glossaries.sty v 1.07: LaTeX 2$_\varepsilon$ Package to Assist Generating Glossaries

Nicola L.C. Talbot

School of Computing Sciences
University of East Anglia
Norwich. Norfolk
NR4 7TJ. United Kingdom.
http://theoval.cmp.uea.ac.uk/~nlct/

13th September 2007

Contents

1 Introduction

The glossaries package is provided to assist generating glossaries. It has a certain amount of flexibility, allowing the user to customize the format of the glossary, define new glossary styles, and multiple glossaries. It also supports acronyms, and glossary styles which include symbols in addition to a name and description for a given glossary entry. There is provision for loading a database of glossary terms, where only those terms used in the text are added to the glossary. This package replaces the glossary package which is now obsolete.

The glossaries package comes with the Perl script `makeglossaries` which will run `makeindex` on all the glossary files using a customized `makeindex` style file (which is created by `\makeglossaries`). The relevant extensions are obtained from the auxiliary file, so you only need to pass the basename as the argument. For example, if your document is called `myfile.tex`, do:

```
latex myfile
makeglossaries myfile
latex myfile
```

You may need to explicitly load `makeglossaries` into Perl:

```
perl makeglossaries myfile
```

There is a batch file called `makeglossaries.bat` which does this for Windows users.

If you don't have Perl installed, you will have to run `makeindex` for each glossary type you have defined. For example, if you have used the acronym package option, so you have both a main glossary as well as a list of acronyms, you will need to do (assuming your document is called `myfile.tex`):

```
makeindex -s myfile.ist -t myfile.glg -o myfile.gls myfile.glo
makeindex -s myfile.ist -t myfile.alg -o myfile.acr myfile.acn
```

This requires remembering all extensions for each of the glossaries defined in your document, so where possible you should use `makeglossaries` instead to reduce the possibility of error.

This documentation is structured as follows: section 2 is for people who want a few quick pointers of how to get started, without having to read through lengthy descriptions, section 3 gives an overview of available commands and their syntax, section 4 contains the documented source code for those who want to know more about how the package works, and how to do more complicated things, such as changing the way glossary entries appear.

2 A Quick Guide For The Impatient

This section is for people who want a few quick pointers of how to get started, without having to read through lengthy descriptions.

1. Load glossaries *after* hyperref:

   ```
   \usepackage{hyperref}
   \usepackage{glossaries}
   ```

   Similarly for the html package:

   ```
   \usepackage{html}
   \usepackage{glossaries}
   ```

2. Always use `\makeglossaries` if you want the glossary entries to be written to the glossary file:

   ```
   \documentclass{article}
   \usepackage{glossaries}
   \makeglossaries
   ```

   If you don't use `\makeglossaries`, your glossaries will not appear in the document!

3. Use `\printglossaries` to make your glossaries appear in the document at that point. For example:

   ```
   \maketitle
   \printglossaries
   \section{Introduction}
   ```

   Note that only the glossary entries that have been used in the document text will appear in the glossary.

4. When you have created your document, run LaTeX on it, then the Perl script `makeglossaries`, then run LaTeX on it again:

   ```
   latex myfile
   makeglossaries myfile
   latex myfile
   ```

   If you use Windows, there is a batch file called `makeglossaries.bat` which you can use, but you will still need Perl installed.

5. New glossaries can be defined using:

   `\newglossary {`⟨*label*⟩`}{`⟨*in-ext*⟩`}{`⟨*out-ext*⟩`} {`⟨*title*⟩`}`

   where ⟨*label*⟩ is an identifying label, ⟨*in-ext*⟩ is the extension of the file to be created by `makeindex` (called by `makeglossaries`), ⟨*out-ext*⟩ is the extension of the file to be read by `makeindex` and ⟨*title*⟩ is the title for this new glossary. Example:

   ```
   \newglossary{notation}{not}{ntn}{Notation}
   ```

   This glossary's label is `notation` and its title will be Notation.

6. Any new glossaries must be defined before `\makeglossaries`

   ```
   \documentclass{article}
   \usepackage{glossaries}
   \newglossary{notation}{not}{ntn}{Notation}
   \makeglossaries
   ```

7. If you use the acronym package option, the glossaries package will automatically create a new glossary type labelled `acronym`:

   ```
   \usepackage[acronym]{glossaries}
   ```

8. If your pages have a hyphen compositor (i.e. your page numbers appear in the form 2-1), redefine `\glscompositor` *before* `\makeglossaries`:

   ```
   \documentclass{article}
   \usepackage{glossaries}
   \renewcommand{\glscompositor}{-}
   \makeglossaries
   ```

9. To add the glossaries to the table of contents use the toc package option:

   ```
   \usepackage[toc]{glossaries}
   ```

10. Define a new entry with:

    `\newglossaryentry{`⟨*label*⟩`}{`⟨*key-val list*⟩`}`

    The ⟨*key-val list*⟩ must at least contain a name key and a description key. For example:

    ```
    \newglossaryentry{perl}{name=Perl,
    description=A scripting language}
    ```

    In this example, I have given the entry the label `perl`. Whenever I want to use this entry, that is the label I need to use to identify it.

11. If the entry name starts with an accented letter, you will need to group the first letter (otherwise it will cause a problem for `\Gls` and `\Glspl`):

    ```
    \newglossaryentry{elite}{name={\'e}lite},
    description={select group or class}}
    ```

12. If you have multiple glossaries, use the type key to specify in which glossary the entry belongs. For example:

```
\newglossary{languages}{lan}{lng}{Index of Languages}

\makeglossaries

\newglossaryentry{perl}{name=Perl,
description=A scripting language,
type=languages}
```

    If type is omitted, the default glossary is used.

13. Remember to group values that have a comma or equal sign. For example:

```
\newglossaryentry{pagelist}{name=page list,
description={A list of individual pages or page ranges
(e.g.\ 1,2,4,7--9)}}
```

14. Plural forms are assumed to be the singular form with an "s" appended, unless otherwise specified. To specify an irregular plural, use the plural key. For example:

```
\newglossaryentry{matrix}{name=matrix,
description=rectangular array of quantities,
plural=matrices}
```

15. The way the term appears in the main text can be different from the way the term appears in the glossary:

```
\newglossaryentry{matrix}{name=Matrix,
description=rectangular array of quantities,
text=matrix,
plural=matrices}
```

    In this example, the entry name appears as Matrix in the glossary, and either matrix or matrices in the text.

16. The way the term appears on first use can be different to the way it appears subsequently:

```
\newglossaryentry{singmtx}{name=Singular Matrix,
description=A matrix with a zero determinant,
first=singular matrix (SM),
text=SM,
firstplural=singular matrices (SMs)}
```

    In this example, the entry name appears as Singular Matrix in the glossary, and in the text it appears as singular matrix (SM) or singular matrices (SMs) the first time the entry is used, and subsequently appears as SM or SMs.

17. The quick and easy way to define an acronym is to use:

    \newacronym[⟨*key-val list*⟩]{⟨*label*⟩}{⟨*abbrev*⟩} {⟨*long*⟩}

    For example:

    ```
    \newacronym{svm}{SVM}{support vector machine}
    ```

    This is equivalent to:

    ```
    \newglossaryentry{svm}{type=\acronymtype,
    name={SVM},
    description={support vector machine},
    text={SVM},
    first={support vector machine (SVM)},
    plural={SVMs},
    firstplural={support vector machines (SVMs)}}
    ```

    (The value of \acronymtype varies depending on whether the glossary package option acronym is used or not. The optional argument ⟨*key-val list*⟩ can be used to override any of the \newglossaryentry keys, for example, if the acronym has an irregular plural.)

18. The font used to display the entry name in the glossary is governed by \glsnamefont. This can be redefined as required. For example, to make the entry names appear in a medium sans-serif font do:

    ```
    \renewcommand{\glsnamefont}[1]{\textsf{\mdseries #1}}
    ```

    Note that the list-like glossary styles defined in the glossary-list package place the entry name in the optional argument to \item, so they will appear in bold, unless you redefine \glsnamefont.

19. In the document use \gls{⟨*label*⟩} to use a predefined term (this will also enter the term into the associated glossary output file.) For example:

    ```
    A \gls{singmtx} is a matrix with a zero determinant.
    ```

20. Other variations:

    - \Gls{⟨*label*⟩} : like \gls, but with first letter in upper case
    - \GLS{⟨*label*⟩} : like \gls, but all upper case.
    - \glspl{⟨*label*⟩} : use plural
    - \Glspl{⟨*label*⟩} : use plural with first letter in upper case
    - \GLSpl{⟨*label*⟩} : use plural but all upper case
    - \glslink{⟨*label*⟩}{⟨*link text*⟩} : use ⟨*link text*⟩ to link to the given entry in the glossary.

    For example, the following will produce the plural form with the first letter in uppercase:

    ```
    \Glspl{singmtx} are matrices with a zero determinant.
    ```

21. Additional text can be appended to the link using the end optional argument. For example, to form the possessive:

```
The \gls{singmtx}['s] dimensions are not necessarily equal.
```

22. The format of the associated entry number can be changed using the format key in the optional argument. Note that the value of the format key should be the name of a command *without* the initial backslash. For example:

```
The primary definition of \glspl[format=textbf]{singmtx}.
```

    In this example the relevant glossary entry will have the page number in bold (since it uses `\textbf`) but it will no longer have a hyperlink (if hyperlinks are enabled.)

23. The glossaries package provides commands to change the font whilst ensuring that the number remains a hyperlink. These are of the form `\hyper`⟨*xx*⟩ and are equivalent to the standard font changing commands of the form `\text`⟨*xx*⟩, as well as `\hyperemph` (which uses `\emph`.) For example:

```
The primary definition of \glspl[format=hyperbf]{singmtx}.
```

24. Entries can be added to the glossary without producing any text using `\glsadd{`⟨*label*⟩`}` or `\glsaddall`. These commands also taken an optional argument where you can specify the format. For example

```
\glsadd[format=hyperbf]{singmtx}
```

    will add a line to the glossary file for the specified term, but will not produce any text where the command occurs.

25. A number range can be entered using `format=(` and `format=)` to mark the beginning and ending of the range[1]. For example:

```
\glsadd[format=(]{singmtx}
This is a very long section all about \glspl{singmtx}.

% lots of text omitted

\glsadd[format=)]{singmtx}
```

    This is equivalent to `makeindex`'s `|(` and `|)` formats.

26. You can combine the range markers with a formatting command (again without the preceding backslash.) For example:

```
This is the start of a very long section all
about \glspl[format=(hyperbf]{singmtx}.

% lots of text omitted

This is the end a very long section all about
\glspl[format=)hyperbf]{singmtx}.
```

[1] This is new to version 1.01

27. Only those terms that have actually been used in the document will be placed in the glossary. If you have defined a term that doesn't appear in the document, then it means you haven't used it in the text (either via `\glslink` or `\gls` and related commands, or via `\glsadd` or `\glsaddall`.)

28. To change the sorting order, use the sort key. For example:

```
\newglossaryentry{universal}{name={\ensuremath{\mathcal{U}}},
description=The universal set,
sort=U}
```

29. You don't need to escape `makeindex`'s special characters:

```
\newglossaryentry{quote}{name={"},
description={Double quote character}}

\newglossaryentry{exclam}{name={!},
description={Exclamation mark}}

\newacronym{rna}{RNA}{ribonukleins\"aure}
```

30. Associated symbols can also be specified, but whether the symbol appears in the glossary depends on the glossary style. For example:

```
\newglossaryentry{metre}{name={metre},
description={A metric measurement of length},
symbol={m}}
```

    The predefined glossary styles that display the entry symbol are: long4col, long4colheader, long4colborder, long4colheaderborder, super4col, super4colheader, super4colborder and super4colheaderborder. All the other styles supplied by this package ignore the associated symbol.

31. Glossary styles can be set using the style package option. For example:

```
\usepackage[style=long3col]{glossary}
```

    or using `\glossarystyle{`⟨*style*⟩`}`. For example:

```
\glossarystyle{altlist}
```

    The predefined glossary styles provided by the glossaries bundle are detailed in subsection 4.15.

32. The list of numbers associated with each glossary entry can be suppressed using the package option nonumberlist:

```
\usepackage[nonumberlist]{glossaries}
```

33. By default, the glossaries will appear in an unnumbered chapter if chapters are defined, otherwise in an unnumbered section. This can be changed using the section package option. For example, to make the glossaries appear in an unnumbered section, even if chapters are defined, do:

```
\usepackage[section]{glossaries}
```

Other sectional units can also be specified as section=⟨*value*⟩. For example, to make the glossaries appear in unnumbered subsections:

```
\usepackage[section=subsection]{glossaries}
```

3 Overview

3.1 Package Options

The glossaries package options are as follows:

toc Add the glossaries to the table of contents

section This is a key=value option. Its value should be the name of a sectional unit (e.g. `chapter`). This will make the glossaries appear in the named sectional unit, otherwise the each glossary will appear in an unnumber chapter, if chapters exists, otherwise in an unnumbered section. Example:

```
\usepackage[section=subsection]{glossaries}
```

You can omit the value if you want to use sections, i.e.

```
\usepackage[section]{glossaries}
```

is equivalent to

```
\usepackage[section=section]{glossaries}
```

Note that the starred form of the sectioning command is always used since glossaries tend to be placed in unnumbered sections or chapters. If you want the glossaries to appeared in a numbered section, you will need to set `\glossarysection` to the relevant sectioning command. For example, to make the glossaries appear in numbered chapters, do:

`\glossarysection`

```
\let\glossarysection\chapter
```

style This is a key=value option. Its value should be the name of the glossary style to use.

nonumberlist This option will suppress the associated number lists in the glossaries (see also subsection 3.3.)

acronym Make a separate glossary for acronyms.

counter This is a key=value option. The value should be the name of the default counter to use in the number lists.

sanitize This is a key=value option whose value is a key=value list. By default, the glossaries package sanitizes the values of the name, description and symbol keys used when defining a new glossary entry. This may lead to unexpected results if you try to display these values within the document text. This sanitization can be switched off using the sanitize package option. (See subsection 4.2 and subsection 4.6 for further details.) For example, to switch

off the sanitization for the description and name keys, but not for the symbol key, do:

```
\usepackage[sanitize={name=false,description=false,%
symbol=true}]{glossaries}
```

Note: this sanitization only applies to the name, description and symbol keys. It doesn't apply to any of the other keys (except the sort key which is always sanitized) so fragile commands contained in the value of the other keys must always be protected using `\protect`. Since the value of the text key is obtained from the name key, you will still need to protect fragile commands in the name key if you don't use the text key.

3.2 Defining Glossary Entries

All glossary entries that are used in a document must be defined in the preamble. Only those entries that occur in the document (using any of the commands described in subsection 3.4 and subsection 3.5) will appear in the glossary. Each time an entry is used in this way, a line is added to an associated glossary (.glo) file, which then needs to be converted into a corresponding .gls file which contains the typeset glossary. The Perl script `makeglossaries` can be used to call `makeindex` using a customised style file for each of the glossaries that are defined in the document.

`\makeglossaries` The command `\makeglossaries` must be placed in the preamble, in order to create the customised `makeindex` style file, and to ensure that glossary entries are written to the appropriate output file.

`\newglossaryentry` New glossary entries are defined using the command:

`\newglossaryentry{`⟨*label*⟩`}{`⟨*key-val list*⟩`}`

The first argument, ⟨*label*⟩, must be a unique label with which to identify this entry. The second argument, ⟨*key-val list*⟩, is a key=value list that supplies the relevant information about this entry. There are two required fields: name and description. Available fields are listed below:

name The name of the entry (as it will appear in the glossary.)

description A brief description of this term (to appear in the glossary.)

text How this entry will appear in the document text when using `\gls` (or one of its uppercase variants.) If this field is omitted, the value of the name key is used.

first How the entry will appear in the document text the first time it is used with `\gls` (or one of its uppercase variants.) If this field is omitted, the value of the text key is used.

plural How the entry will appear in the document text when using `\glspl` (or one of its uppercase variants.) If this field is omitted, the value is obtained by appending an "s" to the value of the text field.

firstplural How the entry will appear in the document text the first time it is used with `\glspl` (or one of its uppercase variants.) If this field is omitted, the value is obtained by appending an "s" to the value of the first field.

symbol This field is provided to allow the user to specify an associated symbol, but most glossary styles ignore this value. If omitted, the value is set to `\relax`.

sort This value indicates how `makeindex` should sort this entry. If omitted, the value is given by the name field. This value is equivalent to `makeindex`'s "actual" character (which is usually the at-sign @.)

type This is the glossary type to which this entry belongs. If omitted, the default glossary is assumed.

`\loadglsentries` You can store all your glossary entry definitions in another file, and use:

`\loadglsentries[`⟨*type*⟩`]{`⟨*filename*⟩`}`

where ⟨*filename*⟩ is the name of the file containing all the `\newglossaryentry` commands. The optional argument ⟨*type*⟩ is the name of the glossary to which those entries should belong, for those entries where the type key has been omitted. Note that only those entries that have been used in the text will appear in the relevant glossaries.

3.3 Number lists

Each entry in the glossary has an associated *number list*. By default, these numbers refer to the pages on which that entry has been used (using any of the commands described in subsection 3.4 and subsection 3.5.) The number list can be suppressed using the nonumberlist package option, or an alternative counter can be set as the default using the counter package option.

3.4 Links to Glossary Entries

Once you have defined a glossary entry using `\newglossaryentry`, you can refer to that entry in the document using one of the commands listed in this section. The text which appears at that point in the document when using one of these commands is referred to as the *link text* (even if there are no hyperlinks.)

`\glstextformat` The way the link text is displayed depends on `\glstextformat{`⟨*text*⟩`}`. For example, to make all link text appear in a sans-serif font, do:

```
\renewcommand*{\glstextformat}[1]{\textsf{#1}}
```

`\glslink` The command:

`\glslink[`⟨*options*⟩`]{`⟨*label*⟩`}{`⟨*text*⟩`}`

will place ⟨*text*⟩ in the document at that point, and add a line into the associated glossary file for the glossary entry given by ⟨*label*⟩. If hyperlinks are supported, ⟨*text*⟩ will be a hyperlink to the relevant line in the glossary. The optional argument ⟨*options*⟩ must be a key=value list which can take any of the following keys:

format This specifies how to format the associated number for this entry in the glossary. This value is equivalent to the `makeindex` encap value, and (as with `\index`) the value needs to be the name of a command *without* the initial backslash. As with `\index`, the characters `(` and `)` can also be used to specify the beginning and ending of a number range. Again, as with `\index` the command should be the name of a command which takes an argument (which will be the associated number.) Be careful not to use a declaration (such as `\bfseries`) instead of a text block command (such as `\textbf`) as the effect will not be localised. If you want to apply more than one style to a given entry (e.g. **bold** and *italic*) you will need to create a command that applies both formats, e.g.

```
\newcommand*{\textbfem}[1]{\textbf{\emph{#1}}}
```

and use that command. (Just as you would have to do with `\index`.)

If you are using hyperlinks, and you want to change the font of the hyperlink, don't use `\hyperpage` (provided by the hyperref package) as the numbers may not refer to a page number. Instead, the glossaries package provides the following number formats:

hyperrm	The number is a serif hyperlink to the relevant part of the document
hypersf	The number is a sans-serif hyperlink to the relevant part of the document
hypertt	The number is a monospaced hyperlink to the relevant part of the document
hyperbf	The number is a bold hyperlink to the relevant part of the document
hypermd	The number is a medium weight hyperlink to the relevant part of the document
hyperit	The number is an italic hyperlink to the relevant part of the document
hypersl	The number is a slanted hyperlink to the relevant part of the document
hyperup	The number is an upright hyperlink to the relevant part of the document
hypersc	The number is a small caps hyperlink to the relevant part of the document
hyperemph	The number is an emphasized hyperlink to the relevant part of the document

Note that if the `\hyperlink` command hasn't been defined, the `hyper`⟨*xx*⟩ formats are equivalent to the analogous `\text`⟨*xx*⟩ font commands. If you want to make a new format, you will need to define a command which takes one argument, for example, if you want the associated number in the glossary to be in a bold sans-serif font, you can define a command called, say, `\hyperbsf`:

```
\newcommand{\hyperbsf}[1]{\textbf{\hypersf{#1}}}
```

and then use `hyperbsf` as the value for the format key. (See also subsection 4.13.)

counter This specifies which counter to use for the associated number for this glossary entry. (See also subsection 3.3.)

hyper This is a boolean key which can be used to enable/disable the hyperlink to the relevant entry in the glossary. (Note that setting `hyper=true` will have no effect if `\hyperlink` has not been defined.) The default value is `hyper=true`.

`\glslink*` There is also a starred version:

`\glslink*[`⟨*options*⟩`]{`⟨*label*⟩`}{`⟨*text*⟩`}`

which is equivalent to `\glslink`, except it sets `hyper=false`.

`\gls` The command:

`\gls[`⟨*options*⟩`]{`⟨*label*⟩`}[`⟨*insert*⟩`]`

is the same as `\glslink`, except that the link text is determined from the values of the text and first keys supplied when the entry was defined using `\newglossaryentry`. There are two uppercase variants:

`\Gls` `\Gls[`⟨*options*⟩`]{`⟨*label*⟩`}[`⟨*insert*⟩`]`

`\GLS` and `\GLS[`⟨*options*⟩`]{`⟨*label*⟩`}[`⟨*insert*⟩`]`

which make the first letter of the link, or all the link text, uppercase, respectively.

The final optional argument ⟨*insert*⟩, allows you to insert some additional text into the link text. By default, this will append ⟨*insert*⟩ at the end of the link text. The first optional argument, ⟨*options*⟩, is the same as the optional argument to `\glslink`.

There are also analogous plural forms:

`\glspl` `\glspl[`⟨*options*⟩`]{`⟨*label*⟩`}[`⟨*insert*⟩`]`

`\Glspl` `\Glspl[`⟨*options*⟩`]{`⟨*label*⟩`}[`⟨*insert*⟩`]`

`\GLSpl` `\GLSpl[`⟨*options*⟩`]{`⟨*label*⟩`}[`⟨*insert*⟩`]`

These determine the link text from the plural and firstplural keys supplied when the entry was first defined.

To make the description or symbol also appear in the link text, you will need to redefine `\glsdisplayfirst` and `\glsdisplay` or use the commands `\defglsdisplayfirst` and `\defglsdisplay`. See subsection 4.9 for further details. (Note that if you want either the description or symbol to appear in the link text, you will have to disable the sanitization of these keys, and protect fragile commands.)

3.5 Adding an entry to the glossary without generating text

`\glsadd` It is also possible to add a line in the glossary file without generating any text at

that point in the document.

`\glsadd[`⟨*options*⟩`]{`⟨*label*⟩`}`

This is similar to `\glslink`, only it doesn't produce any text (so therefore, there is no hyper key available in ⟨*options*⟩.)

`\glsaddall` To add a line for all entries that have been defined, use:

`\glsaddall[`⟨*glossary list*⟩`]`

If there are multiple glossaries, you can specify to add only those entries which belong to the glossaries listed in ⟨*glossary list*⟩ (which must be a comma separated list of glossary names.)

3.6 Displaying a glossary

`\printglossaries` The command `\printglossaries` will display all the defined glossaries. Note that no glossaries will appear until you have either used the Perl script `makeglossaries` or have directly used `makeindex`. If the glossary still does not appear, after you re-LaTeX your document, then check the `makeindex` log files to see if there is a problem.

`\printglossary` An individual glossary is displayed using:

`\printglossary[`⟨*options*⟩`]`

where ⟨*options*⟩ is a key-val list of options. The following keys are available:

type The value of this key specifies which glossary to print. If omitted, the default glossary is assumed.

title This is the glossary's title (overriding the title specified when the glossary was defined.)

toctitle This is the title to use for the table of contents (if the toc package option has been used.)

style This specifies which glossary style to use for this glossary.

3.7 Defining New Glossaries

`\newglossary` A new glossary can be defined using:

`\newglossary[`⟨*log-ext*⟩`]{`⟨*name*⟩`}{`⟨*in-ext*⟩`}{`⟨*out-ext*⟩`}{`⟨*title*⟩`}[`⟨*counter*⟩`]`

where ⟨*name*⟩ is label to assign to this glossary. (Note that the default glossary is labelled `main` and if you use the acronym package option, there will also be a glossary called `acronym`.) The arguments ⟨*in-ext*⟩ and ⟨*out-ext*⟩ specify the extensions to give to the input and output files for that glossary, ⟨*title*⟩ is the default title for this new glossary and the final optional argument ⟨*counter*⟩ specifies which counter to use for the associated number lists (see also subsection 3.3.) The first optional argument specifies the extension for the `makeindex` transcript file (this information is only used by `makeglossaries` which picks up the information from the auxiliary file.)

3.8 Acronyms

\newacronym

As you may have noticed in subsection 3.2, when you specify a new entry, you can specify alternate text to use when the term is first used in the document, this provides a useful means to define acronyms. The glossaries package defines the command:

\newacronym[⟨*key-val list*⟩]{⟨*label*⟩}{⟨*abbrv*⟩}{⟨*long*⟩}

This is equivalent to:

\newglossaryentry{⟨*label*⟩}{type=\acronymtype,
name={⟨*abbrv*⟩},
description={⟨*long*⟩},
text={⟨*abbrv*⟩},
first={⟨*long*⟩ (⟨*abbrv*⟩)},
plural={⟨*abbrv*⟩s},
firstplural={⟨*long*⟩s (⟨*abbrv*⟩s)},
⟨*key-val list*⟩}

The command \acronymtype is the name of the glossary in which the acronyms should appear. If the acronym package option has been used, this will be acronym, otherwise it will be main. The acronyms can then be used in exactly the same way as any other glossary entry.

As you can see from the above, \newacronym sets the description to the long form of the acronym. You may prefer to have a description, and set the name to both the long and abbreviated forms. For example, the following defines the acronym IDN:

```
\newacronym[name={identification number (IDN)},
description={A number uniquely identifying a particular
object}]{idn}{IDN}{identification number}
```

Alternatively:

```
\newacronym[description={identification number --- a number uniquely
identifying a particular object}]{idn}{IDN}{identification number}
```

3.9 Glossary Styles

The glossaries package comes with some pre-defined glossary styles. These are as follows:

list The list style uses the description environment. The entry name is placed in the optional argument of the \item command (so it will appear in bold by default). The description follows, and then the associated number list for that entry.

listgroup The listgroup style is like list, but the glossary groups have headings.

listhypergroup The listhypergroup style is like listgroup, but has a set of links to the glossary groups.

altlist The altlist style is like list but the description is placed on the following line.

altlistgroup The altlistgroup style is like altlist, but the glossary groups have headings.

altlisthypergroup The altlisthypergroup style is like altlistgroup, but has a set of links to the glossary groups.

long The long style uses the longtable environment. It has two columns, the first column contains the entry's name, the second column contains the description followed by the number list.

longborder The longborder style is like long, but has horizontal and vertical lines around it.

longheader The longheader style is like long, but has a header row.

longheaderborder The longheaderborder style is like longheader, but has horizontal and vertical lines around it.

long3col The long3col style is like long but has three columns. The first column contains the entry's name, the second column contains the description and the third column contains the number list.

long3colborder The long3colborder style is like the long3col style but has horizontal and vertical lines around it.

long3colheader The long3colheader style is like long3col, but has a header row.

long3colheaderborder The long3colheaderborder style is like long3colheader, but has horizontal and vertical lines around it.

long4col The long4col style is like long3col but has an additional column in which the entry's associated symbol appears.

long4colborder The long4colborder style is like the long4col style but has horizontal and vertical lines around it.

long4colheader The long4colheader style is like long4col, but has a header row.

long4colheaderborder The long4colheaderborder style is like long4colheader, but has horizontal and vertical lines around it.

super The super style uses the supertabular environment. It has two columns, the first column contains the entry's name, the second column contains the description followed by the number list.

superborder The superborder style is like super, but has horizontal and vertical lines around it.

superheader The superheader style is like super, but has a header row.

superheaderborder The superheaderborder style is like superheader, but has horizontal and vertical lines around it.

super3col The super3col style is like super but has three columns. The first column contains the entry's name, the second column contains the description and the third column contains the .

super3colborder The super3colborder style is like the super3col style but has horizontal and vertical lines around it.

super3colheader The super3colheader style is like super3col, but has a header row.

super3colheaderborder The super3colheaderborder style is like super3colheader, but has horizontal and vertical lines around it.

super4col The super4col style is like super3col but has an additional column in which the entry's associated symbol appears.

super4colborder The super4colborder style is like the super4col style but has horizontal and vertical lines around it.

super4colheader The super4colheader style is like super4col, but has a header row.

super4colheaderborder The super4colheaderborder style is like super4colheader, but has horizontal and vertical lines around it.

The glossary style can be set using the style package option, or using the style key in the optional argument to `\printglossary`, or using the command:

`\glossarystyle{`⟨*style-name*⟩`}`

For further details on creating or modifying glossary styles see subsection 4.13 and subsection 4.15.

`\glspostdescription` All the styles except for the three and four column styles use the command `\glspostdescription` after the description. This simply displays a full stop by default. To eliminate this full stop (or replace it with something else, say a comma), you will need to redefine `\glspostdescription` before the glossary is displayed.

4 Documented Code

4.1 Package Definition

This package requires LaTeX 2_ε.

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2007/09/13 v1.07 (NLCT)]
```

Required packages:

```
3 \RequirePackage{ifthen}
4 \RequirePackage{xkeyval}[2006/11/18]
5 \RequirePackage{xspace}
```

4.2 Package Options

The toc package option will add the glossaries to the table of contents. This is a boolean key, if the value is omitted it is taken to be true.

```
6 \define@boolkey{glossaries.sty}[gls]{toc}[true]{}
```

The sectional unit used to start the glossary is stored in `\@@glossarysec`. If chapters are defined, this is initialised to `chapter`, otherwise it is initialised to `section`.

`\@@glossarysec`

```
\@ifundefined{chapter}{\newcommand*{\@@glossarysec}{section}}{%
\newcommand*{\@@glossarysec}{chapter}}
```

The section key can be used to set the sectional unit. If no unit is specified, use `section` as the default. The starred form of the named sectional unit will be used. If you want some other way to start the glossary section (e.g. a numbered section) you will have to redefined `\glossarysection`.

```
\define@choicekey{glossaries.sty}{section}{part,chapter,section,%
subsection,subsubsection,paragraph,subparagraph}[section]{%
\renewcommand*{\@@glossarysec}{#1}}
```

The default glossary style is stored in `\@glossary@default@style`. This is initialised to `list`. (The list style is defined in the accompanying glossary-list package described in subsection 4.15.)

`\@glossary@default@style`

```
\newcommand*{\@glossary@default@style}{list}
```

The default glossary style can be changed using the style package option. The value can be the name of any defined glossary style. The glossary style is set at the beginning of the document, so you can still use the style key to set a style that is defined in another package. This package comes with some predefined styles that are defined in subsection 4.15.

```
\define@key{glossaries.sty}{style}{%
\renewcommand*{\@glossary@default@style}{#1}}
```

Each entry within a given glossary will have an associated number list. By default, this refers to the page numbers on which that entry has been used, but it can also refer to any counter used in the document (such as the section or equation counters). The default number list format displays the number list "as is":

`\glossaryentrynumbers`

```
\newcommand*{\glossaryentrynumbers}[1]{#1}
```

Note that the entire number list for a given entry will be passed to `\glossaryentrynumbers` so any font changes will also be applied to the delimiters. The nonumberlist package option suppresses the number lists (this simply redefines `\glossaryentrynumbers` to ignores its argument.)

```
\DeclareOptionX{nonumberlist}{%
\renewcommand*{\glossaryentrynumbers}[1]{}}
```

Keep track of the default glossary. This is initialised to the main glossary, but can be changed if for some reason you want to make a secondary glossary the main glossary. This affects any commands that can optionally take a glossary name as an argument (or as the value of the type key in a key-value list.) This was mainly done so that `\loadglsentries` can temporarily change `\glsdefaulttype` while it loads a file containing new glossary entries (see subsection 4.8).

`\glsdefaulttype`

```
\newcommand{\glsdefaulttype}{main}
```

Keep track of which glossary the acronyms are in. This is initialised to `\glsdefaulttype`, but is changed by the acronym package option.

`\acronymtype`

```
\newcommand{\acronymtype}{\glsdefaulttype}
```

The acronym option sets an associated conditional which is used in subsection 4.14 to determine whether or not to define a separate glossary for acronyms.

```
\define@boolkey{glossaries.sty}[gls]{acronym}[true]{}
```

The default counter associated with the numbers in the glossary is stored in `\glscounter`. This is initialised to the page counter. This is used as the default counter when a new glossary is defined, unless a different counter is specified in the optional argument to `\newglossary` (see subsection 4.5).

`\glscounter`

```
\newcommand{\glscounter}{page}
```

The counter option changes the default counter. (This just redefines `\glscounter`.)

```
\define@key{glossaries.sty}{counter}{%
\renewcommand*{\glscounter}{#1}}
```

The glossary keys whose values are written to another file (i.e. sort, name, description and symbol) need to be sanitized, otherwise fragile commands would not be able to be used in `\newglossaryentry`. However, strange results will occur if you then use those fields in the document. As these fields are not normally used in the document, but are by default only used in the glossary, the default is to sanitize them. If however you want to use these values in the document (either by redefining commands like `\glsdisplay` or by using commands like `\glsentrydesc`) you will have to switch off the sanitization using the sanitize package option, but you will then have to use `\protect` to protect fragile commands when defining new glossary entries. The sanitize option takes a key-value list as its value, which can be used to switch individual values on and off. For example:

```
\usepackage[sanitize={description,name,symbol=false}]{glossaries}
```

will switch off the sanitization for the symbol key, but switch it on for the description and name keys. This would mean that you can use fragile commands in the description and name when defining a new glossary entry, but not for the symbol.

The default values are defined as:

`\@gls@sanitizedesc`

```
\newcommand*{\@gls@sanitizedesc}{\@onelevel@sanitize\@glo@desc}
```

`\@gls@sanitizename`

```
\newcommand*{\@gls@sanitizename}{\@onelevel@sanitize\@glo@name}
```

`\@gls@sanitizesymbol`

```
\newcommand*{\@gls@sanitizesymbol}{\@onelevel@sanitize\@glo@symbol}
```

(There is no equivalent for the sort key, since that is only provided for the benefit of `makeindex`, and so will always be sanitized.)

Before defining the sanitize package option, The key-value list for the sanitize value needs to be defined. These are all boolean keys. If they are not given a value, assume `true`.

Firstly the description. If set, it will redefine \@gls@sanitizedesc to use \@onelevel@sanitize, otherwise \@gls@sanitizedesc will do nothing.

```
\define@boolkey[gls]{sanitize}{description}[true]{%
\ifgls@sanitize@description
  \renewcommand*{\@gls@sanitizedesc}{\@onelevel@sanitize\@glo@desc}%
\else
  \renewcommand*{\@gls@sanitizedesc}{}%
\fi
}
```

Similarly for the name key:

```
\define@boolkey[gls]{sanitize}{name}[true]{%
\ifgls@sanitize@name
  \renewcommand*{\@gls@sanitizename}{\@onelevel@sanitize\@glo@name}%
\else
  \renewcommand*{\@gls@sanitizename}{}%
\fi}
```

and for the symbol key:

```
\define@boolkey[gls]{sanitize}{symbol}[true]{%
\ifgls@sanitize@symbol
  \renewcommand*{\@gls@sanitizesymbol}{%
\@onelevel@sanitize\@glo@symbol}%
\else
  \renewcommand*{\@gls@sanitizesymbol}{}%
\fi}
```

Now define the sanitize option. It can either take a key-val list as its value, or it can take the keyword none, which is equivalent to description=false, symbol=false, name=false:

```
\define@key{glossaries.sty}{sanitize}[description=true,symbol=true,
name=true]{%
\ifthenelse{\equal{#1}{none}}{%
\renewcommand*{\@gls@sanitizedesc}{}%
\renewcommand*{\@gls@sanitizename}{}%
\renewcommand*{\@gls@sanitizesymbol}{}%
}{\setkeys[gls]{sanitize}{#1}}%
}
```

Process package options:

```
\ProcessOptionsX
```

If chapters are defined and the user has requested the section counter as a package option, \@chapter will be modified so that it adds a section.⟨*n*⟩.0 target, otherwise entries placed before the first section of a chapter will have undefined links.

The same problem will also occur if a lower sectional unit is used, but this is less likely to happen. If it does, or if you change \glscounter to section later, you will have to specify a different counter for the entries that give rise to a name{⟨*section-level*⟩.⟨*n*⟩.0} non-existent warning (e.g. \gls[counter=chapter]{label}).

```
\ifthenelse{\equal{\glscounter}{section}}{%
\@ifundefined{chapter}{}{%
\let\@gls@old@chapter\@chapter
\def\@chapter[#1]#2{\@gls@old@chapter[{#1}]{#2}%
```

```
\@ifundefined{hyperdef}{}{\hyperdef{section}{\thesection}{}}}}}{}
```

4.3 Default values

This section sets up default values that are used by this package. Some of the names may already be defined (e.g. by babel) so `\providecommand` is used.

Main glossary title:

`\glossaryname`

```
\providecommand*{\glossaryname}{Glossary}
```

The title for the `acronym` glossary type (which is defined if `acronym` package option is used) is given by `\acronymname`. If the `acronym` package option is not used, `\acronymname` won't be used.

`\acronymname`

```
\providecommand*{\acronymname}{Acronyms}
```

The following commands provide text for the headers used by some of the tabular-like glossary styles. Whether or not they get used in the glossary depends on the glossary style.

`\entryname`

```
\providecommand*{\entryname}{Notation}
```

`\descriptionname`

```
\providecommand*{\descriptionname}{Description}
```

`\symbolname`

```
\providecommand*{\symbolname}{Symbol}
```

`\pagelistname`

```
\providecommand*{\pagelistname}{Page List}
```

Labels for `makeindex`'s symbol and number groups:

`\glssymbolsgroupname`

```
\providecommand*{\glssymbolsgroupname}{Symbols}
```

`\glsnumbersgroupname`

```
\providecommand*{\glsnumbersgroupname}{Numbers}
```

The description terminator is given by `\glspostdescription` (except for the 3 and 4 column styles). This is a full stop by default:

`\glspostdescription`

```
\newcommand*{\glspostdescription}{.}
```

The name of the `makeindex` style file is given by `\istfilename`. This file is created by `\writeist` (which is used by `\makeglossaries`) so redefining this command will only have an effect if it is done *before* the `ist` file is created.

`\istfilename`

```
\providecommand*{\istfilename}{\jobname.ist}
```

The makeglossaries Perl script picks up this name from the auxiliary file and passes it to makeindex using the -s option. Since its not required by LaTeX, \@istfilename ignores its argument.

\@istfilename

```
\newcommand*{\@istfilename}[1]{}
```

This command is the value of the page_compositor makeindex key. Again, any redefinition of this command must take place *before* \writeist otherwise it will have no effect.

\glscompositor

```
\newcommand{\glscompositor}{.}
```

(The page compositor is usually defined as a dash when using makeindex, but most of the standard counters used by LaTeX use a full stop as the compositor, which is why I have used it as the default.)

The command \glsnumberformat indicates the default format for the page numbers in the glossary. (Note that this is not the same as \glossaryentrynumbers, but applies to individual numbers or groups of numbers within an entry's associated number list.) If hyperlinks are defined, it will use \glshypernumber, otherwise it will simply display its argument "as is".

\glsnumberformat

```
\@ifundefined{hyperlink}{%
\newcommand*{\glsnumberformat}[1]{#1}}{%
\newcommand*{\glsnumberformat}[1]{\glshypernumber{#1}}}
```

Individual numbers in an entry's associated number list are delimited using \delimN (which corresponds to the delim_n makeindex keyword.) The default value is a comma followed by a space.

\delimN

```
\newcommand{\delimN}{, }
```

A range of numbers within an entry's associated number list is delimited using \delimR (which corresponds to the delim_r makeindex keyword.) The default is an en-dash.

\delimR

```
\newcommand{\delimR}{--}
```

The glossary preamble is given by \glossarypreamble. This will appear after the glossary sectioning command, and before the theglossary environment. It is designed to allow the user to add information pertaining to the glossary (e.g. "page numbers in italic indicate the primary definition") therefore \glossarypremable shouldn't be affected by the glossary style. (So if you define your own glossary style, don't have it change \glossarypreamble.) The preamble is empty by default. If you have multiple glossaries, and you want a different preamble for each glossary, you will need to use \printglossary for each glossary type, instead of \printglossaries, and redefine \glossarypreamble before each \printglossary.

\glossarypreamble

```
\newcommand*{\glossarypreamble}{}
```

The glossary postamble is given by \glossarypostamble. This is provided to allow the user to add something after the end of the theglossary environment (again, this shouldn't be affected by the glossary style.) It is, of course, possible to simply add the text after \printglossary, but if you only want the postamble to appear after the first glossary, but not after subsequent glossaries, you can do something like:

```
\renewcommand{\glossarypostamble}{For a complete list of terms
see \cite{blah}\gdef\glossarypreamble{}}
```

\glossarypostamble

```
\newcommand*{\glossarypostamble}{}
```

The sectioning command that starts a glossary is given by \glossarysection. (This does not form part of the glossary style, and so should not be changed by a glossary style.) If \phantomsection is defined, it uses \p@glossarysection, otherwise it uses \@glossarysection.

\glossarysection

```
\newcommand*{\glossarysection}[2][\@gls@title]{%
\def\@gls@title{#2}%
\@ifundefined{phantomsection}{%
\@glossarysection{#1}{#2}}{\@p@glossarysection{#1}{#2}}%
\@mkboth{\glossarytoctitle}{\glossarytoctitle}%
}
```

The required sectional unit is given by \@@glossarysec which was defined by the section package option. The starred form of the command is chosen. If you want a numbered section for the glossary or if you don't want any sectional command, you will need to redefine \glossarysection.

The command \@glossarysection indicates how to start the glossary section if \phantomsection is not defined.

\@glossarysection

```
\newcommand*{\@glossarysection}[2]{%
\csname\@@glossarysec\endcsname*{#2}%
\@gls@toc{#1}{\@@glossarysec}}
```

As \@glossarysection, but put in \phantomsection, and swap where \@gls@toc goes. If using chapters do a \clearpage. This ensures that the hyper link from the table of contents leads to the line above the heading, rather than the line below it.

\@p@glossarysection

```
\newcommand*{\@p@glossarysection}[2]{%
\gls@doclearpage
\phantomsection\@gls@toc{#1}{\@@glossarysec}%
\csname\@@glossarysec\endcsname*{#2}}
```

The \gls@doclearpage command is used to issue a \clearpage (or \cleardoublepage) depending on whether the glossary sectional unit is a chapter. If the sectional unit is something else, do nothing.

\gls@doclearpage

```
\newcommand{\gls@doclearpage}{%
\ifthenelse{\equal{\@@glossarysec}{chapter}}{%
\@ifundefined{cleardoublepage}{\clearpage}{\cleardoublepage}}{}%
}
```

The glossary is added to the table of contents if glstoc flag set. If it is set, `\@gls@toc` will add a line to the `toc` file, otherwise it will do nothing. (The first argument to `\@gls@toc` is the title for the table of contents, the second argument is the sectioning type.)

\@gls@toc

```
\newcommand*{\@gls@toc}[2]{%
\ifglstoc \addcontentsline{toc}{#2}{#1}\fi}
```

4.4 Loops and conditionals

To iterate through all glossaries (or comma-separated list of glossary names given in optional argument) use:

`\forallglossaries[`⟨*glossary list*⟩`]{`⟨*cmd*⟩`}{`⟨*code*⟩`}`

where ⟨*cmd*⟩ is a control sequence which will be set to the name of the glossary in the current iteration.

\forallglossaries

```
\newcommand*{\forallglossaries}[3][\@glo@types]{%
\@for#2:=#1\do{\ifthenelse{\equal{#2}{}}{}{#3}}}
```

To iterate through all entries in a given glossary use:

`\forglsentries[`⟨*type*⟩`]{`⟨*cmd*⟩`}{`⟨*code*⟩`}`

where ⟨*type*⟩ is the glossary label and ⟨*cmd*⟩ is a control sequence which will be set to the entry label in the current iteration.

\forglsentries

```
\newcommand*{\forglsentries}[3][\glsdefaulttype]{%
\edef\@@glo@list{\csname glolist@#1\endcsname}%
\@for#2:=\@@glo@list\do{%
\ifthenelse{\equal{#2}{}}{}{#3}}}
```

To iterate through all glossary entries over all glossaries listed in the optional argument (the default is all glossaries) use:

`\forallglsentries[`⟨*glossary list*⟩`]{`⟨*cmd*⟩`}{`⟨*code*⟩`}`

Within `\forallglsentries`, the current glossary type is given by `\@@this@glo@`.

\forallglsentries

```
\newcommand*{\forallglsentries}[3][\@glo@types]{%
\expandafter\forallglossaries\expandafter[#1]{\@@this@glo@}{%
\forglsentries[\@@this@glo@]{#2}{#3}}}
```

To check to see if a glossary exists use:

`\ifglossaryexists{`⟨*type*⟩`}{`⟨*true-text*⟩`}{`⟨*false-text*⟩`}`

where ⟨*type*⟩ is the glossary's label.

`\ifglossaryexists`

```
\newcommand{\ifglossaryexists}[3]{%
\@ifundefined{@glotype@#1@out}{#3}{#2}}
```

To check to see if a glossary entry has been defined use:

`\ifglsentryexists{`⟨*label*⟩`}{`⟨*true text*⟩`}{`⟨*false text*⟩`}`

where ⟨*label*⟩ is the entry's label.

`\ifglsentryexists`

```
\newcommand{\ifglsentryexists}[3]{%
\@ifundefined{glo@#1@name}{#3}{#2}}
```

To determine if given glossary entry has been used in the document text yet use:

`\ifglsused{`⟨*label*⟩`}{`⟨*true text*⟩`}{`⟨*false text*⟩`}`

where ⟨*label*⟩ is the entry's label. If true it will do ⟨*true text*⟩ otherwise it will do ⟨*false text*⟩.

`\ifglsused`

```
\newcommand*{\ifglsused}[3]{\ifthenelse{\boolean{glo@#1@flag}}{#2}{#3}}
```

The following two commands will cause an error if the given condition fails:

`\glsdoifexists{`⟨*label*⟩`}{`⟨*code*⟩`}`

Generate an error if entry specified by ⟨*label*⟩ doesn't exists, otherwise do ⟨*code*⟩.

`\glsdoifexists`

```
\newcommand{\glsdoifexists}[2]{\ifglsentryexists{#1}{#2}{%
\PackageError{glossaries}{Glossary entry `#1' has not been
defined.}{You need to define a glossary entry before you
can use it.}}}
```

`\glsdoifnoexists{`⟨*label*⟩`}{`⟨*code*⟩`}`

The opposite: only do second argument if the entry doesn't exists. Generate an error message if it exists.

`\glsdoifnoexists`

```
\newcommand{\glsdoifnoexists}[2]{\ifglsentryexists{#1}{%
\PackageError{glossaries}{Glossary entry `#1' has already
been defined.}{}}{#2}}
```

4.5 Defining new glossaries

A comma-separated list of glossary names is stored in `\@glo@types`. When a new glossary type is created, its identifying name is added to this list. This is used by commands that iterate through all glossaries (such as `\makeglossaries` and `\printglossaries`.)

`\@glo@types`

```
\newcommand*{\@glo@types}{,}
```

A new glossary type is defined using `\newglossary`. Syntax:

`\newglossary[`⟨*log-ext*⟩`]{`⟨*name*⟩`}{`⟨*in-ext*⟩`}{`⟨*out-ext*⟩`} {`⟨*title*⟩`}[`⟨*counter*⟩`]`

where ⟨*log-ext*⟩ is the extension of the `makeindex` transcript file, ⟨*in-ext*⟩ is the extension of the glossary input file (read in by `\printglossary` and created by `makeindex`), ⟨*out-ext*⟩ is the extension of the glossary output file which is read in by `makeindex` (lines are written to this file by the `\glossary` command), ⟨*title*⟩ is the title of the glossary that is used in `\glossarysection` and ⟨*counter*⟩ is the default counter to be used by entries belonging to this glossary. The `makeglossaries` Perl script reads in the relevant extensions from the auxiliary file, and passes the appropriate file names and switches to `makeindex`.)

`\newglossary`

```
\newcommand*{\newglossary}[5][glg]{%
\ifglossaryexists{#2}{%
\PackageError{glossaries}{Glossary type `#2' already exists}{%
You can't define a new glossary called `#2' because it already
exists}%
}{%
```

Add this to the list of glossary types:

```
\toks@{#2}\edef\@glo@types{\@glo@types\the\toks@,}%
```

Define a comma-separated list of labels for this glossary type, so that all the entries for this glossary can be reset with a single command. When a new entry is created, its label is added to this list.

```
\expandafter\gdef\csname glolist@#2\endcsname{,}%
```

Store details of this new glossary type:

```
\expandafter\def\csname @glotype@#2@in\endcsname{#3}%
\expandafter\def\csname @glotype@#2@out\endcsname{#4}%
\expandafter\def\csname @glotype@#2@title\endcsname{#5}%
\protected@write\@auxout{}{\string\@newglossary{#2}{#1}{#3}{#4}}%
```

How to display this entry in the document text (uses `\glsdisplay` and `\glsdisplayfirst` by default). These can be redefined by the user later if required (see `\defglsdisplay` and `\defglsdisplayfirst`)

```
\expandafter\gdef\csname gls@#2@display\endcsname{%
\glsdisplay}%
\expandafter\gdef\csname gls@#2@displayfirst\endcsname{%
\glsdisplayfirst}%
```

Find out if the final optional argument has been specified, and use it to set the counter associated with this glossary. (Uses `\glscounter` if no optional argument is present.)

```
\@ifnextchar[{\@gls@setcounter{#2}}{\@gls@setcounter{#2}[\glscounter]}}
```

Only defined new glossaries in the preamble:

```
\@onlypreamble{\newglossary}
```

`\@newglossary` is used to specify the file extensions for the `makeindex` input, output and transcript files. It is written to the auxiliary file by `\newglossary`. Since it is not used by LaTeX, `\@newglossary` simply ignores its arguments.

`\@newglossary`

```
\newcommand*{\@newglossary}[4]{}
```

Store counter to be used for given glossary type (the first argument is the glossary label, the second argument is the name of the counter):

`\@gls@setcounter`

```
\def\@gls@setcounter#1[#2]{%
\expandafter\def\csname @glotype@#1@counter\endcsname{#2}%
}
```

Get counter associated with given glossary (the argument is the glossary label):

`\@gls@getcounter`

```
\newcommand*{\@gls@getcounter}[1]{%
\csname @glotype@#1@counter\endcsname}
```

Define the main glossary. This will be the first glossary to be displayed when using `\printglossaries`.

```
\newglossary{main}{gls}{glo}{\glossaryname}
```

4.6 Defining new entries

New glossary entries are defined using `\newglossaryentry`. This command requires a label and a key-value list that defines the relevant information for that entry. The definition for these keys follows. Note that the name, description and symbol keys will be sanitized later, depending on the value of the package option sanitize (this means that if some of the keys haven't been defined, they can be constructed from the name and description key before they are sanitized).

The name key indicates the name of the term being defined. This is how the term will appear in the glossary. The name key is required when defining a new glossary entry.

```
\define@key{glossentry}{name}{%
\def\@glo@name{#1}%
}
```

The description key is usually only used in the glossary, but can be made to appear in the text by redefining `\glsdisplay` and `\glsdisplayfirst` (or using `\defglsdisplay` and `\defglsdisplayfirst`), however, you will have to disable the sanitize option (using the sanitize package option, `sanitize={description=false}`, and protect fragile commands.) The description key is required when defining a new glossary entry. (Be careful not to make the description too long, because `makeindex` has a limited buffer. `\@glo@desc` is defined to be a short command to discourage lengthy descriptions for this reason. If you do have a very long

description, or if you require paragraph breaks, define a separate command that contains the description, and use it as the value to the description key.)

```
\define@key{glossentry}{description}{%
\def\@glo@desc{#1}%
}
```

The sort key needs to be sanitized here (the sort key is provided for `makeindex`'s benefit, not for use in the document.) The sort key is optional when defining a new glossary entry. If omitted, the value is given by ⟨*name*⟩ ⟨*description*⟩.

```
\define@key{glossentry}{sort}{%
\def\@glo@sort{#1}%
\@onelevel@sanitize\@glo@sort}
```

The text key determines how the term should appear when used in the document (i.e. outside of the glossary). If omitted, the value of the name key is used instead.

```
\define@key{glossentry}{text}{%
\def\@glo@text{#1}%
}
```

The plural key determines how the plural form of the term should be displayed in the document. If omitted, the plural is constructed by appending an "s" to the value of the text key.

```
\define@key{glossentry}{plural}{%
\def\@glo@plural{#1}%
}
```

The first key determines how the entry should be displayed in the document when it is first used. If omitted, it is taken to be the same as the value of the text key.

```
\define@key{glossentry}{first}{%
\def\@glo@first{#1}%
}
```

The firstplural key is used to set the plural form for first use, in the event that the plural is required the first time the term is used. If omitted, it is constructed by appending an "s" to the value of the first key.

```
\define@key{glossentry}{firstplural}{%
\def\@glo@firstplural{#1}%
}
```

The symbol key is ignored by most of the predefined glossary styles, and defaults to `\relax` if omitted. It is provided for glossary styles that require an associated symbol, as well as a name and description. To make this value appear in the glossary, you need to redefine `\glossaryentryfield` so that it uses its fourth parameter. If you want this value to appear in the text when the term is used by commands like `\gls`, you will need to change `\glsdisplay` and `\glsdisplayfirst` (either explicitly for all glossaries or via `\defglsdisplay` and `\defglsdisplayfirst` for individual glossaries.)

```
\define@key{glossentry}{symbol}{%
\def\@glo@symbol{#1}%
}
```

The type key specifies to which glossary this entry belongs. If omitted, the default glossary is used.

```
\define@key{glossentry}{type}{%
\def\@glo@type{#1}}
```

The counter key specifies the name of the counter associated with this glossary entry:

```
\define@key{glossentry}{counter}{%
\@ifundefined{c@#1}{\PackageError{glossaries}{There is no counter
called '#1'}{The counter key should have the name of a valid
counter as its value}}{%
\def\@glo@counter{#1}}}
```

Define `\newglossaryentry {`⟨*label*⟩`} {`⟨*key-val list*⟩`}`. There are two required fields in ⟨*key-val list*⟩: name and description. (See above.)

`\newglossaryentry`

```
\DeclareRobustCommand{\newglossaryentry}[2]{%
```

Check to see if this glossary entry has already been defined:

```
\glsdoifnoexists{#1}{%
```

Set up defaults. If the name or description keys are omitted, an error will be generated.

```
\def\@glo@name{\PackageError{glossaries}{name key required in
\string\newglossaryentry}{You haven't specified the entry name}}%
\def\@glo@desc{\PackageError{glossaries}{desc key required in
\string\newglossaryentry}{You haven't specified the entry description}}%
\def\@glo@type{\glsdefaulttype}%
\def\@glo@symbol{\relax}%
\def\@glo@text{\@glo@name}%
\def\@glo@plural{\@glo@text s}%
\def\@glo@first{\@glo@text}%
\def\@glo@firstplural{\@glo@plural}%
\def\@glo@sort{\@glo@name}%
\def\@glo@counter{\@gls@getcounter{\@glo@type}}%
```

Extract key-val information from third parameter:

```
\setkeys{glossentry}{#2}%
```

Check to see if this glossary type has been defined, if it has, add this label to the relevant list, otherwise generate an error.

```
\@ifundefined{glolist@\@glo@type}{\PackageError{glossaries}{%
Glossary type '\@glo@type' has not been defined}{%
You need to define a new glossary type, before making entries
in it}}{%
\protected@edef\@glolist@{\csname glolist@\@glo@type\endcsname}%
\expandafter\xdef\csname glolist@\@glo@type\endcsname{\@glolist@{#1},}%
}%
```

Define commands associated with this entry:

```
\expandafter\protected@xdef\csname glo@#1@text\endcsname{\@glo@text}%
\expandafter\protected@xdef\csname glo@#1@plural\endcsname{\@glo@plural}%
\expandafter\protected@xdef\csname glo@#1@first\endcsname{\@glo@first}%
\expandafter\protected@xdef\csname glo@#1@firstpl\endcsname{\@glo@firstplural}%
\expandafter\protected@xdef\csname glo@#1@type\endcsname{\@glo@type}%
\expandafter\protected@xdef\csname glo@#1@counter\endcsname{\@glo@counter}%
\@gls@sanitizename
\expandafter\protected@xdef\csname glo@#1@name\endcsname{\@glo@name}%
```

```
\@gls@sanitizedesc
\expandafter\protected@xdef\csname glo@#1@desc\endcsname{\@glo@desc}%
\expandafter\protected@xdef\csname glo@#1@sort\endcsname{\@glo@sort}%
\@gls@sanitizesymbol
\expandafter\protected@xdef\csname glo@#1@symbol\endcsname{\@glo@symbol}%
```

Define an associated boolean variable to determine whether this entry has been used yet (needs to be defined globally):

```
\expandafter\gdef\csname glo@#1@flagfalse\endcsname{%
\expandafter\global\expandafter
\let\csname ifglo@#1@flag\endcsname\iffalse}%
\expandafter\gdef\csname glo@#1@flagtrue\endcsname{%
\expandafter\global\expandafter
\let\csname ifglo@#1@flag\endcsname\iftrue}%
\csname glo@#1@flagfalse\endcsname
}}
```

Only defined new glossary entries in the preamble:

```
\@onlypreamble{\newglossaryentry}
```

4.7 Resetting and unsetting entry flags

Each glossary entry is assigned a conditional of the form `\ifglo@`⟨*label*⟩`@flag` which determines whether or not the entry has been used (see also `\ifglsused` defined below.) These flags can be set and unset using the following macros:

The command `\glsreset{`⟨*label*⟩`}` can be used to set the entry flag to indicate that it hasn't been used yet. The required argument is the entry label.

`\glsreset`

```
\newcommand*{\glsreset}[1]{%
\glsdoifexists{#1}{%
\expandafter\global\csname glo@#1@flagfalse\endcsname}}
```

As above, but with only a local effect:

`\glslocalreset`

```
\newcommand*{\glslocalreset}[1]{%
\glsdoifexists{#1}{%
\expandafter\let\csname ifglo@#1@flag\endcsname\iffalse}}
```

The command `\glsunset{`⟨*label*⟩`}` can be used to set the entry flag to indicate that it has been used. The required argument is the entry label.

`\glsunset`

```
\newcommand*{\glsunset}[1]{%
\glsdoifexists{#1}{%
\expandafter\global\csname glo@#1@flagtrue\endcsname}}
```

As above, but with only a local effect:

`\glslocalunset`

```
\newcommand*{\glslocalunset}[1]{%
\glsdoifexists{#1}{%
\expandafter\let\csname ifglo@#1@flag\endcsname\iftrue}}
```

Reset all entries for the named glossaries (supplied in a comma-separated list). Syntax: `\glsresetall[`⟨*glossary-list*⟩`]`

`\glsresetall`

```
\newcommand*{\glsresetall}[1][\@glo@types]{%
\forallglsentries[#1]{\@glsentry}{%
\glsreset{\@glsentry}}}
```

As above, but with only a local effect:

`\glslocalresetall`

```
\newcommand*{\glslocalresetall}[1][\@glo@types]{%
\forallglsentries[#1]{\@glsentry}{%
\glslocalreset{\@glsentry}}}
```

Unset all entries for the named glossaries (supplied in a comma-separated list). Syntax: `\glsunsetall[`⟨*glossary-list*⟩`]`

`\glsunsetall`

```
\newcommand*{\glsunsetall}[1][\@glo@types]{%
\forallglsentries[#1]{\@glsentry}{%
\glsunset{\@glsentry}}}
```

As above, but with only a local effect:

`\glslocalunsetall`

```
\newcommand*{\glslocalunsetall}[1][\@glo@types]{%
\forallglsentries[#1]{\@glsentry}{%
\glslocalunset{\@glsentry}}}
```

4.8 Loading files containing glossary entries

Glossary entries can be defined in an external file. These external files can contain `\newglossaryentry` and `\newacronym` commands.[2]

`\loadglsentries[`⟨*type*⟩`]{`⟨*filename*⟩`}`

This command will input the file using `\input`. The optional argument specifies to which glossary the entries should be assigned if they haven't used the `type` key. If the optional argument is not specified, the default glossary is used. Only those entries used in the document (via `\glslink`, `\gls`, `\glspl` and uppercase variants or `\glsadd` and `\glsaddall` will appear in the glossary.) The mandatory argument is the filename (with or without .tex extension.)

`\loadglsentries`

```
\newcommand*{\loadglsentries}[2][\@gls@default]{%
\let\@gls@default\glsdefaulttype
\def\glsdefaulttype{#1}\input{#2}%
\let\glsdefaulttype\@gls@default}
```

`\loadglsentries` can only be used in the preamble:

```
\@onlypreamble{\loadglsentries}
```

[2] and any other valid LaTeX code that can be used in the preamble.

4.9 Using glossary entries in the text

Any term that has been defined using `\newglossaryentry` (or `\newacronym`) can be displayed in the text (i.e. outside of the glossary) using one of the commands defined in this section. Unless you use `\glslink`, the way the term appears in the text is determined by `\glsdisplayfirst` (if it is the first time the term has been used) or `\glsdisplay` (for subsequent use.) Any formatting commands (such as `\textbf` is governed by `\glstextformat`. By default this just displays the link text "as is".

`\glstextformat`

```
250 \newcommand*{\glstextformat}[1]{#1}
```

The first time an entry is used, the way in which it is displayed is governed by `\glsdisplayfirst`. This takes four parameters: `#1` will be the value of the entry's first or firstplural key, `#2` will be the value of the entry's description key, `#3` will be the value of the entry's symbol key and `#4` is additional text supplied by the final optional argument to commands like `\gls` and `\glspl`. The default is to display the first parameter followed by the additional text.

`\glsdisplayfirst`

```
251 \newcommand*{\glsdisplayfirst}[4]{#1#4}
```

After the first use, the entry is displayed according to the format of `\glsdisplay`. Again, it takes four parameters: `#1` will be the value of the entry's text or plural key, `#2` will be the value of the entry's description key, `#3` will be the value of the entry's symbol key and `#4` is additional text supplied by the final optional argument to commands like `\gls` and `\glspl`.

`\glsdisplay`

```
252 \newcommand*{\glsdisplay}[4]{#1#4}
```

When a new glossary is created it uses `\glsdisplayfirst` and `\glsdisplay` as the default way of displaying its entry in the text. This can be changed for the entries belonging to an individual glossary using `\defglsdisplay` and `\defglsdisplayfirst`.

`\defglsdisplay[`⟨*type*⟩`]{`⟨*definition*⟩`}`

The glossary type is given by ⟨*type*⟩ (the default glossary if omitted) and ⟨*definition*⟩ should have at most `#1`, `#2`, `#3` and `#4`. These represent the same arguments as those described for `\glsdisplay`.

`\defglsdisplay`

```
253 \newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
254 \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}}
```

`\defglsdisplayfirst[`⟨*type*⟩`]{`⟨*definition*⟩`}`

The glossary type is given by ⟨*type*⟩ (the default glossary if omitted) and ⟨*definition*⟩ should have at most `#1`, `#2`, `#3` and `#4`. These represent the same arguments as those described for `\glsdisplayfirst`.

\defglsdisplayfirst

```
\newcommand*{\defglsdisplayfirst}[2][\glsdefaulttype]{%
\expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}}
```

4.9.1 Links to glossary entries

The links to glossary entries all have a first optional argument that can be used to change the format and counter of the associated entry number. Except for `\glslink`, the commands like `\gls` have a final optional argument that can be used to insert additional text in the link (this will usually be appended, but can be redefined using `\defglsdisplay` and `\defglsdisplayfirst`). It goes against the LaTeX norm to have an optional argument after the mandatory arguments, but it makes more sense to write, say, `\gls{label}['s]` rather than, say, `\gls[append='s]{label}`. Since these control sequences are defined to include the final square bracket, spaces will be ignored after them. This is likely to lead to confusion as most users would not expect, say, `\gls{`⟨*label*⟩`}` to ignore following spaces, so `\xspace` is used.

The following keys can be used in the first optional argument. The counter key checks that the value is the name of a valid counter.

```
\define@key{glslink}{counter}{%
\@ifundefined{c@#1}{\PackageError{glossaries}{There is no counter
called '#1'}{The counter key should have the name of a valid
counter as its value}}{%
\def\@gls@counter{#1}}}
```

The value of the format key should be the name of a command (without the initial backslash) that has a single mandatory argument which can be used to format the associated entry number.

```
\define@key{glslink}{format}{%
\def\@glsnumberformat{#1}}
```

The hyper key is a boolean key, it can either have the value true or false, and indicates whether or not to make a hyperlink to the relevant glossary entry. If hyper is false, an entry will still be made in the glossary, but the given text won't be a hyperlink.

```
\define@boolkey{glslink}{hyper}[true]{}
```

Syntax:

`\glslink[`⟨*options*⟩`]{`⟨*label*⟩`}{`⟨*text*⟩`}`

Display ⟨*text*⟩ in the document, and add the entry information for ⟨*label*⟩ into the relevant glossary. The optional argument should be a key value list using the `glslink` keys defined above.

There is also a starred version:

`\glslink*[`⟨*options*⟩`]{`⟨*label*⟩`}{`⟨*text*⟩`}`

which is equivalent to `\glslink[hyper=false,`⟨*options*⟩`]{`⟨*label*⟩`}{`⟨*text*⟩`}`

First determine whether or not we are using the starred version:

\glslink

```
\newcommand{\glslink}{%
\@ifstar\@sgls@link\@gls@link}
```

Define the starred version:

\@sgls@link

```
\newcommand*{\@sgls@link}[1][]{\@gls@link[hyper=false,#1]}
```

Define the un-starred version:

\@gls@link

```
\newcommand*{\@gls@link}[3][]{%
\glsdoifexists{#2}{%
\def\@glsnumberformat{glsnumberformat}%
\edef\@gls@counter{\csname glo@#2@counter\endcsname}%
\KV@glslink@hypertrue
\setkeys{glslink}{#1}%
\edef\theglsentrycounter{\expandafter\noexpand\csname the\@gls@counter\endcsname}%
\ifKV@glslink@hyper
\@glslink{glo:#2}{\glstextformat{#3}}%
\else
\glstextformat{#3}\relax
\fi
\protected@edef\@glo@sort{\csname glo@#2@sort\endcsname}%
\@gls@checkmkidxchars\@glo@sort
\protected@edef\@@glo@name{\csname glo@#2@name\endcsname}%
\@gls@checkmkidxchars\@@glo@name
\protected@edef\@glo@name{\string\glsnamefont{\@@glo@name}}%
\protected@edef\@glo@desc{\csname glo@#2@desc\endcsname}%
\@gls@checkmkidxchars\@glo@desc
\protected@edef\@glo@symbol{\csname glo@#2@symbol\endcsname}%
\@gls@checkmkidxchars\@glo@symbol
\@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat
\glossary[\csname glo@#2@type\endcsname]{%
\@glo@sort\@gls@actualchar
\string\glossaryentryfield{#2}{\@glo@name}{\@glo@desc
}{\@glo@symbol}\@gls@encapchar\@glo@numfmt}%
}}
```

Set the formatting information in the format required by `makeindex`:

\@set@glo@numformat

```
\def\@set@glo@numformat#1#2#3{%
\expandafter\@glo@check@mkidxrangechar#3\@nil
\protected@edef#1{\@glo@prefix setentrycounter{#2}%
\expandafter\string\csname\@glo@suffix\endcsname}%
\@gls@checkmkidxchars#1}
```

Check to see if the given string starts with a (or). If it does set `\@glo@prefix` to the starting character, and `\@glo@suffix` to the rest (or `glsnumberformat` if there is nothing else), otherwise set `\@glo@prefix` to nothing and `\@glo@suffix` to all of it.

\@glo@check@mkidxrangechar

```
\def\@glo@check@mkidxrangechar#1#2\@nil{%
\if#1(\relax
  \def\@glo@prefix{(}%
  \if\relax#2\relax
```

```
    \def\@glo@suffix{glsnumberformat}%
  \else
    \def\@glo@suffix{#2}%
  \fi
\else
  \if#1)\relax
    \def\@glo@prefix{)}%
    \if\relax#2\relax
      \def\@glo@suffix{glsnumberformat}%
    \else
      \def\@glo@suffix{#2}%
  \fi
  \else
    \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
  \fi
\fi}
```

Catch `makeindex` special characters:

`\@gls@checkmkidxchars`

```
\newcommand{\@gls@checkmkidxchars}[1]{%
\def\@gls@checkedmkidx{}%
\expandafter\@gls@checkquote#1\@nil""\null%
\expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
\def\@gls@checkedmkidx{}%
\expandafter\@gls@checkescquote#1\@nil\"\"\null%
\expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
\def\@gls@checkedmkidx{}%
\expandafter\@gls@checkescactual#1\@nil\?\?\null%
\expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
\def\@gls@checkedmkidx{}%
\expandafter\@gls@checkactual#1\@nil??\null%
\expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
\def\@gls@checkedmkidx{}%
\expandafter\@gls@checkbar#1\@nil||\null%
\expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
\def\@gls@checkedmkidx{}%
\expandafter\@gls@checkescbar#1\@nil\|\|\null%
\expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
\def\@gls@checkedmkidx{}%
\expandafter\@gls@checklevel#1\@nil!!\null%
\expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
}
```

Update the control sequence and strip trailing `\@nil`:

`\@gls@updatechecked`

```
\def\@gls@updatechecked#1\@nil#2{\def#2{#1}}
```

Replace " with "" since " is a makeindex special character.

```
\toksdef\@gls@tmpb=2
\def\@gls@checkquote#1"#2"#3\null{%
\@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
\toks@={#1}%
\ifx\null#2\null%
```

```
\ifx\null#3\null
 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
 \def\@@gls@checkquote{\relax}%
\else
 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
   \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
 \def\@@gls@checkquote{\@gls@checkquote#3\null}%
\fi
\else
\edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
  \@gls@quotechar\@gls@quotechar}%
\ifx\null#3\null
  \def\@@gls@checkquote{\@gls@checkquote#2""\null}%
\else
  \def\@@gls@checkquote{\@gls@checkquote#2"#3\null}%
\fi
\fi
\@@gls@checkquote}
```

Do the same for `\"`:

```
\def\@gls@checkescquote#1\"#2\"#3\null{%
\@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
\toks@={#1}%
\ifx\null#2\null%
 \ifx\null#3\null
  \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
  \def\@@gls@checkescquote{\relax}%
 \else
  \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
    \@gls@quotechar\string\"\@gls@quotechar
    \@gls@quotechar\string\"\@gls@quotechar}%
  \def\@@gls@checkescquote{\@gls@checkescquote#3\null}%
 \fi
\else
 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
   \@gls@quotechar\string\"\@gls@quotechar}%
\ifx\null#3\null
  \def\@@gls@checkescquote{\@gls@checkescquote#2\"\"\null}%
\else
  \def\@@gls@checkescquote{\@gls@checkescquote#2\"#3\null}%
\fi
\fi
\@@gls@checkescquote}
```

Similarly for `\?` (which is replaces @ as `makeindex`'s special character):

```
\def\@gls@checkescactual#1\?#2\?#3\null{%
\@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
\toks@={#1}%
\ifx\null#2\null%
 \ifx\null#3\null
  \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
  \def\@@gls@checkescactual{\relax}%
 \else
  \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
    \@gls@quotechar\string\"\@gls@actualchar
```

```
    \@gls@quotechar\string\"\@gls@actualchar}%
  \def\@@gls@checkescactual{\@gls@checkescactual#3\null}%
 \fi
\else
 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
   \@gls@quotechar\string\"\@gls@actualchar}%
 \ifx\null#3\null
  \def\@@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
 \else
  \def\@@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
 \fi
\fi
\@@gls@checkescactual}
```

Similarly for `\|`:

```
\def\@gls@checkescbar#1\|#2\|#3\null{%
\@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
\toks@={#1}%
\ifx\null#2\null%
 \ifx\null#3\null
  \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
  \def\@@gls@checkescbar{\relax}%
 \else
  \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
    \@gls@quotechar\string\"\@gls@encapchar
    \@gls@quotechar\string\"\@gls@encapchar}%
  \def\@@gls@checkescbar{\@gls@checkescbar#3\null}%
 \fi
\else
 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
   \@gls@quotechar\string\"\@gls@encapchar}%
 \ifx\null#3\null
  \def\@@gls@checkescbar{\@gls@checkescbar#2\|\|\null}%
 \else
  \def\@@gls@checkescbar{\@gls@checkescbar#2\|#3\null}%
 \fi
\fi
\@@gls@checkescbar}
```

Similarly for `\!`:

```
\def\@gls@checkesclevel#1\!#2\!#3\null{%
\@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
\toks@={#1}%
\ifx\null#2\null%
 \ifx\null#3\null
  \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
  \def\@@gls@checkesclevel{\relax}%
 \else
  \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
    \@gls@quotechar\string\"\@gls@levelchar
    \@gls@quotechar\string\"\@gls@levelchar}%
  \def\@@gls@checkesclevel{\@gls@checkesclevel#3\null}%
 \fi
\else
 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
```

```
   \@gls@quotechar\string\"\@gls@levelchar}%
 \ifx\null#3\null
  \def\@@gls@checkesclevel{\@gls@checkesclevel#2\!\!\null}%
 \else
  \def\@@gls@checkesclevel{\@gls@checkesclevel#2\!#3\null}%
 \fi
\fi
\@@gls@checkesclevel}
```

and for `|`:

```
\def\@gls@checkbar#1|#2|#3\null{%
\@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
\toks@={#1}%
\ifx\null#2\null%
 \ifx\null#3\null
  \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
  \def\@@gls@checkbar{\relax}%
 \else
  \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
    \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
  \def\@@gls@checkbar{\@gls@checkbar#3\null}%
 \fi
\else
 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
   \@gls@quotechar\@gls@encapchar}%
 \ifx\null#3\null
   \def\@@gls@checkbar{\@gls@checkbar#2||\null}%
 \else
   \def\@@gls@checkbar{\@gls@checkbar#2|#3\null}%
 \fi
\fi
\@@gls@checkbar}
```

and for `!`:

```
\def\@gls@checklevel#1!#2!#3\null{%
\@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
\toks@={#1}%
\ifx\null#2\null%
 \ifx\null#3\null
  \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
  \def\@@gls@checklevel{\relax}%
 \else
  \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
    \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
  \def\@@gls@checklevel{\@gls@checklevel#3\null}%
 \fi
\else
 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
   \@gls@quotechar\@gls@levelchar}%
 \ifx\null#3\null
   \def\@@gls@checklevel{\@gls@checklevel#2!!\null}%
 \else
   \def\@@gls@checklevel{\@gls@checklevel#2!#3\null}%
 \fi
\fi
```

```
\@@gls@checklevel}
```

and for `?`:

```
\def\@gls@checkactual#1?#2?#3\null{%
\@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
\toks@={#1}%
\ifx\null#2\null%
 \ifx\null#3\null
  \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
  \def\@@gls@checkactual{\relax}%
 \else
  \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
    \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
  \def\@@gls@checkactual{\@gls@checkactual#3\null}%
 \fi
\else
 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
   \@gls@quotechar\@gls@actualchar}%
 \ifx\null#3\null
   \def\@@gls@checkactual{\@gls@checkactual#2??\null}%
 \else
   \def\@@gls@checkactual{\@gls@checkactual#2?#3\null}%
 \fi
\fi
\@@gls@checkactual}
```

If `\hyperlink` is not defined, `\@glslink` and `\@glstarget` ignore their first argument, and just do the second argument, otherwise they are equivalent to `\hyperlink` and `\hypertarget`.

```
\@ifundefined{hyperlink}{%
\gdef\@glslink#1#2{#2}\gdef\@glstarget#1#2{#2}%
}{\gdef\@glslink#1#2{\hyperlink{#1}{#2}}%
\gdef\@glstarget#1#2{\hypertarget{#1}{#2}}}
```

Glossary hyperlinks can be disabled using `\glsdisablehyper` (effect can be localised):

`\glsdisablehyper`

```
\newcommand{\glsdisablehyper}{%
\renewcommand*\@glslink[2]{##2}%
\renewcommand*\@glstarget[2]{##2}}
```

Glossary hyperlinks can be enabled using `\glsenablehyper` (effect can be localised):

`\glsenablehyper`

```
\newcommand{\glsenablehyper}{%
\renewcommand*\@glslink[2]{\hyperlink{##1}{##2}}%
\renewcommand*\@glstarget[2]{\hypertarget{##1}{##2}}}
```

Syntax:

`\gls[`⟨*options*⟩`]{`⟨*label*⟩`}[`⟨*insert text*⟩`]`

Link to glossary entry using singular form. The link text is taken from the value of the text or first keys used when the entry was defined.

The first optional argument is a key-value list, the same as \glslink, the mandatory argument is the entry label. After the mandatory argument, there is another optional argument to insert extra text in the link text (the location of the inserted text is governed by \glsdisplay and \glsdisplayfirst). As with \glslink there is a starred version which is the same as the unstarred version but with the hyper key set to false. (Additional options can also be specified in the first optional argument.)

First determine if we are using the starred form:

\gls

```
\newcommand*{\gls}{\@ifstar\@sgls\@gls}
```

Define the starred form:

\@sgls

```
\newcommand*{\@sgls}[1][]{\@gls[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

\@gls

```
\newcommand*{\@gls}[2][]{%
\@ifnextchar[{\@gls@{#1}{#2}}{\@gls@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@gls@#1#2[#3]{%
\glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
\ifglsused{#2}{\protected@edef\@glo@text{%
\csname gls@\@glo@type @display\endcsname
{\glsentrytext{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}}{%
\protected@edef\@glo@text{%
\csname gls@\@glo@type @displayfirst\endcsname
{\glsentryfirst{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}}%
```

Call \@gls@link

```
\@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used, and add a space if appropriate

```
\glsunset{#2}}%
\xspace}
```

\Gls behaves like \gls, but the first letter of the link text is converted to uppercase (note that if the first letter has an accent, the accented letter will need to be grouped when you define the entry.) It is mainly intended for terms that start a sentence:

\Gls

```
\newcommand*{\Gls}{\@ifstar\@sGls\@Gls}
```

Define the starred form:

```
\newcommand*{\@sGls}[1][]{\@Gls[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\@Gls}[2][]{%
\@ifnextchar[{\@Gls@{#1}{#2}}{\@Gls@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@Gls@#1#2[#3]{%
\glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in `\@glo@text`)

```
\ifglsused{#2}{\protected@edef\@glo@text{%
\csname gls@\@glo@type @display\endcsname
{\glsentrytext{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}}{%
\protected@edef\@glo@text{%
\csname gls@\@glo@type @displayfirst\endcsname
{\glsentryfirst{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}}%
```

Call `\@gls@link`

```
\@gls@link[#1]{#2}{\expandafter\MakeUppercase\@glo@text}%
```

Indicate that this entry has now been used, and add a space if appropriate

```
\glsunset{#2}}%
\xspace}
```

`\GLS` behaves like `\gls`, but the link text is converted to uppercase:

`\GLS`

```
\newcommand*{\GLS}{\@ifstar\@sGLS\@GLS}
```

Define the starred form:

```
\newcommand*{\@sGLS}[1][]{\@GLS[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\@GLS}[2][]{%
\@ifnextchar[{\@GLS@{#1}{#2}}{\@GLS@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@GLS@#1#2[#3]{%
\glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in `\@glo@text`)

```
\ifglsused{#2}{\protected@edef\@glo@text{%
\csname gls@\@glo@type @display\endcsname
{\glsentrytext{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}}{%
\protected@edef\@glo@text{%
\csname gls@\@glo@type @displayfirst\endcsname
{\glsentryfirst{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}}%
```

Call `\@gls@link`

```
\@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
```

Indicate that this entry has now been used, and add a space if appropriate

```
\glsunset{#2}}%
\xspace}
```

`\glspl` behaves in the same way as `\gls` except it uses the plural form.

\glspl

```
\newcommand*{\glspl}{\@ifstar\@sglspl\@glspl}
```

Define the starred form:

```
\newcommand*{\@sglspl}[1][]{\@glspl[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\@glspl}[2][]{%
\@ifnextchar[{\@glspl@{#1}{#2}}{\@glspl@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@glspl@#1#2[#3]{%
\glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in `\@glo@text`)

```
\ifglsused{#2}{\protected@edef\@glo@text{%
\csname gls@\@glo@type @display\endcsname
{\glsentryplural{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}}{%
\protected@edef\@glo@text{%
\csname gls@\@glo@type @displayfirst\endcsname
{\glsentryfirstplural{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}}%
```

Call `\@gls@link`

```
\@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used, and add a space if appropriate

```
\glsunset{#2}}%
\xspace}
```

`\Glspl` behaves in the same way as `\glspl`, except that the first letter of the link text is converted to uppercase (as with `\Gls`, if the first letter has an accent, it will need to be grouped.)

\Glspl

```
\newcommand*{\Glspl}{\@ifstar\@sGlspl\@Glspl}
```

Define the starred form:

```
\newcommand*{\@sGlspl}[1][]{\@Glspl[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\@Glspl}[2][]{%
\@ifnextchar[{\@Glspl@{#1}{#2}}{\@Glspl@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@Glspl@#1#2[#3]{%
\glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in `\@glo@text`)

```
\ifglsused{#2}{\protected@edef\@glo@text{%
\csname gls@\@glo@type @display\endcsname
{\glsentryplural{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}}{%
\protected@edef\@glo@text{%
\csname gls@\@glo@type @displayfirst\endcsname
{\glsentryfirstplural{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}}%
```

Call \@gls@link

```
\@gls@link[#1]{#2}{\expandafter\MakeUppercase\@glo@text}%
```

Indicate that this entry has now been used, and add a space if appropriate

```
\glsunset{#2}}%
\xspace}
```

\GLSpl behaves like \glspl except that all the link text is converted to uppercase.

\GLSpl

```
\newcommand*{\GLSpl}{\@ifstar\@sGLSpl\@GLSpl}
```

Define the starred form:

```
\newcommand*{\@sGLSpl}[1][]{\@GLSpl[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\@GLSpl}[2][]{%
\@ifnextchar[{\@GLSpl@{#1}{#2}}{\@GLSpl@{#1}{#2}[]}}
```

Read in the final optional argument:

```
\def\@GLSpl@#1#2[#3]{%
\glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
\ifglsused{#2}{\protected@edef\@glo@text{%
\csname gls@\@glo@type @display\endcsname
{\glsentryplural{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}}{%
\protected@edef\@glo@text{%
\csname gls@\@glo@type @displayfirst\endcsname
{\glsentryfirstplural{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}}%
```

Call \@gls@link

```
\@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
```

Indicate that this entry has now been used, and add a space if appropriate

```
\glsunset{#2}}%
\xspace}
```

4.9.2 Displaying entry details without adding information to the glossary

These commands merely display entry information without adding entries in the associated file or having hyperlinks.

Get the entry name (as specified by the name key when the entry was defined.) The argument is the label associated with the entry. Note that unless you used name=false in the sanitize package option you may get unexpected results if the name key contained any commands.

\glsentryname

```
\newcommand*{\glsentryname}[1]{\csname glo@#1@name\endcsname}
```

\Glsentryname

```
\newcommand*{\Glsentryname}[1]{\expandafter
\MakeUppercase\csname glo@#1@name\endcsname}
```

Get the entry description (as specified by the description when the entry was defined.) The argument is the label associated with the entry. Note that unless you used `description=false` in the sanitize package option you may get unexpected results if the description key contained any commands.

\glsentrydesc

```
628 \newcommand*{\glsentrydesc}[1]{\csname glo@#1@desc\endcsname}
```

\Glsentrydesc

```
629 \newcommand*{\Glsentrydesc}[1]{\expandafter
630 \MakeUppercase\csname glo@#1@desc\endcsname}
```

Get the entry text, as specified by the text key when the entry was defined. The argument is the label associated with the entry:

\glsentrytext

```
631 \newcommand*{\glsentrytext}[1]{\csname glo@#1@text\endcsname}
```

\Glsentrytext

```
632 \newcommand*{\Glsentrytext}[1]{\expandafter
633 \MakeUppercase\csname glo@#1@text\endcsname}
```

Get the plural form:

\glsentryplural

```
634 \newcommand*{\glsentryplural}[1]{\csname glo@#1@plural\endcsname}
```

\Glsentryplural

```
635 \newcommand*{\Glsentryplural}[1]{\expandafter
636 \MakeUppercase\csname glo@#1@plural\endcsname}
```

Get the symbol associated with this entry. The argument is the label associated with the entry. Note that unless you used `symbol=false` in the sanitize package option you may get unexpected results if the symbol key contained any commands.

\glsentrysymbol

```
637 \newcommand*{\glsentrysymbol}[1]{\csname glo@#1@symbol\endcsname}
```

\Glsentrysymbol

```
638 \newcommand*{\Glsentrysymbol}[1]{\expandafter
639 \MakeUppercase\csname glo@#1@symbol\endcsname}
```

Get the entry text to be used when the entry is first used in the document (as specified by the first key when the entry was defined.)

\glsentryfirst

```
640 \newcommand*{\glsentryfirst}[1]{\csname glo@#1@first\endcsname}
```

\Glsentryfirst

```
641 \newcommand*{\Glsentryfirst}[1]{\expandafter
642 \MakeUppercase\csname glo@#1@first\endcsname}
```

Get the plural form (as specified by the firstplural key when the entry was defined.)

\glsentryfirstplural

```
\newcommand*{\glsentryfirstplural}[1]{%
\csname glo@#1@firstpl\endcsname}
```

\Glsentryfirstplural

```
\newcommand*{\Glsentryfirstplural}[1]{%
\expandafter\MakeUppercase\csname glo@#1@firstpl\endcsname}
```

Display the glossary type with which this entry is associated (as specified by the type key used when the entry was defined)

\glsentrytype

```
\newcommand*{\glsentrytype}[1]{\csname glo@#1@type\endcsname}
```

Display the sort text used for this entry. Note that the sort key is sanitize, so unexpected results may occur if the sort key contained commands.

\glsentrysort

```
\newcommand*{\glsentrysort}[1]{\csname glo@#1@sort\endcsname}
```

4.10 Adding an entry to the glossary without generating text

The following keys are provided for \glsadd and \glsaddall:

```
\define@key{glossadd}{counter}{\def\@glo@counter{#1}}
\define@key{glossadd}{format}{\def\@glo@format{#1}}
```

This key is only used by \glsaddall:

```
\define@key{glossadd}{types}{\def\@glo@type{#1}}
```

\glsadd[⟨*options*⟩]{⟨*label*⟩}

Add a term to the glossary without generating any link text. The optional argument indicates which counter to use, and how to format it (using a key-value list) the second argument is the entry label. Note that ⟨*options*⟩ only has two keys: counter and format (the types key will be ignored).

\glsadd

```
\newcommand*{\glsadd}[2][]{%
\glsdoifexists{#2}{%
\def\@glo@format{glsnumberformat}%
\edef\@glo@counter{\csname glo@#2@counter\endcsname}%
\setkeys{glossadd}{#1}%
\edef\theglsentrycounter{\expandafter\noexpand\csname the\@glo@counter\endcsname}%
\protected@edef\@glo@sort{\csname glo@#2@sort\endcsname}%
\@gls@checkmkidxchars\@glo@sort
\protected@edef\@@glo@name{\csname glo@#2@name\endcsname}%
\@gls@checkmkidxchars\@@glo@name
\protected@edef\@glo@name{\string\glsnamefont{\@@glo@name}}%
\protected@edef\@glo@desc{\csname glo@#2@desc\endcsname}%
\@gls@checkmkidxchars\@glo@desc
\protected@edef\@glo@symbol{\csname glo@#2@symbol\endcsname}%
\@gls@checkmkidxchars\@glo@symbol
\@set@glo@numformat\@glo@numfmt\@glo@counter\@glo@format
```

```
\glossary[\csname glo@#2@type\endcsname]{%
\@glo@sort\@gls@actualchar\string\glossaryentryfield
{#2}{\@glo@name}{\@glo@desc}{\@glo@symbol}\@gls@encapchar
\@glo@numfmt}%
}}
```

\glsaddall[⟨*glossary list*⟩]

Add all terms defined for the listed glossaries (without displaying any text.) If types key is omitted, apply to all glossary types.

\glsaddall

```
\newcommand*{\glsaddall}[1][]{%
\def\@glo@type{\@glo@types}%
\setkeys{glossadd}{#1}%
\forallglsentries[\@glo@type]{\@glo@entry}{%
\glsadd[#1]{\@glo@entry}}%
}
```

4.11 Creating associated files

The \writeist command creates the associated customized ist makeindex style file. While defining this command, some characters have their catcodes temporarily changed to ensure they get written to the ist file correctly. The makeindex actual character (usually @) is redefined to be a ?, to allow internal commands to be written to the glossary file output file.

The special characters are stored in \@gls@actualchar, \@gls@encapchar, \@glsl@levelchar and \@gls@quotechar to make them easier to use later, but don't change these values, because the characters are encoded in the command definitions that are used to escape the special characters (which means that the user no longer needs to worry about makeindex special characters.)

The symbols and numbers label for group headings are hardwired into the ist file as glssymbols and glsnumbers, the group titles can be translated (so that \glssymbolsgroupname replaces glssymbols and \glsnumbersgroupname replaces glsnumbers) using the command \glsgetgrouptitle which is defined in glossary-hypernav. This is done to prevent any problem characters in \glssymbolsgroupname and \glsnumbersgroupname from breaking hyperlinks.

Some of these lines are too long to fit on the page, but as I have temporarily disabled the comment character, I can't split the lines. If you want to see the code in full, have a look at glossaries.sty.

\writeist

```
\newwrite\istfile
\bgroup
\catcode`\%12\relax
\catcode`\"12\relax
\catcode`\|12\relax
\catcode`\!12\relax
\catcode`\?12\relax
\gdef\@gls@actualchar{?}
\gdef\@gls@encapchar{|}
\gdef\@gls@levelchar{!}
```

```
\gdef\@gls@quotechar{"}
\gdef\writeist{\relax
\protected@write\@auxout{}{\string\@istfilename{\istfilename}}
\openout\istfile=\istfilename
\write\istfile{% makeindex style file created by the glossaries package}
\write\istfile{% for document '\jobname' on \the\year-\the\month-\the\day}
\write\istfile{actual '\@gls@actualchar'}
\write\istfile{encap '\@gls@encapchar'}
\write\istfile{level '\@gls@levelchar'}
\write\istfile{quote '\@gls@quotechar'}
\write\istfile{keyword "\string\\glossaryentry"}
\write\istfile{preamble "\string\\glossarysection[\string\\glossarytoctitle]{\string\\glossary
\write\istfile{postamble "\string\n\string\\end{theglossary}\string\n\string\\glossarypostambl
\write\istfile{group_skip "\string\\glsgroupskip\string\n"}
\write\istfile{item_0 "\string\n"}
\write\istfile{delim_0 "\{\string\\glossaryentrynumbers\{\string\\relax "}
\write\istfile{delim_t "\}\}"}
\write\istfile{delim_n "\string\\delimN "}
\write\istfile{delim_r "\string\\delimR "}
\write\istfile{headings_flag 1}
\write\istfile{heading_prefix "\string\\glsgroupheading\{"}
\write\istfile{heading_suffix "\}"}
\write\istfile{symhead_positive "glssymbols"}
\write\istfile{numhead_positive "glsnumbers"}
\write\istfile{page_compositor "\glscompositor"}
\noist}
\egroup
```

The command `\noist` will suppress the creation of the `ist` file (it simply redefines `\writeist` to do nothing.) Obviously you need to use this command before `\writeist` to have any effect. Since the `ist` file should only be created once, `\noist` is called at the end of `\writeist`.

`\noist`

```
\newcommand{\noist}{\let\writeist\relax}
```

`\@makeglossary` is an internal command that takes an argument indicating the glossary type. This command will create the glossary file required by `makeindex` for the given glossary type, using the extension supplied by the ⟨*out-ext*⟩ parameter used in `\newglossary` (and it will also activate the `\glossary` command, and create the customized `ist` `makeindex` style file.)

Note that you can't use `\@makeglossary` for only some of the defined glossaries. You either need to have a `\makeglossary` for all glossaries or none (otherwise you will end up with a situation where TEX is trying to write to a non-existant file.) The relevant glossary must be defined prior to using `\@makeglossary`.

`\@makeglossary`

```
\newcommand*{\@makeglossary}[1]{%
\ifglossaryexists{#1}{%
\edef\glo@out{\csname @glotype@#1@out\endcsname}%
\expandafter\newwrite\csname glo@#1@file\endcsname
\edef\@glo@file{\csname glo@#1@file\endcsname}%
\immediate\openout\@glo@file=\jobname.\glo@out
\@gls@renewglossary
```

```
\PackageInfo{glossaries}{Writing glossary file \jobname.\glo@out}
\writeist
}{\PackageError{glossaries}{%
Glossary type `#1' not defined}{New glossaries must be defined before
using \string\makeglossary}}}
```

`\makeglossaries` will use `\@makeglossary` for each glossary type that has been defined. New glossaries need to be defined before using `\makeglossary`, so have `\makeglossaries` redefine `\newglossary` to prevent it being used afterwards.

`\makeglossaries`

```
\newcommand*{\makeglossaries}{%
\@for\@glo@type:=\@glo@types\do{%
\ifthenelse{\equal{\@glo@type}{}}{}{%
\@makeglossary{\@glo@type}}}%
\renewcommand*\newglossary[4][]{%
\PackageError{glossaries}{New glossaries
must be created before \string\makeglossaries}{You need
to move \string\makeglossaries\space after all your
\string\newglossary\space commands}}%
\let\@makeglossary\empty
\let\makeglossary\empty}
```

The `\makeglossary` command is redefined to be identical to `\makeglossaries`. (This is done to reinforce the message that you must either use `\@makeglossary` for all the glossaries or for none of them.)

`\makeglossary`

```
\let\makeglossary\makeglossaries
```

4.12 Writing information to associated files

The `\glossary` command is redefined so that it takes an optional argument ⟨*type*⟩ to specify the glossary type (use `\glsdefaulttype` glossary by default). This shouldn't be used at user level as `\glslink` sets the correct format. The associated number should be stored in `\theglsentrycounter` before using `\glossary`.

`\glossary`

```
\renewcommand*{\glossary}[1][\glsdefaulttype]{%
\@glossary[#1]}
```

Define internal `\@glossary` to ignore its argument. This gets redefined in `\@makeglossary`.

`\@glossary`

```
\def\@glossary[#1]{\@bsphack\begingroup\@sanitize\@index}
```

This is a convenience command to set `\@glossary`. It is used by `\@makeglossary` and then redefined to do nothing, as it only needs to be done once.

`@gls@renewglossary`

```
\newcommand{\@gls@renewglossary}{%
\gdef\@glossary[##1]{\@bsphack\begingroup\@wrglossary{##1}}%
\let\@gls@renewglossary\@empty
}
```

The \@wrglossary command is redefined to have two arguments. The first argument is the glossary type, the second argument is the glossary entry (the format of which is set in \glslink.)

\@wrglossary

```
\renewcommand*{\@wrglossary}[2]{%
\expandafter\protected@write\csname glo@#1@file\endcsname{}{%
\string\glossaryentry{#2}{\theglsentrycounter}}\endgroup\@esphack}
```

4.13 Displaying the glossary

An individual glossary is displayed in the text using \printglossary[⟨*key-val list*⟩]. If the type key is omitted, the default glossary is displayed. The optional argument can be used to specify an alternative glossary, and can also be used to set the style, title and entry in the table of contents. Available keys are defined below.

\printglossary

```
\newcommand*{\printglossary}[1][type=\glsdefaulttype]{%
\def\@glo@type{\glsdefaulttype}%
\def\glossarytitle{\csname @glotype@\@glo@type @title\endcsname}%
\def\glossarytoctitle{\glossarytitle}%
\def\@glossarystyle{}%
\setkeys{printgloss}{#1}%
\bgroup
\@glossarystyle
\makeatletter
\@input@{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
\egroup
}
```

The \printglossaries command will do \printglossary for each glossary type that has been defined. It is better to use \printglossaries rather than individual \printglossary commands to ensure that you don't forget any new glossaries you may have created. It also makes it easier to chop and change the value of the acronym package option. However, if you want to list the glossaries in a different order, or if you want to set the title or table of contents entry, or if you want to use different glossary styles for each glossary, you will need to use \printglossary explicitly for each glossary type.

\printglossaries

```
\newcommand*{\printglossaries}{%
\forallglossaries{\@@glo@type}{\printglossary[type=\@@glo@type]}}
```

The keys that can be used in the optional argument to \printglossary are as follows: The type key sets the glossary type.

```
\define@key{printgloss}{type}{\def\@glo@type{#1}}
```

The title key sets the title used in the glossary section header. This overrides the title used in \newglossary.

```
\define@key{printgloss}{title}{\def\glossarytitle{#1}}
```

The toctitle sets the text used for the relevant entry in the table of contents.

```
\define@key{printgloss}{toctitle}{\def\glossarytoctitle{#1}}
```

The style key sets the glossary style (but only for the given glossary.)

```
\define@key{printgloss}{style}{%
\@ifundefined{@glsstyle@#1}{\PackageError{glossaries}{Glossary
style '#1' undefined}{}}{%
\def\@glossarystyle{\csname @glsstyle@#1\endcsname}}}
```

theglossary

If the theglossary environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```
\@ifundefined{theglossary}{%
\newenvironment{theglossary}{}{}}{%
\PackageWarning{glossaries}{overriding 'theglossary' environment}%
\renewenvironment{theglossary}{}{}}
```

The glossary header is given by \glossaryheader. This forms part of the glossary style, and must indicate what should appear immediately after the start of the theglossary environment. (For example, if the glossary uses a tabular-like environment, it may be used to set the header row.) Note that if you don't want a header row, the glossary style must redefine \glossaryheader to do nothing.

\glossaryheader

```
\newcommand*{\glossaryheader}{}
```

\glossaryentryfield{⟨*label*⟩}{⟨*name*⟩}{⟨*description*⟩}{⟨*symbol*⟩}{⟨*page-list*⟩}

This command governs how each entry row should be formatted in the glossary. Glossary styles need to redefine this command. Most of the predefined styles ignore ⟨*symbol*⟩.

\glossaryentryfield

```
\newcommand*{\glossaryentryfield}[5]{%
\@glstarget{glo:#1}{#2} #4 #3. #5\par}
```

Within each glossary, the entries form 28 distinct groups which are determined by the first character of the sort key. There will be a maximum of 28 groups: symbols, numbers, and the 26 alphabetical groups A, ..., Z. The command \glsgroupskip specifies what to do between glossary groups. Glossary styles must redefine this command. (Note that \glsgroupskip only occurs between groups, not at the start or end of the glossary.)

\glsgroupskip

```
\newcommand*{\glsgroupskip}{}
```

Each of the 28 glossary groups described above is preceded by a group heading. This is formatted by the command \glsgroupheading which takes one argument which is the *label* assigned to that group (not the title). The corresponding labels are: glssymbols, glsnumbers, A, ..., Z. Glossary styles must redefined this command. (In between groups, \glsgroupheading comes immediately after \glsgroupskip.)

\glsgroupheading

```
\newcommand*{\glsgroupheading}[1]{}
```

It is possible to "trick" `makeindex` into treating entries as though they belong to the same group, even if the terms don't start with the same letter, by modifying the sort key. For example, all entries belonging to one group could be defined so that the sort key starts with an `a`, while entries belonging to another group could be defined so that the sort key starts with a `b`, and so on. If you want each group to have a heading, you would then need to modify the translation control sequences `\glsgetgrouptitle` and `\glsgetgrouplabel` so that the label is translated into the required title (and vice-versa.)

`\glsgetgrouptitle{`⟨*label*⟩`}`

This command produces the title for the glossary group whose label is given by ⟨*label*⟩. By default, the group labelled `glssymbols` produces `\glssymbolsgroupname`, the group labelled `glsnumbers` produces `\glsnumbersgroupname` and all the other groups simply produce their label. As mentioned above, the group labels are: `glssymbols`, `glsnumbers`, `A`, ..., `Z`. If you want to redefine the group titles, you will need to redefine this command.

`\glsgetgrouptitle`

```
781 \newcommand*{\glsgetgrouptitle}[1]{%
782 \@ifundefined{#1groupname}{#1}{\csname #1groupname\endcsname}}
```

`\glsgetgrouplabel{`⟨*title*⟩`}`

This command does the reverse to the previous command. The argument is the group title, and it produces the group label. Note that if you redefine `\glsgetgrouptitle`, you will also need to redefine `\glsgetgrouplabel`.

`\glsgetgrouplabel`

```
783 \newcommand*{\glsgetgrouplabel}[1]{%
784 \ifthenelse{\equals{#1}{\glssymbolsgroupname}}{glssymbols}{%
785 \ifthenelse{\equals{#1}{\glsnumbersgroupname}}{glsnumbers}{#1}}}
```

The command `\setentrycounter` sets the entry's associated counter (required by `\glshypernumber` etc.) `\glslink` and `\glsadd` encode the `\glossary` argument so that the relevant counter is set prior to the formatting command.

`\setentrycounter`

```
786 \newcommand*{\setentrycounter}[1]{\def\glsentrycounter{#1}}
```

The current glossary style can be set using `\glossarystyle{`⟨*style*⟩`}`.

`\glossarystyle`

```
787 \newcommand*{\glossarystyle}[1]{%
788 \@ifundefined{@glsstyle@#1}{\PackageError{glossaries}{Glossary
789 style '#1' undefined}{}}{%
790 \csname @glsstyle@#1\endcsname}}
```

New glossary styles can be defined using:

`\newglossarystyle{`⟨*name*⟩`}{`⟨*definition*⟩`}`

The ⟨*definition*⟩ argument should redefine theglossary, `\glossaryheader`, `\glsgroupheading`,

\glossaryentryfield and \glsgroupskip (see subsection 4.15 for the definitions of predefined styles.) Glossary styles should not redefine \glossarypreamble and \glossarypostamble, as the user should be able to switch between styles without affecting the pre- and postambles.

\newglossarystyle

```
\newcommand*{\newglossarystyle}[2]{%
\@ifundefined{@glsstyle@#1}{%
\expandafter\def\csname @glsstyle@#1\endcsname{#2}}{%
\PackageError{glossaries}{Glossary style `#1' is already defined}{}}}
```

Glossary entries are encoded so that the second argument to \glossaryentryfield is always specified as \glsnamefont{⟨*name*⟩}. This allows the user to change the font used to display the name term without having to redefine \glossaryentryfield. The default uses the surrounding font, so in the list type styles (which place the name in the optional argument to \item) the name will appear in bold.

\glsnamefont

```
\newcommand*{\glsnamefont}[1]{#1}
```

Each glossary entry has an associated number list (usually page numbers) that indicate where in the document the entry has been used. The format for these number lists can be changed using the format key in commands like \glslink. The default format is given by \glshypernumber. This takes a single argument which may be a single number, a number range or a number list. The number ranges are delimited with \delimR, the number lists are delimited with \delimN.

If the document doesn't have hyperlinks, the numbers can be displayed just as they are, but if the document supports hyperlinks, the numbers should link to the relevant location. This means extracting the individual numbers from the list or ranges. The hyperref package does this with the \hyperpage command, but this is encoded for comma and dash delimiters and only for the page counter, but this code needs to be more general. So I have adapted the code used in the hyperref package.

\glshypernumber

```
\@ifundefined{hyperlink}{%
\def\glshypernumber#1{#1}}{%
\def\glshypernumber#1{%
\@delimR#1\delimR\delimR\\}}
```

\@delimR displays a range of numbers for the counter whose name is given by \@gls@counter (which must be set prior to using \glshypernumber.)

\@delimR

```
\def\@delimR#1\delimR #2\delimR #3\\{%
\ifx\\#2\\%
  \@delimN{#1}%
\else
  \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
\fi}
```

\@delimN displays a list of individual numbers, instead of a range:

\@delimN

```
\def\@delimN#1{\@@delimN#1\delimN \delimN\\}
\def\@@delimN#1\delimN #2\delimN#3\\{%
\ifx\\#3\\%
  \@gls@numberlink{#1}%
\else
  \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
\fi
}
```

The following code is modified from hyperref's `\HyInd@pagelink` where the name of the counter being used is given by `\@gls@counter`.

```
\def\@gls@numberlink#1{%
\begingroup
 \toks@={}%
 \@gls@removespaces#1 \@nil
\endgroup}
```

```
\def\@gls@removespaces#1 #2\@nil{%
 \toks@=\expandafter{\the\toks@#1}%
 \ifx\\#2\\%
   \edef\x{\the\toks@}%
   \ifx\x\empty
   \else
     \hyperlink{\glsentrycounter.\the\toks@}{\the\toks@}%
   \fi
 \else
   \@gls@ReturnAfterFi{%
     \@gls@removespaces#2\@nil
   }%
 \fi
}
\long\def\@gls@ReturnAfterFi#1\fi{\fi#1}
```

The following commands will switch to the appropriate font, and create a hyperlink, if hyperlinks are supported. If hyperlinks are not supported, they will just display their argument in the appropriate font.

\hyperrm

```
\newcommand*{\hyperrm}[1]{\textrm{\glshypernumber{#1}}}
```

\hypersf

```
\newcommand*{\hypersf}[1]{\textsf{\glshypernumber{#1}}}
```

\hypertt

```
\newcommand*{\hypertt}[1]{\texttt{\glshypernumber{#1}}}
```

\hyperbf

```
\newcommand*{\hyperbf}[1]{\textbf{\glshypernumber{#1}}}
```

\hypermd

```
\newcommand*{\hypermd}[1]{\textmd{\glshypernumber{#1}}}
```

\hyperit

```
\newcommand*{\hyperit}[1]{\textit{\glshypernumber{#1}}}
```

\hypersl

```
\newcommand*{\hypersl}[1]{\textsl{\glshypernumber{#1}}}
```

\hyperup

```
\newcommand*{\hyperup}[1]{\textup{\glshypernumber{#1}}}
```

\hypersc

```
\newcommand*{\hypersc}[1]{\textsc{\glshypernumber{#1}}}
```

\hyperemph

```
\newcommand*{\hyperemph}[1]{\emph{\glshypernumber{#1}}}
```

4.14 Acronyms

If the acronym package option is used, a new glossary called `acronym` is created

```
\ifglsacronym
\newglossary[alg]{acronym}{acr}{acn}{\acronymname}
```

and `\acronymtype` is set to the name of this new glossary.

```
\renewcommand{\acronymtype}{acronym}
```

In the event that the user redefines `\glsdisplay` and `\glsdisplayfirst`, the relevant commands for the new acronym glossary are set to match the format given by `\newacronym`. If you redefine `\newacronym` you may need to set these to something else.

```
\defglsdisplay[acronym]{#1#4}\defglsdisplayfirst[acronym]{#1#4}
\fi
```

`\newacronym[`⟨*key-val list*⟩`]{`⟨*label*⟩`}{`⟨*abbrev*⟩`}{`⟨*long*⟩`}`

This is a quick way of defining acronyms, all it does is call `\newglossaryentry` with the appropriate values. It sets the glossary type to `\acronymtype` which will be `acronym` if the package option acronym has been used, otherwise it will be the default glossary. Since `\newacronym` merely calls `\newglossaryentry`, the acronym is treated like any other glossary entry.

If you prefer a different format, you can redefine `\newacronym` as required. The optional argument can be used to override any of the settings.

\newacronym

```
\newcommand{\newacronym}[4][]{%
\newglossaryentry{#2}{type=\acronymtype,%
name={#3},description={#4},text={#3},%
first={#4 (#3)},plural={#3s},firstplural={#4s (#3s)},#1}}
```

New acronyms can only be defined in the preamble:

```
\@onlypreamble{\newacronym}
```

4.15 Predefined Styles

The glossaries bundle comes with some predefined glossary styles which are defined in the following packages:

```
\RequirePackage{glossary-hypernav}
\RequirePackage{glossary-list}
```

```
\RequirePackage{glossary-long}
\RequirePackage{glossary-super}
```

The default glossary style is set according to the style package option, but can be overridden by `\glossarystyle`.

```
\glossarystyle{\@glossary@default@style}
```

4.15.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
\ProvidesPackage{glossary-hypernav}[2007/07/04 v1.01 (NLCT)]
```

The commands defined in this package are provided to help navigate around the groups within a glossary (see subsection 4.13.) `\printglossary` (and `\printglossaries`) set `\@glo@type` to the label of the current glossary. This is used to create a unique hypertarget in the event of multiple glossaries.

`\glsnavhyperlink[`⟨*type*⟩`]{`⟨*label*⟩`}{`⟨*text*⟩`}`

This command makes ⟨*text*⟩ a hyperlink to the glossary group whose label is given by ⟨*label*⟩ for the glossary given by ⟨*type*⟩.

`\glsnavhyperlink`

```
\@ifundefined{hyperlink}{%
\newcommand*{\glsnavhyperlink}[3][]{#3}}{%
\newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
\edef\gls@grplabel{#2}\edef\@gls@grptitle{#3}%
\hyperlink{glsn:#1@#2}{#3}}}
```

`\glsnavhypertarget[`⟨*type*⟩`]{`⟨*label*⟩`}{`⟨*text*⟩`}`

This command makes ⟨*text*⟩ a hypertarget for the glossary group whose label is given by ⟨*label*⟩ in the glossary given by ⟨*type*⟩.

`\glsnavhypertarget`

```
\@ifundefined{hypertarget}{%
\newcommand*{\glsnavhypertarget}[3][]{#3}}{%
\newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
\hypertarget{glsn:#1@#2}{#3}}}
```

The `\glsnavigation` command displays a simple glossary group navigation. The symbol and number elements are defined separately, so that they can be suppressed if need be. Note that this command will produce a link to all 28 groups, but some groups may not be defined if there are groups that do not contain any terms, in which case you will get an undefined hyperlink warning. Now for the whole navigation bit:

`\glsnavigation`

```
\newcommand*{\glsnavigation}{%
\glssymbolnav
\glsnavhyperlink{A}{\glsgetgrouptitle{A}} \textbar\
\glsnavhyperlink{B}{\glsgetgrouptitle{B}} \textbar\
\glsnavhyperlink{C}{\glsgetgrouptitle{C}} \textbar\
```

```
\glsnavhyperlink{D}{\glsgetgrouptitle{D}} \textbar\
\glsnavhyperlink{E}{\glsgetgrouptitle{E}} \textbar\
\glsnavhyperlink{F}{\glsgetgrouptitle{F}} \textbar\
\glsnavhyperlink{G}{\glsgetgrouptitle{G}} \textbar\
\glsnavhyperlink{H}{\glsgetgrouptitle{H}} \textbar\
\glsnavhyperlink{I}{\glsgetgrouptitle{I}} \textbar\
\glsnavhyperlink{J}{\glsgetgrouptitle{J}} \textbar\
\glsnavhyperlink{K}{\glsgetgrouptitle{K}} \textbar\
\glsnavhyperlink{L}{\glsgetgrouptitle{L}} \textbar\
\glsnavhyperlink{M}{\glsgetgrouptitle{M}} \textbar\
\glsnavhyperlink{N}{\glsgetgrouptitle{N}} \textbar\
\glsnavhyperlink{O}{\glsgetgrouptitle{O}} \textbar\
\glsnavhyperlink{P}{\glsgetgrouptitle{P}} \textbar\
\glsnavhyperlink{Q}{\glsgetgrouptitle{Q}} \textbar\
\glsnavhyperlink{R}{\glsgetgrouptitle{R}} \textbar\
\glsnavhyperlink{S}{\glsgetgrouptitle{S}} \textbar\
\glsnavhyperlink{T}{\glsgetgrouptitle{T}} \textbar\
\glsnavhyperlink{U}{\glsgetgrouptitle{U}} \textbar\
\glsnavhyperlink{V}{\glsgetgrouptitle{V}} \textbar\
\glsnavhyperlink{W}{\glsgetgrouptitle{W}} \textbar\
\glsnavhyperlink{X}{\glsgetgrouptitle{X}} \textbar\
\glsnavhyperlink{Y}{\glsgetgrouptitle{Y}} \textbar\
\glsnavhyperlink{Z}{\glsgetgrouptitle{Z}}}
```

The `\glssymbolnav` produces a simple navigation set of links for just the symbol and number groups. This is used at the start of `\glsnavigation`. If your glossary doesn't contain any symbol or navigation groups, you can redefine this command to do nothing.

`\glssymbolnav`

```
\newcommand*{\glssymbolnav}{%
\glsnavhyperlink{glssymbols}{\glsgetgrouptitle{glssymbols}} \textbar\
\glsnavhyperlink{glsnumbers}{\glsgetgrouptitle{glsnumbers}} \textbar\
}
```

4.15.2 List Style (glossary-list package)

The glossary-list package defines glossary styles that use the description environment. Note that since the entry name is placed in the optional argument to the `\item` command, it will appear in a bold font by default.

```
\ProvidesPackage{glossary-list}[2007/07/04 v1.01 (NLCT)]
```

The list glossary style uses the description environment. The group separator `\glsgroupskip` is redefined as `\indexspace` which produces a gap between groups. The glossary heading and the group headings do nothing. This is used as the default style for the glossaries package.

```
\newglossarystyle{list}{%
\renewenvironment{theglossary}{\begin{description}}{\end{description}}%
\renewcommand*{\glossaryheader}{}%
\renewcommand*{\glsgroupheading}[1]{}%
\renewcommand*{\glossaryentryfield}[5]{%
\item[\@glstarget{glo:##1}{##2}] ##3\glspostdescription\space ##5}%
\renewcommand*{\glsgroupskip}{\indexspace}}
```

The listgroup style is like the list style, but the glossary groups have headings.

```
\newglossarystyle{listgroup}{%
\glossarystyle{list}%
\renewcommand*{\glsgroupheading}[1]{\item[##1]}}
```

The listhypergroup style is like the listgroup style, but has a set of links to the groups at the start of the glossary.

```
\newglossarystyle{listhypergroup}{%
\glossarystyle{list}%
\renewcommand*{\glossaryheader}{%
\item[]\glsnavigation}%
\renewcommand*{\glsgroupheading}[1]{%
\item[\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}]}}
```

The altlist glossary style is like the list style, but places the description on a new line.

```
\newglossarystyle{altlist}{%
\glossarystyle{list}%
\renewcommand*{\glossaryentryfield}[5]{%
\item[\@glstarget{glo:##1}{##2}]\mbox{}\newline ##3\glspostdescription\space ##5}%
}
```

The altlistgroup glossary style is like the altlist style, but the glossary groups have headings.

```
\newglossarystyle{altlistgroup}{%
\glossarystyle{altlist}%
\renewcommand*{\glsgroupheading}[1]{\item[##1]}}
```

The altlisthypergroup glossary style is like the altlisthypergroup style, but has a set of links to the groups at the start of the glossary.

```
\newglossarystyle{altlisthypergroup}{%
\glossarystyle{altlist}%
\renewcommand*{\glossaryheader}{%
\item[]\glsnavigation}%
\renewcommand*{\glsgroupheading}[1]{%
\item[\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}]}}
```

4.15.3 Glossary Styles using longtable (the glossary-long package)

The glossary styles defined in the glossary-long package used the longtable environment in the glossary.

```
\ProvidesPackage{glossary-long}[2007/07/04 v1.01 (NLCT)]
```

Requires the longtable package:

```
\RequirePackage{longtable}
```

This is a length that governs the width of the description column.

`\glsdescwidth`

```
\newlength\glsdescwidth
```

This is a length that governs the width of the page list column.

`\glspagelistwidth`

```
\newlength\glspagelistwidth
```

Default values:

```
\setlength{\glsdescwidth}{0.6\linewidth}
\setlength{\glspagelistwidth}{0.1\linewidth}
```

The long glossary style command which uses the longtable environment:

```
\newglossarystyle{long}{%
\renewenvironment{theglossary}{\begin{longtable}{lp{\glsdescwidth}}}{%
\end{longtable}}%
\renewcommand*{\glossaryheader}{}%
\renewcommand*{\glsgroupheading}[1]{}%
\renewcommand*{\glossaryentryfield}[5]{%
\@glstarget{glo:##1}{##2} & ##3\glspostdescription\space ##5\\}%
\renewcommand*{\glsgroupskip}{ & \\}}
```

The longborder style is like the above, but with horizontal and vertical lines:

```
\newglossarystyle{longborder}{%
\glossarystyle{long}%
\renewenvironment{theglossary}{%
\begin{longtable}{|l|p{\glsdescwidth}|}}{\end{longtable}}%
\renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
}
```

The longheader style is like the long style but with a header:

```
\newglossarystyle{longheader}{%
\glossarystyle{long}%
\renewcommand*{\glossaryheader}{%
\bfseries \entryname & \bfseries \descriptionname\\
\endhead}}
```

The longheaderborder style is like the long style but with a header and border:

```
\newglossarystyle{longheaderborder}{%
\glossarystyle{longborder}%
\renewcommand*{\glossaryheader}{%
\hline\bfseries \entryname & \bfseries \descriptionname\\\hline
\endhead
\hline\endfoot}}
```

The long3col style is like long but with 3 columns

```
\newglossarystyle{long3col}{%
\renewenvironment{theglossary}{\begin{longtable}{lp{\glsdescwidth}p{\glspagelistwidth}}}{%
\end{longtable}}%
\renewcommand*{\glossaryheader}{}%
\renewcommand*{\glsgroupheading}[1]{}%
\renewcommand*{\glossaryentryfield}[5]{%
\@glstarget{glo:##1}{##2} & ##3 & ##5\\}%
\renewcommand*{\glsgroupskip}{ & &\\}}
```

The long3colborder style is like the long3col style but with a border:

```
\newglossarystyle{long3colborder}{%
\glossarystyle{long3col}%
\renewenvironment{theglossary}{%
\begin{longtable}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}}{%
\end{longtable}}%
\renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
}
```

The long3colheader style is like long3col but with a header row:

```
\newglossarystyle{long3colheader}{%
\glossarystyle{long3col}%
\renewcommand*{\glossaryheader}{%
\bfseries\entryname&\bfseries\descriptionname&
\bfseries\pagelistname\\\endhead}%
}
```

The long3colheaderborder style is like the above but with a border

```
\newglossarystyle{long3colheaderborder}{%
\glossarystyle{long3colborder}%
\renewcommand*{\glossaryheader}{%
\hline
\bfseries\entryname&\bfseries\descriptionname&
\bfseries\pagelistname\\\hline\endhead
\hline\endfoot}%
}
```

The long4col style has four columns where the third column contains the value of the associated symbol key.

```
\newglossarystyle{long4col}{%
\renewenvironment{theglossary}{%
\begin{longtable}{llll}}{%
\end{longtable}}%
\renewcommand*{\glossaryheader}{}%
\renewcommand*{\glsgroupheading}[1]{}%
\renewcommand*{\glossaryentryfield}[5]{%
\@glstarget{glo:##1}{##2} & ##3 & ##4 & ##5\\}%
\renewcommand*{\glsgroupskip}{ & & &\\}}
```

The long4colheader style is like long4col but with a header row.

```
\newglossarystyle{long4colheader}{%
\glossarystyle{long4col}%
\renewcommand*{\glossaryheader}{%
\bfseries\entryname&\bfseries\descriptionname&
\bfseries \symbolname&
\bfseries\pagelistname\\\endhead}%
}
```

The long4colborder style is like long4col but with a border.

```
\newglossarystyle{long4colborder}{%
\glossarystyle{long4col}%
\renewenvironment{theglossary}{%
\begin{longtable}{|l|l|l|l|}}{%
\end{longtable}}%
\renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
}
```

The long4colheaderborder style is like the above but with a border.

```
\newglossarystyle{long4colheaderborder}{%
\glossarystyle{long4col}%
\renewenvironment{theglossary}{%
\begin{longtable}{|l|l|l|l|}}{%
\end{longtable}}%
\renewcommand*{\glossaryheader}{%
\hline\bfseries\entryname&\bfseries\descriptionname&
```

```
\bfseries \symbolname&
\bfseries\pagelistname\\\hline\endhead\hline\\\endfoot}%
}
```

4.15.4 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the glossary-super package use the supertabular environment.

```
\ProvidesPackage{glossary-super}[2007/07/04 v1.01 (NLCT)]
```

Requires the supertabular package:

```
\RequirePackage{supertabular}
```

The super glossary style uses the supertabular environment (it uses lengths defined in the glossary-long package.)

```
\newglossarystyle{super}{%
\renewenvironment{theglossary}{%
\tablehead{}\tabletail{}%
\begin{supertabular}{lp{\glsdescwidth}}}{%
\end{supertabular}}%
\renewcommand*{\glossaryheader}{}%
\renewcommand*{\glsgroupheading}[1]{}%
\renewcommand*{\glossaryentryfield}[5]{%
\@glstarget{glo:##1}{##2} & ##3\glspostdescription\space ##5\\}%
\renewcommand*{\glsgroupskip}{ & \\}}
```

The superborder style is like the above, but with horizontal and vertical lines:

```
\newglossarystyle{superborder}{%
\glossarystyle{super}%
\renewenvironment{theglossary}{%
\tablehead{\hline}\tabletail{\hline}%
\begin{supertabular}{|l|p{\glsdescwidth}|}}{\end{supertabular}}%
}
```

The superheader style is like the super style, but with a header:

```
\newglossarystyle{superheader}{%
\glossarystyle{super}%
\renewenvironment{theglossary}{%
\tablehead{\bfseries \entryname & \bfseries \descriptionname\\}%
\tabletail{}%
\begin{supertabular}{lp{\glsdescwidth}}}{\end{supertabular}}%
}
```

The superheaderborder style is like the super style but with a header and border:

```
\newglossarystyle{superheaderborder}{%
\glossarystyle{super}%
\renewenvironment{theglossary}{%
\tablehead{\hline\bfseries \entryname & \bfseries \descriptionname\\\hline}%
\tabletail{\hline}
\begin{supertabular}{|l|p{\glsdescwidth}|}}{\end{supertabular}}%
}
```

The super3col style is like the super style, but with 3 columns:

```
\newglossarystyle{super3col}{%
\renewenvironment{theglossary}{%
```

```
\tablehead{}\tabletail{}%
\begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}{%
\end{supertabular}}%
\renewcommand*{\glossaryheader}{}%
\renewcommand*{\glsgroupheading}[1]{}%
\renewcommand*{\glossaryentryfield}[5]{%
\@glstarget{glo:##1}{##2} & ##3 & ##5\\}%
\renewcommand*{\glsgroupskip}{ & &\\}}
```

The super3colborder style is like the super3col style, but with a border:

```
\newglossarystyle{super3colborder}{%
\glossarystyle{super3col}%
\renewenvironment{theglossary}{%
\tablehead{\hline}\tabletail{\hline}%
\begin{supertabular}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}}{%
\end{supertabular}}%
}
```

The super3colheader style is like the super3col style but with a header row:

```
\newglossarystyle{super3colheader}{%
\glossarystyle{super3col}%
\renewenvironment{theglossary}{%
\tablehead{\bfseries\entryname&\bfseries\descriptionname&
\bfseries\pagelistname\\}\tabletail{}%
\begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}{%
\end{supertabular}}%
}
```

The super3colheaderborder style is like the super3col style but with a header and border:

```
\newglossarystyle{super3colheaderborder}{%
\glossarystyle{super3colborder}%
\renewenvironment{theglossary}{%
\tablehead{\hline
\bfseries\entryname&\bfseries\descriptionname&
\bfseries\pagelistname\\\hline}%
\tabletail{\hline}%
\begin{supertabular}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}}{%
\end{supertabular}}%
}
```

The super4col glossary style has four columns, where the third column contains the value of the corresponding symbol key used when that entry was defined.

```
\newglossarystyle{super4col}{%
\renewenvironment{theglossary}{%
\tablehead{}\tabletail{}%
\begin{supertabular}{llll}}{%
\end{supertabular}}%
\renewcommand*{\glossaryheader}{}%
\renewcommand*{\glsgroupheading}[1]{}%
\renewcommand*{\glossaryentryfield}[5]{%
\@glstarget{glo:##1}{##2} & ##3 & ##4 & ##5\\}%
\renewcommand*{\glsgroupskip}{ & & &\\}}
```

The super4colheader style is like the super4col but with a header row.

```
\newglossarystyle{super4colheader}{%
```

```
\glossarystyle{super4col}%
\renewenvironment{theglossary}{%
\tablehead{\bfseries\entryname&\bfseries\descriptionname&
\bfseries\symbolname &
\bfseries\pagelistname\\}\tabletail{}%
\begin{supertabular}{llll}}{%
\end{supertabular}}%
}
```

The super4colborder style is like the super4col but with a border.

```
\newglossarystyle{super4colborder}{%
\glossarystyle{super4col}%
\renewenvironment{theglossary}{%
\tablehead{\hline}\tabletail{\hline}%
\begin{supertabular}{|l|l|l|l|}}{%
\end{supertabular}}%
}
```

The super4colheaderborder style is like the super4col but with a header and border.

```
\newglossarystyle{super4colheaderborder}{%
\glossarystyle{super4col}%
\renewenvironment{theglossary}{%
\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
\bfseries\symbolname &
\bfseries\pagelistname\\}\tabletail{\hline}%
\begin{supertabular}{|l|l|l|l|}}{%
\end{supertabular}}%
}
```

Index

H

I