The glossaries package v4.14: a guide for beginners

Nicola L.C. Talbot

2015-02-28

Abstract

The glossaries package is very flexible, but this means that it has a lot of options, and since a user guide is supposed to provide a complete list of all the high-level user commands, the main user manual is quite big. This can be rather daunting for beginners, so this document is a brief introduction just to help get you started. If you find yourself saying, “Yeah, but how can I do...?” then it’s time to move on to the main user manual (glossaries-user.pdf).

I’ve made some statements in this document that don’t actually tell you the full truth, but it would clutter the document and cause confusion if I keep writing “except when...” or “but you can also do this, that or the other” or “you can do it this way but you can also do it that way, but that way may cause complications under certain circumstances”.

Contents

1 Getting Started 1
2 Defining Terms 5
3 Using Entries 8
4 Displaying a List of Entries 9
5 Customising the Glossary 15
6 Multiple Glossaries 17
7 glossaries and hyperref 19
8 Cross-References 21
1 Getting Started

As with all packages, you need to load `glossaries` with `\usepackage`, but there are certain packages that must be loaded before `glossaries`, if they are required: `hyperref`, `babel`, `polyglossia`, `inputenc` and `fontenc`. (You don’t have to load these packages, but if you want them, you must load them before `glossaries`.)

If you require multilingual support you must also install the relevant language module. Each language module is called `glossaries-⟨language⟩`, where ⟨language⟩ is the root language name. For example, `glossaries-french` or `glossaries-german`. If a language module is required, the `glossaries` package will automatically try to load it and will give a warning if the module isn’t found.

Once you have loaded `glossaries`, you need to define your terms in the preamble and then you can use them throughout the document. Here’s a simple example:

\documentclass{article}
\usepackage{glossaries}
\newglossaryentry{ex}{name={sample},description={an example}}
\begin{document}
Here’s my \gls{ex} term.
\end{document}

This produces:

Here’s my sample term.

Here’s another example:

\documentclass{article}
\usepackage{glossaries}
\setacronymstyle{long-short}
\newacronym{svm}{SVM}{support vector machine}
\begin{document}

2
First use: \gls{svm}. Second use: \gls{svm}.
\end{document}

This produces:

First use: support vector machine (SVM). Second use: SVM.

In this case, the text produced by \gls{svm} changed after the first use. The first use produced the long form followed by the short form in parentheses because I set the acronym style to long-short. I suggest you try the above two examples to make sure you have the package correctly installed. If you get an undefined control sequence error, check that the version number at the top of this document matches the version you have installed. (Open the .log file and search for the line that starts with Package: glossaries followed by a date and version.)

If you like, you can put all your definitions in another file (for example, defns.tex) and load that file in the preamble using \loadglsentries with the filename as the argument. For example:

\loadglsentries{defns}

Don’t try inserting formatting commands into the definitions as they can interfere with the underlying mechanism. Instead, the formatting should be done by the style. For example, suppose I want to replace SVM in the above to \textsc{svm}, then I need to select a style that uses \textsc, like this:

\documentclass{article}
\usepackage{glossaries}
\setacronymstyle{long-sc-short}
\newacronym{svm}{svm}{support vector machine}
\begin{document}
First use: \gls{svm}. Second use: \gls{svm}.
\end{document}

As you can hopefully see from the above examples, there are two main ways of defining a term. In both cases, the term is given a label. In the first case the label was \texttt{ex} and in the second case the label was \texttt{svm}. The label is just a way of identifying the term (like the standard \texttt{\label/\ref} mechanism). It’s best to just use the following alphanumerics in the labels: a, ..., z, A, ..., Z, 0, ..., 9. Sometimes you can also use punctuation characters but not if another package (such as babel) meddles with them. Don’t try using any characters outside of the basic Latin set (for example, é or ß)
or things will go horribly wrong. This warning only applies to the label.
It doesn’t apply to the text that appears in the document.

Don’t use `\gls` in chapter or section headings as it can have some
unpleasant side-effects. Instead use `\glsentrytext` for regular
entries and one of `\glsentryshort`, `\glsentrylong` or
`\glsentryfull` for acronyms.

The above examples are reasonably straightforward. The difficulty
comes if you want to display a sorted list of all the entries you have used
in the document. The glossaries package provides three options: use \TeX
to perform the sorting; use `makeindex` to perform the sorting; use `xindy`
to perform the sorting.

The first option (using \TeX) is the simplest and easiest method, but it’s
inefficient. To use this method, add `\makenoidxglossaries` to the
preamble and put `\printnoidxglossaries` at the place where you
want your glossary. For example:

```latex
\documentclass{article}
\usepackage{glossaries}
\makenoidxglossaries
\newglossaryentry{potato}{name={potato},plural={potatoes},
  description={starchy tuber}}
\newglossaryentry{cabbage}{name={cabbage},
  description={vegetable with thick green or purple leaves}}
\newglossaryentry{carrot}{name={carrot},
  description={orange root}}
\begin{document}
Chop the \gls{cabbage}, \glspl{potato} and \glspl{carrot}.
\printnoidxglossaries
\end{document}
```

Try this out and run \TeX (or pdf\TeX) \textit{twice}. The first run won’t show the
glossary. It will only appear on the second run. The glossary has a vertical
gap between the “carrot” term and the “potato” term. This is because the
entries in the glossaries are grouped according to their first letter. If you
don’t want this gap, just add `\nogroupskip` to the package options:

```latex
\usepackage[nogroupskip]{glossaries}
```

4
If you try out this example you may also notice that the description is followed by a full stop (period) and a number. The number is the location in the document where the entry was used, so you can lookup the term in the glossary and be directed to the relevant pages. It may be that you don’t want this back-reference, in which case you can suppress it using the nonumberlist package option:
\usepackage[nonumberlist]{glossaries}

If you don’t like the terminating full stop, you can suppress that with the nopostdot package option:
\usepackage[nopostdot]{glossaries}

You may have noticed that I’ve used another command in the above example: \glspl. This displays the plural form. By default, this is just the singular form with the letter “s” appended, but in the case of “potato” I had to specify the correct plural using the plural key.

As I mentioned earlier, using \TeX to sort the entries is the simplest but least efficient method. If you have a large glossary or if your terms contain non-Latin characters, then you will have a much faster build time if you use makeindex or xindy. If you are using extended Latin or non-Latin characters, then xindy is the recommended method. These two methods are described in more detail in Section 4.

The rest of this document briefly describes the main commands provided by the glossaries package.

2 Defining Terms

When you use the glossaries package, you need to define glossary entries in the document preamble. These entries could be a word, phrase, acronym or symbol. They’re usually accompanied by a description, which could be a short sentence or an in-depth explanation that spans multiple paragraphs. The simplest method of defining an entry is to use:

\newglossaryentry{⟨label⟩}
{
   name=⟨name⟩,
   description=⟨description⟩,
   ⟨other options⟩
}

where ⟨label⟩ is a unique label that identifies this entry. (Don’t include the angle brackets ⟨ ⟩. They just indicate the parts of the code you need to change when you use this command in your document.) The ⟨name⟩
is the word, phrase or symbol you are defining, and \textit{description} is the description to be displayed in the glossary.

This command is a “short” command, which means that \textit{description} can’t contain a paragraph break. If you have a long description, you can instead use:

\begin{Verbatim}
\longnewglossaryentry{(label)}
{
  name={\langle name \rangle},
  \langle other options \rangle
}
\langle description \rangle
\end{Verbatim}

Examples:

1. Define the term “set” with the label \textit{set}:

\begin{verbatim}
\newglossaryentry{set}
{
  name={set},
  description={a collection of objects}
}
\end{verbatim}

2. Define the symbol $\emptyset$ with the label \textit{emptyset}:

\begin{verbatim}
\newglossaryentry{emptyset}
{
  name={\ensuremath{\emptyset}},
  description={the empty set}
}
\end{verbatim}

3. Define the phrase “Fish Age” with the label \textit{fishage}:

\begin{verbatim}
\longnewglossaryentry{fishage}
{name={Fish Age}}
{%
  A common name for the Devonian geologic period spanning from the end of the Silurian Period to the beginning of the Carboniferous Period.

  This age was known for its remarkable variety of fish species.
} %
\end{verbatim}

(The percent character discards the end of line character that would otherwise cause an unwanted space to appear at the start of the description.)
4. Take care if the first letter is an extended Latin or non-Latin character (either specified via a command such as `\'e` or explicitly via the `inputenc` package such as `é`). This first letter must be placed in a group:

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

or

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

(For further details, see the section “UTF-8” (mfirstuc-manual.pdf) in the `mfirstuc` user manual.)

Acronyms or abbreviations can be defined using

```
\newacronym{\langle label\rangle}{\langle short\rangle}{\langle long\rangle}
```

where \langle label\rangle is the label (as with the `\newglossaryentry` and the `\longnewglossaryentry` commands), \langle short\rangle is the abbreviation or acronym and \langle long\rangle is the long form. For example:

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

This defines a glossary entry with the label `svm`. By default, the \langle name\rangle is set to \langle short\rangle (“svm” in the above example) and the \langle description\rangle is set to \langle long\rangle (“support vector machine” in the above example). If, instead, you want to be able to specify your own description you can do this using the optional argument:

```
\newacronym[description={statistical pattern recognition technique}]{svm}{svm}{support vector machine}
```

Before you define your acronyms, you need to specify which acronym style to use with

```
\setacronymstyle{\langle style name\rangle}
```
where \(<style name>\) is the name of the style. There are a number of pre-defined styles, such as: \text{long-short} (on first use display the long form with the short form in parentheses); \text{short-long} (on first use display the short form with the long form in parentheses); \text{long-short-desc} (like \text{long-short} but you need to specify the description); or \text{short-long-desc} (like \text{short-long} but you need to specify the description). There are some other styles as well that use \textsc{\texttt{\texttt{/textsc}}} to typeset the acronym or that use a footnote on first use. See the main user guide for further details.

There are other keys you can use when you define an entry. For example, the name key used above indicates how the term should appear in the list of entries (glossary). If the term should appear differently when you reference it in the document, you need to use the text key as well.

For example:

\begin{verbatim}
\newglossaryentry{latinalph} {
    name={Latin Alphabet},
    text={Latin alphabet},
    description={alphabet consisting of the letters a, \ldots, z, A, \ldots, Z}
}
\end{verbatim}

This will appear in the text as “Latin alphabet” but will be listed in the glossary as “Latin Alphabet”.

Another commonly used key is \text{plural} for specifying the plural of the term. This defaults to the value of the text key with an “s” appended, but if this is incorrect, just use the \text{plural} key to override it:

\begin{verbatim}
\newglossaryentry{oesophagus} {
    name={\oe}sophagus,
    plural={\oe}sophagi,
    description={canal from mouth to stomach}
}
\end{verbatim}

(Remember from earlier that the initial ligature \oe needs to be grouped.)

The plural forms for acronyms can be specified using the \text{longplural} and \text{shortplural} keys. For example:

\begin{verbatim}
\newacronym{longplural={diagonal matrices}}{dm}{DM}{diagonal matrix}
\end{verbatim}

If omitted, the defaults are again obtained by appending an “s” to the singular versions.

It’s also possible to have both a name and a corresponding symbol. Just use the name key for the name and the symbol key for the symbol. For example:
3 Using Entries

Once you have defined your entries, as described above, you can reference them in your document. There are a number of commands to do this, but the most common one is:

\gls{⟨label⟩}

where ⟨label⟩ is the label you assigned to the entry when you defined it. For example, \gls{fishage} will display “Fish Age” in the text (given the definition from the previous section).

If you are using the hyperref package (remember to load it before glossaries) \gls will create a hyperlink to the corresponding entry in the glossary. If you want to suppress the hyperlink for a particular instance, use the starred form \gls* for example, \gls*{fishage}. The other commands described in this section all have a similar starred form.

If the entry was defined as an acronym (using \newacronym described above), then \gls will display the full form the first time it’s used and just the short form on subsequent use. For example, if the acronym style is set to long-short, then \gls{svm} will display “support vector machine (svm)” the first time it’s used, but the next occurrence of \gls{svm} will just display “svm”.

If you want the plural form, you can use:

\glspl{⟨label⟩}

instead of \gls{⟨label⟩}. For example, \glspl{set} displays “sets”.

If the term appears at the start of a sentence, you can convert the first letter to upper case using:

\Gls{⟨label⟩}

for the singular form or

\Glspl{⟨label⟩}
for the plural form. For example:
\Glspl{set} are collections.
produces “Sets are collections”.

If you’ve specified a symbol using the symbol key, you can display it using:
\glssymbol{⟨label⟩}

4 Displaying a List of Entries

In Section 1 I mentioned that there are three options you can choose from to create your glossary. Here they are again in a little more detail:

Option 1:
This is the simplest option but it’s slow and if you want a sorted list, it doesn’t work for non-Latin alphabets.

1. Add \makenoidxglossaries to your preamble (before you start defining your entries, as described in Section 2).
2. Put
\printnoidxglossary[sort=⟨order⟩, ⟨other options⟩]

where you want your list of entries to appear. The sort ⟨order⟩ may be one of: word (word ordering), letter (letter ordering), case (case-sensitive letter ordering), def (in order of definition) or use (in order of use). Alternatively, use
\printnoidxglossaries
to display all your glossaries (if you have more than one). This command doesn’t have any arguments.
3. Run \LaTeX{} twice on your document. (As you would do to make a table of contents appear.) For example, click twice on the “typeset” or “build” or “PDF\LaTeX{}” button in your editor.

Option 2:
This option uses an application called makeindex to sort the entries. This application comes with all modern \TeX{} distributions, but it’s hard-coded for the non-extended Latin alphabet. This process
involves making \LaTeX{} write the glossary information to a temporary file which makeindex reads. Then makeindex writes a new file containing the code to typeset the glossary. \LaTeX{} then reads this file on the next run.

1. Add `\makeglossaries` to your preamble (before you start defining your entries).

2. Put

\begin{verbatim}
\printglossary[⟨options⟩]
\end{verbatim}

where you want your list of entries (glossary) to appear. (The sort key isn’t available in ⟨options⟩.) Alternatively, use

\begin{verbatim}
\printglossaries
\end{verbatim}

which will display all glossaries (if you have more than one). This command doesn’t have any arguments.

3. Run \LaTeX{} on your document. This creates files with the extensions .glo and .ist (for example, if your \LaTeX{} document is called myDoc.tex, then you’ll have two extra files called myDoc.glo and myDoc.ist). If you look at your document at this point, you won’t see the glossary as it hasn’t been created yet.

4. Run makeindex with the .glo file as the input file and the .ist file as the style so that it creates an output file with the extension .gls. If you have access to a terminal or a command prompt (for example, the MSDOS command prompt for Windows users or the bash console for Unix-like users) then you need to run the command:

\begin{verbatim}
makeindex -s myDoc.ist -o myDoc.gls myDoc.glo
\end{verbatim}

(Replace myDoc with the base name of your \LaTeX{} document file. Avoid spaces in the file name.) If you don’t know how to use the command prompt, then you can probably access makeindex via your text editor, but each editor has a different method of doing this, so I can’t give a general description. You will have to check your editor’s manual.

The default sort is word order (“sea lion” comes before “seal”). If you want letter ordering you need to add the \texttt{-l} switch:

\begin{verbatim}
makeindex -l -s myDoc.ist -o myDoc.gls myDoc.glo
\end{verbatim}
5. Once you have successfully completed the previous step, you can now run \LaTeX on your document again.

**Option 3:**
This option uses an application called *xindy* to sort the entries. This application is more flexible than *makeindex* and is able to sort extended Latin or non-Latin alphabets. It comes with \TeX Live but not with MiK\TeX. Since *xindy* is a Perl script, if you are using MiK\TeX you will not only need to install *xindy*, you will also need to install Perl. In a similar way to **Option 2**, this option involves making \LaTeX write the glossary information to a temporary file which *xindy* reads. Then *xindy* writes a new file containing the code to typeset the glossary. \LaTeX then reads this file on the next run.

1. Add the *xindy* option to the *glossaries* package option list:
   \usepackage*[xindy]{glossaries}

2. Add \makeglossaries to your preamble (before you start defining your entries).

3. Put
   \printglossary[⟨options⟩]
   where you want your list of entries (glossary) to appear. (The sort key isn’t available in ⟨options⟩.) Alternatively, use
   \printglossaries

4. Run \LaTeX on your document. This creates files with the extensions .glo and .xdy (for example, if your \LaTeX document is called myDoc.tex, then you’ll have two extra files called myDoc.glo and myDoc.xdy). If you look at your document at this point, you won’t see the glossary as it hasn’t been created yet.

5. Run *xindy* with the .glo file as the input file and the .xdy file as a module so that it creates an output file with the extension .gls. You also need to set the language name and input encoding. If you have access to a terminal or a command prompt (for example, the MSDOS command prompt for Windows users or the bash console for Unix-like users) then you need to run the command (all on one line):
   xindy -L english -C utf8 -I xindy -M myDoc -t myDoc.glg -o myDoc.gls myDoc.glo

12
(Replace myDoc with the base name of your \LaTeX{} document file. Avoid spaces in the file name. If necessary, also replace \texttt{english} with the name of your language and \texttt{utf8} with your input encoding.) If you don’t know how to use the command prompt, then you can probably access \texttt{xindy} via your text editor, but each editor has a different method of doing this, so I can’t give a general description. You will have to check your editor’s manual.

The default sort is word order (“sea lion” comes before “seal”). If you want letter ordering you need to add the \texttt{order=letter} package option:

\begin{verbatim}
\usepackage[xindy,order=letter]{glossaries}
\end{verbatim}

6. Once you have successfully completed the previous step, you can now run \LaTeX{} on your document again.

For Options 2 and 3, it can be difficult to remember all the parameters required for \texttt{makeindex} or \texttt{xindy}, so the \texttt{glossaries} package provides a script called \texttt{makeglossaries} that reads the \texttt{.aux} file to determine what settings you have used and will then run \texttt{makeindex} or \texttt{xindy}. Again, this is a command line application and can be run in a terminal or command prompt. For example, if your \LaTeX{} document is in the file \texttt{myDoc.tex}, then run:

\begin{verbatim}
makeglossaries myDoc
\end{verbatim}

(Replace \texttt{myDoc} with the base name of your \LaTeX{} document file. Avoid spaces in the file name.) If you don’t know how to use the command prompt, you can probably access \texttt{makeglossaries} via your text editor. Check your editor’s manual for advice. If you are using \texttt{arara} then you can just use the directives:

\begin{verbatim}
arara: pdflatex
arara: makeglossaries
arara: pdflatex
\end{verbatim}

The \texttt{makeglossaries} script is written in Perl, so you need a Perl interpreter installed. If you are using a Unix-like operating system then you most likely have one installed. If you are using Windows with the \TeX{} Live distribution, then you can use the Perl interpreter that comes with \TeX{} Live. If you are using Windows and MiK\TeX{} then you need to install a Perl distribution for Windows. If you are using \texttt{Option 3}, then you need to do this anyway as \texttt{xindy} is also written in Perl. If you are using \texttt{Option 2} and can’t work out how to install Perl (or for some reason don’t want to install it) then just use \texttt{makeindex} directly, as described above.

As a last resort you can try the package option \texttt{automake}:

\begin{verbatim}
\usepackage[automake]{glossaries}
\end{verbatim}
This will attempt to use \TeX's \texttt{\write18} mechanism to run \texttt{makeindex} or \texttt{xindy}. It probably won’t work for \texttt{xindy} and won’t work at all if the shell escape has been disabled in your \TeX distribution. Most \TeX distributions will allow a restricted shell escape, which will only allow trusted applications to be run. If the \texttt{automake} option is successful, you will still need to run \TeX twice for the glossaries to be displayed.

When sorting the entries, the string comparisons are made according to each entry’s \texttt{sort} key. If this is omitted, the \texttt{name} key is used. For example, recall the earlier definition:

\begin{verbatim}
\newglossaryentry{elite}
{  
  name={\"e\}lite},
  description={select group or class}
}
\end{verbatim}

No \texttt{sort} key was used, so it’s set to the same as the \texttt{name} key: {\"e\}lite.

How this is interpreted depends on which option you have used:

\textbf{Option 1:} By default, the accent command will be stripped so the sort value will be \texttt{elite}. This will put the entry in the “E” letter group. However if you use the \texttt{sanitizesort=true} package option, the sort value will be interpreted as the sequence of characters: \texttt{\'e l i t} and \texttt{e}. This will place this entry before the “A” letter group since it starts with a symbol.

\textbf{Option 2:} The sort key will be interpreted the sequence of characters: {\texttt{ \'e \}l i t} and \texttt{e}. The first character is an opening curly brace { so \texttt{makeindex} will put this entry in the “symbols” group.

\textbf{Option 3:} \texttt{xindy} disregards \LaTeX commands so it sorts on \texttt{elite}, which puts this entry in the “E” group.

If the \texttt{inputenc} package is used and the entry is defined as:

\begin{verbatim}
\newglossaryentry{elite}
{  
  name={\élite},
  description={select group or class}
}
\end{verbatim}

then:

\textbf{Option 1:} By default the sort value will be interpreted as \texttt{elite} so the entry will be put in the “E” letter group. If you use the \texttt{sanitizesort=true} package option, the sort value will be interpreted as \texttt{élite} where \texttt{é} has been sanitized (so it’s no longer an active character) which will put this entry before the “A” letter group.
**Option 2:** makeindex doesn’t recognise é as a letter so it will be put in the symbols group.

**Option 3:** xindy will correctly recognise the sort value élite and will place it in whatever letter group is appropriate for the given language setting. (In English, this would just be the “E” letter group.)

Therefore if you have extended Latin or non-Latin characters, your best option is to use xindy (Option 3) with the inputenc package. If you use makeindex (Option 2) you need to specify the sort key like this:

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

If you use Option 1, you may or may not need to use the sort key, but you will need to be careful about fragile commands in the name key if you don’t set the sort key.

Table 1 summarises the main pros and cons of three options described above.

**Table 1: Glossary Options: Pros and Cons**

<table>
<thead>
<tr>
<th></th>
<th>Option 1</th>
<th>Option 2</th>
<th>Option 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requires an external application?</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Requires Perl?</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Can sort extended Latin or non-Latin alphabets?</td>
<td>✗†</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Efficient sort algorithm?</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Can form ranges in the location lists?</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Can have non-standard locations?</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
</tr>
</tbody>
</table>

† Strips standard \LaTeX accents so, for example, \textbackslash AA is treated the same as A.


5 Customising the Glossary

The default glossary style uses the description environment to display the entry list. Each entry name is set in the optional argument of \item which means that it will typically be displayed in bold. You can switch to medium weight by redefining \glsnamefont:

\renewcommand*{\glsnamefont}{\textmd{#1}}

By default, a full stop is appended to the description. To prevent this from happening use the nopostdot package option:

\usepackage[nopostdot]{glossaries}

By default, a location list is displayed for each entry. This refers to the document locations (for example, the page number) where the entry has been referenced. If you use Options 2 or 3 described in Section 4 location ranges will be compressed. For example, if an entry was used on pages 1, 2 and 3, with Options 2 or 3 the location list will appear as 1–3, but with Option 1 it will appear as 1, 2, 3. If you don’t want the locations displayed you can hide them using the nonumberlist package option:

\usepackage[nonumberlist]{glossaries}

Entries are grouped according to the first letter of each entry’s sort key. By default a vertical gap is placed between letter groups. You can suppress this with the nogroupskip package option:

\usepackage[nogroupskip]{glossaries}

If the default style doesn’t suit your document, you can change the style using:

\setglossarystyle{⟨style name⟩}

There are a number of predefined styles. Glossaries can vary from a list of symbols with a terse description to a list of words or phrases with descriptions that span multiple paragraphs, so there’s no “one style fits all” solution. You need to choose a style that suits your document. Examples:

1. You have entries where the name is a symbol and the description is a brief phrase or short sentence. Try one of the “mcol” styles defined in the glossary-mcols package. For example:

\usepackage[nogroupskip,nopostdot]{glossaries}
\usepackage{glossary-mcols}
\setglossarystyle{mcolindex}
2. You have entries where the name is a word or phrase and the description spans multiple paragraphs. Try one of the “altlist” styles. For example:

\usepackage[nopostdot]{glossaries}
\setglossarystyle{altlist}

3. You have entries where the name is a single word, the description is brief, and an associated symbol has been set. Use one of the styles that display the symbol (not all of them do). For example, one of the tabular styles:

\usepackage[nopostdot,nonumberlist]{glossaries}
\setglossarystyle{long4col}

or one of the “tree” styles:

\usepackage[nopostdot,nonumberlist]{glossaries}
\setglossarystyle{tree}

If your glossary consists of a list of acronyms and you also want to specify a description as well as the long form, then you need to use an acronym style that will suit the glossary style. For example, use the long-short-desc acronym style:

\setacronymstyle{long-short-desc}

Define the acronyms with a description:

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

Choose a glossary style that suits wide entry names:

\setglossarystyle{altlist}

6 Multiple Glossaries

The glossaries package predefines a default main glossary. When you define an entry (using one of the commands described in Section 2), that entry is automatically assigned to the default glossary, unless you indicate otherwise using the type key. However you first need to make sure the desired glossary has been defined. This is done using:

\newglossary[⟨glg⟩]{⟨label⟩}{⟨gls⟩}{⟨glo⟩}{⟨title⟩}
The \langle label \rangle is a label that uniquely identifies this new glossary. As with other types of identifying labels, be careful not to use active characters in \langle label \rangle. The final argument \langle title \rangle is the section or chapter heading used by \texttt{printglossary} or \texttt{printnoidxglossary}. The other arguments indicate the file extensions used by makeindex/xindy (described in Section 4). If you use Option 1 described above, the \langle glg \rangle, \langle gls \rangle and \langle glo \rangle arguments are ignored, in which case you may prefer to use the starred version where you don’t specify the extensions:

\begin{verbatim}
\newglossary*{\langle label \rangle}{\langle title \rangle}
\end{verbatim}

In the case of Options 2 or 3, all glossary definitions must come before \texttt{makeglossaries}.

Since it’s quite common for documents to have both a list of terms and a list of acronyms, the glossaries package provides the package option acronym (or acronyms), which is a convenient shortcut for

\begin{verbatim}
\newglossary[alg]{acronym}{acr}{acn}{\acronymname}
\end{verbatim}

The option also changes the behaviour of \texttt{newacronym} so that acronyms are automatically put in the list of acronyms instead of the main glossary.

For example, suppose you want a main glossary for terms, a list of acronyms and a list of notation:

\begin{verbatim}
\usepackage[acronyms]{glossaries}
\newglossary[nlg]{notation}{not}{ntn}{Notation}
\end{verbatim}

After \texttt{makeglossaries} (or \texttt{makenoidxglossaries}) you can define the entries in the preamble. For example:

\begin{verbatim}
\newglossaryentry{gls:set}
{% This entry goes in the 'main' glossary
  name=set,
  description={A collection of distinct objects}
}

This entry goes in the ‘acronym’ glossary:
\newacronym{svm}{svm}{support vector machine}

\newglossaryentry{not:set}
{% This entry goes in the 'notation' glossary:
  type=notation,
  name={\ensuremath{\mathcal{S}}},
  description={A set},
  sort={S}}

or if you don't like using \ensuremath:

\newglossaryentry{not:set}
\end{verbatim}
Each glossary is displayed using:

\printnoidxglossary[type=type]

(Option 1) or

\printglossary[type=type]

(Options 2 and 3). Where \textit{type} is the glossary label. If the type is omitted the default main glossary is assumed.

There's a convenient shortcut that will display all the defined glossaries:

\printnoidxglossaries

(Option 1) or

\printglossaries

(Options 2 and 3).

If you use Option 1, you don't need to do anything else. If you use Options 2 or 3 with the \texttt{makeglossaries} Perl script, you similarly don't need to do anything else. If you use Options 2 or 3 without the \texttt{makeglossaries} Perl script then you need to make sure you run \texttt{makeindex/xindy for each defined glossary}. The \texttt{.gls} and \texttt{.glo} arguments of \texttt{\newglossary} specify the file extensions to use instead of \texttt{.gls} and \texttt{.glo}. The optional argument \texttt{.glg} is the file extension for the transcript file. This should be different for each glossary in case you need to check for \texttt{makeindex/xindy errors or warnings} if things go wrong.

For example, suppose you have three glossaries in your document (main, acronym and notation), specified using:

\usepackage[acronyms]{glossaries}
\newglossary[nlg]{notation}{not}{ntn}{Notation}

Then (assuming your \LaTeX document is in a file called \texttt{myDoc.tex}):  

\textbf{Option 2:}

You need to run \texttt{makeindex} three times:
Option 3:
You need to run `xindy` three times (be careful not to insert line breaks where the line has wrapped.)

```
xindy -L english -C utf8 -I xindy -M myDoc -t myDoc.glg
    -o myDoc.gls myDoc.glo
xindy -L english -C utf8 -I xindy -M myDoc -t myDoc.alg
    -o myDoc.acr myDoc.acn
xindy -L english -C utf8 -I xindy -M myDoc -t myDoc.nlg
    -o myDoc.not myDoc.ntn
```

7 glossaries and hyperref

Take care if you use the glossaries package with hyperref. Contrary to the usual advice that hyperref should be loaded last, glossaries must be loaded after hyperref:

```
\usepackage[colorlinks]{hyperref}
\usepackage{glossaries}
```

If you use hyperref make sure you use PDFLaTeX rather than the \TeX to DVI engine. The DVI format can’t break hyperlinks across a line so long glossary entries (such as the full form of acronyms) won’t line wrap with the DVI engine. Also, hyperlinks in sub- or superscripts aren’t correctly sized with the DVI format.

By default, if the hyperref package has been loaded, commands like `\gls` will form a hyperlink to the relevant entry in the glossary. If you to disable this for all your glossaries, then use

```
\glsdisablehyper
```

If you want hyperlinks suppressed for entries in specific glossaries, then use the nohypertypes package option. For example, if you don’t want hyperlinks for entries in the acronym and notation glossaries but you do want them for entries in the main glossary, then do:

```
\usepackage[colorlinks]{hyperref}
\usepackage[acronym,nohypertypes={acronym,notation}]{glossaries}
\newglossary[ngl]{notation}{not}{ntn}{Notation}
```

If you want the hyperlinks suppressed the first time an entry is used, but you want hyperlinks for subsequence references then use the `hyperfirst=false` package option:
Take care not to use non-expandable commands in PDF bookmarks. This isn’t specific to the glossaries package but is a limitation of PDF bookmarks. Non-expandable commands include commands like \gls, \glspl, \Gls and \Glspl. The \hyperref package provides a way of specifying alternative text for the PDF bookmarks via \texorpdfstring. For example:

\section{The \texorpdfstring{\gls{fishage}}{Fish Age}}

However, it’s not a good idea to use commands like \gls in a section heading as you’ll end up with the table of contents in your location list. Instead you can use

\glsentrytext{⟨label⟩}

This is expandable provided that the text key doesn’t contain non-expandable code. For example, the following works:

\section{The \glsentrytext{fishage}}

and it doesn’t put the table of contents in the location list.

8 Cross-References

You can add a reference to another entry in a location list using the see=⟨⟨label⟩⟩ key when you define an entry. The referenced entry must also be defined.

For example:

\longnewglossaryentry{devonian}{name={Devonian}}%
{%
   The geologic period spanning from the end of the
   Silurian Period to the beginning of the Carboniferous Period.

   This age was known for its remarkable variety of
   fish species.
%}

\newglossaryentry{fishage}
{
   name={Fish Age},
   description={Common name for the Devonian period},
   see={devonian}
}
The cross-reference will appear as “see Devonian”. You can change the “see” tag using the format see=\{\langle tag\rangle\}\langle label\rangle. For example:

\newglossaryentry{latinalph}
{
  name={Latin alphabet},
  description={alphabet consisting of the letters a, \ldots, z, A, \ldots, Z},
  see=\{see also\}\{exlatinalph\}
}
\newglossaryentry{exlatinalph}
{
  name={extended Latin alphabet},
  description={The Latin alphabet extended to include other letters such as ligatures or diacritics.}
}

If you use the see key in the optional argument of \newacronym, make sure you enclose the value in braces. For example:

\newacronym{ksvm}{kernel support vector machine}
\newacronym{svm}{support vector machine}
\newacronym{svm}{svm}{see=\{\{see also\}\{ksvm\}\}}

Since the cross-reference appears in the location list, if you suppress the location list using the nonumberlist package option, then the cross-reference will also be suppressed.

9 Further Information

Further information can be found in the main glossaries user manual (glossaries-user.pdf) or in the glossaries FAQ. There is also an article on the glossaries package on the \LaTeX\ Community’s\footnote{\url{http://www.latex-community.org/}} Know How section and a chapter on the glossaries package in \LaTeX\ to Write a PhD Thesis.