite` command should find that `\b@xyz` is undefined.

If the author-year option is in effect, the “label” contains the author last names instead of a label:

```
\b@xyz -> \citesel 00{\name{Smith}\name{Jones}}{...}{...}
```

Full name information is included in the data because some citation styles give full names at the first citation and abbreviated forms for subsequent instances.

### 6.3 Data structures

The result of scanning the key/value pairs of a `\bib` command is an assignment statement for `\rsk@toks`. (Cf. the `rkeyval` package.) For example, consider the entry

```
\bib{miller83}{article}{
 author={Miller, G.},
 title={Eine Bemerkung zur Darstellung von Polynomen "\{u}ber
 Verb\{a}nden"}*{language={german}},
 journal={J. Math. Sent.},
 volume={10},
 year={1983},
 pages={26\ndash 30},
}
```

The scanned result is to assign

```
\global\rsk@toks{%
 \set:bib'author{Miller, G.}{}%
 \set:bib'title{Eine Bemerkung zur Darstellung von Polynomen
 \{"u}ber Verb\{"a}nden}{language={german}}%
 \set:bib'journal{J. Math. Sent.}{}%
 \set:bib'volume{10}{}%
 \set:bib'year{1983}{}%
 \set:bib'pages{26\ndash 30}{}%
}
```

The code in the last arg of `\RestrictedSetKeys` then invokes `\bib@exec` to do something with the value of `\rsk@toks`.

```
\bib@exec{miller83}{\the\rsk@toks}{\setbib@article}{}
```

## 6.4 Preliminaries

```
1 (*pkg)
```

Standard declaration of package name and date.

```
2 \NeedsTeXFormat{LaTeX2e}[1995/12/01]
```

```
\amsrefs@warning@nl
```

```
3 \def\amsrefs@warning@nl{\PackageWarningNoLine{amsrefs}}
```

Backward handling for beta version.

```
4 \ifpackagewith{amsrefs}{beta}{%
```

```
5 \amsrefs@warning@nl{The beta option is deprecated^^J%
```

```
6 and will be removed in a future release of amsrefs}
```

```
7 \expandafter\edef\csname opt@amsrbeta.sty\endcsname
```

```
8 {\@ptionlist{amsrefs.sty}}%
```

```
9 \def\@currname{amsrbeta}%
```

```
10 \expandafter\let\csname amsrbeta.sty-h@k\endcsname\@empty
```

```
11 \def\@tempa{\input{amsrbeta.sty}\endinput}%
```

```
12 }{%
```

```
13 \let\@tempa\@empty
```

```
14 }
```

```
15 \@tempa
```

```
16 \IfFileExists{url.sty}{%
```

```
17 \RequirePackage{url}\relax
```

```
18 \@gobble
```

```
19 }{%
```

```
20 \@firstofone
```

```
21 }
```

```
22 {
```

```
23 \DeclareRobustCommand{\url}[1]{%
```

```
24 \def\@tempa{#1}%
```

```
25 \texttt{\@urlsetup $\expandafter\strip@prefix\meaning\@tempa$}%
```

```
26 }%
```

```
27 \def\@urlsetup{%
```

```
28 \check@mathfonts \textfont\@ne\the\font \textfont\z@\the\font
```

```

29 \@apply\@urlfix{\do\+\do\=\do\:\do\-\do\.\do\,\do\;}%
30 \@apply\@urlbreak{\do\&\do\/\do\?}%
31 }%
32 \def\@urlbreak#1{%
33 \mathcode'#1="8000
34 \begingroup \lccode'\~='#1 \lowercase{\endgroup \edef~}%
35 {\mathchar\number'#1\penalty\hyphenpenalty}%
36 }%
37 \def\@urlfix#1{%
38 \mathcode'#1='#1\relax
39 }%
40 }

41 \ifundefined{NormalCatcodes}{\RequirePackage{pcatcode}\relax}{}
42 \PushCatcodes\NormalCatcodes
43 \ProvidesPackage{amsrefs}[2004/06/07 v1.71]
44 %% WARNING WARNING WARNING: Catcode of apostrophe ' is letter
45 %% throughout this file.
46 \catcode'\'=11 % letter

```

## 6.5 Utilities

Some of these useful functions are also found in AMS document classes.

`\after@deleting@token` Similar in concept to `\afterassignment`, except it deletes the next token in the stream before putting its argument back into the input. Useful for skipping past tokens during parsing.

```

47 \def\after@deleting@token#1{%
48 \afterassignment#1%
49 \let\@let@token= % Don't delete this space!
50 }

```

`\@ifempty` Some frequently used tests for empty arguments. Note that an argument consisting entirely of spaces (e.g., `\@ifempty{\_}`) counts as empty.

`\@ifnotempty`

```

51 \long\def\@ifempty#1{\@xifempty#1@..\@nil}
52
53 \long\def\@xifempty#1#2@#3#4#5\@nil{%
54 \ifx#3#4\@xp\@firstoftwo\else\@xp\@secondoftwo\fi
55 }
56
57 \long\def\@ifnotempty#1{\@ifempty{#1}{}}

```

`\macrotext`

```

58 \def\macrotext{\expandafter\strip@prefix\meaning}

```

`\vdef` “Verbatim” def.

```

59 \def\vdef#1#2{%
60 \def#1{#2}%
61 \edef#1{\macrotext#1}%
62 }

```

`\auto@protect` Sometimes it's convenient to render a given control sequence unexpandable for a time. `\auto@protect` provides a way to do that.<sup>2</sup>

An earlier version of this code read simply `\let#1\relax` but that had the disadvantage of making all `\auto@protected` macros compare equal via `\ifx`. This version allows macros to keep their identities under comparisons.

```
63 \def\auto@protect#1{\def#1{\@nx#1}}
```

`\g@undef` Globally undefine a control sequence.

```
64 \def\g@undef#1{\global\let#1\relax}
```

`\@concat` Concatenate onto the end of a token list. Expands everything.

```
65 \def\@concat#1#2{\edef#1{#1#2}}
```

`\add@toks@` This saves a few tokens of main memory and a lot of typing.

```
66 \def\add@toks@{\addto@hook\toks@}
```

`\@lappend` Append an element to a `\do`-delimited list. As long as the element to be appended (`#2`) is a single token, nothing is expanded. If it contains multiple tokens, all tokens after the first will be expanded.

```
67 \def\@lappend#1#2{%
68 \begingroup
69 \def\do{\@nx\do\@nx}%
70 \edef\@tempa{\def\@nx#1{#1\do#2}}%
71 \@xp\endgroup
72 \@tempa
73 }
```

`\@apply` Apply a macro to each element of a `\do`-delimited list.

```
74 \def\@apply#1#2{%
75 \let\do#1%
76 #2%
77 }
```

`\get@numberof` This is a generic macro for counting the number of elements in a L<sup>A</sup>T<sub>E</sub>X-style list. The first argument is a `\count` register that will receive the final count; the second argument is the control sequence that separates elements of the list, and the third argument is the list itself. So, for example,

```
\get@numberof\@tempcnta\do\dospecials
```

would count the number of special characters in `\dospecials` and store the number in `\@tempcnta`.

```
78 \def\get@numberof#1#2#3{%
79 \begingroup
80 \def#2{\advance\@tempcnta\@ne \gobble}%
```

---

<sup>2</sup>There really should be a special name for macros that, like `\auto@protect`, take a control sequence as an argument and redefine that control sequence in order to achieve some special effect. Pending happier inspiration, I'm going to call them "wrapper" macros.

```

81 \@tempcnta\z@
82 #3\relax
83 \edef\@tempb{#1=\the\@tempcnta\relax}%
84 \@xp\endgroup
85 \@tempb
86 }

```

`\safe@set` This is a quick and dirty way of extracting an integer prefix from a string and assigning it to a counter. If the string does not begin with an integer, the counter receives the value 0. The suffix after the integer prefix is discarded. (But bad things will happen if the string contains the token `\@nil`.)

```

87 \def\safe@set#1#2{%
88 \afterassignment\@nilgobble
89 #1=0#2\relax\@nil
90 }

```

`\@chomp` Vaguely reminiscent of Perl's `chomp` function, which removes a substring from the end of a variable, but ours works with tokens (more-or-less) and takes the substring to be removed as its second argument. Note the use of `\@empty` to anchor the chomped substring to the end of the string. Note also that the second argument will be fully expanded during the chomping.

```

91 \def\@chomp#1#2{%
92 \begingroup
93 \toks@\@emptytoks
94 \def\@chomper##1##2#2\@empty##3\@nil{%
95 \ifx\@let@token\bgroup
96 \toks@{##1}##2}%
97 \else
98 \toks@{##1##2}%
99 \fi
100 }%
101 \@xp\chomp@ #1\@empty#2\@empty\@nil
102 \edef\@tempa{\def\@nx#1\@xp{\the\toks@}}%
103 \@xp\endgroup
104 \@tempa
105 }

```

`\chomp@` Before passing control to `\@chomper`, we peek ahead at the next token in the stream. That way, if the next token is an open brace, we know we need to surround `\@chomper`'s first argument with braces. Unfortunately, this might still remove braces from the second argument, but I think that's ok for our purposes.

```

106 \def\chomp@{%
107 \futurelet\@let@token
108 \@chomper
109 }

```

`\amsrefs@warning`

```

110 \def\amsrefs@warning{\PackageWarning{amsrefs}}

```

```

\amsrefs@error
111 \def\amsrefs@error{\PackageError{amsrefs}}

\MessageBreakNS This suppresses the leading space in \on@line in error and warning messages.
112 \def\MessageBreakNS{\MessageBreak\romannumeral'\^^@}

\@addpunct The \@addpunct function is defined by AMS document classes and the amsgen
package. But if we find it undefined we had better define it.
113 \ifundefined{@addpunct}{%
114 \def\@addpunct#1{%
115 \relax\ifhmode
116 \ifnum\spacefactor>\@m \else#1\fi
117 \fi
118 }
119 \def\frenchspacing{%
120 \sfcode'\.1006
121 \sfcode'\?1005
122 \sfcode'\!1004
123 \sfcode'\:1003
124 \sfcode'\;1002
125 \sfcode'\,1001\relax
126 }
127 }{}}

\nopunct Omit any following punctuation that would normally be inserted by \@addpunct.
128 \providecommand{\nopunct}{\spacefactor \@nopunctsfcode}

\@nopunctsfcode
129 \def\@nopunctsfcode{1007 }

6.6 Declaring package options
We call the ifoption package to facilitate some option tests.
130 \RequirePackage{ifoption}[2000/02/15]
The sorted option is a no-op and is no longer documented. I'm only leaving
it here for backwards compatibility.
131 \DeclareExclusiveOptions{sorted,citation-order}
The alphabetic option corresponds to the standard alpha biblio style with
labels like Knu66 (three letters from name plus two digits of year). Maybe
should provide an alias LIIYY for this option. Numeric is the default since it is
commoner in AMS publications.
132 \DeclareExclusiveOptions{alphabetic,shortalphabetic,author-year,numeric}

y2k
133 \DeclareBooleanOption{y2k}

nobysame
134 \DeclareBooleanOption{nobysame}

```

The standard `abbrv` bibliography style uses abbreviations for month names and journal names, and first names of people are abbreviated to their initials. Since the second test bibliography that I tested with had unabbreviated month names but abbreviated journal names, perhaps it is a good idea to let these choices be specified separately.

```
135 \DeclareBooleanOption{short-journals}
136 \DeclareBooleanOption{short-publishers}
```

The `short-journals` and `short-publishers` options only affect journal and publisher names that are defined with `\DefineJournal` and `\DefinePublisher` commands.

```
137 \DeclareBooleanOption{short-months}
138 \DeclareBooleanOption{initials}
```

Nevertheless, it's to be expected that the preceding four options would typically be used together, so we provide a short-hand for requesting them all.

```
139 \DeclareOption{abbrev}{%
140 \@pass@ptions
141 \@currentx
142 {initials,short-months,short-journals,short-publishers}%
143 \@currname
144 }
```

In the bibliography, if a title or something is enclosed in quotes, should the closing quotes go inside the punctuation (logical position) rather than outside (traditional)? These options give you a choice.

```
145 \DeclareExclusiveOptions{traditional-quotes,logical-quotes}
```

A sequence of cites will be sorted and ranges of length three or greater will be compressed if these options so indicate. Note that the `non-sorted-cites` option automatically disables compression. This is probably a feature.

```
146 \DeclareExclusiveOptions{sorted-cites,non-sorted-cites}
147 \DeclareExclusiveOptions{non-compressed-cites,compressed-cites}
```

In the bibliography, print page numbers showing where each entry was cited.

```
148 \DeclareBooleanOption{backrefs}
```

Option for giving information about the available options:

```
149 \DeclareBooleanOption{?}
```

This option means to forgo loading of the `textcmds` and `mathscinet` packages.

```
150 \DeclareBooleanOption{lite}
```

This option can be used by later releases as a sign that fall-back adaptations need to be done.

```
151 \DeclareBooleanOption{beta}
```

This one is obsolete now.

```
152 \DeclareBooleanOption{jpa}
```

```

153 \DeclareBooleanOption{bibtex-style}
154 \ExecuteOptions{numeric,traditional-quotes,sorted-cites,compressed-cites}
155
156 \ProcessOptions\relax
157
158 \ProcessExclusiveOptions
159 \IfOption{backrefs}{%
160 \IfFileExists{hyperref.sty}{%
161 \RequirePackage{hyperref}[1999/07/08]
162 }{}%
163 \IfFileExists{backref.sty}{%
164 \RequirePackage{backref}[1999/05/30]
165 }{}%
166 }{}

```

### 6.6.1 The ? option

Note that in the following auxiliary package list, `getwidth` is not (yet) included.

```

167 \IfOption{?}{%
168 \typeout{^^J%
169 Documentation for the amsrefs package is found in amsrdoc.dvi^^J%
170 (or .pdf or .tex).
171 ^^J%
172 }%
173 }{}%

```

## 6.7 Loading auxiliary packages

Now, if these other packages make use of the `pcatcode` package like they should, then we don't need to make any fuss here about the special catcode of `'`. Just load the packages.

```

174 \RequirePackage{rkeyval}[2001/12/22]

```

### 6.7.1 The lite option

In my opinion, this is misguided, since `amsrefs` shouldn't be loading these packages to begin with. But it's too late to change it now.

```

175 \IfOption{lite}{% True? Then don't load the next two packages.
176 }{% False? OK, let's load them:
177 \RequirePackage{textcmds}[2001/12/14]
178 \RequirePackage{mathscinet}[2002/01/01]
179 }

```

## 6.8 Key-value setup

`\BibField` This provides easy access to individual fields for user-defined formatting functions.

```

180 \newcommand{\BibField}[1]{\csname bib'#1\endcsname}

```

`\IfEmptyBibField` A convenient partial application of `\rkvIfEmpty`.

```

181 \newcommand{\IfEmptyBibField}{\rkvIfEmpty{bib}}

```

### 6.8.1 Standard field names (the bib group)

And here are the predefined key names. You could always add some more if you needed them. Only worry is about compatibility if you want to share your data with other people.

`\fld@elt` We want the list macros used above to be unexpandable except when special  
`\name` processing is done. (It's not clear to me there's any real benefit to using these instead of just using `\do.—dmj`)

```
182 \let\fld@elt=?
183 \let\name=?
```

First the fields that could be repeated more than once in a single entry. Maybe publisher should be allowed to repeat also, for co-published works. But then need to worry about the address handling.

```
184 \DefineAdditiveKey{bib}{author}{\name}
185 \DefineAdditiveKey{bib}{editor}{\name}
186 \DefineAdditiveKey{bib}{translator}{\name}
187 \DefineAdditiveKey{bib}{contribution}{\fld@elt}
188 \DefineAdditiveKey{bib}{isbn}{\fld@elt}
189 \DefineAdditiveKey{bib}{issn}{\fld@elt}
190 \DefineAdditiveKey{bib}{review}{\fld@elt}
191 \DefineAdditiveKey{bib}{partial}{\fld@elt}

192 \DefineSimpleKey{bib}{address}
193 \DefineSimpleKey{bib}{book}
194 \DefineSimpleKey{bib}{booktitle}
195 \DefineSimpleKey{bib}{conference}
196 %\DefineSimpleKey{bib}{contributor}
197 \DefineSimpleKey{bib}{copula}
198 \DefineSimpleKey{bib}{date}
199 \DefineSimpleKey{bib}{doi}
200 \DefineSimpleKey{bib}{edition}
201 \DefineSimpleKey{bib}{eprint}
202 \DefineSimpleKey{bib}{fulljournal}
203 \DefineSimpleKey{bib}{hyphenation}
204 \DefineSimpleKey{bib}{institution}
205 \DefineSimpleKey{bib}{journal}
206 \DefineSimpleKey{bib}{label}
207 \DefineSimpleKey{bib}{language}
208 \DefineSimpleKey{bib}{name}
209 \DefineSimpleKey{bib}{note}
210 \DefineSimpleKey{bib}{number}
211 \DefineSimpleKey{bib}{organization}
212 \DefineSimpleKey{bib}{pages}
213 \DefineSimpleKey{bib}{part}
214 \DefineSimpleKey{bib}{place}
215 \DefineSimpleKey{bib}{publisher}
216 \DefineSimpleKey{bib}{reprint}
217 \DefineSimpleKey{bib}{school}
```

```

218 \DefineSimpleKey{bib}{series}
219 \DefineSimpleKey{bib}{setup}
220 \DefineSimpleKey{bib}{status}
221 \DefineSimpleKey{bib}{subtitle}
222 \DefineSimpleKey{bib}{title}
223 \DefineSimpleKey{bib}{translation}
224 \DefineSimpleKey{bib}{type}
225 \DefineSimpleKey{bib}{url}
226 \DefineSimpleKey{bib}{volume}
227 \DefineSimpleKey{bib}{xref}
228 \DefineSimpleKey{bib}{year}

```

The `transition` key is used when we want to insert punctuation or other material at a given point in the sequence unconditionally. The key appears to have a non-empty value to `\IfEmptyBibField`, but its value (expansion) is empty.

```
229 \DefineDummyKey{bib}{transition}
```

### 6.8.2 Auxiliary properties (the prop group)

```

230 \DefineSimpleKey{prop}{inverted}
231 \DefineSimpleKey{prop}{language}

```

## 6.9 Bibliography type specifications

`\BibSpec` Accumulate specification material in `\toks@`, then define `\setbib@TYPE` from it.

```

232 \newcommand{\BibSpec}[2]{%
233 \toks@ \@emptytoks
234 \@ifnotempty{#2}{%

```

The `\@ifnextchar` removes an optional `+` at the beginning of a specification. From then on, each time `\bibspeg@scan` is invoked, it expects to find four arguments. The four `\@emptys` appended to the specification (`#2`) below ensure that this is so.

```

235 \@ifnextchar+{\@xp\bibspeg@scan@gobble}{\bibspeg@scan}%
236 #2\@empty\@empty\@empty\@empty
237 }%
238 \@xp\edef\csname setbib@#1\endcsname{\the\toks@}%
239 }

```

`\bibspeg@scan` The `\bibspeg@scan` function scans one field specification from the second arg of `\BibSpec`. Each field specification has the form

```
+{punctuation}{prelim material}{field name}
```

Note however that because the initial `+` is stripped off by `\BibSpec` (see above), the actual order that `\bibspeg@scan` reads the field specification is

```
#1={punctuation} #2={prelim material} #3={field name} #4=+
```

where the fourth argument is actually expected to be either the `+` from the following specification, or one of the special `\@empty` tokens inserted by `\BibSpec`.

If it is neither of these special values, it means we have a malformed specification; so, we issue an error and then try to pick up where we left off.

```

240 \def\bibspec@scan#1#2#3#4{%
241 \add@toks@{\bib@append{#1}{#2}}%
242 \edef\@tempa{%
243 \toks@{\the\toks@ \exp\nx\cscname bib'#3\endcscname}%
244 }%
245 \@tempa
246 \ifx\@empty#4%
247 \exp@gobble % end the recursion
248 \else
249 \ifx +#4\else\bibspec@scan@error\fi
250 \fi
251 \bibspec@scan
252 }

```

`\bibspec@scan@error`

```

253 \def\bibspec@scan@error{\amsrefs@error{Bad BibSpec: Expected '+'}}

```

`\bib@append` The function `\bib@append` prints the value of a field, together with associated punctuation and font changes, unless the value is empty. Arg 1 is punctuation (that may need to be swapped with a preceding line break), arg 2 gives the space to be added after the punctuation, and possibly a function to be applied to the contents of arg 3, which is a macro containing the field value. So if we have `\moo` and `\bib'pages`, from `pages={21\ndash 44}`, then we want to arrange to call

```
\moo{21\ndash 44}
```

We don't want to simply call `\moo\bib'bar` because that makes it rather difficult for `\moo` to look at the contents of `\bib@bar`.

```

254 \def\bib@append#1#2#3{%
255 \ifx\@empty#3%
256 \else

```

*Known bug:* Need better error message here.

```

257 \ifx\relax#3%
258 \errmessage{#3=\relax}%
259 \else
260 \begingroup
261 \series@index\m@ne
262 \def\current@bibfield{#3}%
263 \@ifempty{#1}{%
264 \@temptokena{\ifnum\lastkern=\@ne\ignorespaces\fi #2}%
265 }{%
266 \@temptokena{\SwapBreak{#1}#2}%
267 }%
268 \toks@\exp{#3}%
269 \edef\@tempa{\the\@temptokena{\the\toks@}}%

```

```

270 \rkvIfAdditive#3}{}%
271 \get@current@properties
272 \select@auxlanguage
273 }%
274 \@tempa
275 \endgroup
276 \fi
277 \fi
278 }

```

`\select@auxlanguage`

```

279 \def\select@auxlanguage{%
280 \ifx\prop'language\@empty
281 \else
282 \xp\selectlanguage\@xp{\prop'language}%
283 \fi
284 }

```

`\erase@field` There are some fields that can appear in more than one place in a reference, depending on context. For example, if a book has an editor but no author, the editor appears at the beginning of the entry, but if the book has both an editor and an author, the editor appears at the end of the entry. A simple way to handle this is to “erase” the editor field after printing it, which is what `\erase@field` is for.

The obvious definition of `\erase@field` is

```
\def\erase@field#1{\global\let#1\@empty}
```

but that doesn’t work because the top-level value of `rkeyval` fields isn’t `\@empty`; instead, it contains a setter function used by `\RestrictedSetKeys` when processing a key-value list (see `\rkv@DSAK`, `\rsk@set@a` and `\rsk@set@b`).

On the other hand, rewriting the field locally won’t work either, since `\erase@field` will typically be executed inside the group established by `\bib@append`. Instead, we want to rewrite the value right after `\bib@append`’s group ends. One way to do this would be to keep a list of fields to be erased and have `\bib@append` iterate over the list after its `\endgroup`.

However, as long as the call to `\erase@field` is never nested within any deeper groups, it’s simpler just to use `\aftergroup`, which is what we’ll do (“Sufficient unto the day is the evil thereof” and all that).

```

285 \def\erase@field#1{%
286 \aftergroup\let\aftergroup#1\aftergroup\@empty
287 }

```

`\get@current@properties` This retrieves the auxiliary properties for the current field value, as defined by `\current@bibfield` and `\series@index`.

```

288 \def\get@current@properties{%
289 \begingroup
290 \xp\get@nth@property\@xp\@tempa\current@bibfield\series@index

```

```

291 \edef\@tempa{%
292 \@nx\RestrictedSetKeys}{prop}{%
293 \def\@nx\@tempa{\@nx\prop@reset \@nx\the\@nx\rsk@toks}%
294 }\@tempa}%
295 }%
296 \@tempa
297 \@xp\endgroup
298 \@tempa
299 }

```

`\BibSpecAlias` This is a `\def` rather than a `\let` because using `\let` would make `\BibSpecAlias` statements order-sensitive in a way that seems frequently to be a stumbling block to unwary package writers. But then we should probably do at least the simplest kind of infinite loop check.

```

300 \newcommand{\BibSpecAlias}[2]{%
301 \@xp\def\@xp\@tempa\@xp{\csname setbib@#1\@xp\endcsname}%
302 \@xp\ifx\csname setbib@#2\endcsname\@tempa
303 \amsrefs@error{%
304 Mirror alias #1->#2 not allowed (infinite loop)}\@ehc
305 \else
306 \@xp\def\csname setbib@#1\@xp\endcsname
307 \@xp{\csname setbib@#2\endcsname}%
308 \fi
309 }

```

## 6.10 The standard bibliography types

```

310 \BibSpec{article}{%
311 +{} { \PrintAuthors} {author}
312 +{,} { \textit} {title}
313 +{.} { } {part}
314 +{:} { \textit} {subtitle}
315 +{,} { \PrintContributions} {contribution}
316 +{.} { \PrintPartials} {partial}
317 +{,} { } {journal}
318 +{} { \textbf} {volume}

```

The date form is tricky depending on presence or absence of DOI.

```

319 +{} { \PrintDatePV} {date}
320 +{,} { \issuetext} {number}
321 +{,} { \eprintpages} {pages}
322 +{,} { } {status}
323 +{,} { \PrintDOI} {doi}
324 +{,} { available at \eprint} {eprint}
325 +{} { \parenthesize} {language}
326 +{} { \PrintTranslation} {translation}
327 +{;} { \PrintReprint} {reprint}
328 +{.} { } {note}
329 +{.} {} {transition}
330 +{} { \SentenceSpace \PrintReviews} {review}
331 }

```

```

332
333 \BibSpec{partial}{%
334 +{} {} {part}
335 +{:} { \textit} {subtitle}
336 +{,} { \PrintContributions} {contribution}
337 +{,} { } {journal}
338 +{} { \textbf} {volume}
339 +{} { \PrintDatePV} {date}
340 +{,} { \issuetext} {number}
341 +{,} { \eprintpages} {pages}
342 }
343
344 \BibSpec{contribution}{%
345 +{} {} {type}
346 +{} { by \PrintNameList} {author}
347 }
348
349 \BibSpec{book}{%
350 +{} { \PrintPrimary} {transition}
351 +{,} { \textit} {title}
352 +{.} { } {part}
353 +{:} { \textit} {subtitle}
354 +{,} { \PrintEdition} {edition}
355 +{} { \PrintEditorsB} {editor}
356 +{,} { \PrintTranslatorsC} {translator}
357 +{,} { \PrintContributions} {contribution}
358 +{,} { } {series}
359 +{,} { \voltext} {volume}
360 +{,} { } {publisher}
361 +{,} { } {organization}
362 +{,} { } {address}
363 +{,} { \PrintDateB} {date}
364 +{,} { } {status}
365 +{} { \parenthesize} {language}
366 +{} { \PrintTranslation} {translation}
367 +{;} { \PrintReprint} {reprint}
368 +{.} { } {note}
369 +{.} {} {transition}
370 +{} { \SentenceSpace \PrintReviews} {review}
371 }
372
373 \BibSpec{collection.article}{%
374 +{} { \PrintAuthors} {author}
375 +{,} { \textit} {title}
376 +{.} { } {part}
377 +{:} { \textit} {subtitle}
378 +{,} { \PrintContributions} {contribution}
379 +{,} { \PrintConference} {conference}
380 +{} { \PrintBook} {book}
381 +{,} { } {booktitle}

```

```

382 +{,} { \PrintDateB} {date}
383 +{,} { pp.~} {pages}
384 +{,} { } {status}
385 +{,} { \PrintDOI} {doi}
386 +{,} { available at \eprint} {eprint}
387 +{} { \parenthesize} {language}
388 +{} { \PrintTranslation} {translation}
389 +{;} { \PrintReprint} {reprint}
390 +{.} { } {note}
391 +{.} {} {transition}
392 +{} { \SentenceSpace \PrintReviews} {review}
393 }
394
395 \BibSpec{conference}{%
396 +{} {} {title}
397 +{} { \PrintConferenceDetails} {transition}
398 }
399
400 \BibSpec{innerbook}{%
401 +{,} { } {title}
402 +{.} { } {part}
403 +{:} { } {subtitle}
404 +{,} { \PrintEdition} {edition}
405 +{} { \PrintEditorsB} {editor}
406 +{,} { \PrintTranslatorsC} {translator}
407 +{,} { \PrintContributions} {contribution}
408 +{,} { } {series}
409 +{,} { \voltext} {volume}
410 +{,} { } {publisher}
411 +{,} { } {organization}
412 +{,} { } {address}
413 +{,} { \PrintDateB} {date}
414 +{.} { } {note}
415 }
416
417 \BibSpec{report}{%
418 +{} { \PrintPrimary} {transition}
419 +{,} { \textit} {title}
420 +{.} { } {part}
421 +{:} { \textit} {subtitle}
422 +{,} { \PrintEdition} {edition}
423 +{,} { \PrintContributions} {contribution}
424 +{,} { Technical Report } {number}
425 +{,} { } {series}
426 +{,} { } {organization}
427 +{,} { } {address}
428 +{,} { \PrintDateB} {date}
429 +{,} { \eprint} {eprint}
430 +{,} { } {status}
431 +{} { \parenthesize} {language}

```

```

432 +{} { \PrintTranslation} {translation}
433 +{;} { \PrintReprint} {reprint}
434 +{.} { } {note}
435 +{.} {} {transition}
436 +{} {\SentenceSpace \PrintReviews} {review}
437 }
438
439 \BibSpec{thesis}{%
440 +{} {\PrintAuthors} {author}
441 +{,} { \textit} {title}
442 +{:} { \textit} {subtitle}
443 +{,} { \PrintThesisType} {type}
444 +{,} { } {organization}
445 +{,} { } {address}
446 +{,} { \PrintDateB} {date}
447 +{,} { \eprint} {eprint}
448 +{,} { } {status}
449 +{} { \parenthesize} {language}
450 +{} { \PrintTranslation} {translation}
451 +{;} { \PrintReprint} {reprint}
452 +{.} { } {note}
453 +{.} {} {transition}
454 +{} {\SentenceSpace \PrintReviews} {review}
455 }

456 \BibSpecAlias{periodical}{book}
457 \BibSpecAlias{collection}{book}
458 \BibSpecAlias{proceedings}{book}
459 \BibSpecAlias{manual}{book}
460 \BibSpecAlias{miscellaneous}{book}
461 \BibSpecAlias{misc}{miscellaneous}
462 \BibSpecAlias{unpublished}{book}
463 \BibSpecAlias{proceedings.article}{collection.article}
464 \BibSpecAlias{techreport}{report}

\setbib@incollection
465 \edef\setbib@incollection{%
466 \@xp\@nx\csname setbib@collection.article\endcsname
467 }

\setbib@inproceedings
468 \edef\setbib@inproceedings{%
469 \@xp\@nx\csname setbib@collection.article\endcsname
470 }

Some more entry types for implementing abbreviations.
471 \BibSpec{name}{%
472 +{} {\PrintAuthors} {name}
473 }
474
475 \BibSpec{publisher}{%

```

```

476 +{,} { } {publisher}
477 +{,} { } {address}
478 }

```

### 6.11 The biblist environment

The `biblist` environment can be used with a section or chapter heading.

Use a standard L<sup>A</sup>T<sub>E</sub>X counter for numbering bibliography items.

```

479 \newcounter{bib}

```

`biblist`

```

480 \newenvironment{biblist}{%
481 \setcounter{bib}\z@
482 \@biblist
483 }{%
484 \@endbiblist
485 }

```

`biblist*`

```

486 \newenvironment{biblist*}{%
487 \@biblist
488 }{%
489 \@endbiblist
490 }

```

`\@biblist`

```

491 \newcommand\@biblist [1] [] {%
492 \stepcounter{bib@env}
493 \normalfont
494 \footnotesize
495 \labelsep .5em\relax
496 \list{\BibLabel}{%
497 \restore@labelwidth
498 \@maxlabelwidth\z@
499 \@nbrlisttrue
500 \def\@listctr{bib}%
501 \let\makelabel\bib@mklab
502 #1\relax
503 }%
504 \sloppy

```

Discourage page breaks within bibliography entries and disable them completely for entries that are less than four lines long.

```

505 \interlinepenalty\@m
506 \clubpenalty\@M
507 \widowpenalty\clubpenalty
508 \frenchspacing
509 \ResetCapSFCodes
510 }

```

`\@endbiblist` Change error for empty list (no items) to warning, to allow authors to leave their bibliography temporarily empty during writing:

```
511 \def\@endbiblist{%
512 \save@labelwidth
513 \def\@noitemerr{\@latex@warning{Empty bibliography list}}%
514 \endlist
515 }
```

`\@maxlabelwidth`

```
516 \newdimen\@maxlabelwidth
```

`\bib@mklab`

```
517 \def\bib@mklab#1{%
518 \settowidth\@tempdima{#1}%
519 \ifdim \@tempdima > \@maxlabelwidth
520 \global\@maxlabelwidth\@tempdima
521 \fi
522 #1\hfil
523 }
```

```
524 \newcounter{bib@env}
```

`\save@labelwidth`

```
525 \def\save@labelwidth{%
526 \if@filesw
527 \immediate\write\@auxout{%
528 \string\newlabel{[bibenv:\the\c@bib@env]}{\the\@maxlabelwidth}%
529 }%
530 \fi
531 }
```

`\restore@labelwidth`

```
532 \def\restore@labelwidth{%
533 \@xp\ifx \c@name r@[bibenv:\the\c@bib@env]\endcsname \relax
534 \resetbiblist{00}%
535 \else
536 \@xp\labelwidth\c@name r@[bibenv:\the\c@bib@env]\endcsname
537 \leftmargin\labelwidth
538 \advance\leftmargin\labelsep
539 \fi
540 }
```

`\ResetCapSFCodes` Presumably this is here because there has been a problem in the past with packages that change the `\catcodes` of capital letters.

```
541 \providecommand{\ResetCapSFCodes}{%
542 \count@='A
543 \def\@tempa{%
544 \sfcode\count@=\@m
545 \advance\count@\@ne
```

```

546 \ifnum\count@>'\Z\relax \expandafter\@gobble \fi
547 \@tempa
548 }%
549 \@tempa
550 }

```

`\CurrentBib` In case this is undefined sometimes.

```
551 \def\CurrentBib{??}
```

`\BibLabel`

```

552 \newcommand{\BibLabel}{%
553 [\hyper@anchorstart{cite.\CurrentBib}\relax\thebib\hyper@anchorend]%
554 }

```

`\resetbiblist`

```

555 \newcommand{\resetbiblist}[1]{%
556 \settowidth\labelwidth{\def\thebib{#1}\BibLabel}%
557 \leftmargin\labelwidth
558 \ifdim\labelwidth=\z@
559 \leftmargin=1em
560 \itemindent=-\leftmargin
561 \else
562 \advance\leftmargin\labelsep
563 \fi
564 }

```

## 6.12 Processing bibliography entries

There are several things one might want to do when a `\bib` entry is encountered:

1. Format and print it. This corresponds to the direct entry of bibliography items as described in section 2.1 of the users's guide.
2. Copy it into a `.bb1` file. This corresponds to the use of `\bibselect` and an external `.ltb` database as described in section 2.2 of the user's guide.
3. Store the full information in memory. This is done by `\bib*`.

`\bib` Here is where the rubber hits the road.

```

565 \newcommand{\bib}{%
566 \begingroup
567 \@ifstar{%
568 \@tempswatru
569 \let\@bibdef\star@bibdef
570 \BibItem
571 }{%
572 \@tempswafalse
573 \BibItem
574 }%
575 }

```

```

\BibItem Arguments:
 #1 <- citekey.
 #2 <- bibtype.

576 \newcommand{\BibItem}[2]{%
577 \def\@tempa{#1}%
578 \edef\@tempb{%
579 \@nx\@bibdef\@xp\@nx\csname setbib@#2\endcsname{#2}%
580 {\macrotext\@tempa}%
581 }%
582 \@tempb
583 }

```

`\@bibdef` `\@bibdef` is a pointer to the procedure that should be handed the entry's key-value pairs. It has one of four values:

1. `\star@bibdef`
2. `\normal@bibdef`
3. `\copy@bibdef`
4. `\selective@bibdef`

*Arguments:*

```

 #1 <- \setbib@bibtype.
 #2 <- bibtype.
 #3 <- citekey.

```

```
584 \AtBeginDocument{\let\@bibdef\normal@bibdef}
```

`\bib@exec` And `\bib@exec` is a pointer to the procedure that `\normal@bibdef` will invoke to process the key-value pairs after they've been parsed. It has one of these values:

1. `\bib@store`
2. `\bib@print`

*Arguments:*

```

 #1 <- citekey.
 #2 <- \the\rsk@toks.
 #3 <- \setbib@bibtype.

```

```
585 \AtBeginDocument{\let\bib@exec\bib@print}
```

### 6.12.1 `\@bibdef` Implementations

```

\normal@bibdef Arguments:
 #1 <- \setbib@bibtype.
 #2 <- bibtype.
 #3 <- citekey.

586 \def\normal@bibdef#1#2#3{%

```

`\CurrentBibType` is used by `export-bibtex`, but there might be a better way to handle it. (dmj)

```

587 \def\CurrentBibType{#2}%
588 \ifx\relax#1%
589 \amsrefs@error{Undefined entry type: #2}\@ehc
590 \let#1\setbib@misc
591 \fi
592 \RestrictedSetKeys{}{bib}%
593 {\bib@exec{#3}{\the\rsk@toks}{#1}\endgroup}%
594 }
595
596 \let\@bibdef\normal@bibdef

```

`\star@bibdef` *Arguments:*

```

#1 <- \setbib@bibtype.
#2 <- bibtype.
#3 <- citekey.

```

```

597 \def\star@bibdef{%
598 \let\bib@exec\bib@store
599 \normal@bibdef
600 }

```

`\copy@bibdef` This is a variation that copies everything into the `.bbl` file. Used by `\bibselect*` and `\bib*` inside `.ltx` files.

```

601 \def\copy@bibdef{%
602 \if@tempwa
603 \xp\defer@bibdef
604 \else
605 \xp\copy@bibdef@a
606 \fi
607 }

```

`\copy@bibdef@a`

```

608 \def\copy@bibdef@a#1#2#3#4{%
609 \@open@bbl@file
610 \process@xrefs{#4}%
611 \bbl@write{%
612 \string\bib\if@tempwa*\fi{#3}{#2}\string{\iffalse}\fi
613 }%

```

Since we're supplying our own definition of `\rsk@set`, we don't actually need the group argument, so we leave it out to save a few tokens.

```

614 \RestrictedSetKeys{\global\let\rsk@set\bbl@copy}\@empty
615 {\bbl@write{\iffalse\fi\string}^^J}%
616 \endgroup}{#4}%
617 }

```

```

618 \catcode'\:=11
619

```

```

620 \def\modify@xref@fields{%
621 \let\set:bib'author\output@xref@a
622 \let\set:bib'editor\output@xref@a
623 \let\set:bib'translator\output@xref@a
624 \let\set:bib'journal\output@xref@a
625 \let\set:bib'publisher\output@xref@a
626 \def\set:bib'xref##1##2{\output@xref@{##1}\@empty}%
627 }
628
629 \catcode'\:=12
630
631 \def\process@xrefs#1{%
632 \begingroup
633 \RestrictedSetKeys{\modify@xref@fields}{bib}{\the\rsk@toks}{#1}%
634 \endgroup
635 }
636
637 \def\output@xref@a#1#2{%
638 \def\@tempa{#1}%
639 \lowercase{\def\@tempb{#1}}%
640 \ifx\@tempa\@tempb
641 \output@xref@{#1}%
642 \fi
643 }
644
645 \def\output@xref@#1{%
646 \@ifnotempty{#1}{%
647 \@ifundefined{bi@#1}{%
648 \begingroup
649 \let\star@bibdef\copy@bibdef@a
650 \csname bi@#1\endcsname
651 \endgroup
652 }%
653 \xp@g@undef\csname bi@#1\endcsname
654 }%
655 }

```

\bbl@copy

```

656 \def\bbl@copy#1\endcsname#2{%
657 \begingroup
658 \def\@tempa{#1}%
659 \toks@{#2}%
660 \star@{\bbl@copy@a}{%
661 }

```

\bbl@copy@a

```

662 \def\bbl@copy@a#1{%
663 \@ifnotempty{#1}{%
664 \add@toks@{*#1}%
665 }%

```

```

666 \bbl@writef \space\@tempa=\the\toks@,}%
667 \endgroup
668 \rsk@resume
669 }

```

`\selective@bibdef` This is a variation that ignores anything not having a known citation key. Used by `\bibselect`.

*Arguments:*

#1 <- `\setbib@bibtype`.

#2 <- `bibtype`.

#3 <- `citekey`.

```

670 \def\selective@bibdef#1#2#3{%
671 \xp\selbibdef@a\csname b@#3\endcsname-#1-#2-#3}%
672 }

```

`\selbibdef@a`

```

673 \def\selbibdef@a#1{%
674 \def\@tempa{\endgroup\@gobblefour}%
675 \ifx\relax#1\else \xp\selbibdef@b#1\@nil \fi
676 \@tempa
677 }

```

`\selbibdef@b`

```

678 \def\selbibdef@b#1#2#3\@nil{%
679 \ifx 1#2\let\@tempa\copy@bibdef\fi
680 }

```

`\defer@bibdef` This is a variation that ignores anything not having a known citation key. Used by `\bibselect`.

*Arguments:*

#1 <- `\setbib@bibtype`.

#2 <- `bibtype`.

#3 <- `citekey`.

#4 <- `key-val pairs`.

```

681 \def\defer@bibdef#1#2#3#4{%
682 \xp\gdef\csname bi@#3\endcsname{%
683 \bib*{#3}{#2}{#4}%
684 }%
685 \xp\addto@defer@list \csname bi@#3\endcsname
686 \endgroup
687 }

```

`\bibdefer@list`

```

688 \let\bibdefer@list\@empty

```

`\addto@defer@list`

```
689 \def\addto@defer@list#1{%
690 \begingroup
691 \def\do{\@nx\do\@nx}%
692 \xdef\bibdefer@list{\bibdefer@list\do#1}%
693 \endgroup
694 }
```

### 6.12.2 `\bib@exec` Implementations

`\bib@store` This is the easy one. It just stores the entire set of key-value pairs in `\bi@citekey`.

```
695 \def\bib@store#1{%
696 \afterassignment\@gobble
697 \@xp\xdef\csname bi@#1\endcsname
698 }
```

`\numeric@refs`

```
699 \def\numeric@refs{00}
```

`\bib@print` *Arguments:*

```
#1 <- citekey.
#2 <- \the\rsk@toks.
#3 <- \setbib@bibtype.
```

```
700 \def\bib@print#1#2#3{%
701 \bib@start{#1}%
702 \let\setbib@@#3%
703 #2\relax % execute definitions locally
704 \bib@resolve@xrefs
705 \bib@field@patches
706 \bib@selectlanguage
707 \generate@label
708 \bib'setup
709 \bib@cite{#1}%
710 \kern\@ne sp
711 \ifx\setbib@@\setbib@article
712 \ifx\bib'booktitle\@empty
713 \ifx\bib'book\@empty
714 \ifx\bib'conference\@empty
715 \else
716 \let\setbib@@\setbib@incollection
717 \fi
718 \else
719 \let\setbib@@\setbib@incollection
720 \fi
721 \else
722 \let\setbib@@\setbib@incollection
723 \fi
724 \fi
725 \setbib@@
```

```
726 \bib@end
727 }
```

`\bib@print@inner` Note that the order of the arguments is reversed with respect to `\bib@print`. Maybe that isn't such a great idea.

*Arguments:*

```
#1 <- \setbib@bibtype.
#2 <- \the\rsk@toks.
```

```
728 \def\bib@print@inner#1#2{%
729 \begingroup
730 #2\relax % execute definitions locally
731 \bib@field@patches
732 \bib'setup
733 #1%
734 \endgroup
735 }
```

`\current@citekey`

```
736 \let\current@citekey\@empty
```

`\prev@citekey`

```
737 \let\prev@citekey\@empty
```

`\bib@start` There used to be more to it.

```
738 \def\bib@start#1{%
739 \begingroup
740 \def\current@citekey{#1}%
741 }
```

`\bib@end` Instead of being handled by `\bib@end`, ending punctuation is normally handled via the `transition` field (q.v.)

```
742 \def\bib@end{%
743 \relax
744 \@xp\PrintBackRefs\@xp{\CurrentBib}%
745 \par
746 \save@primary
747 \global\let\prev@citekey\current@citekey
748 \endgroup
749 }
```

### 6.12.3 Resolving cross-references

`\bib@resolve@xrefs`

```
750 \def\bib@resolve@xrefs{%
751 \xref@check@c\bib'xref
752 \xref@check@a\bib'author
753 \xref@check@a\bib'editor
754 \xref@check@a\bib'translator
```

```

755 \xref@check@b\bib' journal
756 \xref@check@b\bib'publisher
757 }

```

`\xref@check@a` Resolve a contributor (typically a `\DefineName`) alias. Requires rebuilding the list.

```

758 \def\xref@check@a#1{%
759 \ifx\@empty#1\relax
760 \else
761 \begingroup
762 \toks@\@emptytoks
763 \@temptokenb\@emptytoks
764 \series@index\z@
765 \def\name{\xref@check@aa#1}%
766 #1\relax
767 \edef\@tempa{%
768 \def\@nx#1{\the\toks@}%
769 \the\@temptokenb
770 }%
771 \@xp@endgroup
772 \@tempa
773 \fi
774 }

```

`\xref@check@aa`

```

775 \def\xref@check@aa#1#2{%
776 \advance\series@index\@ne
777 \def\@tempa{#2}%
778 \lowercase{\def\@tempb{#2}}%
779 \ifx\@tempa\@tempb
780 \ifx\@tempa\@empty
781 \add@toks@\{ \name{ } }%
782 \else
783 \@ifundefined{bi@#2}{%
784 \BibAbbrevWarning{#2}%
785 \add@toks@\{ \name{#2} }%
786 }{%
787 \xref@check@ab#1{#2}%
788 }%
789 \fi
790 \else
791 \add@toks@\{ \name{#2} }%
792 \fi
793 }

```

`\xref@check@ab`

```

794 \def\xref@check@ab#1#2{%
795 \csname bi@#2\endcsname
796 \ifx\@empty\bib'name

```

```

797 \@temptokena{#2}%
798 \else
799 \@temptokena\@xp{\bib'name}%
800 \get@property\@tempa\bib'name
801 \edef\@tempa{%
802 \@nx\addto@hook\@temptokenb{%
803 \@nx\reset@nth@property\@nx#1\the\series@index{\@tempa}%
804 }%
805 }%
806 \@tempa
807 \fi
808 \edef\@tempa{\@nx\add@toks@{\@nx\name{\the\@temptokena}}}%
809 \@tempa
810 }

```

`\xref@check@b` Resolve a journal or publisher alias (typically a `\DefinePublisher` or `\DefineJournal` alias).

```

811 \def\xref@check@b#1{%
812 \ifx\@empty#1%
813 \else
814 \toks@\@xp{#1}%
815 \edef\@tempb{\lowercase{\def\@nx\@tempa{\the\toks@}}}%
816 \@tempb
817 \ifx\@tempa#1\relax % all lowercase
818 \ifundefined{bi@#1}{%
819 \BibAbbrevWarning{#1}%
820 }{%

```

We pass control to `\xref@check@c` here to handle inheritance of multiple fields properly. This means some of the checking we've just done gets done again, but I can live with that.

```

821 \let#1\@empty
822 \xref@check@c\@tempa
823 }%
824 \fi
825 \fi
826 }

```

`\xref@check@c` Resolve an xref field.

```

827 \def\xref@check@c#1{%
828 \ifx#1\@empty
829 \else
830 \begingroup
831 \let\DSK@def\xref@add@toks
832 \let\DSK@append\xref@append
833 \toks@\@emptytoks
834 \let\bib@reset\@empty

```

The `\@for` here is just a fancy way of expanding `#1`. (Or is it?)

```

835 \@for\xref@ID:=#1\do{%

```

```

836 \@ifundefined{bi@\xref@ID}{%
837 \XRefWarning{\xref@ID}%
838 }{%
839 \csname bi@\xref@ID\endcsname
840 }%
841 }%
842 \edef\@tempa{\endgroup\the\toks@}%
843 \@tempa
844 \fi
845 }

```

`\xref@add@toks` If any title occurs in an `xrefed` item, assume that it is a book title. This might not always be the best assumption? Let's see how it goes though. [mjd,2001-12-11]

*Arguments:*

```

#1 <- \bib' field.
#2 <- value.

```

```

846 \def\xref@add@toks#1#2#3{%
847 \ifx#1\@empty
848 \edef\@tempa{%
849 \@nx\add@toks@{\@xp\@nx\csname\rkv@setter#1\endcsname{#2}{#3}}%
850 }%
851 \@tempa
852 \else
853 \in@\bib'title{#1}%
854 \ifin@
855 \ifx\bib'booktitle\@empty
856 \edef\@tempa{%
857 \@nx\add@toks@{%
858 \@xp\@nx\csname set:bib'booktitle\endcsname
859 }%
860 }%
861 \@tempa
862 \add@toks@{{#2}{#3}}%
863 \fi
864 \fi
865 \fi
866 }

867 \def\xref@append#1#2#3#4{%
868 \edef\@tempa{%
869 \@nx\add@toks@{\@xp\@nx\csname\rkv@setter#2\endcsname{#3}{#4}}%
870 }%
871 \@tempa
872 }

```

`\BibAbbrevWarning`

```

873 \def\BibAbbrevWarning#1{\amsrefs@warning{Abbreviation '#1' undefined}}

```

\XrefWarning

```
874 \def\XrefWarning#1{\amsrefs@warning{Xref '#1' undefined}}
```

#### 6.12.4 Bib field preprocessing

\current@primary

```
875 \let\current@primary\@empty
```

\previous@primary

```
876 \let\previous@primary\@empty
```

\save@primary

```
877 \IfOption{nobysame}{%
878 \let\save@primary\@empty
879 }{%
880 \def\save@primary{%
881 \global\let\previous@primary\current@primary
882 }%
883 }
```

\bib@field@patches

```
884 \def\bib@field@patches{%
885 \ifx\bib'author\@empty
886 \ifx\bib'editor\@empty
887 \let\current@primary\bib'translator
888 \let\print@primary\PrintTranslatorsA
889 \else
890 \let\current@primary\bib'editor
891 \let\print@primary\PrintEditorsA
892 \fi
893 \else
894 \let\current@primary\bib'author
895 \let\print@primary\PrintAuthors
896 \fi
897 \ifx\bib'address\@empty
898 \let\bib'address\bib'place
899 \fi
900 \ifx\bib'organization\@empty
901 \ifx\bib'institution\@empty
902 \let\bib'organization\bib'school
903 \else
904 \let\bib'organization\bib'institution
905 \fi
906 \fi
907 \ifx\bib'date\@empty
908 \ifx\bib'year\@empty
909 \let\bib@year\bib'status
910 \else
911 \bib@parsedate\bib'year
912 \fi
```

```

913 \else
914 \bib@parsedate\bib'date
915 \fi

```

Example 21 on page 74 of *Mathematics into Type* [?SOS99] seems to indicate that when the year serves as the volume number, the date should be suppressed. If so, this is where that is done.

```

916 \def\@tempa{year}%
917 \ifx\bib'volume\@tempa
918 \let\bib'volume\bib@year
919 \let\bib'date\@empty
920 \fi

```

`\bib'language` is used for producing the printed rendition of the language. `\bib@language` needs to be in the form required by `\selectlanguage`.

```

921 \bib@language@fixup
922 }

```

### 6.12.5 Date setup

`\bib@year`

```

923 \let\bib@year\@empty

```

`\bib@month`

```

924 \let\bib@month\@empty

```

`\bib@day`

```

925 \let\bib@day\@empty

```

`\bib@parsedate` Parse an ISO 8601 date into its year, month and day components, but without actually verifying that any of the components are numeric. Hmmm.

```

926 \def\bib@parsedate#1{%
927 \@xp\bib@parsedate@a#1---\@nil
928 }

```

`\bib@parsedate@a`

```

929 \def\bib@parsedate@a#1-#2-#3-#4\@nil{%
930 \def\bib@year{#1}%
931 \def\bib@month{#2}%
932 \def\bib@day{#3}%

```

The rest of this macro tries to rewrite `\bib'date` into a normalized form. I'm not sure if this is a good idea.

```

933 \ifx\@empty\bib@day
934 \ifx\@empty\bib@month
935 \let\bib'date\bib@year
936 \else
937 \def\bib'date{#1-#2}%
938 \fi
939 \else

```

```

940 \def\bib'date{#1-#2-#3}%
941 \fi
942 }

```

### 6.12.6 Language setup

`\bib@language@fixup`

```

943 \def\bib@language@fixup{%
944 \ifx\bib'hyphenation\@empty
945 \ifx\bib'language\@empty
946 \let\bib@language\biblanguagedefault
947 \else
948 \let\bib@language\bib'language
949 \fi
950 \else
951 \let\bib@language\bib'hyphenation
952 \fi
953 \def\@tempa##1 ##2\@nil{\lowercase{\def\bib@language{##1}}}%

```

The mysterious `\@firstofone` here is to preserve the space before the `\@nil`.

```

954 \@firstofone{\@xp\@tempa\bib@language} \@nil
955 }

```

`\bib@selectlanguage`

For `\bib` purposes we are interested mainly in testing whether the hyphenation patterns are the same. So we use an if-same-patterns test (by which `babel`'s 'english' and 'american' compare as equal) rather than an if-same-language test. Also, the way that the `\selectlanguage` command checks to see whether a language has been properly defined for `babel` use is to see if `\dateLANGUAGE` is defined. And if we tried to select an undefined language, the result would be a `LATEX` error.

```

956 \def\bib@selectlanguage{%
957 \@ifsame@patterns{\languagename}{\bib@language}{-}{-}%
958 \@ifundefined{date\bib@language}{-}{-}%
959 \@xp\selectlanguage\@xp{\bib@language}%
960 }%
961 }%
962 }

```

`\@ifsame@patterns`

```

963 \def\@ifsame@patterns#1#2{%
964 \@xp\@ifsamepat\csname l@#1\@xp\endcsname\csname l@#2\endcsname
965 }

```

`\@ifsamepat`

```

966 \def\@ifsamepat#1#2{%
967 \ifnum \ifx\relax#1\m@ne\else#1\fi = \ifx\relax#2\m@ne\else#2\fi
968 \@xp\@firstoftwo
969 \else
970 \@xp\@secondoftwo
971 \fi
972 }

```

```

\languagename
\biblanguageEnglish 973 \providecommand{\languagename}{english}
\biblanguagedefault 974 \def\biblanguageEnglish{english}
 \bib@language 975 \let\biblanguagedefault\biblanguageEnglish
 976 \let\bib@language\@empty

```

### 6.12.7 Citation label setup

```

\generate@label
 977 \let\generate@label\relax

\cite@label
 978 \def\cite@label{\@currentlabel}

\alpha@label
 979 \let\alpha@label\relax

\bib@cite When \bib@cite is called, author name and year are available in \bib@author
 and \bib@year.

 Arguments:
 #1 <- citekey.

 980 \def\bib@cite#1{%
 981 \def\CurrentBib{#1}%
 982 \alpha@label % modify \thebib if necessary
 983 \item\leavevmode
 984 \SK@\SK@@label{#1}%
 985 \@xp\bib@cite@a\csname b@#1\endcsname
 986 \bibcite@write{#1}%
 987 }

 988 \def\bib@cite@a#1{%
 989 \ifx\relax#1%
 990 \begingroup
 991 \auto@protect\etaltext
 992 \protected@edef\@tempa{%
 993 \gdef\@nx#1{%
 994 \@nx\citesel 01{\cite@label}{\bib@label@year}{}%
 995 }%
 996 }%
 997 \@xp\endgroup
 998 \@tempa
 999 \else
 1000 \@xp\bib@cite@check\@xp#1#1\@empty\@empty\@empty\@empty\@empty
 1001 \fi
 1002 }

```

**\bib@cite@check** For the citation key we want to check if it is already defined. But there is a slight problem. There is already one control sequence in use for each bibliography entry, to store the label or the author/year information needed by `\cite`. If

we introduce another control sequence to check whether a particular cite is multiply defined, then we double the number of control sequences used. For a large bibliography in a book this is fairly serious. This is addressed by using a `\citesel` function.

*Arguments:*

```
#1 <- \b@citekey.
#2 <- \citesel.
#3 <- cited?.
#4 <- used?.
#5 <- label.
#6 <- year.
#7 <- backrefs.
```

```
1003 \def\bib@cite@check#1#2#3#4#5#6#7{%
1004 \ifx 1#4\relax
1005 \DuplicateBibKeyWarning
1006 \else
```

This has gotten *way* out of hand.

```
1007 \begingroup
1008 \auto@protect\etaltex
1009 \@apply\auto@protect\amsrefs@textsymbols
1010 \@apply\auto@protect\amsrefs@textaccents
1011 \@tempswafalse
1012 \in\CitePrintUndefined{#5}%
1013 \ifin@
1014 \let\@tempa\@empty
1015 \else
1016 \def\@tempa{#5}%
1017 \fi
1018 \ifx\@tempa\@empty
1019 \else
1020 \@xp\ifx\@xp\@currentlabel\cite@label
1021 \edef\@tempb{\cite@label}%
1022 \else
1023 \let\@tempb\cite@label
1024 \fi
1025 \ifx\@tempa\@tempb
1026 \def\@tempa{#6}%
1027 \ifx\@tempa\bib@label@year
1028 \else
1029 \@tempwatrue
1030 \fi
1031 \else
1032 \@tempwatrue
1033 \fi
1034 \fi
1035 \if@tempswa
1036 \@ifempty{#6}{%
```

```

1037 \def\@tempa{#5}%
1038 \let\@tempb\cite@label
1039 }{%
1040 \def\@tempa{#5, #6}%
1041 \def\@tempb{\cite@label, \bib@label@year}%
1042 }%
1043 \amsrefs@warning{Citation label for \extr@cite#1 is
1044 changing from '\@tempa ' to '\@tempb '}%
1045 \fi
1046 \protected@edef\@tempa{%
1047 \gdef\@nx#1{%
1048 \@nx\citesel #31{\cite@label}{\bib@label@year}{#7}%
1049 }%
1050 }%
1051 \xp@endgroup
1052 \@tempa
1053 \fi
1054 }

```

\bib@label@year

```
1055 \let\bib@label@year\@empty
```

\DuplicateBibKeyWarning

```

1056 \def\DuplicateBibKeyWarning{%
1057 \amsrefs@warning{%
1058 Duplicate \protect\bib\space key
1059 '\CurrentBib ' detected\MessageBreakNS}%
1060 }

```

\bibcite@write

```

1061 \def\bibcite@write#1{%
1062 \if@filesw
1063 \let\citesel\citesel@write
1064 \csname b@#1\endcsname
1065 \fi
1066 }

```

\citesel@write

```

1067 \def\citesel@write#1#2#3#4#5{%
1068 \begingroup
1069 \toks@{#3}{#4}%
1070 \immediate\write\@auxout{\string\bibcite{\CurrentBib}{\the\toks@}}%
1071 \endgroup
1072 }

```

Because duplicate bibs are caught immediately, we don't need \bibcite to run \@testdef.

```
1073 \AtEndDocument{\let\bibcite\@gobbletwo}
```

**6.12.8 Printing the bibliography****\bibname**

1074 \providecommand{\bibname}{Bibliography}

**\refname**

1075 \providecommand{\refname}{References}

**bibchapter** We need to take a little extra trouble here to pre-expand the `\bibname`.

```

1076 \newenvironment{bibchapter}[1][\bibname]{%
1077 \begingroup
1078 \protected@edef\@{\endgroup\protect\chapter*{#1}}%
1079 \@
1080 }{\par}

```

**bibsection** And here to pre-expand the `\refname`.

```

1081 \newenvironment{bibsection}[1][\refname]{%
1082 \begingroup
1083 \protected@edef\@{\endgroup\protect\section*{#1}}%
1084 \@
1085 }{\par}

```

**bibdiv** Here we try to guess whether this is a book-like document or an article-like document.

```

1086 \@ifundefined{chapter}{%
1087 \newenvironment{bibdiv}{\bibsection}{\endbibsection}
1088 }{%
1089 \newenvironment{bibdiv}{\bibchapter}{\endbibchapter}
1090 }

```

This is what the standard book class has for the bibliography title:

```

\newenvironment{thebibliography}[1]
{
 \chapter*{\bibname
 \@mkboth{\MakeUppercase\bibname}{\MakeUppercase\bibname}}%
 \list{\@biblabel{\@arabic{c@enumiv}}}%

```

**thebibliography**

```

1091 \renewenvironment{thebibliography}[1]{%
1092 \bibdiv
1093 \biblist[\resetbiblist{#1}]%
1094 }{%
1095 \endbiblist
1096 \endbibdiv
1097 }

```

### 6.13 Name, journal and publisher abbreviations

The commands `\DefineName`, `\DefinePublisher`, and `\DefineJournal` are provided to make abbreviations a little easier.

`\DefineName`

```
1098 \newcommand{\DefineName}[2]{%
1099 \bib*{#1}{name}{name={#2}}%
1100 }
```

`\DefineJournal`

```
1101 \newcommand{\DefineJournal}[4]{%
1102 \bib*{#1}{periodical}{
1103 issn={#2},
1104 journal={#4}
1105 }%
1106 }
```

`\DefinePublisher` Note that an explicit address field in a `\bib` entry will override the address supplied as part of a `\DefinePublisher`.

```
1107 \newcommand{\DefinePublisher}[4]{%
1108 \bib*{#1}{publisher}{%
1109 publisher={#3},
1110 address={#4}
1111 }%
1112 }
```

### 6.14 Processing .ltb files

If you have a file that contains `amsrefs`-style `\bib` entries, you can use it as a database and extract items from it for use in another document. In typical relatively simple scenarios, the extraction can be done by `LATEX` itself on the first pass, so that citations in the text will be successfully resolved on the second pass (possibly even the first, depending on what kind of bibliography sorting is used).

`\bibselect`

```
1113 \newcommand{\bibselect}{%
1114 \@ifstar{%
1115 \let\@bibdef\copy@bibdef
1116 \BibSelect
1117 }{%
1118 \let\@bibdef\selective@bibdef
1119 \BibSelect
1120 }%
1121 }
```

`\BibSelect`

```
1122 \newcommand{\BibSelect}[2][\bblname]{%
1123 \if@filesw
```

```

1124 \typeout{Trying to create bbl file '#1.bbl' ...}%
1125 \def\bibselect@msg{%
1126 \typeout{ ... rats. Unable to create bbl file.}%
1127 }%
1128 \let\@open@bbl@file\OpenBBLFile
1129 \@for\@tempa:=#2\do{\ReadBibData{\@tempa}}%
1130 \fi
1131 \close@bbl@file
1132 \@apply\g@undef\bibdefer@list
1133 \global\let\bibdefer@list\@empty
 Now read the .bbl file we just created.
1134 \let\@bibdef\normal@bibdef
1135 \@input{#1.bbl}%
1136 \let\BibSelect\MultipleBibSelectWarning
1137 }

```

\MultipleBibSelectWarning

```

1138 \newcommand\MultipleBibSelectWarning[2] [] {%
1139 \amsrefs@warning{%
1140 Multiple \string\bibselect 's found (only one
1141 \string\bibselect\space per biblist environment is allowed)%
1142 }%
1143 }

```

\bblname

```

1144 \def\bblname{\jobname}

```

\bib@dbfile

```

1145 \newread\bib@dbfile

```

\ReadBibData

```

1146 \newcommand{\ReadBibData}[1] {%
1147 \IfFileExists{#1.ltb}{%
1148 \openin\bib@dbfile=\@filef@und \relax
1149 }{%
1150 \IfFileExists{#1.ltx}{%
1151 \openin\bib@dbfile=\@filef@und \relax
1152 }{%
1153 \IfFileExists{#1.tex}{%
1154 \openin\bib@dbfile=\@filef@und \relax
1155 }{%
1156 \begingroup
1157 \NoBibDBFile{#1}%
1158 \let\ReadBibData@a\endgroup
1159 }%
1160 }%
1161 }%
1162 \ReadBibData@a
1163 }

```

`\NoBibDBFile`

```

1164 \def\NoBibDBFile#1{%
1165 \amsrefs@warning{No data file #1.ltb (.ltx, .tex) found}%
1166 }

```

`\ReadBibData@a`

```

1167 \def\ReadBibData@a{%
1168 \ProvidesFile{\@filef@und}\relax
1169 \begingroup
1170 \let\star@bibdef\defer@bibdef
1171 \ReadBibLoop
1172 \endgroup
1173 \closein\bib@dbfile
1174 }

```

`\ReadBibLoop`

```

1175 \def\ReadBibLoop{%
1176 \ifeof\bib@dbfile
1177 \@xp@gobble
1178 \else
1179 \read\bib@dbfile to\CurLine
1180 The \@empty is in case \CurLine is empty.
1181 \@xp\ReadBibLoop@a\CurLine\@empty\@nil
1182 \fi
1183 }

```

`\ReadBibLoop@e` This traps top-level `\bib` commands. Note that:

- If `\CurLine` doesn't contain a complete `\bib` entry, the code chokes.
- If `\bib` is not the very first non-space token in a line, it will not be recognized.

```

1184 \long\def\ReadBibLoop@a#1#2\@nil{%
1185 \ifx\bib#1%
1186 \CurLine % just exec it
1187 \else

```

We're not done yet. The line may contain something like `\DefineName`, so we need to expand the first macro in the line and see if it starts with `\bib`. But first we check to make sure that the token we're about to expand isn't `\endinput`.

```

1188 \ifx\endinput#1%
1189 \let\ReadBibLoop\@empty
1190 \else

```

And this `\@empty` is for the admittedly unlikely case that `\CurLine` isn't empty, but its expansion is.

```

1191 \@xp\ReadBibLoop@b#1#2\@empty\@nil
1192 \fi
1193 \fi
1194 }

```

`\ReadBibLoop@b`

```

1195 \long\def\ReadBibLoop@b#1#2\@nil{%
1196 \ifx\bib#1%
1197 \CurLine % just exec it
1198 \fi
1199 }

1200 \let\bbl@out=\relax
1201 \let\bbl@write\@gobble
1202 \let\@open@bbl@file\relax
1203 \let\@close@bbl@file\relax

```

`\OpenBBLFile`

```

1204 \def\OpenBBLFile{%
1205 \if@filesw
1206 % Just use the next unused output stream
1207 \count@\count17
1208 \advance\count@\@ne
1209 \ifnum\count@<\sist@\@n
1210 \global\chardef\bbl@out=\count@
1211 \immediate\openout\bbl@out=\bblname.bbl\relax
1212 \global\let\@close@bbl@file\CloseBBLFile
1213 \gdef\bbl@writef\immediate\write\bbl@out}%
1214 \else
1215 \ch@ck\count@\sist@\@n\write
1216 \fi
1217 \fi
1218 \global\let\@open@bbl@file\relax
1219 }

```

`\CloseBBLFile`

```

1220 \def\CloseBBLFile{%
1221 \immediate\closeout\bbl@out\relax
1222 \global\let\@close@bbl@file\relax
1223 \global\let\bbl@write\@gobble
1224 \global\let\bbl@out\relax
1225 }

```

## 6.15 Citation processing

### 6.15.1 The `\citesel` structure

The information used by `\cite` for key `moo` is stored in `\b@moo` in the form

```
\citesel{status1}{status2}{label}{year}{backref-info}
```

The first status flag is 1 if this key has already been cited earlier in the same document; 0 otherwise. This is used in some bibliography schemes to print a full list of author names for the first citation and an abbreviated author list for subsequent citations.

The second status flag is 1 if this key has already been used by a `define-cite` command (such as `\bib`); 0 otherwise. This makes it possible to issue a warning

message as soon as the conflict is seen, on the first L<sup>A</sup>T<sub>E</sub>X run, instead of on a subsequent run during the processing of the .aux file.

When an author/year citation scheme is in use, args 3 and 4 hold respectively author names and year. Otherwise arg 3 simply holds a cite label and arg 4 is empty.

And finally, arg 5 holds a list of backref pointers indicating the locations in the document where this entry has been cited.

```
\citesel@update
1226 \def\citesel@update#1#2#3#4#5#6{%
1227 \gdef#6{\citesel 1#2{#3}{#4}{#5}}%
1228 }
```

```
\citesel@number
1229 \def\citesel@number#1#2#3#4#5{#3}
```

```
\citesel@year
1230 \def\citesel@year#1#2#3#4#5{#4}
```

```
\citesel
1231 \let\citesel\citesel@number
```

### 6.15.2 The basic \cite command

Here is the difference between the various optional forms of \cite:

```
\cite{xyz} -> \cite@a\citesel{xyz}{ }
 -> \cite@bc\b@xyz\citesel{ }

\cite{xyz}*{blub} -> \cite@a\citesel{xyz}{blub}
 -> \cite@bc\b@xyz\citesel{blub}

\cite[blub]{xyz} -> \cite@a\citesel{xyz}{blub}
 -> \cite@bc\b@xyz\citesel{blub}
```

Canceling the old L<sup>A</sup>T<sub>E</sub>X definition of \cite<sub>l</sub> prevents certain problems that could arise with the showkeys package.

```
1232 \expandafter\let\csname cite \endcsname\relax
```

\cite Need to handle the standard [...] option for compatibility's sake.

```
1233 \renewcommand{\cite}[2] [] {%
1234 \if\cite@single#2,\@gobble \else\MultipleCiteKeyWarning{#2}{#1}\fi
1235 \@ifempty{#1}{%
1236 \cites@o{#2}%
1237 }{%
1238 \ObsoleteCiteOptionWarning
1239 \cites@a{*{#1}}{#2}%
1240 }%
1241 }
```

## \MultipleCiteKeyWarning

```

1242 \def\MultipleCiteKeyWarning#1#2{%
1243 \amsrefs@warning{%
1244 Use of \string\cites\space is recommended instead of %
1245 \string\cite\space\MessageBreak
1246 for multiple cites '#1'}%
1247 \@ifnotempty{#2}{%
1248 \amsrefs@warning{Star option requires \string\citelist\space here}%
1249 }%
1250 \global\let\MultipleCiteKeyWarning@gobbletwo
1251 }

```

## \ObsoleteCiteOptionWarning

```

1252 \def\ObsoleteCiteOptionWarning{%
1253 \amsrefs@warning{%
1254 The form \string\cite{...}*{...} is recommended\MessageBreak
1255 instead of \string\cite[...]{...}%
1256 \global\let\ObsoleteCiteOptionWarning@empty
1257 }

```

## \cite@single

```

1258 \edef\cite@single#1,#2{\iffalse{\fi\iffalse{\fi\string}#2.\string}}

```

## \cites@o

```

1259 \def\cites@o#1{\star@\cites@oo{#1}}

```

## \cites@oo

```

1260 \def\cites@oo#1#2{\@ifempty{#2}{\cites@a}{#1}}{\cites@a*{#2}}{#1}}

```

## \cites@a

```

1261 \def\cites@a#1#2{%
1262 \begingroup
1263 \toks@{\endgroup \cites@b{#1}}%
1264 \vdef\@tempa{#2}%
1265 \edef\@tempa{%
1266 \the\toks@ \@firstofone{\@xp\zap@space\@tempa} \@empty
1267 }%
1268 \@tempa,\@empty
1269 \edef\@tempa{\endgroup\@nx\citelist{\the\toks@}}%
1270 \@tempa
1271 }

```

## \cites@b

```

1272 \def\cites@b#1#2,#3{%
1273 \begingroup
1274 \toks@{\InnerCite{#2}#1}%
1275 \ifx\@empty#3\@xp@gobble\fi
1276 \cites@c#3%
1277 }

```

`\cites@c`

```
1278 \def\cites@c#1,#2{%
1279 \add@toks@{\InnerCite{#1}}%
1280 \ifx\@empty#2\@xp\@gobble\fi
1281 \cites@c#2%
1282 }
```

`\citeleft` These variables are named to follow the precedent set by Arseneau’s `cite` package. `\citeright` `\citewidth` is used to separate a citation label from additional information such as “Theorem 4.9”. `\citepunct` is used to separate multiple cites, unless one of the cites has additional associated information, in which case `\CiteAltPunct` is used.

```
1283 \def\citeleft{[]
1284 \def\citeright{]}
1285 \def\citewidth{\penalty9999 \space}
1286 \def\citepunct{\penalty9999 \hskip.13em plus.1em minus.05em\relax}
```

`\citeAltPunct` When a citation list contains one or more citations with optional arguments, we replace `\citewidth` by `\CiteAltPunct`.

```
1287 \def\citeAltPunct{;\ }
```

`\citeform` This is used for formatting the citation label. It can be used, for example, to bolden the labels (as in `amsbook` and `amsproc`) or to do more elaborate things such as convert the numbers to roman numerals. By default, it’s just a no-op.

Note that currently there is no corresponding macro for changing the formatting of `\cite`’s optional argument. This is probably a bug.

```
1288 \providecommand{\citeform}{\@firstofone}
```

`\citelist` The `\@citelist` indirection turns out to be helpful in implementing the `\ocites` command for the author-year option.

```
1289 \DeclareRobustCommand{\citelist}{\@citelist}
```

`\@citelist`

```
1290 \def\@citelist#1{%
1291 \leavevmode
1292 \begingroup
1293 \@citestyle
1294 \citeleft\nopunct % suppress first \citepunct
1295 \cite@begingroup
1296 \in*{#1}%
1297 \ifin@
1298 \let\citepunct\CiteAltPunct
1299 \fi
1300 \let\cite@endgroup\@empty
1301 \cites@init
1302 \def\citeleft{\@addpunct{\citepunct}}%
1303 \let\citeright\ignorespaces
```

```

1304 \def\cite{\InnerCite}%
1305 \process@citelist{#1}%
1306 \endgroup
1307 \citeright
1308 \endgroup
1309 }

```

`\@citestyle` Reset the font to an upright, medium font (e.g. `cmr`), per AMS style. Also set `\mathsurround = 0pt` just in case there are subscripts in the cite numbers (from `\etalchar`, for example).

```
1310 \providecommand{\@citestyle}{\m@th\upshape\mdseries}
```

`\cite@begingroup` Grouping that encloses an entire cite block (a single cite or a list of cites).

```
1311 \def\cite@begingroup{\begingroup\let\cite@begingroup\relax}
```

`\cite@endgroup`

```
1312 \let\cite@endgroup\endgroup
```

`\cites@init` This needs to be called at the beginning of a list of cites to reset a few things.

```

1313 \def\cites@init{%
1314 \gdef\prev@names{???}%
1315 \let\cites@init\@empty
1316 }

```

`\InnerCite`

```
1317 \newcommand{\InnerCite}[1]{\star@{\cite@a\citesel{#1}}{}}
```

`\cite@a` The job of `\cite@a` is to convert the cite key to all catcode-12 characters and remove any spaces it might contain before passing it on to `\cite@b`.

*Arguments:*

```

#1 <- \CITASEL.
#2 <- citekey.

```

```

1318 \def\cite@a#1#2{%
1319 \BackCite{#2}%
1320 \cite@begingroup
1321 \cites@init
1322 \let\citesel#1\relax
1323 \ifx\citesel\citesel@author
1324 \let\citeleft\@empty
1325 \let\citeright\@empty
1326 \fi
1327 \begingroup
1328 \toks@{\endgroup \cite@b}%
1329 \vdef\@tempa{#2}%
1330 \edef\@tempa{%
1331 \the\toks@{\@firstofone{\@xp\zap@space\@tempa} \@empty}%
1332 }%
1333 \@tempa
1334 }

```



`\CPU@normal` This has to be a `\let`, not a `\def`.

```
1360 \let\CPU@normal\CitePrintUndefined
```

`\cite@cj` *Arguments:*

```
 #1 <- \b@citekey.
 #2 <- star-optional-arg.
```

```
1361 \def\cite@cj#1#2{%
1362 \leavevmode
1363 \begingroup
1364 \cite@cb#1% write info to aux file
1365 \ar@SK@cite#1%
1366 \citeleft
1367 \ar@hyperlink{#1}%
1368 \@ifnotempty{#2}{\citamid{#2}}%
1369 \citeright
1370 \endgroup
1371 \ignorespaces % ignore spaces inside \citelist
1372 \cite@endgroup
1373 }
```

`\@citeleft` The following definition provides some indirection that helps to deal with author-year object cites.

```
1374 \def\@citeleft{\citeleft}
```

`\cite@cb`

```
1375 \def\cite@cb#1{%
1376 \if@filesw
1377 \immediate\write\@auxout{\string\citation{\extr@cite#1}}%
1378 \fi
 Define \citesel to make \b@whatever update itself.
1379 \begingroup
1380 \let\citesel\citesel@update
1381 #1#1%
1382 \endgroup
1383 }
```

`\extr@cite` Extract *citekey* from `\b@citekey`.

```
1384 \def\extr@cite{\@xp\@gobblethree\string}
```

### 6.15.3 Fancier `\cite` commands

`\cites` A list of simple cites. Make it robust in case used inside a figure caption. (But then also, by the way, `listoffigures` should provide special handling.)

```
1385 \DeclareRobustCommand{\cites}{\cites@a{}}
```

`\citen` This is just to keep the `showkeys` package from clobbering the wrong part of our definition of `\cite`:

```
1386 \providecommand{\citen}{\ocite}
```

`\ycite` `\cite` gets redefined inside of `\citelist`, so we need to `\def \ycite` here instead of just `\letting` everything to `\cite`.

```
1387 \def\ycite{\cite}
```

```
\ycites
```

```
1388 \let\ycites\cites
```

```
\ocite
```

```
1389 \let\ocite\ycite
```

```
\ocites
```

```
1390 \let\ocites\cites
```

```
\fullcite
```

```
1391 \let\fullcite\cite
```

```
\fullocite
```

```
1392 \let\fullocite\ocite
```

```
\citeauthor
```

```
1393 \let\citeauthor\ycite
```

```
\citeauthorhy
```

```
1394 \let\citeauthorhy\ycite
```

#### 6.15.4 The `\nocite` command

```
\nocite
```

```
1395 \renewcommand{\nocite}[1]{\othercites{#1}}
```

```
\othercites
```

```
1396 \newcommand{\othercites}[1]{%
```

```
1397 \cite@begingroup
```

```
1398 \let\cite@endgroup\@empty
```

```
1399 \def\citelist{\othercitelist}%
```

```
1400 \cites{#1}%
```

```
1401 }
```

```
\othercitelist
```

```
1402 \newcommand{\othercitelist}[1]{%
```

```
1403 \cite@begingroup
```

```
1404 \let\cite@endgroup\@empty
```

```
1405 \cites@init
```

```
1406 \let\citeleft\relax
```

```
1407 \let\citeright\ignorespaces
```

```
1408 \def\InnerCite{\OtherCite}%
```

```
1409 \def\cite@cj ##1##2{%
```

```
1410 \begingroup
```

```
1411 \@xp\citesel##1%
```

```
1412 \cite@cb ##1%
```

```
1413 \endgroup
```

If we detect `\nocite{*}`, we globally alias `\selective@bibdef` to `\copy@bibdef` so that all succeeding `\bibselect` commands act like `\bibselect*`.

```

1414 \@xp\ifx\csname b@*\endcsname ##1%
1415 \global\let\selective@bibdef\copy@bibdef
1416 \fi
1417 \ignorespaces
1418 \cite@endgroup
1419 }%
1420 #1\relax
1421 \endgroup
1422 }

```

`\OtherCite`

```

1423 \def\OtherCite#1{\cite@a\citesel@other{#1}{}}

```

`\citesel@other`

```

1424 \def\citesel@other#1#2#3#4#5#6{}

```

`\b@*` This provides a dummy definition to keep things like `\nocite{*}` from generating an error message.

```

1425 \@namedef{b@*}{\citesel 11{*}{*}{*}}

```

### 6.15.5 Citation sorting

`\process@citelist@sorted`

```

1426 \def\process@citelist@sorted#1{%
1427 \ifx\citesel\citesel@number
1428 \cite@sorted@s #1\cite@sorted@e
1429 \else
1430 \NonNumericCiteWarning
1431 \process@citelist@unsorted{#1}%
1432 \fi
1433 }

```

`\NonNumericCiteWarning`

```

1434 \def\NonNumericCiteWarning{%
1435 \amsrefs@warning{%
1436 Unable to confirm that cite keys are numeric: not sorting%
1437 }%
1438 }

```

`\process@citelist@unsorted`

```

1439 \def\process@citelist@unsorted#1{%
1440 \ignorespaces#1\relax
1441 }

```

`\process@citelist` By default, citation lists will be sorted.

```

1442 \let\process@citelist\process@citelist@sorted

```

`\CPU@sort` By defining this as TeX's maxint, undefined cites migrate to the end of a sorted list.

```
1443 \def\CPU@sort#1{2147483647}
```

`\cite@sorted@s` Here's where we prepare to sort the citations and (optionally) compress ranges.

```
1444 \def\cite@sorted@s{%
1445 \begingroup
1446 \let\CitePrintUndefined\CPU@sort
1447 \let\cite@cjs\cite@cj
1448 \let\cite@cj\cite@compress
1449 \begingroup
1450 \toks@\@emptytoks
1451 \let\cite@cj\cite@sort
1452 \ignorespaces
1453 }
```

`\cite@sorted@e`

```
1454 \def\cite@sorted@e{%
1455 \@xp\endgroup
1456 \the\toks@
1457 \cite@dash
1458 \prev@cite
1459 \endgroup
1460 }
```

`\cite@sort` This is essentially an insertion sort. I think.

*Arguments:*

`#1` <- `\b@citekey`.

`#2` <- *optional arg*.

```
1461 \def\cite@sort#1#2{%
1462 \safe@set\@tempcnta#1% highest number so far
1463 \toks@\{\cite@cj#1{#2}}%
1464 \@temptokena\toks@
1465 \let\cite@cj\cite@sort@a
1466 \ignorespaces
1467 }
```

`\cite@sort@a`

```
1468 \def\cite@sort@a#1#2{%
1469 \safe@set\@tempcntb#1%
1470 \ifnum\@tempcntb > \@tempcnta
1471 \add@toks@\{\cite@cj#1{#2}}%
1472 \@tempcnta\@tempcntb
1473 \else
1474 \let\cite@cj\cite@sort@b
1475 \toks@\@emptytoks
1476 \def\@tempb{\add@toks@\{\cite@cj#1{#2}}}%
1477 \the\@temptokena
```

```

1478 \@tempb
1479 \let\cite@cj\cite@sort@a
1480 \fi
1481 \@temptokena\toks@
1482 \ignorespaces
1483 }

\cite@sort@b
1484 \def\cite@sort@b#1#2{%
1485 \safe@set\count@#1%
1486 \ifnum\@tempcntb < \count@
1487 \@tempb
1488 \let\@tempb\@empty
1489 \fi
1490 \add@toks@\{\cite@cj#1{#2}}%
1491 \ignorespaces
1492 }

```

### 6.15.6 Range compression

When the time comes to apply compression, we have at our disposal a list of internal cite calls that looks like this:

```
\cite@cj\b@aaa{opta}\cite@cj\b@bbb{optb}...\cite@cj\b@zzz{optz}
```

where

$$\b@aaa < \b@bbb < \dots < \b@zzz$$

and the *opt* arguments are possibly null. To print the citations while collapsing sequences of 3 or more contiguous numbers into ranges of the form  $n-m$ , we bind `\cite@cj` to a suitably clever function and then execute the list. In the absence of optional arguments, here's the algorithm:

Begin. Enter state 0. This is done by `\cite@sorted@s`.

State 0. The current citation is the beginning of a range (possibly a singleton range). Print it. Then, set  $prevnum := number$  and enter state 1.

State 1. The current citation might be the second element of a range.

Case a)  $number = prevnum + 1$ . Then the current item is definitely the second element of a range. It might be the last element of the range, but we won't know until we examine the following citation. So, save the current citation in `\prev@cite`, set  $prevnum := number$ , and go to state 2.

Case b)  $number \neq prevnum + 1$ . The current citation is the beginning of a new range. Print it, set  $prevnum := number$  and remain in state 1. (This is essentially identical to stage 0.)

State 2. The current citation might be the third (or later) element of a range.

Case a)  $number = prevnum + 1$ . The current element is definitely part of a range. It might be the last element of the range, but again we won't know until we examine the following citation. Save the current citation in `\prev@cite` and set  $prevnum := number$ . Remain in state 2.

Case b)  $number \neq prevnum + 1$ . The previous citation was the end of a range and the current citation is the beginning of a new range. Print a dash followed by `\prev@cite`, then set  $prevnum := number$  and enter state 1.

End. If `\prev@cite` is not empty, print it, preceded by a dash if we were in the middle of a range. (This is done by `\cite@sorted@e`.)

The presence of optional arguments complicates things somewhat, since a citation with an optional argument should never participate in range compression. In other words, when we come across an optional argument, we should finish off the preceding range, print the current citation, and then return to the initial state. More precisely, here are the actions taken in each state when there is an optional argument:

State 0. Print the current citation and remain in state 0.

State 1. Print the current citation and return to state 0.

State 2. Print a dash followed by `\prev@cite`. Then print the current citation and return to state 0.

`\prev@cite`

```
1493 \let\prev@cite\@empty
```

`\prev@cite@cb` There's one further complication: Even though we're suppressing some of the citation numbers, we need to make sure that each citation is recorded in the `.aux` file. So, in case 2a, before we overwrite `\prev@cite`, we first invoke `\prev@cite@cb` to record the previous citation (if any).

```
1494 \def\prev@cite@cb{%
1495 \ifx\@prev@cite\@empty
1496 \else
1497 \begingroup
1498 \def\cite@print##1##2{%
1499 \cite@cb##1%
1500 }%
1501 \prev@cite
1502 \endgroup
1503 \fi
1504 }
```

`\cite@print`

```
1505 \def\cite@print#1#2{%
1506 \begingroup
1507 \let\CitePrintUndefined\CPU@normal
1508 \cite@cjs#1{#2}%
1509 \endgroup
1510 }
```

`\cite@dash` Ok, I lied. There was more than one further complication. Suppose that when we hit the end of the list, we're in state 2. We need to know whether to output

a dash or a comma. (For example, both the sequences [2, 3] and [1, 2, 3] will end in state 2 with *prevcite* = 3, but in the former case we want a comma before the 3 and in the latter case we want a dash.) So, rather than printing the dash explicitly, we use `\cite@dash` to keep track of whether a dash is needed.

```
1511 \let\cite@dash\@empty
```

```
\print@one@dash
```

```
1512 \def\print@one@dash{%
1513 \textendash \nopunct
1514 \let\cite@dash\@empty
1515 }
```

State 0, 1 and 2 each correspond to a different binding for `\cite@cj`. Here they are. The role of *prevnum* is played by `\@tempcnta`, with `\@tempcntb` assisting as *number* at times.

```
\cite@compress State 0:
```

```
1516 \def\cite@compress#1#2{%
1517 \cite@print#1{#2}%
1518 \@ifempty{#2}{%
1519 \safe@set\@tempcnta#1%
1520 \let\cite@cj\cite@compress@a
1521 }{%
1522 }
```

```
\cite@compress@a State 1:
```

```
1523 \def\cite@compress@a#1#2{%
1524 \@ifempty{#2}{%
1525 \advance\@tempcnta\@ne
1526 \safe@set\@tempcntb#1%
1527 \ifnum\@tempcnta=\@tempcntb
1528 \def\prev@cite{\cite@print#1{}}%
1529 \let\cite@cj\cite@compress@b
1530 \else
1531 \cite@print#1{}%
1532 \@tempcnta\@tempcntb
1533 \fi
1534 }{%
1535 \cite@print#1{#2}%
1536 \let\cite@cj\cite@compress
1537 }%
1538 }
```

```
\cite@compress@b State 2:
```

```
1539 \def\cite@compress@b#1#2{%
1540 \@ifempty{#2}{%
1541 \advance\@tempcnta\@ne
1542 \safe@set\@tempcntb#1%
1543 \ifnum\@tempcnta=\@tempcntb
```

```

1544 \let\cite@dash\print@one@dash
1545 \prev@cite@cb
1546 \def\prev@cite{\cite@print#1{}}%
1547 \else
1548 \cite@dash
1549 \prev@cite
1550 \let\prev@cite\@empty
1551 \cite@print#1{}%
1552 \@tempcnta\@tempcntb
1553 \let\cite@cj\cite@compress@a
1554 \fi
1555 }{%
1556 \cite@dash
1557 \prev@cite
1558 \let\prev@cite\@empty
1559 \cite@print#1{#2}%
1560 \let\cite@cj\cite@compress
1561 }%
1562 }

```

### 6.15.7 Munging the .aux file

`\bibcite` When processing the .aux file at begin-document, this is what `\bibcite` will do:

```

1563 \def\bibcite#1{\@xp\bibcite@a\csname b@#1\endcsname}

```

`\bibcite@a` *Arguments:*

```

#1 <- \b@citekey.
#2 <- {label}{ } or {author}{year}.

```

```

1564 \def\bibcite@a#1#2{%

```

Most of the time arg 1 will already be defined, by an earlier `\citedest` command in the .aux file. Then we just need to change the number.

```

1565 \ifx\relax#1%
1566 \gdef#1{\citesel 00#2{}}%
1567 \else
1568 \begingroup
1569 \@xp\bibcite@b\@xp#1#1{#2}%
1570 \endgroup
1571 \fi
1572 }

```

`\bibcite@b` *Arguments:*

```

#1 <- \b@citekey.
#2 <- \citesel.
#3 <- cited?.
#4 <- used?.
#5 <- label.

```

```

#6 <- year.
#7 <- backrefs.
#8 <- {newlabel}{newyear}.
1573 \def\bibcite@b#1#2#3#4#5#6#7#8{\gdef#1{\citesel#3#4#8{#7}}}

\citedest The \citedest command goes into the .aux file to provide back-reference sup-
port.
1574 \newcommand{\citedest}[1]{\@xp\cite@dest\csname b@#1\endcsname}

\cite@dest
1575 \def\cite@dest#1{%
1576 \ifx\relax#1%
1577 \gdef#1{\citesel 00{}{}{}}%
1578 \fi
1579 \@xp\cite@dest@b\@xp#1#1%
1580 }

\cite@dest@b Arguments:
#1 <- \b@citekey.
#2 <- \citesel.
#3 <- cited?.
#4 <- used?.
#5 <- label.
#6 <- year.
#7 <- backrefs.
#8 <- {more backrefs}.
1581 \def\cite@dest@b#1#2#3#4#5#6#7#8{%
1582 \@ifempty{#7}{%
1583 \def#1{\citesel #3#4{#5}{#6}{#8}}%
1584 }{%
1585 \gdef#1{\citesel #3#4{#5}{#6}{#7,#8}}%
1586 }%
1587 }

6.15.8 Back references

\ifBR@verbose
1588 \@ifundefined{ifBR@verbose}{\let\ifBR@verbose\iffalse \let\fi\fi}{%

\BackCite
1589 \let\BackCite@gobble

\back@cite
1590 \def\back@cite#1{%
1591 \ifBR@verbose
1592 \PackageInfo{backref}{back cite \string '\extr@cite#1'}%
1593 \fi
1594 \Hy@backout{#1}%
1595 }

```

`\print@backrefs` In an AMS-style bibliography, the backref info might follow the final period of the reference, or it might follow some *Mathematical Reviews* info, without a period.

```
1596 \def\print@backrefs#1{%
1597 \space\SentenceSpace\uparrow\csname br@#1\endcsname
1598 }
```

`\PrintBackRefs`

```
1599 \let\PrintBackRefs@gobble
```

### 6.15.9 hyperref and showkeys support

`\ar@hyperlink`

```
1600 \def\ar@hyperlink#1{\hyper@@link [cite]{}{cite.\extr@cite#1}{#1}}
```

`\ar@SK@cite`

```
1601 \def\ar@SK@cite#1{\@bsphack\@xp\SK@\@xp\SK@@ref\@xp{\extr@cite#1}\@esphack}
```

Turn off `hyperref` and `showkeys` support if those packages don't appear to be loaded.

```
1602 \AtBeginDocument{%
1603 \@ifundefined{hyper@@link}{%
1604 \let\ar@hyperlink\@firstofone
1605 \let\hyper@anchorstart@gobble
1606 \let\hyper@anchorend\relax
1607 }{}%
1608 \@ifundefined{SK@@label}{%
1609 \let\ar@SK@cite@gobble
1610 \let\SK@@label@gobble
1611 \let\SK@\@gobbletwo
1612 }{}%
1613 }
```

## 6.16 Lexical structure of names

Before we can begin parsing names, we need to give some thought to the lexical structure of names. For the remainder of this document, when we refer to a “name” and especially when we speak of a name as a macro argument, we assume that the only tokens contained in the name are

- letters and punctuation (i.e., characters with catcode 11 or 12),
- ties (the token `~13`),
- accent commands, such as `\` or `\k`,
- text symbol macros, such as `\i`, `\ae` or `\cprime`,
- grouping characters (braces).

In addition to their normal function of delimiting macro arguments, braces inside names have the following special functions:

1. They are used to indicate that multiple characters should be considered a single “compound” character when extracting initials. For example, `Yuri` becomes `Y.`, but `{Yu}ri` becomes `Yu`.

An important aspect of this use of braces is that it only applies to the first characters of a given name. As we’ll see below, this has important implications for our parsing code, which must preserve braces at the beginning of given names, but can be more cavalier with braces in other positions.

2. Spaces and commas are ordinarily interpreted as name separators, rather than name components. Similarly, periods and hyphens usually have a special interpretation. All these characters can be stripped of their special meanings by putting them within braces.

In practice, it might be possible to insert other tokens (such as macros) into names as long as they either (a) are non-expandable or (b) expand into a series of tokens of the above enumerated types. However, in such cases it will probably be safer to declare the macro in question as either a text accent or a text symbol.

### 6.16.1 Text accents

Syntactically, a text accent is a macro that takes a single, undelimited argument, i.e, it has a “prototype” of `macro:#1->`. Semantically, the implication is that it takes a letter (the *base*) as an argument and produces a glyph that for certain purposes can be considered equivalent to the base (see the discussion of stem comparison on page ??).<sup>3</sup>

`\amsrefs@textaccents` This will contain a list of accent commands in standard  $\text{\LaTeX}$  format (i.e., separated by the token `\do`). For example, after registering the `\"` and `\'` accents, it will contain

```
\do \"\do \'
```

```
1614 \let\amsrefs@textaccents\@empty
```

`\DeclareNameAccent` *Arguments:*  
`#1 <- accent.`

```
1615 \def\DeclareNameAccent{%
1616 \@lappend\amsrefs@textaccents
1617 }
```

Here are all the standard  $\text{\LaTeX}$  accents, as well as a few nonstandard accents from the `mathscinet` package.

```
1618 \DeclareNameAccent\"
1619 \DeclareNameAccent\'
1620 \DeclareNameAccent\
1621 \DeclareNameAccent\=
1622 \DeclareNameAccent\^
1623 \DeclareNameAccent\`
```

---

<sup>3</sup>Note that this is meant to be a pragmatic definition for the purposes of this package. No claim is made to greater generality.

```

1624 \DeclareNameAccent\~%
1625 \DeclareNameAccent\b
1626 \DeclareNameAccent\c
1627 \DeclareNameAccent\d
1628 \DeclareNameAccent\H
1629 \DeclareNameAccent\k
1630 \DeclareNameAccent\r
1631 \DeclareNameAccent\t
1632 \DeclareNameAccent\u
1633 \DeclareNameAccent\v

```

From mathscinet:

```

1634 \DeclareNameAccent\utilde
1635 \DeclareNameAccent\uarc
1636 \DeclareNameAccent\dudot
1637 \DeclareNameAccent\lfhook
1638 \DeclareNameAccent\udot
1639 \DeclareNameAccent\polhk
1640 \DeclareNameAccent\soft

```

`\etalchar` and `\etaltext` are sort of accent-like if you look at them in the right light.

```

1641 \DeclareNameAccent\etalchar
1642 %\DeclareNameAccent\etaltext

```

### 6.16.2 Text symbols

Syntactically, a text symbol is a macro with a empty parameter text, i.e., a prototype of `macro:->`. Semantically, it's a letter-like glyph that should not be considered equivalent to any other glyph or group of glyphs. In addition, it may exist in both upper- and lowercase variants, unlike text accents, where we consider the case to be an attribute of the base letter, not of the accent.<sup>4</sup>

`\amsrefs@textsymbols` This is analogous to `\amsrefs@textaccents` but a little more complicated due to the need to store lowercase equivalents. It consists of a list of double entries of the form

```
\do \symbol \do \lcsymbol
```

which means that `\symbol` is a text symbol whose corresponding lowercase version is `\lcsymbol`. (Note that nothing is implied about whether `\symbol` is to be considered as uppercase or lowercase.) For example, in

```
\do \ae \do \ae \do \OE \do \oe
```

the first four tokens indicate that `\ae` is a text symbol with lowercase equivalent `\ae`, while the last four tokens indicate that `\OE` is a text symbol with lowercase equivalent `\oe`. This scheme is somewhat redundant, but pleasingly simple.

This also duplicates some of the information in `\@uclclist`, but it seems safer to do this than to modify `\@uclclist`.

```
1643 \let\amsrefs@textsymbols\@empty
```

<sup>4</sup>As with text accents, this is not intended as a fully general definition.

```

\DeclareNameSymbol Arguments:
 #1 <- symbol.
 #2 <- lowercase.

1644 \def\DeclareNameSymbol#1#2{%
1645 \@lappend\amsrefs@textsymbols#1%
1646 \@lappend\amsrefs@textsymbols#2%
1647 \ifx#1#2\else
1648 \@lappend\amsrefs@textsymbols#2%
1649 \@lappend\amsrefs@textsymbols#2%
1650 \fi
1651 }

```

Here are the standard L<sup>A</sup>T<sub>E</sub>X and mathscinet text symbols.

Note that `\i` and `\j` are anomalous in being syntactically like text symbols, but semantically more like text accents.

```

1652 \DeclareNameSymbol\i\i
1653 \DeclareNameSymbol\j\j
1654 \DeclareNameSymbol\AE\ae
1655 \DeclareNameSymbol\OE\oe
1656 \DeclareNameSymbol\O\o
1657 \DeclareNameSymbol\DH\dh
1658 \DeclareNameSymbol\DJ\dj
1659 \DeclareNameSymbol\L\l
1660 \DeclareNameSymbol\NG\ng
1661 \DeclareNameSymbol\SS\ss
1662 \DeclareNameSymbol\TH\th

```

From mathscinet:

```

1663 \DeclareNameSymbol\Dbar\dbar
1664 \DeclareNameSymbol\lasp\lasp
1665 \DeclareNameSymbol\rasp\rasp
1666 \DeclareNameSymbol\cprime\cprime
1667 \DeclareNameSymbol\cdprime\cdprime
1668 \DeclareNameSymbol\bud\bud
1669 \DeclareNameSymbol\cydot\cydot

```

`~` can be considered a text symbol in much the same way that `\etalchar` can be considered an accent.

```

1670 \DeclareNameSymbol~~%

```

### 6.16.3 \edef-like macros for names

The following macros all behave sort of like `\edef`, in the sense that

```
\X@edef\foo{name}
```

defines `\foo` to be the result of expanding `name` and applying a certain transformation to it.

`\normalize@edef` This converts accents in the name to a normalized form where the accent and its argument are surrounded by braces. E.g., after

```
\normalize@edef\cs{P\'olya}
```

`\cs` will contain `P{\’o}lya`. (This might result in a redundant layer of braces if the original text contained, say, “`P{\’o}lya`”, but that’s ok.) This lets us extract the first  $n$  characters from a name by using TeX’s macro argument-gobbling mechanism without worrying that an accent will be separated from its base letter. As a bonus, it also replaces ties (`~`) by spaces.

```
1671 \def\normalize@edef#1#2{%
1672 \begingroup
1673 \@apply\auto@protect\amsrefs@textsymbols
1674 \@apply\wrap@accent\amsrefs@textaccents
 Redefine \@tabacckludge in case someone wants to use this with the inputenc
 package.
1675 \let\@tabacckludge\use@accent
1676 \let~\space
1677 \edef\@tempa{\def\@nx#1{#2}}%
1678 \@xp\endgroup
1679 \@tempa
1680 }
```

`\use@accent` This is identical to `\@nameuse` except for the addition of the `\string`, which, as per `ltoutenc.dtx`, guards against the eventuality that something like `’` might be active at the point of use. We don’t expect to find a `\bib` in the middle of a `tabbing` environment (do we?) so we

```
1681 \def\use@accent#1{\csname\string#1\endcsname}
```

`\wrap@accent` Here’s a wrapper macro that causes an accent to become auto-wrapping. E.g., after `\wrap@accent\’`, `\’o` will expand to `{\’o}`.

```
1682 \def\wrap@accent#1{%
1683 \def###1{{\@nx###1}}%
1684 }
```

`\lc@edef` This converts all the characters in a name to all lowercase, using the mapping defined by `\amsrefs@textsymbols`. So, after

```
\lc@edef\cs{P\’olya}
```

`\cs` will contain `p\’olya`. Note that accents are not wrapped and ties are passed through unmolested.

```
1685 \def\lc@edef#1#2{%
1686 \begingroup
1687 \let\@tabacckludge\use@accent %??
1688 \@apply\auto@protect\amsrefs@textaccents
1689 \@apply\lc@do\amsrefs@textsymbols
1690 \edef\@tempa{\lowercase{\def\@nx#1{#2}}}%
1691 \@xp\endgroup
1692 \@tempa
1693 }
```

`\lc@do` This is a slightly more complicated wrapper macro than previous ones. The first argument is a text symbol; the second argument is the lowercase variant of the symbol. If they're the same (i.e., the first argument is a lowercase text symbol), we `\auto@protect` it. Otherwise we define the first symbol to expand to the second.

```
1694 \def\lc@do#1\do#2{%
1695 \ifx#1#2%
1696 \auto@protect#1%
1697 \else
1698 \def#1{#2}%
1699 \fi
1700 }
```

`\purge@edef` Removes accents and braces from a name and converts ties to spaces, leaving only letters, punctuation and text symbols. For example,

```
\lc@edef\cs{P{\'}lya}
```

will put Polyá in `\cs`.

```
1701 \def\purge@edef#1#2{%
1702 \begingroup
1703 \@apply\auto@protect\amsrefs@textsymbols
1704 \let~\space
1705 \@apply\purge@accent\amsrefs@textaccents
1706 \let\@tabacckludge\@gobble
```

As mentioned above (page ??), `\i` and `\j` are semantically like text accents; hence, they require special treatment here.

```
1707 \def\i{i}%
1708 \def\j{j}%
1709 \edef\@tempa{#2}%
1710 \toks@\@emptytoks
1711 \@xp\purge@edef@ \@tempa \@nil
1712 \edef\@tempa{\def\@nx#1{\the\toks@}}%
1713 \@xp@endgroup
1714 \@tempa
1715 }
```

`\purge@edef@` Peek ahead so `\purge@edef@a` will know whether its argument was originally surrounded by braces.

```
1716 \def\purge@edef@{%
1717 \futurelet\@let@token
1718 \purge@edef@a
1719 }
```

`\purge@edef@a` Process a single “chunk” (i.e., one macro-argument’s worth) of the name.

```
1720 \def\purge@edef@a#1{%
```

If we’ve run into the `\@nil` terminator, we’re done.

```
1721 \ifx\@let@token\@nil
```

```
1722 \let\@tempa\@empty
1723 \else
```

Otherwise, if the argument was originally surrounded by braces, process it recursively before processing the remainder of the token stream.

```
1724 \ifx\@let\@token\@bgroup
1725 \def\@tempa{%
1726 \purge@edef@ #1\@nil
1727 \purge@edef@
1728 }%
1729 \else
```

If the argument is a single unbracketed token, just copy it into the output.

```
1730 \add@toks@{#1}%
1731 \let\@tempa\purge@edef@
1732 \fi
1733 \fi
1734 \@tempa
1735 }
```

`\purge@accent` This is similar to `\wrap@accent` but it removes the accent command (and possibly a layer of braces surrounding the accent's argument).

```
1736 \def\purge@accent#1{%
1737 \def#1##1{##1}%
1738 }
```

### 6.17 Name parsing

Parsing names is somewhat complicated because parts of the name can (in principle) be empty (G=given, S=surname, J=jr):

```
author={Doe, John, Jr.}: G={John} S={Doe} J={Jr.}
author={Doe, John}: G={John} S={Doe} J={}
author={Doe, , Jr.}: G={} S={Doe} J={Jr.}
author={Doe}: G={} S={Doe} J={}
author={, John, Jr.}: G={John} S={} J={Jr.}
author={, John}: G={John} S={} J={}
author={, , Jr.}: G={} S={} J={Jr.}
author={}: G={} S={} J={}
```

Not all of these forms are legal, of course, but that's no excuse for not parsing them correctly.

We also want to be somewhat lenient about the placement of spaces:

```
author={ Doe,John,Jr.}: G={John} S={Doe} J={Jr.}
```

However, because one must have some standards, we assume there are no spaces in the following positions in the input:

1. before periods,
2. before commas,
3. at the end of the name,

## 4. before or after hyphens.

Thus, we make no attempt to compensate for the misplaced spaces in examples like these:

```
author={Doe , J . , Jr. } : G={J .} S={Doe } J={Jr. }
author={Doe, J. - M.} : G={J. - M.} S={Doe} J={}
```

Also, unless we are generating initials, we don't try to normalize spaces *after* periods:

```
author={Doe, J.M.} : G={J.M.} S={Doe} J={}
(not G={J. M.})
```

Finally, since we allow authors to group together characters that should be treated as a single unit, we need to be careful to preserve the author's markup in cases like these:

```
author={Doe, {Yu}ri} : G={{Yu}ri} S={Doe} J={}
author={Doe, {Yu}} : G={{Yu}} S={Doe} J={}
```

This is harder than it seems. For example, consider a naive implementation that uses delimited arguments to pull the name apart:

```
\def\parsename#1,#2\@nil{%
 \def\bib'surname{#1}%
 \def\bib'given{#2}%
}
```

```
\parsename Doe, {Yu}ri\@nil
```

Unfortunately, this results in the space after the comma becoming part of `\bib'given`: “ {Yu}ri”.

Our next thought would be to modify the definition slightly to trick `TEX` into gobbling the space:

```
\def\parsename#1,#2#3\@nil{%
 \def\bib'surname{#1}%
 \def\bib'given{#2#3}%
}
```

Now the space is gone, but—surprise!—so are the braces: “Yuri”. In addition, this approach makes it difficult to handle empty name parts correctly.

To sidestep these problems, instead of blindly gobbling macro arguments, we use `\futurelet` to look ahead at certain strategic moments so we can take the appropriate action (see `\get@namepart@d-f`). We only really care about preserving braces at the start of names (page ??), which simplifies things somewhat.

`\name@split` `\name@split` parses a name into its three parts and stores them in `\bib'surname`, `\bib'given` and `\bib'jr`. If the `initials` option is in force, it also extracts the initials from the given name and stores them in `\bib'initials`.

It expects the name to be parsed to be terminated by `\@nil` and to contain at least three commas. Thus the usual way to invoke it is

```
\name@split <name>,,, \@nil
```

\name@split just uses \get@namepart to peel off the surname and then passes control to \name@split@given. (Note the spiffy continuation-passing programming style.)

```
1739 \def\name@split{%
1740 \get@namepart\bib'surname\name@split@given
1741 }
```

\name@split@given Pretty much the same, *mutatis mutandis*...

```
1742 \def\name@split@given{%
1743 \get@namepart\bib'given\name@split@jr
1744 }
```

\name@split@jr And again...

```
1745 \def\name@split@jr{%
1746 \get@namepart\bib'jr\name@split@finish
1747 }
```

\name@split@finish We have all three parts now. Do some consistency checking, extract the initials from the given name, and then call \@nilgobble to remove anything (such as extra commas) left on the stack.

```
1748 \def\name@split@finish{%
1749 \ifx\bib'surname\@empty \EmptyNameWarning \fi
```

Theoretically, we could try to check for uninverted names here, but only at the risk of producing spurious warnings when the name really does only have one part (author={Arvind}).

A possible solution: Now that we have the *inverted* attribute, we could issue a warning if the given name is empty and the family name contains a space. I'm sure someone could find valid input that would still generate a spurious warning, but this would take care of the most common cases. This bears more thinking about.

```
1750 %% \ifx\@empty\bib'given
1751 %% \NameCheck \bib'surname ??\@nil
1752 %% \else
1753 \extract@initials\bib'given
1754 %% \fi
1755 \@nilgobble
1756 }
```

\get@namepart Now for the fun part. \get@namepart takes two arguments. The first (the destination) should be a control sequence; the second (the continuation) will normally also be a control sequence, though technically we only require that it be a single token. \get@namepart scans everything up to the next level-0 comma, places it in the destination, and then calls the continuation.

```
1757 \def\get@namepart#1#2{%
```

Save the destination in `\toks@` and the continuation in `\@temptokena`. It's unfortunate that this trashes the previous contents of those token lists (as well as the contents of `\@tempa` later on), but preliminary attempts to rewrite the code to leave the calling environment unchanged were not encouraging.

```
1758 \toks@{#1}%
1759 \@temptokena{#2}%
1760 \get@namepart@a
1761 }
```

`\get@namepart@a` Now peek ahead at the next token in the stream and call `\get@namepart@b` to examine it.

```
1762 \def\get@namepart@a{%
1763 \futurelet\@let@token
1764 \get@namepart@b
1765 }
```

`\get@namepart@b` If the next token is a space token, we want to delete it. Otherwise we're ready to read the name.

```
1766 \def\get@namepart@b{%
1767 \ifx\@let@token\@sptoken
1768 \xp\get@namepart@c
1769 \else
1770 \xp\get@namepart@d
1771 \fi
1772 }
```

`\get@namepart@c` The next token is a space; we delete it and restart `\get@namepart@a`, in case there are multiple spaces.

```
1773 \def\get@namepart@c{%
1774 \after@deleting@token\get@namepart@a
1775 }
```

`\get@namepart@d` We're at the beginning of the name part. However, there are still two special cases we have to watch out for. First, the next token might be a comma, meaning that this name part is empty. Second, the next token might be an open brace (`{`), which we have to be sure to copy into the destination. So, we peek ahead again before proceeding.

```
1776 \def\get@namepart@d{%
1777 \futurelet\@let@token
1778 \get@namepart@e
1779 }
```

`\get@namepart@e` If the next token is a comma, it means the name part is empty; so, we set the destination to an empty list and then arrange to execute the continuation after deleting the comma. Otherwise we call `\get@namepart@f` to read a non-empty name, leaving `\@let@token` undisturbed so that `\get@namepart@f` knows what's coming up.

```

1780 \def\get@namepart@e{%
1781 \ifx\@let@token,%
1782 \xp\let\the\toks@\@empty
1783 \edef\@tempa{%
1784 \@nx\after@deleting@token\the\@temptokena
1785 }%
1786 \xp\@tempa
1787 \else
1788 \xp\get@namepart@f
1789 \fi
1790 }

```

`\get@namepart@f` We know whether or not the name begins with a brace, but we don't know if the corresponding group contains the entire name or only part of it. By reading the name as two arguments, we can handle all cases correctly.<sup>5</sup>

Note that the arguments are not expanded.

```

1791 \def\get@namepart@f#1#2,{%
1792 \ifx\@let@token\bgroup
1793 \xp\def\the\toks@{#1#2}%
1794 \else
1795 \xp\def\the\toks@{#1#2}%
1796 \fi
1797 \the\@temptokena
1798 }

```

`\EmptyNameWarning` Or translator or contributor or...

```

1799 \def\EmptyNameWarning{\amsrefs@warning{Empty contributor name}}

```

## 6.18 Extracting initials

Extracting initials from the author's given name is tricky because of the numerous special cases that need to be handled. Consider the following examples, some of which are admittedly contrived:

```

author={Arvind}: I={ }
author={Bing, R H}: I={R H}
author={Harish, \ 'Etienne}: I={É.}
author={Harish, \ 'E.}: I={É.}
author={Harish, \ {E}.}: I={É.}
author={Harish, {\ 'E}.}: I={É.}
author={Harish, \ 'E}: I={É}
author={Harish, \ 'Etienne-P\ ^{\i }erre}: I={É.-P.}
author={Jones, David}: I={D.}
author={Jones, David-Michael}: I={D.-M.}
author={Katzenbach, Nicholas {deB}elleville}: I={N. deB.}
author={Katzenbach, Nicholas deB.}: I={N. deB.}

```

<sup>5</sup>More or less. If the second argument is brace-delimited, the braces will be lost. But as mentioned above (page ??), we don't really care.

```
author={Matiyasevich, {Yu}ri}: I={Yu.}
author={Matiyasevich, {Yu}}: I={Yu}
author={Matiyasevich, Yu.}: I={Yu.}
```

When processing initials, we loosen our strictures on spaces inside the given name by not requiring spaces after periods and tolerating them around hyphens and after the name:

```
author={Jones, D.M.}: I={D. M.}
author={Jones, David - Michael}: I={D.-M.}
author={Jones, David , Jr.}: I={D.}
```

(Strictly speaking, only the support for the first of these examples was a deliberate design decision; the other two are side-effects of the implementation. In any case, toleration of these quirks is in no way an endorsement of them, especially since they may make it more difficult for third-party software to correctly process bibliography entries.)

### 6.18.1 The algorithm

As a running example, consider the following contrived input:

```
\'E.-P\^{\i}erre J.K. M
```

which we want to turn into “É.-P. J. K. M”.

We precede by stages.

1. Normalize the name by surrounding accents and their arguments by braces:

```
{\'E}.-P{\^{\i }erre J.K. M
```

We also replace ~s by spaces at this stage.

2. Replace each hyphen (-) by “\ini@hyphen”:

```
{\'E}. \ini@hyphen P{\^{\i }erre J.K. M
```

3. Add a space after each period:

```
{\'E}. \ini@hyphen P{\^{\i }erre J. K. M
```

4. Now we have the name as a list of space-separated components. (In our example, the components are “{\'E}.”, “\ini@hyphen”, “P{\^{\i }erre”, “J.”, “K.”, and “M”.) We loop through the components and replace each one by its “initialized” form. There are four cases:

- (a) The component ends in a period. Copy it and add the token ~. (In our example, these are the components “{\'E}.”, “J.” and “K.”.)
- (b) The component consists of a single (possibly compound) character without a period. Again, copy it and add ~. (In our example, this is the component “M”.)
- (c) The component is the token \ini@hyphen. Copy it.
- (d) The component consts of two or more (possibly compound) characters without a period (e.g., “P{\^{\i }erre”). Copy the first character and add the tokens .~.

5. The token list generated above will end with an unwanted ~. Delete it.

The end result is

```
{\’E}.~\ini@hyphen P.~J.~K.~M
```

which, when typeset, does indeed produce “É.-P. J. K. M”.<sup>6</sup>

### 6.18.2 The implementation

`\extract@initials` This is pretty straightforward.

```
1800 \def\extract@initials#1{%
1801 \begingroup
1802 \auto@protect\ini@hyphen
1803 \auto@protect\nobreakspace
1804 \let~\relax
1805 \@apply\auto@protect\amsrefs@textsymbols
1806 \@apply\auto@protect\amsrefs@textaccents
1807 \normalize@edef\@tempa{#1}%
1808 \ifx\@tempa\@empty
1809 \else
```

It would be nice if `\process@hyphens` and `\process@dots` commuted, and they almost do. However, suppose you have the (admittedly contrived) name `Yu.-{Yu}`, which should be turned into “Yu.-Yu”. If `\process@dots` is applied first, the braces around the second “Yu” get removed, so the output is “Yu.-Y.”. (Even worse would be `P.-\’E`, which would produce “P.-?”)

```
1810 \process@hyphens\@tempa
1811 \process@dots\@tempa
1812 \process@names\@tempa
1813 \@chomp\@tempa{~}%
1814 \fi
1815 \edef\@tempa{\def\@nx\bib’initials{\@tempa}}%
1816 \@xp\endgroup
1817 \@tempa
1818 }
```

`\ini@hyphen` The `\unskip` removes the space at the end of a potential (and probable) preceding `~`, but leaves the `\nobreak` penalty.

```
1819 \def\ini@hyphen{\unskip-\nobreak}
```

`\process@hyphens` This follows the same general pattern as `\get@namepart`, but with an extra layer of grouping to avoid unwanted side-effects. Otherwise, it uses the same parsing techniques.

One difference is that there is no explicit continuation: instead, we iterate by repeatedly calling `\process@one@hyphen@d` until we run into the `\@nil` marker.

```
1820 \def\process@hyphens#1{%
1821 \begingroup
1822 \toks@\@emptytoks
1823 \@xp\process@one@hyphen #1-\@nil
1824 \edef\@tempa{\the\toks@}%
```

<sup>6</sup>Tying all the characters together is potentially undesirable when, as in the example, there are a large number of pieces in the given name.

Because of the - we have to stick in as a delimiter above, `\process@one@hyphen` will always generate unwanted code at the end of the name. We now delete it. (This also has the necessary side-effect of expanding the `\space` macros into space characters.)

```
1825 \@chomp\@tempa{ \ini@hyphen\space}%
1826 \edef\@tempa{\def\@nx#1{\@tempa}}%
1827 \@xp\endgroup
1828 \@tempa
1829 }
```

`\process@one@hyphen` Cf. `\get@namepart@a`.

```
1830 \def\process@one@hyphen{%
1831 \futurelet\@let@token
1832 \process@one@hyphen@a
1833 }
```

`\process@one@hyphen@a` Cf. `\get@namepart@b` and `\extract@initial@a`.

The tests for `\@nil` and - here are purely to supply better error recovery. Without them, a hyphen at the end of the given name (e.g. `author={Doe, John-}`) would produce a very mysterious error message. Since it's unlikely the hyphen really belongs there, we delete it, but we also issue a warning to the author. (It will still show up as part of the full given name, though.)

We borrow `\fsa@n` from `rkeyval` to keep track of the appropriate next action.

```
1834 \def\process@one@hyphen@a{%
1835 \ifx\@let@token\@nil
1836 \let\fsa@n\@gobble
1837 \else
1838 \ifx\@let@token -%
1839 \TrailingHyphenWarning
1840 \let\fsa@n\process@one@hyphen@b
1841 \else
1842 \ifx\@let@token\@sptoken
1843 \let\fsa@n\process@one@hyphen@b
1844 \else
1845 \let\fsa@n\process@one@hyphen@c
1846 \fi
1847 \fi
1848 \fi
1849 \fsa@n
1850 }
```

`\process@one@hyphen@b` Cf. `\get@namepart@c`.

```
1851 \def\process@one@hyphen@b{%
1852 \after@deleting@token\process@one@hyphen
1853 }
```

`\process@one@hyphen@c` Cf. `\get@namepart@f`.

```
1854 \def\process@one@hyphen@c#1#2-{%
1855 \ifx\bgroup\@let@token
1856 \add@toks@{#1}#2 \ini@hyphen\space}%
1857 \else
1858 \add@toks@{#1#2 \ini@hyphen\space}%
1859 \fi
1860 \futurelet\@let@token
1861 \process@one@hyphen@d
1862 }
```

`\process@one@hyphen@d` Here we just check for `\@nil` and terminate if we detect it. Otherwise, we start over.

```
1863 \def\process@one@hyphen@d{%
1864 \ifx\@let@token\@nil
1865 \xp\@gobble
1866 \else
1867 \xp\process@one@hyphen
1868 \fi
1869 }
```

`\TrailingHyphenWarning` Or translator or contributor or...

```
1870 \def\TrailingHyphenWarning{%
1871 \amsrefs@warning{Trailing hyphen deleted from name}%
1872 }
```

`\process@dots` This is almost completely parallel to `\process@hyphens`.

```
1873 \def\process@dots#1{%
1874 \begingroup
1875 \toks@\@emptytoks
1876 \xp\process@one@dot #1.\@nil
1877 \edef\@tempa{\the\toks@}%
1878 \@chomp\@tempa{. }%
```

Since it's legitimate for names to end in periods, we might still have an unwanted space at the end of the name, so we delete it too.

```
1879 \@chomp\@tempa{ }%
1880 \edef\@tempa{\def\@nx#1{\@tempa}}%
1881 \xp\endgroup
1882 \@tempa
1883 }
```

`\process@one@dot`

```
1884 \def\process@one@dot{%
1885 \futurelet\@let@token
1886 \process@one@dot@a
1887 }
```

`\process@one@dot@a` This is a bit different from `\process@one@hyphen@a` since we expect names sometimes to end in a period—or even two periods—not least because of the `.` we add as a delimiter when invoking `\process@one@dot`.

```

1888 \def\process@one@dot@a{%
1889 \ifx\@let@token .%
1890 \def\fsa@n{\after@deleting@token\process@bare@dot}%
1891 \else
1892 \ifx\@let@token\@sptoken
1893 \let\fsa@n\process@one@dot@b
1894 \else
1895 \let\fsa@n\process@one@dot@c
1896 \fi
1897 \fi
1898 \fsa@n
1899 }

```

`\process@bare@dot`

```

1900 \def\process@bare@dot{%
1901 \add@toks@{. }%
1902 \futurelet\@let@token
1903 \process@one@dot@d
1904 }

```

`\process@one@dot@b`

```

1905 \def\process@one@dot@b{%
1906 \after@deleting@token\process@one@dot
1907 }

```

`\process@one@dot@c`

```

1908 \def\process@one@dot@c#1#2.{%
1909 \ifx\bgroup\@let@token
1910 \add@toks@{#1}#2. }%
1911 \else
1912 \add@toks@{#1#2. }%
1913 \fi
1914 \futurelet\@let@token
1915 \process@one@dot@d
1916 }

```

`\process@one@dot@d`

```

1917 \def\process@one@dot@d{%
1918 \ifx\@let@token\@nil
1919 \@xp\@gobble
1920 \else
1921 \@xp\process@one@dot
1922 \fi
1923 }

```

`\process@names` This is very similar to `\process@hyphens` and `\process@dots`, but with a couple of twists, as noted below.

```
1924 \def\process@names#1{%
1925 \begingroup
1926 \toks@\emptytoks
1927 \@xp\extract@initial #1 \@nil
1928 \edef\@tempa{\def\@nx#1{\the\toks@}}%
1929 \@xp\endgroup
1930 \@tempa
1931 }
```

`\extract@initial` Scan through the token stream replacing words by their initials until we hit the terminating `'11`

```
1932 \def\extract@initial{%
1933 \futurelet\@let@token
1934 \extract@initial@a
1935 }
```

`\extract@initial@a` As with `\process@one@hyphen@a`, the test for `'11` here is purely to provide better recovery, this time in case the given name has a trailing space (e.g, `author={Doe, John }`). But since we're just deleting whitespace, we don't bother issuing a warning.

```
1936 \def\extract@initial@a{%
1937 \ifx\@let@token\@nil
1938 \let\fsa@n\@gobble
1939 \else
1940 \ifx\@let@token\@sptoken
1941 \let\fsa@n\extract@initial@b
1942 \else
1943 \let\fsa@n\extract@initial@c
1944 \fi
1945 \fi
1946 \fsa@n
1947 }
```

`\extract@initial@b`

```
1948 \def\extract@initial@b{%
1949 \after@deleting@token\extract@initial
1950 }
```

`\extract@initial@c` Here, instead of just copying the name, we extract its initials and copy those.

```
1951 \def\extract@initial@c#1#2 {%
1952 \ifx\@let@token\bgroup
```

Note that we double-brace the first argument to avoid having to test `\@let@token` again inside `\@extract@initial`.

```
1953 \@extract@initial {{#1}}#2\@nil
1954 \else
1955 \@extract@initial #1#2\@nil
```

```

1956 \fi
1957 \futurelet\@let@token
1958 \extract@initial@d
1959 }

```

`\extract@initial@d`

```

1960 \def\extract@initial@d{%
1961 \ifx\@let@token\@nil
1962 \exp\@gobble
1963 \else
1964 \exp\extract@initial
1965 \fi
1966 }

```

`\@extract@initial` This handles the four cases mentioned on page ??.

```

1967 \def\@extract@initial#1#2\@nil{%
1968 \ifx\ini@hyphen#1%
1969 \add@toks@\ini@hyphen}%
1970 \else
1971 \in@{.\@nil}{#1#2\@nil}% Look for a period at the end of the name
1972 \ifin@
1973 \add@toks@{#1#2~}%
1974 \else
1975 \count@chars\@tempcnta{#1#2}%
1976 \ifnum\@tempcnta > \@ne
1977 \add@toks@{#1.~}%
1978 \else
1979 \add@toks@{#1~}%
1980 \fi
1981 \fi
1982 \fi
1983 }

```

`\count@chars` This sets its first argument (which is assumed to be a count register) to the number of characters in the second argument. Compound characters are counted as a single character.

```

1984 \def\count@chars#1#2{%
1985 \begingroup
1986 \@tempcnta\z@
1987 \@count@chars#2\@nil
1988 \edef\@tempb{#1=\the\@tempcnta\relax}%
1989 \@xp\endgroup
1990 \@tempb
1991 }

```

`\@count@chars`

```

1992 \def\@count@chars#1{%
1993 \ifx #1\@nil
1994 \else

```

```

1995 \advance\@tempcnta\@ne
1996 \exp\@count@chars
1997 \fi
1998 }

```

## 6.19 Generating alphabetic labels

### 6.19.1 The algorithm

Like Gaul, an alphabetic label is divided into three parts.

1. The author part. In the simplest case, this is formed by extracting the first character of each word of each last name of each author. Thus, if there were two authors with last names “Vaughan Williams” and “Tallis”, the author part would be “VWT”.

If there are more than four authors, only the first three names are used, and a superscript “+” is appended to represent the elided names. Similarly, if an author name is “others”, it is replaced by a superscript “+” and any following author names (of which there shouldn’t be any) are ignored.

Finally, if there is only one author and the author’s last name consists of a single word, the first three characters of that name are used.

2. The year part. If the `y2k` option is in force, or if the year is less than 1901, the entire year is used. Otherwise the last two digits of the year are used.<sup>7</sup> The combination of author part and year part will be referred to as the *stem*.

3. The suffix. If two or more items have the same stems, a suffix consisting of a lowercase latin letter will be appended to each label to make it unique.

This third part is more subtle than it might first appear. First, case is ignored when comparing stems, so that, for example, “Ahl1999” and “AHL1999” are considered identical. Second, existing practice (in English, at least), is to ignore diacritics so that, for example, “Ahl1999” and “Ähl1999” are considered identical.

Note that when checking for duplicate stems, we assume that bibliography items appear sorted by label, which means that all items with the same stem will be adjacent. This means we can use the naive algorithm (check to see if the current item has the same stem as the previous item and, if so, append a suffix) to detect clashes. This sorting will be done automatically by `amsxport`, but the document author is responsible for ensuring the appropriate order if `amsxport` is not used. This is why it’s an error to mix the `alphabetic` and `citation-order` options.

### 6.19.2 The implementation

```

1999 \let\previous@stem\@empty
2000 \let\current@stem\@empty

2001 \let\previous@year\@empty
2002 \let\current@year\@empty

```

---

<sup>7</sup>Years with more than 4 digits are not currently handled correctly. *Caveat lector*.

```

\append@to@stem
2003 \def\append@to@stem{\global\@concat\current@stem}

\generate@alphalabel
2004 \def\generate@alphalabel{%
 If the user supplied an explicit label field, we use it. Otherwise, we generate
 our own.
2005 \ifx\bib'label\@empty
2006 \begingroup
 We begin by saving the previous stem and initializing the current stem to the
 empty string.
2007 \global\let\previous@stem\current@stem
2008 \global\let\current@stem\@empty
 The list of primary contributors is available to us in \current@primary in the
 form
 \name{Last1,First1} \name{Last2,First2} ... \name{Lastn,Firstn}
 We will be executing this list multiple times with various definitions of \name.
 So the first thing we want to do is establish a safe environment and normalize
 the names.
2009 \@apply\auto@protect\amsrefs@textsymbols
2010 \@apply\auto@protect\amsrefs@textaccents
2011 \auto@protect\name
2012 \auto@protect\etaltext
2013 \normalize@edef\@tempa\current@primary
 Now we count the number of authors in the list and invoke the appropriate
 macro to calculate the author part of the reference label.
2014 \get@numberof\@tempcnta\name\@tempa
2015 \calc@author@part
 Next append the year part.
2016 \append@label@year
 At this point, the \current@stem is complete and we're ready to determine
 what (if any) suffix is needed to disambiguate it from the previous label.
2017 \calc@alpha@suffix
 We have all the pieces now. Arrange to end the current group and then define
 \bib@label in the enclosing group. (This keeps \bib@label from being defined
 outside the group started by \bib@start. This isn't strictly necessary, but it
 provides a bit of compartmentalization.)
2018 \edef\@tempa{%
2019 \def\@nx\bib'label{%
2020 \current@stem
2021 \alpha@label@suffix
2022 }%
2023 }%

```

```

2024 \exp\endgroup
2025 \@tempa
2026 \fi
2027 }

```

#### \calc@author@part

```

2028 \def\calc@author@part{%
2029 \ifnum \@tempcnta = 1
2030 \exp\@oneauthorlabel\exp{\@tempa}%
2031 \else
2032 \exp\@multiauthorlabel\exp{\@tempa}%
2033 \fi
2034 }

```

`\@firstone` This extracts the first character from a properly prepared author name (i.e., one in which accents are properly wrapped).

```
2035 \def\@firstone#1{\@car#1\@empty\@nil}
```

`\@firstthree` And this extracts the first three characters.

```
2036 \def\@firstthree#1{\@carcube#1\@empty\@empty\@empty\@nil}
```

#### \@nametoken

```
2037 \let\@nametoken\@firstone
```

#### \hyph@to@space

```
2038 \def\hyph@to@space#1-#1 \hyph@to@space}
```

`\@marknames` Since we have a ' with funny catcode already, let's use it (being able to easily put a space after the ' makes things easier).

```

2039 \def\@marknames#1{%
2040 \@ifnotempty{#1}{\surround@names#1 ' }%
2041 }

```

#### \surround@names

```

2042 \def\surround@names#1 {%
2043 \ifx '#1%
2044 \else
2045 \@nx\@nametoken{#1}%
2046 \exp\surround@names
2047 \fi
2048 }

```

#### \extract@surnames

```

2049 \def\extract@surnames#1#2{%
2050 \get@namepart\@tempb\@nilgobble #2,\@nil
2051 \edef\@tempb{\@nx\@marknames{\exp\hyph@to@space\@tempb\@gobble-}}%
2052 \edef#1{\@tempb}%
2053 }

```

`\@oneauthorlabel` This is the easy case.

```

2054 \newcommand{\@oneauthorlabel}[1]{%
2055 \def\name##1{%
2056 \extract@surnames\@tempa{##1}%
2057 \get@numberof\@tempcnta\@nametoken\@tempa
2058 \ifnum \@tempcnta = 1
2059 \let\@nametoken\@firstthree
2060 \fi
2061 \append@to@stem{\@tempa}%
2062 }%
2063 #1%
2064 }

```

`\@threeauthors`

```

2065 \def\@threeauthors\name#1\name#2\name#3#4\@empty{%
2066 \name{#1}\name{#2}\name{#3}%
2067 \append@to@stem{\etalchar{+}}%
2068 }

```

`\@multiauthorlabel`

```

2069 \newcommand{\@multiauthorlabel}[1]{%
2070 \def\name##1{%
2071 \ifx\etaltext ##1%
2072 \def\@tempa{\@nx\etalchar{+}}%
2073 \let\name\@gobble
2074 \else
2075 \extract@surnames\@tempa{##1}%
2076 \fi
2077 \append@to@stem{\@tempa}%
2078 }%
2079 \ifnum \@tempcnta > 4 \exp \@threeauthors \fi
2080 #1\@empty
2081 }

```

`\etalchar`

```

2082 \newcommand{\etalchar}[1]{${#1}$}

```

`\year@short` For alphanumeric labels, we want to extract the last 2 digits of the year. Here's a way to do that, assuming a 4-digit year.

```

2083 \def\year@short#1#2#3#4\@nil{#3#4}

```

`\append@label@year`

```

2084 \def\append@label@year{%
2085 \safe@set\@tempcnta\bib@year
2086 \edef\bib@citeyear{\the\@tempcnta}%
2087 \append@to@stem{%
2088 \ifx\bib@year\@empty
2089 \else
2090 \exp\year@short \bib@citeyear \@nil

```

```

2091 \fi
2092 }%
2093 }

2094 \let\alpha@label@suffix\@empty
2095
2096 \newcount\alpha@suffix
2097 \alpha@suffix\@ne
2098 \let\@suffix@format\@alph

```

\calc@alpha@suffix

```

2099 \def\calc@alpha@suffix{%
2100 \@tempwafalse
2101 \compare@stems\previous@stem\current@stem
2102 \ifsame@stems

```

Under the alphabetic option, \previous@year and \current@year will always be the same (namely, both will be empty), but including the test allows this code to work with the author-year option as well.

```

2103 \ifx\previous@year\current@year
2104 \@tempwattrue
2105 \fi
2106 \fi
2107 \if@tempswa
2108 \global\advance\alpha@suffix\@ne
2109 \edef\alpha@label@suffix{\@suffix@format\alpha@suffix}%
2110 \ifnum\alpha@suffix=\tw@
2111 \immediate\write\@auxout{%
2112 \string\ModifyBibLabel{\prev@citekey}%
2113 }%
2114 \fi
2115 \else
2116 \let\alpha@label@suffix\@empty
2117 \global\alpha@suffix\@ne
2118 \@xp\ifx \csname b@\current@citekey @suffix\endcsname \relax
2119 \else
2120 \edef\alpha@label@suffix{\@suffix@format\alpha@suffix}%
2121 \fi
2122 \fi
2123 }

```

\ifsame@stems

```

2124 \newif\ifsame@stems

```

\compare@stems

```

2125 \def\compare@stems#1#2{%
2126 \begingroup
2127 \purge@edef\@tempa{#1}%
2128 \purge@edef\@tempb{#2}%
2129 \lc@edef\@tempa{\@tempa}%

```

```

2130 \lc@edef\@tempb{\@tempb}%
2131 \ifx\@tempa\@tempb
2132 \def\@tempa{\same@stemstrue}%
2133 \else
2134 \def\@tempa{\same@stemsfalse}%
2135 \fi
2136 \@xp@endgroup
2137 \@tempa
2138 }

```

`\ModifyBibLabel`

```

2139 \def\ModifyBibLabel#1{%
2140 \global\@xp\let\csname b@#1@suffix\endcsname\@empty
2141 }

```

## 6.20 Generating short alphabetic labels

This style for alphabetic labels is somewhat simpler than the regular alphabetic style. The stem consists only of an author part without a year part. The author part is formed in the same way, except that even when there is only a single author with a one-word last name, only the first letter of the name is used, not the first three. Finally, the suffix used to disambiguate identical stems is numeric rather than alphabetic.

See section ?? on page ?? for the implementation.

## 6.21 Formatting series

The `\PrintSeries` command prints a list of objects in series form. The essential idea is to produce something like “A, B, and C” when we are given three elements “A”, “B”, and “C”, with suitable variations in the punctuation and other intervening material depending on the number of elements.

More precisely, we can envision `\PrintSeries` being called as

$$\text{\PrintSeries}\{S\}\{i_1\}\{i_2\}\{i_3\}\{E\}\{\text{\do}\{T_1\} \dots \text{\do}\{T_n\}\}$$

where  $S$  and  $E$  are material to be interpolated before the start and after the end of the list, respectively,  $i_1, \dots, i_3$  are material to be interpolated between the elements, and the final argument is a list of indeterminate length where each element consists of a macro and its argument. If there are exactly two elements,  $i_1$  is inserted between them; otherwise,  $i_2$  is inserted between each pair of items except for the last pair, where  $i_3$  is inserted. Thus,

| $n$ | output                            |
|-----|-----------------------------------|
| 1   | $S T_1 E$                         |
| 2   | $S T_1 i_1 T_2 E$                 |
| 3   | $S T_1 i_2 T_2 i_3 T_3 E$         |
| 4   | $S T_1 i_2 T_2 i_2 T_3 i_3 T_4 E$ |

and so forth. For example, a standard comma-separated list could be formatted by

$$\text{\PrintSeries}\{\text{\and}\}\{, \}\{, \text{\and}\}\{\dots\}$$

That is the simple case but in practice there are additional complications. What if user-supplied line breaks have to be supported at the boundaries between elements? What if in addition to adding material between elements we also want to apply some handy function to each element (e.g., `\textsc`)? Even worse, what if we want the function to be different depending on the position of the element in the list? Indeed if this did not happen to be the case with the current application I would not have gone to the extra trouble of supporting it. But if it must be so, then the output that we need from a list `\do{A}\do{B}...` is

```
f0{A}
f0{A} p1 i1 f1{B}
f0{A} p2 i2 f2{B} p3 i3 f3{B}
```

and so on, where

- $f_n$  is a macro taking one argument,
- $p_n$  is punctuation—material that must precede a line break if one occurs at this boundary,
- $i_n$  other interpolated material, as before.

To reduce the number of distinct required objects we decree that each element will get braces wrapped around it as a matter of course; then it is possible for  $f_1$ ,  $f_2$ ,  $f_3$  to be assimilated onto the tail end of  $i_1$ ,  $i_2$ ,  $i_3$ . Since we also have to specify the macro that delimits the elements of the list, we end up with the following rather formidable signature:

```
\PrintSeries{\m} {f0} {p1}{i1f1} {p2}{i2f2} {p3}{i3f3}
{S} {\m{T1}...\m{Tn}} {E}
```

and our comma-separated list example becomes

```
\PrintSeries{\do}{ }{ and } {,}{ } {,}{ and } {...}{ }
```

`\series@index` First we define a dedicated count register to be used in tracking the ordinal number of the item currently being processed.

```
2142 \newcount\series@index
```

```
\PrintSeries
```

```
2143 \def\PrintSeries#1#2#3#4#5#6#7#8{%
2144 \begingroup
2145 \def\series@add@a{#2}%
2146 \def\series@add@b{\SwapBreak{#3}#4}%
2147 \def\series@add@c{\SwapBreak{#5}#6}%
2148 \def\series@add@d{\SwapBreak{#7}#8}%
2149 \def\series@add@e{\SwapBreak{#7}}%
2150 \PrintSeries@a{#1}%
2151 }
```

`\PrintSeries@a` For `\PrintSeries@a` the first arg is the iterator function present in the list which is arg 3. Args 2 and 4 are extra material to be added before and after the list that may require the use of `\Plural` or `\SingularPlural`.

```

2152 \def\PrintSeries@a#1#2#3#4{%
2153 \get@numberof\@tempcnta#1{#3}%
2154 \chardef\series@total=\@tempcnta
2155 \ifnum\series@total=\@ne
2156 \let\SingularPlural\@firstoftwo
2157 \else
2158 \let\SingularPlural\@secondoftwo
2159 \fi
2160 \series@index=\z@
2161 \let#1\series@add
2162 #2#3#4\relax
2163 \endgroup
2164 }

```

`\series@add` This is the inner function called by `\PrintSeries` that carefully distributes all the material stored previously in `\series@add@...` macros.

Note that the handling of “et al.” cases is somewhat hardcoded. This seemed preferable to adding yet another argument (or two!) to `\PrintSeries`.

```

2165 \def\series@add#1{%
2166 \advance\series@index\@ne
2167 \ifx\etaltext#1\relax
2168 \ifnum\series@index=\tw@
2169 \def\@tempa{\space\SubEtal}%
2170 \else
2171 \def\@tempa{\series@add@e\space\SubEtal}%
2172 \fi

```

We assume there are fewer than 20,000 items in the list.

```

2173 \series@index\@MM
2174 \else
2175 \ifcase\series@index
2176 \or

```

Material before name 1:

```

2177 \let\@tempa\series@add@a
2178 \or

```

Material before name 2:

```

2179 \ifnum\series@total<\thr@@
2180 \let\@tempa\series@add@b
2181 \else
2182 \let\@tempa\series@add@c
2183 \fi
2184 \else

```

Material before names 3, 4, 5,...

```

2185 \ifnum\series@index=\series@total
2186 \let\@tempa\series@add@d
2187 \else
2188 \ifnum\series@index<\series@total
2189 \let\@tempa\series@add@c

```

```

2190 \else
2191 \let\@tempa\@gobble
2192 \fi
2193 \fi
2194 \fi
2195 \fi
2196 \@tempa{#1}%
2197 }

```

**\SwapBreak** This takes a single argument, which should begin with a punctuation character, and conditionally appends it to the current horizontal list after removing any preceding whitespace. If there was also a penalty at the end of the hlist (presumed to be the result of a `\linebreak` at the end of a field value), it moves the penalty to *after* the argument.

*Known bug:* `\SwapBreak` interferes with `TEX`'s kerning mechanism. For example, consider a field value that ends with a “y” and that should have a comma automatically appended. `amsrefs` generates the equivalent of `y\SwapBreak{,}`, which results in “y,” (no kern before the comma) rather than “y,”. Unfortunately, fixing this would likely require a disproportionate effort. In cases where the lack of kerning is unacceptable, a workaround is to add the punctuation mark to the field value manually. For example, `title={...y,}` would generate the equivalent of `y,\SwapBreak{,}`, which in turn would produce “y,” since `\SwapBreak` is careful not to add duplicate punctuation.

```

2198 \def\SwapBreak#1{%
2199 \relax\ifvmode\leavevmode\fi
2200 \@tempcnta\@MM
2201 \toks@{#1}%

```

First, remove any preceding glue. (There usually shouldn't be any of this.)

```
2202 \unskip
```

There might be also be kern, typically an italic correction left there by a previous `TextFontCommand` like `\textit`. But don't remove the special 1 sp kern used to mark the beginning of a bibliography entry.

*Known bug:* Sometimes we want to keep the italic correction.

```
2203 \ifnum \lastkern>\@ne \unkern \fi
```

And now look for a penalty and stash it in a safe place.

```

2204 \ifnum\lastpenalty=\z@
2205 \else
2206 \@tempcnta\lastpenalty
2207 \unpenalty
2208 \fi

```

Now we add the punctuation, *unless* one of the following conditions is true:

1. The last item on the horizontal list was a kern of 1sp, indicating that we're at the very beginning of a bibliography item.
2. The current space factor is equal to the `\sfcode` of the punctuation mark we are adding, meaning that the mark is already on the list.
3. The current space factor is equal to the special value `\@nopunctsfcode`, meaning that `\nopunct` was specified.

This relies on distinct punctuation marks having distinct space factors, as established by our definition of `\frenchspacing`.

```

2209 \edef\@tempa{%
2210 \@nx\deferredquoteslogical
2211 \ifnum\lastkern=\@ne
2212 \else
2213 \ifnum\spacefactor=\sfcode\@xp\@xp\@xp'\@xp\@car\string#1)\@nil
2214 \else
2215 \ifnum\spacefactor=\@nopunctsfcode
2216 \else
2217 \the\toks@
2218 \fi
2219 \fi
2220 \fi
2221 \@nx\deferredquotes
2222 \ifnum\@tempcnta=\@MM \else \penalty\number\@tempcnta\space \fi
2223 \ifnum\lastkern=\@ne \ignorespaces \fi
2224 }%
2225 \@tempa
2226 }
```

`\Plural` `\Plural` takes one argument and prints it if there were two or more elements in the current list. So, to get “editors” instead of “editor” after printing a list of editor names, write `editor\Plural{s}`.

`\SingularPlural` takes two arguments and prints the first if there was only one element, otherwise prints the second arg.

```

2227 \newcommand{\SingularPlural}[2]{#1}
2228 \newcommand{\Plural}{\SingularPlural{}}
```

## 6.22 Formatting names and series of names

Now that we have a general mechanism for formatting series, we can easily specialize to the common case of a comma-separated list of names. First we provide specifications for the three most common name formats.

`\setbib@nameLE` This sets a name in standard western uninverted order, e.g., “John Doe Jr.” (The “LE” stands for little-endian.)

```

2229 \BibSpec{nameLE}{
2230 +{}{}{given}
2231 +{}{\IfEmptyBibField{given}{}{ }}{surname}
2232 +{}{ }{jr}
2233 }
```

`\setbib@nameBE` Big-endian order, as used for example in traditional Chinese, Japanese, Vietnamese, and Hungarian names: “Doe John”. Big-endian formatting can be requested for name by setting the “inverted” property to “yes.”

```
2234 \BibSpec{nameBE}{
2235 +{}{}{surname}
2236 +{}{ }{given}
```

I don’t know what should happen if there’s a suffix, so I’m going to just leave it out for now (although I should probably issue a warning). I suspect that either (a) it never comes up or (b) if it does come up, there’s no set standard for how it should be handled.

```
2237 % +{}{ }{jr}
2238 }
```

`\setbib@nameinverted` Inverted western-style names: “Doe, John, Jr.”

```
2239 \BibSpec{nameinverted}{
2240 +{} {} {surname}
2241 +{,}{ } {given}
2242 +{,}{ } {jr}
2243 }
```

Incidentally, it would probably be cleaner if names had their own namespace like properties do, i.e., something like

```
\DefineSimpleKey{name}{given}
\DefineSimpleKey{name}{initials}
\DefineSimpleKey{name}{surname}
\DefineSimpleKey{name}{jr}
```

followed by

```
\NameSpec{nameLE}{...}
```

or

```
\BibSpec[name]{nameLE}{...}
```

But this seems a little extravagant at this stage, so I’ve decided to leave things as-is for now.

`\PrintNames` `\PrintNames` is a simplified interface to `\PrintSeries` that takes only the last three arguments:

```
\PrintNames {S} {E} {\name{T1}... \name{Tn}}
```

The order of the last two arguments is reversed to make it moderately easier to use; cf. `\PrintEditorsA`, etc.

The first name in a series is treated differently than the other names in the `author-year` style, so we use a separate formatting macro for it.

```
2244 \newcommand{\PrintNames}{%
2245 \@ifstar{\PrintNames@a\set@othername}{\PrintNames@a\set@firstname}%
2246 }
```

`\PrintNames@a`

```
2247 \newcommand{\PrintNames@a}[4]{%
2248 \PrintSeries{\name}
2249 {#1}
2250 }{ and \set@othername}
2251 {,}{ \set@othername}
2252 {,}{ and \set@othername}
2253 {#2}{#4}{#3}%
2254 }
```

`\set@firstname` By default, the first name is formatted in little-endian format. The `author-year` option changes this to inverted order.

```
2255 \def\set@firstname#1{%
2256 \set@name{#1}\setbib@nameLE
2257 }
```

`\set@othername` The rest of the names are set in little-endian format by default.

```
2258 \def\set@othername#1{%
2259 \set@name{#1}\setbib@nameLE
2260 }
```

`\set@name` Parse the name into its components and then pass control to `\set@name@a`, which will decide what format to use for the name.

```
2261 \def\set@name#1{%
2262 \name@split#1,,\@nil
2263 \set@name@a
2264 }
```

`\set@namea` Use the requested format unless the `order` property has been set to “inverted.”

```
2265 \def\set@name@a#1{%
2266 \begingroup
2267 \get@current@properties
2268 \select@auxlanguage
2269 \def\@tempa{yes}%
2270 \ifx\@tempa\prop'inverted
2271 \setbib@nameBE
2272 \else
2273 #1%
2274 \fi
2275 \endgroup
2276 }
```

`\PrintPrimary`

```
2277 \def\PrintPrimary{%
2278 \ifx\current@primary\@empty
2279 \EmptyPrimaryWarning
2280 \else
2281 \print@primary\current@primary
2282 \fi
2283 }
```



## \PrintEditorsB

```

2313 \newcommand{\PrintEditorsB}{%
2314 \PrintNames*{ }\{\SwapBreak{,}~ed\Plural{s}.)}%
2315 }

```

## \PrintContributions

```

2316 \newcommand{\PrintContributions}[1]{%
2317 \PrintSeries
2318 {\fld@elt}
2319 {\print@contribution}
2320 {}{ and \print@contribution}
2321 {,}{ \print@contribution}
2322 {,}{ and \print@contribution}{#1}{%
2323 }

```

## \print@contribution

```

2324 \newcommand{\print@contribution}[1]{%
2325 \in@=#1}%
2326 \ifin@
2327 \ifnum\series@index=\@one with \fi
2328 \RestrictedSetKeys{}{bib}{%
2329 \bib@print@inner\setbib@contribution{\the\rsk@toks}%
2330 }{#1}%
2331 \else
2332 #1%
2333 \fi
2334 }

```

## \resolve@inner

```

2335 \def\resolve@inner#1#2{%
2336 \in@=#2}%
2337 \ifin@
2338 \RestrictedSetKeys{}{bib}{#1{\the\rsk@toks}}{#2}%
2339 \else
2340 \@ifundefined{bi@#2}{%
2341 \XRefWarning{#2}%
2342 }{%
2343 #1{\csname bi@#2\endcsname}%
2344 }%
2345 \fi
2346 }

```

## \PrintConference

```

2347 \def\PrintConference{%
2348 \resolve@inner{\bib@print@inner\setbib@conference}
2349 }

```

## \PrintConferenceDetails

```

2350 \def\PrintConferenceDetails#1{%

```

```

2351 \ifx\@empty\bib'address
2352 \ifx\@empty\bib'date
2353 \else
2354 \PrintConferenceDetails@
2355 \fi
2356 \else
2357 \PrintConferenceDetails@
2358 \fi
2359 }

```

\PrintConferenceDetails@

```

2360 \def\PrintConferenceDetails@{%
2361 \ifnum\lastkern=\@ne\else\space\fi(\kern 1sp
2362 \ifx\@empty\bib'address
2363 \else
2364 \bib'address
2365 \fi
2366 \ifx\@empty\bib'date
2367 \else
2368 \SwapBreak{,}\space
2369 \print@date
2370 \fi
2371)%\spacefactor\sfcode'\,%
2372 }

```

\PrintBook

```

2373 \def\PrintBook{%
2374 \resolve@inner{\bib@print@inner\setbib@innerbook}
2375 }

```

\PrintReprint

```

2376 \def\PrintReprint{%
2377 \resolve@inner{\bib@reprint}
2378 }

```

\bib@reprint

```

2379 \def\bib@reprint#1{%
2380 \begingroup
2381 #1\relax % execute definitions locally
2382 \bib@field@patches
2383 \bib'setup
2384 \IfEmptyBibField{copula}{reprinted in}{\bib'copula} \nopunct
2385 \let\bib'language\@empty
2386 \setbib@book
2387 \endgroup
2388 }

```

\PrintTranslation

```

2389 \def\PrintTranslation{%

```

```
2390 \resolve@inner{\bib@translation}
2391 }
```

**\bib@translation**

```
2392 \def\bib@translation#1{%
2393 \begingroup
2394 #1\relax % execute definitions locally
2395 \bib@field@patches
2396 \bib'setup
2397 \let\PrintPrimary\@empty
2398 \bib@append{;}{ % keep this space!
2399 \IfEmptyBibField{language}{English}{\bib'language} transl.%
2400 \IfEmptyBibField{pages}{ in \kern\@one sp}{, }%
2401 }\bib'transition
2402 \let\bib'language\@empty
2403 \setbib@@
2404 \endgroup
2405 }
```

**\PrintTranslatorsC**

```
2406 \newcommand{\PrintTranslatorsC}[1]{%
2407 \PrintNames{translated by }{ }{#1}%
2408 }
```

**\PrintTranslatorsA**

```
2409 \newcommand{\PrintTranslatorsA}[1]{%
2410 \ifx\previous@primary\current@primary
2411 \sameauthors{(trans.)}%
2412 \else
2413 \def\current@bibfield{\bib'translator}%
2414 \PrintNames{}{ (trans.)}{#1}%
2415 \fi
2416 \erase@field\bib'translator
2417 }
```

**\PrintTranslatorsB**

```
2418 \newcommand{\PrintTranslatorsB}[1]{
2419 \PrintNames*{ }{\SwapBreak{,}~tran\Plural{s}.)}%
2420 }
```

Some special handling for “et alii” or “and others”.

```
2421 \DefineName{alii}{\etaltext}
2422 \DefineName{others}{\etaltext}
```

**\etaltext** The Chicago Manual of Style suggests that it is slightly better not to italicize  
**\SubEtal** ‘et al’ and some other extremely common abbreviations inherited from Latin.  
 (Compare ‘etc’.)

```
2423 \newcommand{\etaltext}{et al.}
2424 \newcommand{\SubEtal}[1]{\etaltext}
```

### 6.23 The partial field

`\print@partial`

```
2425 \newcommand{\print@partial}{%
2426 \resolve@inner{\bib@print@inner\setbib@partial}
2427 }
```

### 6.24 Special formatting for other fields

`\parenthesize` The `\parenthesize` function adds parentheses around its argument, calling `\upn` to optionally prevent italic parentheses from being used.

```
2428 \newcommand{\parenthesize}[1]{%
2429 \leavevmode\push@bracket\upn{ }#1\pop@bracket
2430 }
```

`\upn` By default, `\upn` is a no-op, meaning that this refinement lies dormant unless the `upref` package or other activation is done. (Probably better done via special fonts, anyway.)

```
2431 \providecommand{\upn}[1]{#1}
```

`\push@bracket`

```
\pop@bracket 2432 \let\bracket@stack\@empty
2433
2434 \def\push@bracket#1{%
2435 \xdef\bracket@stack{#1\bracket@stack}%
2436 }
2437
2438 \def\pop@bracket{%
2439 \iffalse{\fi
2440 \xp\pop@bracket@a\bracket@stack \@empty}%
2441 }
2442
2443 \def\pop@bracket@a#1{%
2444 \leavevmode/>\upn{#1}%
2445 \xdef\bracket@stack{\iffalse}\fi
2446 }
```

`\bibquotes`

```
2447 \newcommand{\bibquotes}[1]{%
2448 \textquotedblleft#1%
2449 \gdef\deferredquotes{%
2450 \global\let\deferredquotes\@empty
2451 \textquotedblright
2452 }%
2453 }
```

`\mdash` Cf. `textcmds`, where there's also a penalty added.

```
\ndash 2454 \providecommand{\mdash}{\textemdash}
2455 \providecommand{\ndash}{\textendash}
```



```

2483
2484 \def\pp@scan#1-#2@-#3#4\@nil{#3}
2485
2486 \def\pp@scan@a#1\ndash#2@\ndash#3#4\@nil{#3}

```

`\eprintpages` If we have eprint info and pages info and no journal name, the pages information is presumably the number of pages in the eprint.

```

2487 \newcommand{\eprintpages}[1]{%
2488 #1\IfEmptyBibField{eprint}{}\IfEmptyBibField{journal}{ pp.}{}}%
2489 }

```

`\PrintThesisType`

```

2490 \def\PrintThesisType#1{%
2491 \thesis@type#1?\@nil{#1}%
2492 }
2493
2494 \def\thesis@type#1#2\@nil#3{%
2495 \ifx p#1%
2496 Ph.D. Thesis%
2497 \else
2498 \ifx m#1%
2499 Master's Thesis%
2500 \else
2501 #3%
2502 \fi
2503 \fi
2504 }

```

`\PrintDOI` Perhaps need to add allowbreak penalties at the parentheses in a DOI. Also what about prohibiting a break after the leading S?

```

2505 \newcommand{\PrintDOI}[1]{%
2506 DOI #1%
2507 \IfEmptyBibField{volume}{, (to appear in print)}{}}%
2508 }

```

`\PrintDatePV` Print date in different forms depending on DOI and volume information.

```

2509 \newcommand{\PrintDatePV}[1]{%
2510 \IfEmptyBibField{doi}{%
2511 \let\@tempa\PrintDate
2512 }{%
2513 \IfEmptyBibField{volume}{%
2514 \let\@tempa\PrintDatePosted
2515 }{%
2516 \let\@tempa\PrintDate
2517 }%
2518 }%
2519 \@tempa{#1}%
2520 }

```

`\PrintDate` The intent is to handle variations such as 1987, August 1987, 1987-08, and 1987-08-14. If the month is present, print August or Aug. or 08 or nothing, at the behest of the bib style.

We've taken some special care to parse out the date info ahead of time, so this function just discards arg 1 and uses the already-parsed value.

```
2521 \newcommand{\PrintDate}[1]{\print@date}
```

`\PrintDateB` The same, but without the parentheses.

```
2522 \newcommand{\PrintDateB}[1]{\print@date}
```

`\print@date`

```
2523 \def\print@date{%
2524 \ifx\bib@month\@empty
2525 \else
2526 \print@month@day
2527 \fi
2528 \bib@year
2529 }
```

`\print@month@day`

```
2530 \def\print@month@day{%
2531 \bib@monthname
2532 \ifx\@empty\bib@day \else \nobreakspace\number 0\bib@day,\fi
2533 \space
2534 }
```

`\bib@monthname` With the Babel package, month names for a given language are typically available in a macro `\month@language`:

```
\def\month@german{\ifcase\month\or
 Januar\or Februar\or M"arz\or April\or Mai\or Juni\or
 Juli\or August\or September\or Oktober\or November\or Dezember\fi}
```

However this is not true for English.

```
2535 \newcommand{\bib@monthname}{%
2536 \ifcase 0\bib@month
2537 \or January\or February\or March\or April\or May\or June\or
2538 July\or August\or September\or October\or November\or December\or
2539 Winter\or Spring\or Summer\or Fall\else Unknown Month%
2540 \fi
2541 }
```

`\PrintYear` You can use `\PrintYear` if you want to suppress month/day even when supplied in the data.

```
2542 \newcommand{\PrintYear}[1]{\bib@year}
```

`\PrintDatePosted` This one is special for AMS use.

```
2543 \newcommand{\PrintDatePosted}[1]{\unskip, posted on \print@date}
```

`\PrintEdition`

```
2544 \newcommand{\PrintEdition}[1]{%
2545 \afterassignment\print@edition
2546 \count@ 0#1\relax\@nil
2547 }
```

`\print@edition` If the number assignment swept up all the contents, produce a cardinal number from `\count@`.

```
2548 \def\print@edition#1#2\@nil{%
2549 \ifx\relax#1\relax
2550 \ifnum\count@>\z@
2551 \CardinalNumeric\count@
2552 \else
2553 ??th%
2554 \fi
2555 \ \editiontext
2556 \else
2557 \ifnum \count@>\z@ \number\count@ \fi
2558 #1#2\relax
2559 \fi
2560 }
```

`\editiontext`

```
2561 \newcommand{\editiontext}{ed.}
```

`\CardinalNumber`

```
2562 \newcommand{\CardinalNumeric}[1]{%
2563 \number#1\relax
2564 \if
2565 \ifnum#1<14
2566 \ifnum#1>\thr@@ T\else F\fi
2567 \else
2568 F%
2569 \fi
2570 T%
2571 th%
2572 \else
2573 \@xp\keep@last@digit\@xp#1\number#1\relax
2574 \ifcase#1th\or st\or nd\or rd\else th\fi
2575 \fi
2576 }
```

`\keep@last@digit`

```
2577 \def\keep@last@digit#1#2{%
2578 \ifx\relax#2%
2579 \@xp\@gobbletwo
2580 \else
2581 #1=#2\relax
2582 \fi
```

```
2583 \keep@last@digit#1%
2584 }
```

`\SentenceSpace` Note how careful we are here to preserve `\frenchspacing`.

```
2585 \newcommand{\SentenceSpace}{\relax\ifhmode\spacefactor'\. \fi}
```

`\eprint` For now, this does nothing. Could do a url/hyperlink or something.

```
2586 \newcommand{\eprint}[1]{\url{#1}}
```

The [www.arXiv.org](http://www.arXiv.org) recommendations for citing their eprints are found at <http://xxx.lanl.gov/help/faq/references>, including these examples:

```
arXiv:hep-th/9910001
arXiv:math.AT/9910001
arXiv:physics.acc-ph/9911027
```

## 6.25 Bib<sub>T</sub><sub>E</sub>X support

`\bibliographystyle` Disable `\bibliographystyle` since we're going to handle that behind the scenes.

```
2587 \let\bibliographystyle@gobble
```

`\bibtex@style`

```
2588 \def\bibtex@style{amsrn}
```

```
2589 \AtBeginDocument{
```

```
2590 \if@filesw
```

```
2591 \immediate\write\@auxout{\string\bibstyle{\bibtex@style}}%
```

```
2592 \fi
```

```
2593 }
```

## 6.26 Implementing package options

### 6.26.1 The alphabetic option

```
2594 \IfOption{alphabetic}{%
```

```
2595 \def\bibtex@style{amsra}%
```

```
2596 \def\alpha@label{%
```

```
2597 \ifx\@empty\bib'label
```

```
2598 \def\thebib{\CurrentBib}%
```

```
2599 \else
```

```
2600 \let\thebib\bib'label
```

```
2601 \fi
```

```
2602 }%
```

```
2603 \let\generate@label\generate@alphalabel
```

```
2604 \let\process@citelist\process@citelist@unsorted
```

```
2605 \def\numeric@refs{01}%
```

```
2606 }{}
```

### 6.26.2 The shortalphabetic option

```
2607 \IfOption{shortalphabetic}{%
```

```
2608 \def\bibtex@style{amsrs}%
```

```
2609 \def\alpha@label{%
```

```

2610 \ifx\@empty\bib'label
2611 \def\thebib{\CurrentBib}%
2612 \else
2613 \let\thebib\bib'label
2614 \fi
2615 }%
2616 \let\@suffix@format\@arabic
2617 \def\calc@author@part{%
2618 \xp\@multiauthorlabel\@xp{\@tempa}%
2619 }%
2620 \let\append@label@year\@empty
2621 \let\generate@label\generate@alphalabel
2622 \let\process@citelist\process@citelist@unsorted
2623 \def\numeric@refs{01}%
2624 }{}

```

### 6.26.3 The backrefs option

```

2625 \IfOption{backrefs}{%
2626 \let\PrintBackRefs\print@backrefs
2627 \@ifundefined{Hy@backout}{%
2628 \amsrefs@warning{backref option requires hyperref package}%
2629 }{%
2630 \let\BackCite\back@cite
2631 \AtBeginDocument{\@starttoc{brf}{}}%
2632 }%
2633 }{}%
2634 }

```

### 6.26.4 The citation-order option

```

2635 \IfOption{citation-order}{%
2636 \IfOption{alphabetic}{%
2637 \amsrefs@warning@nl{%
2638 The 'citation-order' and 'alphabetic' options are
2639 incompatible%
2640 }%
2641 }{
2642 \def\bibtex@style{amsru}%
2643 }
2644 }{}

```

### 6.26.5 The initials option

```

2645 \IfOption{initials}{% TRUE:
2646 \BibSpec{nameLE}{
2647 +{}{}{initials}
2648 +{}{\IfEmptyBibField{initials}{}{ }}{surname}
2649 +{}{ }{jr}
2650 }
2651
2652 \BibSpec{nameBE}{
2653 +{}{}{surname}
2654 +{}{ }{initials}

```

```

2655 % +{}{ }{jr}
2656 }
2657
2658 \BibSpec{nameinverted}{
2659 +{} {} {surname}
2660 +{,}{ } {initials}
2661 +{,}{ } {jr}
2662 }
2663 }{% initials? FALSE:
2664 % \let\extract@initials@gobble
2665 } % end conditional code for initials option

```

### 6.26.6 The jpa option

```

2666 \IfOption{jpa}{%
2667 \amsrefs@warning{The 'jpa' option is obsolete}%
2668 \typeout{Trying \string\usepackage{amsjpa} instead ...}%
2669 \RequirePackage{amsjpa}[2000/02/02]
2670 }{}

```

### 6.26.7 The logical-quotes option

\deferredquotes

```

2671 \let\deferredquotes@empty

```

\deferredquoteslogical

```

2672 \IfOption{logical-quotes}{%
2673 \def\deferredquoteslogical{\deferredquotes}%
2674 }{%
2675 \let\deferredquoteslogical\relax
2676 }

```

### 6.26.8 The non-compressed-cites option

```

2677 \IfOption{non-compressed-cites}{%
2678 \let\cite@compress\cite@print
2679 }{}

```

### 6.26.9 The non-sorted-cites option

```

2680 \IfOption{non-sorted-cites}{%
2681 \let\process@citelist\process@citelist@unsorted
2682 }{}

```

### 6.26.10 The short-journals option

```

2683 \IfOption{short-journals}{%
2684 \renewcommand{\DefineJournal}[4]{%
2685 \bib*{#1}{periodical}{
2686 issn={#2},
2687 journal={#3},
2688 }%
2689 }
2690 }{}

```

### 6.26.11 The short-publishers option

```

2691 \IfOption{short-publishers}{%
2692 \renewcommand{\DefinePublisher}[4]{%
2693 \bib*{#1}{publisher}{%
2694 publisher={#2},%

```

Maybe `short-publishers` should suppress the `address`? Or is that a separate option? I sense a combinatorial explosion coming on...

```

2695 address={#4},
2696 }%
2697 }%
2698 }{}

```

#### 6.26.12 The short-months option

```

2699 \IfOption{short-months}{%
2700 \renewcommand{\bib@monthname}{%
2701 \ifcase 0\bib@month
2702 \or Jan.\or Feb.\or Mar.\or Apr.\or May\or June\or
2703 July\or Aug.\or Sep.\or Oct.\or Nov.\or Dec.\or
2704 Winter\or Spring\or Summer\or Fall\else Unknown Month%
2705 \fi
2706 }%
2707 }{}

```

#### 6.26.13 The y2k option

```

2708 \IfOption{y2k}{%
2709 \IfOption{alphabetic}{%
2710 \def\year@short#1\@nil{#1}%
2711 \def\bibtex@style{amsry}%
2712 }{%
2713 \amsrefs@warning@nl{%
2714 The 'y2k' option can only be used with the^^J%
2715 'alphabetic' option%
2716 }%
2717 }
2718 }{}

```

#### 6.26.14 The bibtex-style option

```

2719 \IfOption{bibtex-style}{%
2720 \RequirePackage{amsbst}
2721 }{}

```

#### 6.26.15 The author-year option

Here ends the `amsrefs` package, unless the `author-year` option is in effect; then we want to use some different bibspecs.

```

2722 \IfOption{author-year}{\PopCatcodes \endinput}

```

`\generate@label`

```

2723 \def\generate@label{%

```

If the user supplied an explicit `label` field, we use it. Otherwise, we generate our own.

```
2724 \ifx\bib'label\@empty
2725 \begingroup
```

We begin by saving the previous stem and initializing the current stem to the empty string.

```
2726 \global\let\previous@stem\current@stem
2727 \global\let\current@stem\@empty
2728 \global\let\previous@year\current@year
2729 \global\let\current@year\bib@year
```

The list of primary contributors is available to us in `\current@primary` in the form

```
\name{Last1, First1} \name{Last2, First2} ... \name{Lastn, Firstn}
```

We will be executing this list multiple times with various definitions of `\name`. So the first thing we want to do is establish a safe environment and normalize the names.

```
2730 \@apply\auto@protect\amsrefs@textsymbols
2731 \@apply\auto@protect\amsrefs@textaccents
2732 \def\name##1{\@nx\name{\lncan@a##1,\@nil}}%
2733 \auto@protect\etaltext
2734 \normalize@edef\current@stem{\current@primary}%
2735 \xdef\current@stem{\current@stem}%
```

At this point, the `\current@stem` is complete and we're ready to determine what (if any) suffix is needed to disambiguate it from the previous label.

```
2736 \calc@alpha@suffix
```

We have all the pieces now. Arrange to end the current group and then define `\bib@label` in the enclosing group. (This keeps `\bib@label` from being defined outside the group started by `\bib@start`. This isn't strictly necessary, but it provides a bit of compartmentalization.)

```
2737 \edef\@tempa{%
2738 \def\@nx\cite@label{\current@stem}%
2739 \def\@nx\bib@label@year{%
2740 \current@year
2741 \alpha@label@suffix
2742 }%
2743 }
2744 \@xp@endgroup
2745 \@tempa
2746 \fi
2747 }
```

```
\lncan@a
```

```
2748 \def\lncan@a#1,#2\@nil{#1}
```

```
\citesel@author
```

```
2749 \def\citesel@author#1#2#3#4#5{\PrintCiteNames{#3}}
```

```

\citesel@authoryear
2750 \def\citesel@authoryear#1#2#3#4#5{\PrintCNY{#3}-{#4}}

\citesel@object
2751 \def\citesel@object#1#2#3#4#5{\PrintCiteNames{#3} \citeleft#4}

\citesel
2752 \let\citesel\citesel@authoryear

\numeric@refs
2753 \def\numeric@refs{01}%

\citeleft
2754 \def\citeleft{()%

\citeright
2755 \def\citeright{)%

@citeleft
2756 \def@citeleft{\ifx\citesel\citesel@object\else\citeleft\fi}%

\citepunct
2757 \def\citepunct{; }

\BibLabel
2758 \let\BibLabel\@empty

\process@citelist
2759 \let\process@citelist\process@citelist@unsorted

\ycite
2760 \DeclareRobustCommand{\ycite}[1]{%
2761 \star@{\cite@a\citesel@year{#1}}{}}%
2762 }

\ycites
2763 \DeclareRobustCommand{\ycites}[1]{%
2764 \begingroup
2765 \def\citepunct{, }%
2766 \let\citesel\citesel@year
2767 \cites{#1}%
2768 \endgroup
2769 }

\ocite
2770 \DeclareRobustCommand{\ocite}[1]{%
2771 \star@{\cite@a\citesel@object{#1}}{}}%
2772 }

```

`\ocites`

```
2773 \DeclareRobustCommand{\ocites}[1]{%
2774 \begingroup
2775 \let\@citelist\@ocitelist
2776 \cites{#1}%
2777 \endgroup
2778 }
```

`\ocitelist`

```
2779 \def\@ocitelist#1{%
2780 \PrintSeries{\InnerCite}%
2781 {\ocite}%
2782 }{ and \ocite}%
```

For three or more names: print ‘et al’ instead of the last name. Have to putz around with the space factor a bit or the comma between name and year will not be applied.

```
2783 {,}{ \ocite}%
2784 {,}{ and \ocite}%
2785 }{%
2786 {#1}%
2787 }{%
2788 }
```

`\citeauthor`

```
2789 \DeclareRobustCommand{\citeauthor}[1]{%
2790 \star@\{cite@a\citesel@author{#1}\}%
2791 }
```

`\citeauthority`

```
2792 \DeclareRobustCommand{\citeauthority}[1]{%
2793 \citeauthor{#1} \ycite{#1}%
2794 }
```

`\fullcite`

```
2795 \DeclareRobustCommand{\fullcite}[1]{%
2796 \begingroup
2797 \let\print@citenames\CiteNamesFull
2798 \star@\{cite@a\citesel@authoryear{#1}\}%
2799 \endgroup
2800 }
```

`\fullocite`

```
2801 \DeclareRobustCommand{\fullocite}[1]{%
2802 \begingroup
2803 \let\print@citenames\CiteNamesFull
2804 \star@\{cite@a\citesel@object{#1}\}%
2805 \endgroup
2806 }
```

Invert the first author's name.

```
2807 \def\set@firstname#1{%
2808 \set@name{#1}\setbib@nameinverted
2809 }
```

\PrintCNY

```
2810 \def\PrintCNY#1#2{%
2811 \PrintCiteNames{#1}%
2812 \@ifnotempty{#2}{\@addpunct{,} #2}%
2813 }
```

\PrintCiteNames

```
2814 \def\PrintCiteNames#1{%
2815 \leavevmode
2816 \def\@tempa{#1}%
2817 \ifx\@tempa\prev@names
2818 \else
2819 \gdef\prev@names{#1}%
2820 \@xp\ifx\@car#1.\@nil\CitePrintUndefined
2821 #1\relax
2822 \else
2823 \print@citenames{#1}%
2824 \fi
2825 \fi
2826 }
```

\CiteNames

```
2827 \newcommand{\CiteNames}[1]{%
2828 \PrintSeries{\name}%
2829 {}%
2830 {}{ and }%
```

For three or more names: print 'et al' instead of the last name. Have to putz around with the space factor a bit or the comma between name and year will not be applied.

```
2831 {}{\@gobble}%
2832 {}{ \etaltext\@gobble}%
2833 {}%
2834 {#1}%
2835 {}%
2836 }
```

\print@citenames

```
2837 \let\print@citenames\CiteNames
```

\CiteNamesFull

```
2838 \newcommand{\CiteNamesFull}[1]{%
2839 \PrintSeries{\name}%
2840 {}%
2841 {}{ and }%
```

For three or more names: print ‘et al’ instead of the last name. Have to putz around with the space factor a bit or the comma between name and year will not be applied.

```
2842 {,}{ }%
2843 {,}{ and }%
2844 }}%
2845 {#1}%
2846 }}%
2847 }
```

`\PrintDate` No parentheses around the year.

```
2848 \renewcommand{\PrintDate}[1]{\bib@label@year}
```

`\print@date` Only print the year, not the month or day.

```
2849 \def\print@date{%
2850 \IfEmptyBibField{date}{%
2851 \IfEmptyBibField{year}{\BibField{status}}{\bib@year}%
2852 }{%
2853 \bib@year
2854 }%
2855 }

2856 \BibSpec{article}{%
2857 +{} {\PrintAuthors} {author}
2858 +{.} { \PrintDate} {date}
2859 +{.} { \textit} {title}
2860 +{.} { } {part}
2861 +{:} { \textit} {subtitle}
2862 +{,} { \PrintContributions} {contribution}
2863 +{.} { \PrintPartials} {partial}
2864 +{,} { } {journal}
2865 +{} { \textbf} {volume}
2866 +{,} { \issuetext} {number}
2867 +{,} { \eprintpages} {pages}
2868 +{,} { } {status}
2869 +{,} { \PrintDOI} {doi}
2870 +{,} { available at \eprint} {eprint}
2871 +{} { \parenthesize} {language}
2872 +{} { \PrintTranslation} {translation}
2873 +{;} { \PrintReprint} {reprint}
2874 +{.} { } {note}
2875 +{.} {} {transition}
2876 +{} {\SentenceSpace \PrintReviews} {review}
2877 }
2878
2879 \BibSpec{book}{%
2880 +{} {\PrintPrimary} {transition}
2881 +{.} { \PrintDate} {date}
2882 +{.} { \textit} {title}
```

```

2883 +{.} { } {part}
2884 +{:} { \textit} {subtitle}
2885 +{,} { \PrintEdition} {edition}
2886 +{} { \PrintEditorsB} {editor}
2887 +{,} { \PrintTranslatorsC} {translator}
2888 +{,} { \PrintContributions} {contribution}
2889 +{,} { } {series}
2890 +{,} { \volumetext} {volume}
2891 +{,} { } {publisher}
2892 +{,} { } {organization}
2893 +{,} { } {address}
2894 +{,} { } {status}
2895 +{} { \parenthesize} {language}
2896 +{} { \PrintTranslation} {translation}
2897 +{;} { \PrintReprint} {reprint}
2898 +{.} { } {note}
2899 +{.} {} {transition}
2900 +{} {\SentenceSpace \PrintReviews} {review}
2901 }
2902
2903 \BibSpec{collection.article}{%
2904 +{} {\PrintAuthors} {author}
2905 +{.} { \PrintDate} {date}
2906 +{.} { \textit} {title}
2907 +{.} { } {part}
2908 +{:} { \textit} {subtitle}
2909 +{,} { \PrintContributions} {contribution}
2910 +{,} { \PrintConference} {conference}
2911 +{} {\PrintBook} {book}
2912 +{,} { } {booktitle}
2913 +{,} { pp.~} {pages}
2914 +{,} { } {status}
2915 +{,} { \PrintDOI} {doi}
2916 +{,} { available at \eprint} {eprint}
2917 +{} { \parenthesize} {language}
2918 +{} { \PrintTranslation} {translation}
2919 +{;} { \PrintReprint} {reprint}
2920 +{.} { } {note}
2921 +{.} {} {transition}
2922 +{} {\SentenceSpace \PrintReviews} {review}
2923 }
2924
2925 \BibSpec{report}{%
2926 +{} {\PrintPrimary} {transition}
2927 +{.} { \PrintDate} {date}
2928 +{.} { \textit} {title}
2929 +{.} { } {part}
2930 +{:} { \textit} {subtitle}
2931 +{,} { \PrintEdition} {edition}
2932 +{,} { \PrintContributions} {contribution}

```

```

2933 +{,} { Technical Report } {number}
2934 +{,} { } {series}
2935 +{,} { } {organization}
2936 +{,} { } {address}
2937 +{,} { \eprint } {eprint}
2938 +{,} { } {status}
2939 +{} { \parenthesize } {language}
2940 +{} { \PrintTranslation } {translation}
2941 +{;} { \PrintReprint } {reprint}
2942 +{.} { } {note}
2943 +{.} {} {transition}
2944 +{} { \SentenceSpace \PrintReviews } {review}
2945 }
2946
2947 \BibSpec{thesis}{%
2948 +{} { \PrintAuthors } {author}
2949 +{.} { \PrintDate } {date}
2950 +{.} { \textit } {title}
2951 +{:} { \textit } {subtitle}
2952 +{,} { \PrintThesisType } {type}
2953 +{,} { } {organization}
2954 +{,} { } {address}
2955 +{,} { \eprint } {eprint}
2956 +{,} { } {status}
2957 +{} { \parenthesize } {language}
2958 +{} { \PrintTranslation } {translation}
2959 +{;} { \PrintReprint } {reprint}
2960 +{.} { } {note}
2961 +{.} {} {transition}
2962 +{} { \SentenceSpace \PrintReviews } {review}
2963 }

2964 \PopCatcodes
2965 </pkg>

```

## 6.27 The amsbst package

```

2966 <*bst>
2967 \NeedsTeXFormat{LaTeX2e}[1995/12/01]
2968 \ProvidesPackage{amsbst}[2004/03/29 v1.68]
2969 %\RequirePackage{amsrefs}[2004/03/29]
2970 \BibSpec{article}{%
2971 +{} { \PrintAuthors } {author}
2972 +{.} { } {title}
2973 +{.} { } {part}
2974 +{:} { } {subtitle}
2975 +{.} { \PrintContributions } {contribution}
2976 +{.} { \PrintPartials } {partial}
2977 +{.} { \emph } {journal}
2978 +{} { } {volume}
2979 +{} { \parenthesize } {number}

```

```

2980 +{:} {} {pages}
2981 +{,} { \PrintDateB} {date}
2982 +{,} { } {status}
2983 +{.} { \PrintTranslation} {translation}
2984 +{.} { Reprinted in \PrintReprint} {reprint}
2985 +{.} { } {note}
2986 +{.} {} {transition}
2987 }
2988
2989 \BibSpec{partial}{%
2990 +{} {} {part}
2991 +{:} { } {subtitle}
2992 +{.} { \PrintContributions} {contribution}
2993 +{.} { \emph} {journal}
2994 +{} { } {volume}
2995 +{} { \parenthesize} {number}
2996 +{:} {} {pages}
2997 +{,} { \PrintDateB} {date}
2998 }
2999
3000 \BibSpec{book}{%
3001 +{} { \PrintPrimary} {transition}
3002 +{.} { \emph} {title}
3003 +{.} { } {part}
3004 +{:} { \emph} {subtitle}
3005 +{.} { } {series}
3006 +{,} { \voltext} {volume}
3007 +{.} { Edited by \PrintNameList} {editor}
3008 +{.} { Translated by \PrintNameList} {translator}
3009 +{.} { \PrintContributions} {contribution}
3010 +{.} { } {publisher}
3011 +{.} { } {organization}
3012 +{,} { } {address}
3013 +{,} { \PrintEdition} {edition}
3014 +{,} { \PrintDateB} {date}
3015 +{.} { } {note}
3016 +{.} {} {transition}
3017 +{.} { \PrintTranslation} {translation}
3018 +{.} { Reprinted in \PrintReprint} {reprint}
3019 +{.} {} {transition}
3020 }
3021
3022 \BibSpec{collection.article}{%
3023 +{} { \PrintAuthors} {author}
3024 +{.} { } {title}
3025 +{.} { } {part}
3026 +{:} { } {subtitle}
3027 +{.} { \PrintContributions} {contribution}
3028 +{.} { \PrintConference} {conference}
3029 +{.} { \PrintBook} {book}

```

```

3030 +{.} { In } {booktitle}
3031 +{,} { pages~} {pages}
3032 +{.} { \PrintDateB} {date}
3033 +{.} { \PrintTranslation} {translation}
3034 +{.} { Reprinted in \PrintReprint} {reprint}
3035 +{.} { } {note}
3036 +{.} {} {transition}
3037 }
3038
3039 \BibSpec{conference}{%
3040 +{} {} {title}
3041 +{} {\PrintConferenceDetails} {transition}
3042 }
3043
3044 \BibSpec{innerbook}{%
3045 +{.} { \emph} {title}
3046 +{.} { } {part}
3047 +{:} { \emph} {subtitle}
3048 +{.} { } {series}
3049 +{,} { \voltext} {volume}
3050 +{.} { Edited by \PrintNameList} {editor}
3051 +{.} { Translated by \PrintNameList} {translator}
3052 +{.} { \PrintContributions} {contribution}
3053 +{.} { } {publisher}
3054 +{.} { } {organization}
3055 +{,} { } {address}
3056 +{,} { \PrintEdition} {edition}
3057 +{,} { \PrintDateB} {date}
3058 +{.} { } {note}
3059 +{.} {} {transition}
3060 }
3061
3062 \BibSpec{report}{%
3063 +{} {\PrintPrimary} {transition}
3064 +{.} { \emph} {title}
3065 +{.} { } {part}
3066 +{:} { \emph} {subtitle}
3067 +{.} { \PrintContributions} {contribution}
3068 +{.} { Technical Report } {number}
3069 +{,} { } {series}
3070 +{.} { } {organization}
3071 +{,} { } {address}
3072 +{,} { \PrintDateB} {date}
3073 +{.} { \PrintTranslation} {translation}
3074 +{.} { Reprinted in \PrintReprint} {reprint}
3075 +{.} { } {note}
3076 +{.} {} {transition}
3077 }
3078
3079 \BibSpec{thesis}{%

```

```

3080 +{ } { \PrintAuthors} {author}
3081 +{,} { \emph} {title}
3082 +{:} { \emph} {subtitle}
3083 +{.} { \PrintThesisType} {type}
3084 +{.} { } {organization}
3085 +{,} { } {address}
3086 +{,} { \PrintDateB} {date}
3087 +{.} { \PrintTranslation} {translation}
3088 +{.} { Reprinted in \PrintReprint} {reprint}
3089 +{.} { } {note}
3090 +{.} { } {transition}
3091 }

```

`\PrintEditorsA` When we consider editor names we have to think about some further complications. First, for the case of a book where editor names are listed in place of author names, just copy the same style with a bit of added text at the end.

```

3092 \renewcommand{\PrintEditorsA}[1]{%
3093 \def\current@bibfield{\bib'editor}%
3094 \PrintNames{ }{, editor\Plural{s}}{#1}%
3095 \erase@field\bib'editor
3096 }

```

`\PrintTranslatorsA`

```

3097 \renewcommand{\PrintTranslatorsA}[1]{%
3098 \def\current@bibfield{\bib'translator}%
3099 \PrintNames{ }{, translator\Plural{s}}{#1}%
3100 \erase@field\bib'translator
3101 }

```

```
3102 </bst>
```

The usual `\endinput` to ensure that random garbage at the end of the file doesn't get copied by `docstrip`.

```
3103 \endinput
```

## References

- [1] David M. Jones, *User's Guide to the amsrefs Package*. distributed with the `amsrefs` code.
- [2] Ellen Swanson, Arlene O'Sean, and Antoinette Schleyer, *Mathematics into Type*, updated, American Mathematical Society, 1999.