

Glossaries, Nomenclature, Lists of Symbols and Acronyms

★★★★☆ (93 votes, average: 4.04 out of 5)

LaTeX - General

WRITTEN BY NICOLA TALBOT

WEDNESDAY, 18 MARCH 2009 16:58



The `glossaries` package can be used to define terms, symbols and abbreviations that can be used throughout a document. You can then use an indexing application to collate and sort the entries that have been used in the document. It is a highly versatile package, but because of this it has a large manual that some beginners find daunting. This article is an introductory guide to help you get started. (Updated to include the `glossaries-extra` package.)

What do I need installed?

The `glossaries` package principally requires:

- ▶ `ifthen`
- ▶ `xkeyval` at least version 2.5f (2006/11/18)
- ▶ `xfor`
- ▶ `amsgen` (part of `amstex`)
- ▶ `etoolbox`
- ▶ `mfistuc`
- ▶ `translator`
- ▶ `datatool-base`

Some of the features described here are only available in more recent releases. If you get any "undefined control sequence" errors or unknown options, check that you have the latest version.

If you use a language package, such as `babel` or `polyglossia`, you'll also need to install the appropriate `glossaries` language module. For example, `glossaries-french` or `glossaries-german`.

If you want to use `glossaries-extra` you'll need both `glossaries` and `glossaries-extra` installed. (The `glossaries-extra` package is an extension to not a replacement for `glossaries`.)

If you want to use any of the glossary styles that use the `longtable` environment, you will also need to have the `longtable` package installed, and if you want to use any of the glossary styles that use the `supertabular` environment, you will also need to have the `supertabular` package installed.

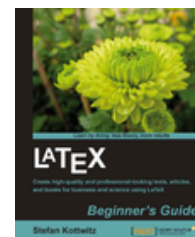
The above packages, including `glossaries` and `glossaries-extra`, are all distributed with MikTeX and TeX Live. (If you have MikTeX or TeX Live, you can install or update packages using the [MikTeX Package Manager](#) or the [TeX Live Package Manager](#), respectively.)

Before you get started, check that you have all the required packages installed using the following test document:

```
\documentclass{article}
\usepackage{glossaries}
\begin{document}
Test.
\end{document}
```

If you want multilingual support, load the appropriate language package before loading `glossaries`. For example:

Featured Book



Supporters and Partners

Sponsored by [DANTE e.V.](#): The German speaking TeX Users Group

TEXNIQUE.fr

TeX_WLT Fragen & Antworten

TeXblog

TeXample.net
& the TikZ Gallery

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{glossaries}
\begin{document}
Test \entryname.
\end{document}
```

The English definition of `\entryname` is “Notation”. If the appropriate language module is installed, then this should be translated into the relevant language. (Not all languages are supported.)

If you want to use the `glossaries-extra` package, replace `glossaries` with `glossaries-extra` in the above examples. If the test document compiles without error, the packages are correctly installed. If you have any errors, such as ``mfistuc.sty'` not found, then use your TeX package manager to install the missing packages.

The version number for the packages will be listed in the log file. For example, my log file contains the line:

```
Package: glossaries 2016/04/19 v4.22 (NLCT)
```

and also (for documents that use `glossaries-extra`):

```
Package: glossaries-extra 2016/04/25 v1.02 (NLCT)
```

You can compare this version information in your log file against the latest version listed on CTAN for [glossaries](#) and [glossaries-extra](#).

In addition to the above LaTeX packages, if you want a sorted glossary (list of terms or abbreviations) you’ll also need to have an indexing application installed to collate and sort the entries (unless you want TeX to do this, which is possible but has limitations). The `glossaries` package is configured to work with [makeindex](#) or [xindy](#). `Makeindex` is distributed with most TeX distributions, including MikTeX and TeX Live, so it should be readily available on your system. `Xindy` is distributed with TeX Live and (more recently) with MikTeX, but it requires Perl.



There is a common misconception that you must have [Perl](#) installed to use the `glossaries` package. You need Perl for `xindy`, but you can use `makeindex` instead. The `glossaries` package comes with a Perl script called `makeglossaries` that acts as a convenient interface to `makeindex` or `xindy`, but it is not essential. If you don’t want to install Perl, there are alternatives, which are discussed below.

Getting Started

The first thing that you need to do in your document is indicate that you want to use the `glossaries` package:

```
\usepackage{glossaries}
```

Note that if you want to use `xindy` instead of `makeindex`, you must add the `xindy` package option:

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

However you will need to have `xindy` and Perl installed if you want to use this option.

If you want to use [glossaries-extra](#), then replace this with

```
\usepackage{glossaries-extra}
```

or

```
\usepackage[xindy]{glossaries-extra}
```

If you are also using the [hyperref](#) package, you must load the `glossaries` or `glossaries-extra` package after the `hyperref` package:

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



or

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

```
\usepackage{glossaries-extra}
```

This is an exception to the general rule that the hyperref package should be loaded last.

If you want your glossaries (list of terms, abbreviations, symbols, etc) to appear in the document's table of contents, you also need to use the toc package option:

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

This is the default for glossaries-extra, so you just need

```
\usepackage{glossaries-extra}
```

If you want to use this extension package and you *don't* want the glossaries to appear in the table of contents, then use the toc=false package option:

```
\usepackage[ toc=false]{glossaries-extra}
```

Defining a Term or Symbol


Terms and symbols are defined using

```
\newglossaryentry{label}{definition}
```

The first argument is a label that uniquely defines this entry.

 It is best to use only non-special characters in the label. Avoid special characters such as ^ or _ and **active** characters like ~ or any of the babel shortcuts. For regular LaTeX or PDFLaTeX, also avoid extended Latin or non-Latin characters, but those may be used with XeLaTeX or LuaLaTeX.


The second argument is a comma-separated list of *key=value* pairs. The most important keys are name and description. These are used to assign the term and a brief description to appear in the glossary, list of symbols or list of abbreviations.

 This article only covers the commonly used keys. See the section "[Defining Glossary Entries](#)" in the manual for a complete list.

For example:

```
\newglossaryentry{culdesac}{name=cul-de-sac,description={passage  
or street closed at one end}}
```

The above defines an entry whose label is culdesac.

 It is generally best to define the terms in the preamble as **problems can occur if you define them in the document environment**. You can put all your definitions in a separate file and input it in the preamble using `\loadglsentries{filename}`, which you may find more convenient. You may also use `\input` but **don't use \include**.

Referencing Terms

Once a term has been defined using `\newglossaryentry`, it can be referenced in the document using:

```
\gls{label}
```

Returning to the example above, I can reference it using:


```
\gls{culdesac}
```


If I want the first letter to be converted to upper case, I can do:

```
\Gls{culdesac}
```

or if I want the whole term to appear in capitals, I can do:

```
\GLS{culdesac}
```

 The above commands also have optional arguments that aren't discussed in this article. For further information, see the section "[Links to Glossary Entries](#)" in the manual.

 Don't use these commands in chapter or section headings or captions as they can cause unexpected problems.

If you use glossaries-extra, you can use

```
\glsfmttext{label}
```

in sectioning headings or captions instead. This isn't available for the base glossaries package.

Let's put it together to make a simple document:

```
\documentclass{article}

\usepackage{glossaries}

\newglossaryentry{culdesac}{name=cul-de-sac,description={passage
or street closed at one end},plural=culs-de-sac}

\begin{document}
I used to live in a \gls{culdesac}. \Gls{culdesac} is another name
for a no-through road.
\end{document}
```

This produces a document that contains the following sentences:

I used to live in a cul-de-sac. Cul-de-sac is another name for a no-through road.

You can replace glossaries with glossaries-extra in the above example. There will be no difference in the PDF, but there will be a difference if we make a slight modification:

```
\documentclass{article}
\usepackage{glossaries-extra}
\begin{document}
\newglossaryentry{culdesac}{name=cul-de-sac,description={passage
or street closed at one end},plural=culs-de-sac}

I used to live in a \gls{culdesac}. \Gls{culdesac} is another name
for a no-through road.
\end{document}
```


This now produces an error message:

```
! Package glossaries-extra Error: Glossary entries must be
(glossaries-extra)                defined in the preamble with
(glossaries-extra)                package option `docdef=false'.
```

This error doesn't occur with just glossaries, so what's gone wrong?

The glossaries package allows entries to be defined within the document environment, but, as mentioned above, there are problems associated with this. These issues are described in [section 4.10 of the user guide](#), but a lot of users prefer to learn from examples rather than reading the manual (which is understandable given how large it is, but this is necessary since it has to cover every available user command). This means that a lot of users don't realise the pitfalls of document definitions until things behave strangely and they ask for help.

Therefore, glossaries-extra by default disallows document definitions to raise user awareness. If you really need to define terms within your document, you can enable it, but first consider whether the need outweighs the drawbacks.

 Consider the `docdef` option a form of "I understand and accept the risks" check box.

First Letter Case-Changing

The commands that convert the first letter to upper case (such as `\Gls`) internally use `\makefirstuc` provided by the [mfirstuc](#)

[package](#). This tries to be intuitive in the event that the text to be modified starts with formatting commands, such as `\emph`, but there are cases where it fails, so take care if you use them.

Common pitfalls:

- The word starts with an accented or UTF-8 character but (PDF)LaTeX is being used. (Not a problem with XeLaTeX.) You need to enclose the character in braces. For example:

```
name={é}lite
```

or

```
name={\'e}lite
```

(More on this below.)

- The word starts with a command that has arguments, but the first argument doesn't contain the text to modify, then a suitable replacement needs to be found. For example, the following **won't work**:

```
name={\textcolor{red}{word}}
```

Instead you need to do, something like:

```
\newcommand*{\strong}[1]{\textcolor{red}{#1}}
```

and then use

```
name={\strong{word}}
```

Similarly for commands like `\foreignlanguage`.

- The word starts with a math-shift. Note that mathematical content shouldn't be capitalised at the start of a sentence. Lower case and upper case variables usually have different meanings. Insert an empty brace at the start to prevent any modification if you happen to use commands like `\Gls`. For example:

```
name={{}}$x$
```

Similarly for any other content that shouldn't be altered. For example:

```
name={{}}\si{\ohm}
```

- The word starts with a command that takes a label, such as `\gls` or `\cite`. The text that requires modifying can't be obtained, and the label will end up with the case-change, which will prevent it from being recognised.

✗ In general, don't use commands like `\gls` inside the value of any key that may result in nested use of `\gls` (or similar commands).

Plural Forms

The plural of a term can be obtained using

```
\glspl{label}
```

The default behaviour is to append the letter "s" to the name to create the plural. In the above example, that would produce "cul-de-sacs" which is incorrect. The correct plural is "culs-de-sac". To fix this, I need to modify the definition:

```
\newglossaryentry{culdesac}{name=cul-de-sac,description={passage  
or street closed at one end},plural=culs-de-sac}
```

As with the singular, I can also make the first letter of the plural form upper case:

```
\Glspl{culdesac}
```

or I can make the entire plural form appear in capitals:

```
\GLSpl{culdesac}
```

Alternative plurals or other grammatical constructs

There are six keys provided that you can use to store additional information, such as alternative plurals or other constructs. For example, you could store alternative plurals in `user1` and gerunds in `user2`:

```
\newglossaryentry{cow}{name=cow,
  description={a fully grown female of any bovine animal (plural cows, archaic plural kine)},
  user1={kine}}

\newglossaryentry{low}{name=low,
  description={long, deep sound made by a cow},
  user2={lowing}}
```

You can access the value of the user keys using `\glsuseri{label}` and `\glsuserii{label}`, but it's better to define easy to remember synonyms. For example:

```
\let\glsaltpl\glsuseri
\let\glsing\glsuserii
```

You can now use `\glsaltpl` to produce the alternative plural and `\glsing` to produce the gerund:

```
\documentclass{article}

\usepackage{glossaries}

\newglossaryentry{cow}{name=cow,
  description={a fully grown female of any bovine animal (plural
cows, archaic plural kine)},
  user1={kine}}

\newglossaryentry{low}{name=low,
  description={long, deep sound made by a cow},
  user2={lowing}}

\let\glsaltpl\glsuseri
\let\glsing\glsuserii

\begin{document}
The \glsing{low} was so loud it could be heard a mile away from the
\glsaltpl{cow}.

\end{document}
```

Another possibility is to add your own custom key using `\glsaddkey`, but that's too complicated for this introductory article. See [Additional Keys](#) for further details.

The sort key

[Later on](#) in this article, I will cover sorting the entries using an indexing application, but when you define an entry, you need to bear in mind that the indexing application won't know how to sort terms that contain LaTeX commands. Consider the following example:

```
\newglossaryentry{pi}{name={\ensuremath{\pi}},description={ratio
of circumference of circle to its diameter}}
```

This entry should ideally belong to the “p” group (that is, the group of terms whose first letter is a “p”) however neither `makeindex` nor `xindy` will be able to sort this entry correctly as they can't interpret LaTeX code. Therefore, we need to tell the indexing application that this entry should be sorted according to “pi”:

```
\newglossaryentry{pi}{name={\ensuremath{\pi}},sort=pi,
description={ratio of circumference of circle to its diameter}}
```

Note that in the above example I have used `\ensuremath` instead of using `$` to switch in and out of math mode. This means that I don't need to keep track of what mode I'm in when using the entry, which is convenient for the auto-generated glossary. For example:


```
The ratio of the circumference of a circle to its diameter is given by \gls{pi}:
\begin{equation}
\gls{pi} = \frac{C}{d}
```

```
\end{equation}
```

For those who prefer not to use `\ensuremath`, the alternative is

```
\newglossaryentry{pi}{%
  name={\pi},
  text={\pi},
  sort=pi,
  description={ratio of circumference of circle to its diameter}}
```

This uses the `text` key. The commands `\gls`, `\Gls` and `\GLS` are all actually using the value of the `text` key. The value of the `name` key is used in the glossary, but if the `text` key isn't supplied, it's assumed to be the same as the name, which is why I haven't used it in any of the other examples so far.

 If the `name` key contains commands, always supply the `sort` key if you want an alphabetically ordered glossary.

Alternatively, you can use the `glossaries-extra` package with the `symbols` option and use `\glsxtrnewsymbol` (described later), which sets the `sort` key to the label by default. For example:

```
\glsxtrnewsymbol{pi}{%
  name={\pi},
  text={\pi},
  description={ratio of circumference of circle to its diameter}}
```

(This will add the entry to the “symbols” glossary, rather than the default “main” glossary.)

Accents and non-Latin characters

`Makeindex` is hard-coded for the basic Latin set of characters (A,a-Z,z), so you will need to use the `sort` key if the entry's name contains accented or non-Latin characters. For example:

```
\newglossaryentry{attache}{name=attach'e,sort=attache,
description={person with special diplomatic responsibilities}}
```

`Xindy` was designed to overcome `makeindex`'s inflexibility and has rules for various (but not all) languages, so if you are using `xindy` with the above example, you don't need the `sort` key with the `inputenc` package (or `XeLaTeX`):

```
\newglossaryentry{attache}{name=attaché,
description={person with special diplomatic responsibilities}}
```

(Remember that the label can't include characters outside the basic Latin set, unless you're using `XeLaTeX`.)

As mentioned earlier, care must be taken with regular (PDF)`LaTeX` when the first character has an accent or is a non-Latin character, regardless of whether you use `makeindex` or `xindy`. This character must be enclosed in braces if you want to use it with `\Gls` or `\Glspl`. For example

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

or if you are using the `inputenc` package:

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

Remember that if you are using `makeindex`, you also need to use the `sort` key:

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

or

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

Defining an entry with both a name and a symbol

It is also possible to define an entry that has both a name and a symbol. For example:

```
\newglossaryentry{ohm}{name=ohm,symbol={\ensuremath{\Omega}},
description=unit of electrical resistance}
```

Or using the [siunitx](#) package:

```
\newglossaryentry{ohm}{name=ohm,symbol={\si{\ohm}},
description=unit of electrical resistance}
```

I can reference the name as usual:

```
\gls{ohm}
```

or I can reference the symbol using:

```
\glssymbol{ohm}
```

Categories

The glossaries-extra package provides an extra key called `category`, which isn't available with the basic glossaries package. This is set to `general` by default when you use `\newglossaryentry`. Take care not to confuse this with the `type` key, which separates entries into different glossaries (described later) or the `parent` key (used for hierarchical entries). Entries within the same glossary may have different categories.

The `category` key provides a way of making some minor adjustments to the way particular entries are displayed. This is much harder to achieve with the basic glossaries package. One of these types of adjustments is the post-link hook. This is used after all the commands like `\gls` and `\glssymbol` (outside of the hyperlink, if you're using `hyperref`).

Consider first the following example:

```
\documentclass{article}

\usepackage{siunitx}
\usepackage{glossaries-extra}

\newglossaryentry{resistor}{name={resistor},
description={component that implements electrical resistance}}

\newglossaryentry{ohm}{name=ohm,symbol={\si{\ohm}},
description=unit of electrical resistance}

\begin{document}

A \gls{resistor} is an electrical component.
The \gls{ohm} is unit of resistance.

\end{document}
```

This produces the text:

A resistor is an electrical component. The ohm is unit of resistance.

So far this is no different from the basic glossaries package. Neither entry has been explicitly assigned a category, so the category has been set to `general`. I can define `\glstrpostlinkgeneral` to automatically insert extra material after each use of commands like `\gls`, but only for entries with the category set to `general`. For example:

```
\documentclass{article}

\usepackage{siunitx}
\usepackage{glossaries-extra}

\newcommand*{\glstrpostlinkgeneral}{(foo)}

\newglossaryentry{resistor}{name={resistor},
description={component that implements electrical resistance}}

\newglossaryentry{ohm}{name=ohm,symbol={\si{\ohm}},
description=unit of electrical resistance}
```



```

\begin{document}

A \gls{resistor} is an electrical component.
The \gls{ohm} is a unit of resistance.

\end{document}

```

This now produces the text:

A resistor(foo) is an electrical component. The ohm(foo) is a unit of resistance.

This example is, of course, a little silly. It would be more practical to, say, access one of the fields. Suppose I want to insert the description instead. I can reference an entry's description using:

```
\glsentrydesc{label}
```

For example `\glsentrydesc{resistor}`. The post-link hook can access the label of the last referenced entry using `\glslabel`. So here's the updated example:

```

\documentclass{article}

\usepackage{siunitx}
\usepackage{glossaries-extra}

\newcommand*{\glsxtrpostlinkgeneral}{ (\glsentrydesc{\glslabel})}

\newglossaryentry{resistor}{name={resistor},
  description={component that implements electrical resistance}}

\newglossaryentry{ohm}{name=ohm,symbol={\si{\ohm}},
  description=unit of electrical resistance}

\begin{document}

A \gls{resistor} is an electrical component.
The \gls{ohm} is a unit of resistance.

\end{document}

```

This now produces the text:

A resistor (component that implements electrical resistance) is an electrical component. The ohm (unit of electrical resistance) is a unit of resistance.

Now let's suppose I want to designate the ohm entry as a unit of measurement. I can use the `category` key. The value is just another form of label that you can make up, but (as with all the labelling systems) it mustn't contain any special characters. Remember that this includes Unicode characters, unless you're using XeLaTeX.

```

\documentclass{article}

\usepackage{siunitx}
\usepackage{glossaries-extra}

\newcommand*{\glsxtrpostlinkgeneral}{ (\glsentrydesc{\glslabel})}

\newglossaryentry{resistor}{name={resistor},
  description={component that implements electrical resistance}}

\newglossaryentry{ohm}{name=ohm,symbol={\si{\ohm}},
  category=unit,% category label
  description=unit of electrical resistance}

\begin{document}

A \gls{resistor} is an electrical component.
The \gls{ohm} is a unit of resistance.

\end{document}

```

This produces the text:

A resistor (component that implements electrical resistance) is an electrical component. The ohm is a unit of resistance.

Only the resistor entry has the post-link text because that has the category set to general. I can change this around so that only the unit category has some post-link text, and I'm now going to use the value of the `symbol` field instead of the description:

```
\documentclass{article}

\usepackage{siunitx}
\usepackage{glossaries-extra}

\newcommand*{\glsxtrpostlinkunit}{ (\glsentrysymbol{\glslabel})}% post-link for the "unit"

\newglossaryentry{resistor}{name={resistor},
  description={component that implements electrical resistance}}

\newglossaryentry{ohm}{name=ohm,symbol={\si{\ohm}},
  category=unit,% category label
  description=unit of electrical resistance}

\begin{document}
A \gls{resistor} is an electrical component.
The \gls{ohm} is a unit of resistance.

\end{document}
```

Note that I've used `\glsentrysymbol` here, instead of `\glsymbol`. The `\glsentry...` commands are less problematic in this context. This now produces the text:

A resistor is an electrical component. The ohm (Ω) is a unit of resistance.

A slight modification to the example document shows up some flaws:

```
\documentclass{article}

\usepackage{siunitx}
\usepackage{glossaries-extra}

\newcommand*{\glsxtrpostlinkunit}{ (\glsentrysymbol{\glslabel})}% post-link for the "unit"

\newglossaryentry{resistor}{name={resistor},
  description={component that implements electrical resistance}}

\newglossaryentry{ohm}{name=ohm,symbol={\si{\ohm}},
  category=unit,
  description=unit of electrical resistance}

\begin{document}

A \gls{resistor} is an electrical component.
The \gls{ohm} is a unit of resistance.

Symbol: \glsymbol{ohm} and let's mention \gls{ohm} again.

\end{document}
```

This produces the text:

*A resistor is an electrical component. The ohm (Ω) is a unit of resistance.
Symbol: Ω (Ω) and let's mention ohm (Ω) again.*

The `\glsymbol` command also uses the post-link hook, and it's a bit repetitive to insert the symbol on every use of `\gls`.

The first problem can be overcome by testing `\glscustomtext`. If it's empty, then `\gls` (or `\glspl`, `\Gls`, etc) has been used. It's not empty if one of the other commands, such as `\glsymbol`, has been used. The [etoolbox](#) package provides `\ifdefempty` and conveniently the glossaries (and therefore glossaries-extra) package loads etoolbox.

So, to only insert the symbol after the `\gls`-like commands:

```
\newcommand*\glsxtrpostlinkunit}{%
\ifdefempty\glscustomtext{ (\glsentrysymbol{\glslabel})}{}}
```

The other problem requires testing if the entry was just used for the first time. Each entry has a “first use” flag, which is unset by `\gls`, `\glspl`, `\Gls`, `\Glspl`, `\GLS`, `\GLSpl` and `\glsdisp`. If this flag is false (unset), it means the entry has already been referenced using one of those commands. (It’s been *used*.) If this flag is true (set), it means the entry hasn’t yet been referenced using one of those commands. (The next instance will be the *first use* of that entry.)

The glossaries package provides a command for testing this flag, but by the time the post-link hook occurs, the flag has already been unset. There’s another command specifically available for the post-link hook and that’s `\glsxtrifwasfirstuse{true}{false}`, so I could do:

```
\glsxtrifwasfirstuse{ (\glsentrysymbol{\glslabel})}{}
```

However, there’s a shortcut command that makes the code simpler:

```
\glsxtrpostlinkAddSymbolOnFirstUse
```

This will append the symbol (if defined) in parentheses on first use. So the modified example is now:

```
\documentclass{article}

\usepackage{siunitx}
\usepackage{glossaries-extra}

\newcommand*\glsxtrpostlinkunit){%
\ifdefempty\glscustomtext{\glsxtrpostlinkAddSymbolOnFirstUse}{}}

\newglossaryentry{resistor}{name={resistor},
description={component that implements electrical resistance}}

\newglossaryentry{ohm}{name=ohm,symbol={\si{\ohm}},
category=unit,% category label
description=unit of electrical resistance}

\begin{document}
A \gls{resistor} is an electrical component.
The \gls{ohm} is a unit of resistance.

Symbol: \glssymbol{ohm} and let’s mention \gls{ohm} again.

\end{document}
```

This now produces the text:

A resistor is an electrical component. The ohm (Ω) is a unit of resistance.
Symbol: Ω and let’s mention ohm again.

With just the basic glossaries package, without the category post-link hook, this example could be done using the `first` key, which is used by `\gls` on first use (and the `text` key is then used on subsequent use).

```
\documentclass{article}

\usepackage{siunitx}
\usepackage{glossaries}

\glsnoexpandfields

\newglossaryentry{resistor}{name={resistor},
description={component that implements electrical resistance}}

\newglossaryentry{ohm}{name=ohm,symbol={\si{\ohm}},
first={ohm (\si{\ohm})},
description=unit of electrical resistance}

\begin{document}
A \gls{resistor} is an electrical component.
The \gls{ohm} is a unit of resistance.

Symbol: \glssymbol{ohm} and let’s mention \gls{ohm} again.
```

```
\end{document}
```

This is less flexible and, if you include the `hyperref` package, the entire contents of the `first` key will be included in the hyperlink. Whereas the `glossaries-extra` example will have the parenthetical material (`\si{\ohm}`) outside the hyperlink.

Abbreviations

The `first` key mentioned above can also be used for abbreviations. For example:

```
\newglossaryentry{led}{name=LED,description={light-emitting diode},first={light-emitting diode (LED)}}
```

However, hard-coding the abbreviation style like this makes it harder to change the style. You might, for example, want to have the abbreviation first followed by the long form in parentheses. It's a bit of a nuisance having to edit all the first fields, especially if you have a lot of entries.

The `glossaries` package provides an easier way of doing this with

```
\newacronym[options]{label}{short}{long}
```

This internally uses `\newglossaryentry`. If the optional argument is present, those options are appended to the second argument of `\newglossaryentry`.

Here's an example document using the basic `glossaries` package:

```
\documentclass{article}

\usepackage{glossaries}

\setacronymstyle{long-short}

\newacronym{led}{LED}{light-emitting diode}

\begin{document}

First use: \gls{led}. Next use: \gls{led}.


\end{document}
```

This produces the text:

First use: light-emitting diode (LED). Next use: LED.

Now the first use format can be changed by changing the style. For example:

```
\setacronymstyle{short-long}
```

 This must be set before the abbreviations are defined.

Another style is `long-sc-short` which uses `\textsc` to typeset the short form. Remember that this command uses small capitals for *lowercase* letters, so the short form needs to be converted to lowercase. You can either do this manually, for example

```
\newacronym{led}{led}{light-emitting diode}
```

or you can redefine `\acronymfont` to do the conversion. For example:

```
\documentclass{article}

\usepackage[T1]{fontenc}
\usepackage{glossaries}

\setacronymstyle{long-sc-short}

\renewcommand*{\acronymfont}[1]{\textsc{\MakeLowercase{#1}}}

\newacronym{led}{LED}{light-emitting diode}
```

```
\begin{document}

First use: \gls{led}. Next use: \gls{led}.

\end{document}
```

(This redefinition must be done after the style is set.)

You can access the short and long forms using

```
\acrlong{label}
```

and

```
\acrshort{label}
```

(analogous to the `\glssymbol` command used earlier) or using

```
\glsentryshort{label}
```

and

```
\glsentrylong{label}
```

(analogous to the `\glsentrysymbol` command used earlier).

There are some drawbacks to the glossaries package's abbreviation management. One of which is that only one style can be selected for all the defined abbreviations. So you can only have, for example, long-short or only short-long, but not a mixture of some with long-short and others with short-long.

The glossaries-extra package provides a different abbreviation management system that allows mixed styles. The abbreviations are now defined using

```
\newabbreviation[options]{label}{short}{long}
```

This sets the category to `abbreviation` by default. The `\newacronym` command provided by the base glossaries package is changed to use this new interface, but the category is set to `acronym`. So

```
\newacronym[options]{label}{short}{long}
```

is now the same as

```
\newabbreviation[category=acronym,options]{label}{short}{long}
```

The style is set (before the abbreviations are defined) using

```
\setabbreviationstyle[category]{style}
```

If the optional argument is omitted, `abbreviation` is used, so

```
\setabbreviationstyle{long-short}
```

sets the `long-short` style for all abbreviations with the category set to `abbreviation` and

```
\setabbreviationstyle[acronym]{short-long}
```

sets the `short-long` style for all abbreviations with the category set to `acronym`.

For example:

```
\documentclass{article}

\usepackage{glossaries-extra}

\setabbreviationstyle{long-short}
```

```

\setabbreviationstyle[acronym]{short-long}

\newabbreviation{led}{LED}{light-emitting diode}
\newacronym{laser}{laser}{light amplification by stimulated
emission of radiation}

\begin{document}

First use: \gls{led}. Next use: \gls{led}.

First use: \gls{laser}. Next use: \gls{laser}.
\end{document}

```

This produces the text:

First use: light-emitting diode (LED). Next use: LED.
First use: laser (light amplification by stimulated emission of radiation). Next use: laser.

The abbreviation style names provided by glossaries-extra don't always have the same names as the closest equivalent acronym style provided by glossaries. The equivalent to the `long-sc-short` (glossaries) acronym style used earlier is the `long-short-sc` (glossaries-extra) abbreviation style, but now the font used by the short form is accessed through `\glstrscfont`. As in the earlier example, this can be redefined to automatically lowercase the short version if required:

```

\documentclass{article}

\usepackage[T1]{fontenc}
\usepackage{glossaries-extra}

\setabbreviationstyle{long-short-sc}
\setabbreviationstyle[acronym]{short-long}

\renewcommand*{\glstrscfont}[1]{\textsc{\MakeLowercase{#1}}}

\newabbreviation{led}{LED}{light-emitting diode}
\newacronym{laser}{laser}{light amplification by stimulated
emission of radiation}

\begin{document}

First use: \gls{led}. Next use: \gls{led}.

First use: \gls{laser}. Next use: \gls{laser}.
\end{document}

```

The short and long forms are now accessed using

```
\glstrshort{label}
```

and

```
\glstrlong{label}
```

(analogous to `\acrshort` and `\acrlong`). They can also still be accessed using `\glstryshort` and `\glstrylong`. If you want to use the short or long form in a chapter or section title or caption, use:

```
\glsfmshort{label}
```

and

```
\glsfmlong{label}
```

instead. These commands aren't available with the base glossaries package.

✗ Don't use any of the commands like `\gls`, `\acrshort` or `\acrlong` in chapter or section headings or captions.

Again, the warning about nested links still applies. Don't use commands like `\gls` within the long or short form as they can cause multiple problems. For example:

```

\documentclass{article}

\usepackage{glossaries}

\setacronymstyle{long-short}

\newacronym{ssi}{SSI}{server-side includes}
\newacronym{html}{HTML}{hypertext markup language}
\newacronym{shtml}{SHTML}{SSI enabled HTML}

\begin{document}

First use: \gls{ssi}, \gls{html} and \gls{shtml}.

Next use: \gls{ssi}, \gls{html} and \gls{shtml}.

\end{document}

```

If you want the same font as the short forms (for example, you're using the small-caps style), then you can incorporate the font changing command, for example, with just glossaries:

```

\documentclass{article}

\usepackage[T1]{fontenc}
\usepackage{glossaries}

\setacronymstyle{long-sc-short}

\renewcommand*{\acronymfont}[1]{\textsc{\MakeLowercase{#1}}}

\newacronym{ssi}{SSI}{server-side includes}
\newacronym{html}{HTML}{hypertext markup language}
\newacronym{shtml}{SHTML}{\acronymfont{SSI} enabled \acronymfont{HTML}}

\begin{document}

First use: \gls{ssi}, \gls{html} and \gls{shtml}.

Next use: \gls{ssi}, \gls{html} and \gls{shtml}.

\end{document}

```

Or with glossaries-extra:

```

\documentclass{article}

\usepackage[T1]{fontenc}
\usepackage{glossaries-extra}

\setabbreviationstyle{long-short-sc}

\renewcommand*{\glxtrscfont}[1]{\textsc{\MakeLowercase{#1}}}

\newabbreviation{ssi}{SSI}{server-side includes}
\newabbreviation{html}{HTML}{hypertext markup language}
\newabbreviation{shtml}{SHTML}{\glxtrscfont{SSI} enabled \glxtrscfont{HTML}}

\begin{document}

First use: \gls{ssi}, \gls{html} and \gls{shtml}.

Next use: \gls{ssi}, \gls{html} and \gls{shtml}.

\end{document}

```


The glossaries-extra package performs some patches that overcomes some, but not all, of the problems with nested entry references, so with glossaries-extra, if you really need to, you may be able to use \glxtrshort, for example:

```

\newabbreviation{shtml}{SHTML}{{}\glxtrshort{ssi} enabled \glxtrshort{html}}

```

The initial empty group is protection against any use of the first letter uppercasing commands, such as \Gls. You can't do the equivalent with just the basic glossaries package.

 The acronym styles provided by the base glossaries package aren't the same as the abbreviation styles provided by the glossaries-extra package so, although you can use `\newacronym` with glossaries-extra (as above), you can't use `\setacronymstyle`, but must use `\setabbreviationstyle[acronym]` instead.

The available acronym styles provided by glossaries are described in [Predefined Acronym Styles](#) and the available abbreviation styles provided by the glossaries-extra package are described in [Predefined Abbreviation Styles](#). In some cases, the styles share the same name (such as `long-short`), but they are implemented differently.

Both the basic glossaries package and the glossaries-extra package provide shorter synonyms for commands. In both packages they can be enabled using the `shortcuts` package option. For example:

```
\documentclass{article}

\usepackage[shortcuts]{glossaries}

\setacronymstyle{long-short}

\newacronym{led}{LED}{light-emitting diode}
\newacronym{laser}{laser}{light amplification by stimulated
emission of radiation}

\begin{document}
First use: \ac{led}. Next use: \ac{led}.

First use: \ac{laser}. Next use: \ac{laser}.

Long form: \acl{led}. Short form: \acs{led}.

\end{document}
```

Or

```
\documentclass{article}

\usepackage[shortcuts=abbr]{glossaries-extra}

\setabbreviationstyle{long-short}
\setabbreviationstyle[acronym]{short-long}

\newabbr{led}{LED}{light-emitting diode}
\newacronym{laser}{laser}{light amplification by stimulated
emission of radiation}

\begin{document}
First use: \ab{led}. Next use: \ab{led}.

First use: \ab{laser}. Next use: \ab{laser}.

Long form: \al{led}. Short form: \as{led}.

\end{document}
```

Other related commands are described in the [glossaries manual](#) and the [glossaries-extra manual](#).

Multiple Glossaries

You may have as many glossaries, lists of symbols or lists of abbreviations as you like in your document. Each glossary has an associated label that can be used to reference it. The default is to just have one glossary (labelled `main`). When you define an entry, you can specify which glossary it belongs to using the `type` key in the second argument of `\newglossaryentry` (or the optional argument for commands like `\newacronym` or `\newabbreviation`). If omitted, `\newglossaryentry` assumes `type=\glsdefaulttype` (where `\glsdefaulttype` is initially set to `main`).

You can add a list of acronyms using the `acronym` package option (for glossaries):

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

(You can also use `acronym` instead of `acronyms`.)

Any acronyms defined using `\newacronym` will automatically be added to the list of acronyms rather than the `main` glossary if the `acronyms` (or `acronym`) package option is used.

If you use glossaries-extra, then the `abbreviations` package option works in a similar manner:


```
\usepackage[abbreviations]{glossaries-extra}
```

Any abbreviations defined using `\newabbreviation` (or `\newacronym`) will automatically be added to the list of abbreviations.

There are some other options for common types of lists: `symbols`, `numbers` and `index`. These options are available for both `glossaries` and `glossaries-extra`, but with `glossaries-extra`, the `symbols` option will additionally define

```
\glstrnewsymbol[options]{label}{symbol}
```

which is equivalent to

```
\newglossaryentry{label}{name={symbol},
sort={label},type=symbols,category=symbol,options}
```

The `numbers` package option (with `glossaries-extra`) will also define:

```
\glstrnewnumber[options]{number}
```

which is equivalent to

```
\newglossaryentry{label}{name={number},
sort={label},type=numbers,category=number,options}
```

If you need an additional glossary, for example a list of units, you need to define it using:

```
\newglossary[log-ext]{glossary-label}{in-ext}{out-ext}{title}
```

The *glossary-label* is a label that uniquely identifies this glossary. Again, it is best to use non-active characters such as alphanumerics (a-z,A-Z,0-9) in the label. The title is used as the section or chapter header for that glossary.

The remaining arguments specify the extensions for the files that contain the glossary information. The *in-ext* argument indicates the extension of the file that is created by the indexing application. This file is read in by the `glossaries` package's `\printglossaries` command, but obviously it can only be read once it has been created by the indexing application. (This is covered in more detail below.) The *out-ext* argument indicates the extension of the file that is read in by the indexing application and written out by the `glossaries` package. The first optional argument *log-ext* is the extension for the indexing transcript file.

Take care that you don't use any extensions that might already be in use. There's a starred version:

```
\newglossary*{glossary-label}{title}
```

which sets the extensions to *glossary-label-glg*, *glossary-label-gls* and *glossary-label-glo* which makes it more convenient for removing these temporary files if you are able to use wildcards. For example, in Unix-like operating systems

```
rm *.--{glg,gls,glo}
```

will remove all the temporary glossary input and output files.

Each glossary creates associated resources, so if you don't want the `main` glossary, you can prevent these unnecessary resources from being used with the `nomain` package option, but you must provide an alternative glossary either implicitly (through packages options like `symbols` or `acronyms/abbreviations`) or explicitly (using `\newglossary`).

❌ Don't use the `nomain` package option if you need the `main` glossary.

This may sound obvious, but from time to time users copy and paste example code without realising what the package options do. If you try to define any entries when there are no defined glossaries, you'll get the following error message:

```
No default glossary type
```

📖 There is a second optional argument to `\newglossary` that isn't covered here. See the section [Defining New Glossaries](#) in the manual for further details.

For example, suppose I want to create a list of units. I can define a glossary labelled “units” as follows:

```
\newglossary[glg-u]{units}{gls-u}{glo-u}{List of Units}
```

(the file extensions are .glg-u, .gls-u and .glo-u) or just

```
\newglossary*{units}{List of Units}
```

(the file extensions are .units-glg, .units-gls and .units-glo).

When I define an entry to go in this glossary, I need to use the `type` key to specify the glossary label. For example:

```
\newglossaryentry{ohm}{name=ohm,
  symbol={\si{\ohm}},
  type=units,
  description={unit of electrical resistance}}
```

Remember that the `type` and `category` keys are independent of each other, so I might also do (with `glossaries-extra`):

```
\newglossaryentry{ohm}{name=ohm,
  symbol={\si{\ohm}},
  type=units,
  category=unit,
  description={unit of electrical resistance}}
```

As mentioned earlier, if the `type` key is omitted, `\glsdefaulttype` (the default glossary) is assumed. If you load entries from a file using `\loadglsentries`, you can assign all the entries to a particular glossary using the optional argument, but you must set the `type` to `\glsdefaulttype` if you are using commands like `\newacronym` or `\newabbreviation`.

For example, if you have a file called `myabbrevs.tex`:


```
\newacronym[type=\glsdefaulttype]{led}{LED}{light-emitting diode}
```

Then you can put the following, for example, in the preamble:

```
\newglossary*{components}{Components}

\makeglossaries

\loadglsentries[components]{myabbrevs}
```

 Note the use of `\makeglossaries`, which must come after `\newglossary`, but should come *before* any entry definitions. This is discussed next.

Using `makeindex` or `xindy`

So far I have only covered how to define a term, symbol or abbreviation and how to use it in the document, but it is likely that you will also want a sorted list of these entries. This is where things start to get a bit complicated and is often a source of confusion for beginners.

The first thing that you must do is add the command

```
\makeglossaries
```

to your preamble. (This needs to come after any occurrence of `\newglossary`.)

The next step is to put the command

```
\printglossaries
```

where you want your glossaries, lists of symbols or abbreviations to appear. This will typically be either in your front matter (with the table of contents, list of figures etc) or back matter (with the bibliography etc).

If you want to rearrange your glossaries or have some in the front matter and some in the back matter, you can use

```
\printglossary[options]
```


The optional argument can include the `type` key to indicate the glossary to display. The `acronym` package option (provided by `glossaries`) defines a shortcut command:

```
\printacronyms[options]
```

which sets the type to the acronym glossary, and the abbreviations package option (provided by glossaries-extra) defines a shortcut command

```
\printabbreviations[options]
```

which sets the type to the abbreviations glossary.

 Further details about the available options are listed in the section [Displaying a glossary](#) in the glossaries manual.

For example (using glossaries):

```
\documentclass{article}

\usepackage{siunitx}
% Load hyperref before glossaries
\usepackage[colorlinks]{hyperref}
\usepackage[acronym,symbols]{glossaries}

\makeglossaries

% The following definitions will go in the main glossary

\newglossaryentry{culdesac}{name=cul-de-sac,description={passage
or street closed at one end},plural=culs-de-sac}

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

\newglossaryentry{elitism}{name={\`e}litism,description={advocacy
of dominance by an \gls{elite}},sort=elitism}

\newglossaryentry{attache}{name=attach\`e,
description={person with special diplomatic responsibilities}}

% The following definitions will go in the list of acronyms

\newacronym{led}{LED}{light-emitting diode}

\newacronym{eeprom}{EEPROM}{electrically erasable programmable
read-only memory}

% The following definitions will go in the list of symbols

\newglossaryentry{ohm}{type=symbols,name=ohm,
symbol={\si{\ohm}},
description=unit of electrical resistance}

\newglossaryentry{angstrom}{type=symbols,name={\aa}ngstr"om,
symbol={\si{\angstrom}},sort=angstrom,
description={non-SI unit of length}}

\begin{document}
\tableofcontents

\section{Diplomatic Memoirs}

When I was an \gls{attache}, I lived in a \gls{culdesac}, but
I didn't much care for it as I found there was a fair amount
of \gls{elitism} amongst my neighbours.

\section{Student Memoirs}

When I was a student I often left bits of electronic circuitry
in my pockets, such as \glspl{led} and \glspl{eeprom}, which
often ended up in the washing machine. The \glspl{led} didn't
fair too badly, but the \glspl{eeprom} frequently broke.

\section{Symbols}

The \gls{angstrom} is commonly used in structural biology,
whereas the \gls{ohm} is used in electronics.
```

```
\printglossaries

\end{document}
```

This produces the document shown below on the first LaTeX run:

Contents

1 Diplomatic Memoirs

When I was an **attaché**, I lived in a **cul-de-sac**, but I didn't much care for it as I found there was a fair amount of **elitism** amongst my neighbours.

2 Student Memoirs

When I was a student I often left bits of electronic circuitry in my pockets, such as **light-emitting diodes (LEDs)** and **electrically erasable programmable read-only memorys (EEPROMs)**, which often ended up in the washing machine. The **LEDs** didn't fair too badly, but the **EEPROMs** frequently broke.

3 Symbols

The **ångström** is commonly used in structural biology, whereas the **ohm** is used in electronics.

There are no glossaries here, and this often confuses new users as they're expecting the glossaries at the place where `\printglossaries` occurs.

Now let's look what happens if we change this example to use `glossaries-extra` instead:

```
\documentclass{article}

\usepackage{siunitx}

% Load hyperref before glossaries
\usepackage[colorlinks]{hyperref}
\usepackage[abbreviations,symbols]{glossaries-extra}

\makeglossaries

% The following definitions will go in the main glossary

\newglossaryentry{culdesac}{name=cul-de-sac,description={passage
or street closed at one end},plural=culs-de-sac}

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

\newglossaryentry{elitism}{name={\e}litism,description={advocacy
of dominance by an \gls{elite}},sort=elitism}

\newglossaryentry{attache}{name=attach\e,
description={person with special diplomatic responsibilities}}

% The following definitions will go in the list of acronyms
\newabbreviation{led}{LED}{light-emitting diode}

\newabbreviation{eeprom}{EEPROM}{electrically erasable programmable
read-only memory}

% The following definitions will go in the list of symbols

\newglossaryentry{ohm}{type=symbols,name=ohm,
symbol={\si{\ohm}},
description=unit of electrical resistance}

\newglossaryentry{angstrom}{type=symbols,name={\aa}ngstr"om,
symbol={\si{\angstrom}},sort=angstrom,
```

```

description={non-SI unit of length}}

\begin{document}
\tableofcontents

\section{Diplomatic Memoirs}

When I was an \gls{attache}, I lived in a \gls{culdesac}, but
I didn't much care for it as I found there was a fair amount
of \gls{elitism} amongst my neighbours.

\section{Student Memoirs}

When I was a student I often left bits of electronic circuitry
in my pockets, such as \glspl{led} and \glspl{eeprom}, which
often ended up in the washing machine. The \glspl{led} didn't
fair too badly, but the \glspl{eeprom} frequently broke.

\section{Symbols}

The \gls{angstrom} is commonly used in structural biology,
whereas the \gls{ohm} is used in electronics.

\printglossaries

\end{document}

```

The only difference is that I've replaced

```
\usepackage[acronyms,symbols]{glossaries}
```

with

```
\usepackage[abbreviations,symbols]{glossaries-extra}
```

and `\newacronym` with `\newabbreviation`.

This version produces a two page document on the first LaTeX run, but it doesn't look quite how you might expect.

The first page is shown below:

Contents

1 Diplomatic Memoirs

When I was an **attache**, I lived in a **cul-de-sac**, but I didn't much care for it as I found there was a fair amount of **elitism** amongst my neighbours.

2 Student Memoirs

When I was a student I often left bits of electronic circuitry in my pockets, such as **LEDs** and **EEPROMs**, which often ended up in the washing machine. The **LEDs** didn't fair too badly, but the **EEPROMs** frequently broke.

3 Symbols

The **ångström** is commonly used in structural biology, whereas the **ohm** is used in electronics.

Glossary

This document is incomplete. The external file associated with the glossary 'main' (which should be called `sample2-xtr.gls`) hasn't been created.

Check the contents of the file `sample2-xtr.gls`. If it's empty, that means you haven't indexed any of your entries in this glossary (using commands like `\gls` or `\glsadd`) so this list can't be generated. If the file isn't empty, the document build process hasn't been completed.

If you don't want this glossary, add `nomain` to your package option list when you load `glossaries-extra.sty`. For example:

```
\usepackage[nomain]{glossaries-extra}
```

Try one of the following:

- Add `automake` to your package option list when you load `glossaries-extra.sty`. For example:

```
\usepackage[automake]{glossaries-extra}
```

- Run the external (Lua) application:
`makeglossaries-lite "sample2-xtr"`
- Run the external (Perl) application:
`makeglossaries "sample2-xtr"`

Then rerun `LATEX` on this document.
This message will be removed once the problem has been fixed.

and the second page is as follows:

Symbols

This document is incomplete. The external file associated with the glossary ‘symbols’ (which should be called `sample2-xtr.sls`) hasn’t been created.

Check the contents of the file `sample2-xtr.sls`. If it’s empty, that means you haven’t indexed any of your entries in this glossary (using commands like `\gls` or `\glsadd`) so this list can’t be generated. If the file isn’t empty, the document build process hasn’t been completed.

Try one of the following:

- Add `automake` to your package option list when you load `glossaries-extra.sty`.
For example:

```
\usepackage[automake]{glossaries-extra}
```
- Run the external (Lua) application:

```
makeglossaries-lite "sample2-xtr"
```
- Run the external (Perl) application:

```
makeglossaries "sample2-xtr"
```

Then rerun \LaTeX on this document.

This message will be removed once the problem has been fixed.

Abbreviations

This document is incomplete. The external file associated with the glossary ‘abbreviations’ (which should be called `sample2-xtr.gls-abr`) hasn’t been created.

Check the contents of the file `sample2-xtr.gls-abr`. If it’s empty, that means you haven’t indexed any of your entries in this glossary (using commands like `\gls` or `\glsadd`) so this list can’t be generated. If the file isn’t empty, the document build process hasn’t been completed.

Try one of the following:

- Add `automake` to your package option list when you load `glossaries-extra.sty`.
For example:

```
\usepackage[automake]{glossaries-extra}
```
- Run the external (Lua) application:

```
makeglossaries-lite "sample2-xtr"
```
- Run the external (Perl) application:

```
makeglossaries "sample2-xtr"
```

Then rerun \LaTeX on this document.

This message will be removed once the problem has been fixed.

The first part of the document looks the same as the previous example, but information text has appeared at the point where my document contains `\printglossaries`. The text is as follows (the file was called `sample2-xtr.tex`):

Glossary

This document is incomplete. The external file associated with the glossary ‘main’ (which should be called `sample2-xtr.gls`) hasn’t been created. Check the contents of the file `sample2-xtr.gls`. If it’s empty, that means you haven’t indexed any of your entries in this glossary (using commands like `\gls` or `\glsadd`) so this list can’t be generated. If the file isn’t empty, the document build process hasn’t been completed.

If you don’t want this glossary, add `nomain` to your package option list when you load `glossaries-extra.sty`. For example:

```
\usepackage[nomain]{glossaries-extra}
```

Try one of the following:

- *Add `automake` to your package option list when you load `glossaries-extra.sty`. For example:*

```
\usepackage[automake]{glossaries-extra}
```

- *Run the external (Lua) application:*

```
makeglossaries-lite "sample2-xtr"
```

- *Run the external (Perl) application:*

```
makeglossaries "sample2-xtr"
```

Then rerun \LaTeX on this document.

This message will be removed once the problem has been fixed.

The second page continues with similar text for the list of symbols and list of abbreviations. This text is an indicator that the document build process hasn't been completed. (Just as ?? indicates a problem with a cross-reference and [?] indicates a problem with a citation.) Once the build process has been completed successfully, this text will be replaced with the appropriate glossary.

The problem is the same for both the glossaries and the glossaries-extra documents, but the glossaries-extra package tries to be helpful with some explanatory text about what has gone wrong (a necessary file is missing) and possible ways of fixing the problem.

The missing file (sample2-xtr.gls in this case) needs to be created by an indexing application. If you've used bibtex or biber, it's a similar process. In this case, I need to use makeindex. If I add xindy to the package option list

```
\usepackage[acronyms,symbols,xindy]{glossaries}
```


or

```
\usepackage[abbreviations,symbols,xindy]{glossaries-extra}
```

then I need to run xindy instead of makeindex.

This can be a bit fiddly, as there are a number of command line switches that need to be set. Instead, you can use the makeglossaries Perl script. This reads the auxiliary file generated by LaTeX (sample2-xtr.aux in the above example) and works out whether to call makeindex or xindy and what settings to use. It also performs some diagnostics so that it can provide more informative messages if things go wrong. The “Using makeglossaries” section below, describes how to run this script.

If you don't want to install Perl, there's a light-weight Lua alternative called makeglossaries-lite.lua (if it's not installed, your version of glossaries may be too old). This is run in a similar manner to makeglossaries, so just follow the instructions below for using makeglossaries and replace “makeglossaries” with “makeglossaries-lite.lua”.

 Be careful not to confuse \makeglossaries (the command used in your LaTeX document) with makeglossaries (an external tool).

If you're still stuck, you can use the automake package option:

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

or

```
\usepackage[automake]{glossaries-extra}
```

This will only work if you have the shell escape enabled. If the option isn't recognised, your version of glossaries is too old.

The automake option will try to run makeindex or xindy using TeX's shell escape (\write18) mechanism. Since this can be a security risk, this function may be disabled or it may be running in restricted mode which will only allow trusted applications to run. Check your log file to determine which mode is in use. For example, if the log file contains

```
restricted \write18 enabled
```

then the restricted mode is on. Since makeindex is a trusted application, the automake option should work in restricted mode.

If you're still stuck, you can make TeX sort and collate the entries, but this is much slower (especially if there are a large number of entries). If you want to use this method, replace

```
\makeglossaries
```

with

```
\makenoidxglossaries
```

and replace

```
\printglossaries
```

with

```
\printnoidxglossaries
```

or replace

```
\printglossary
```

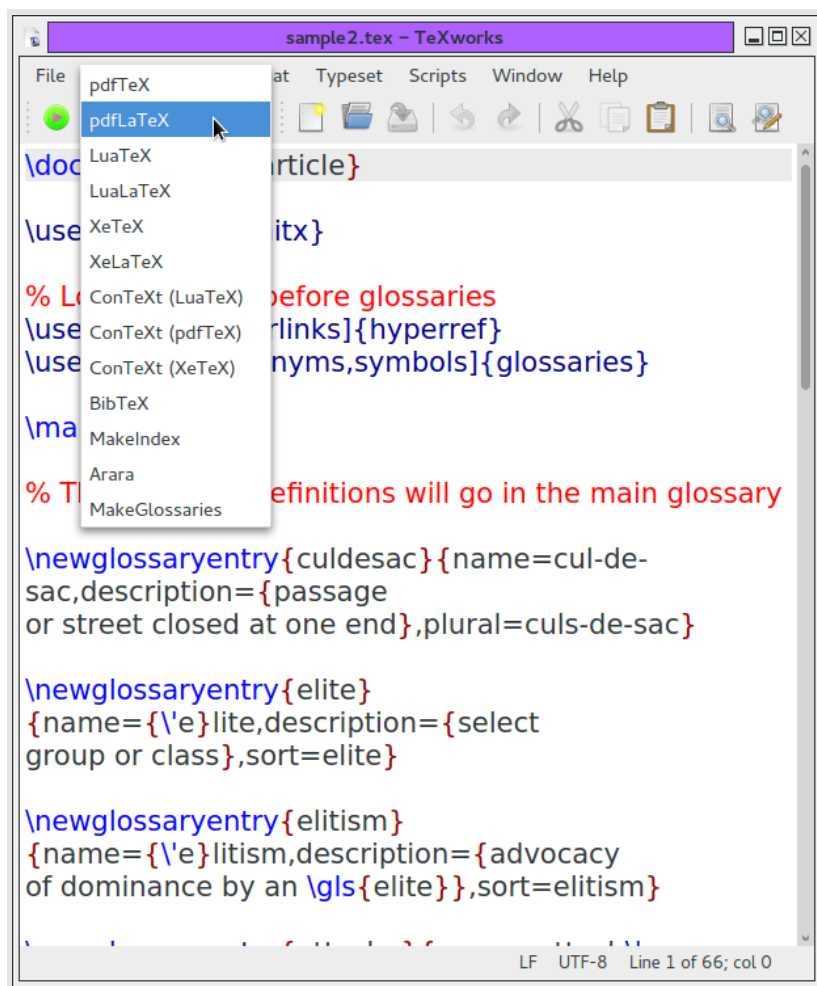
with

```
\printnoidxglossary
```

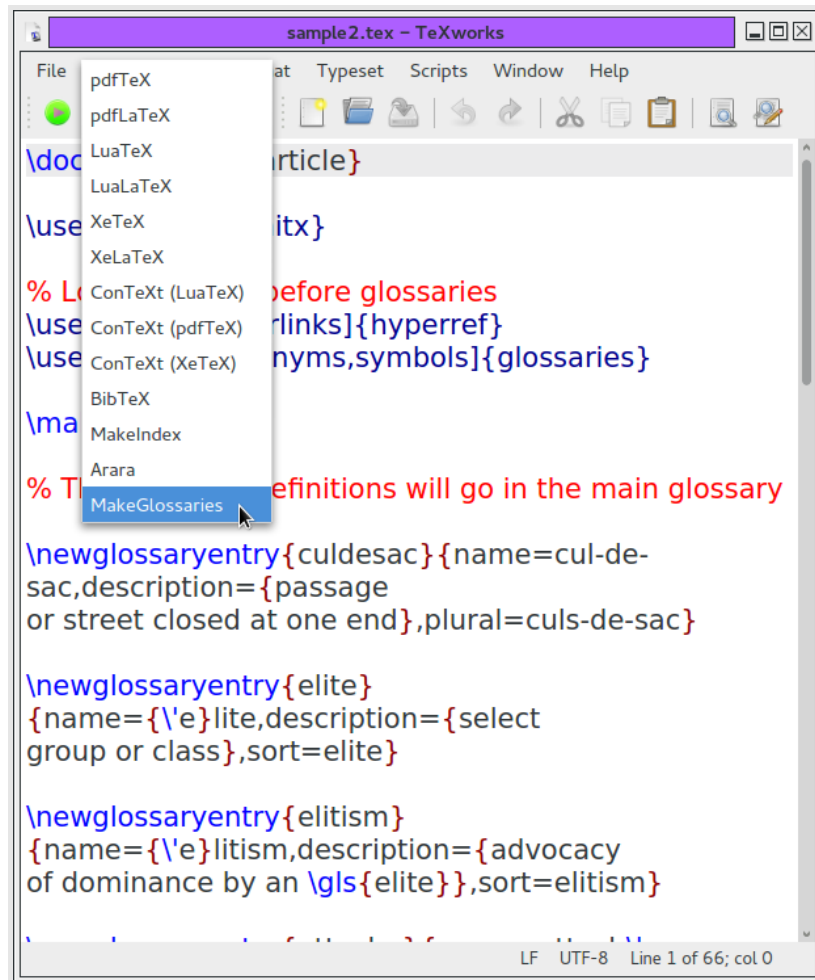
 The pros and cons of using TeX vs makeindex vs xindy are discussed in the [Introduction](#) of the glossaries user manual.

Unless you use a command prompt or terminal, you will probably need to add a tool to your text editor to allow you to run makeglossaries (or makeindex etc). There is some advice for common editors below, but if you're finding this a bit troublesome try out the automake option.

Here's my example document showing in TeXworks:



1. I've used the dropdown menu to select PDFLaTeX. Then I clicked on the green "Typeset" button next to it. This creates a PDF file, but it's missing the glossary.
2. Next I used the dropdown menu to select the MakeGlossaries tool. I've already added this tool using the instructions below.



- Once this MakeGlossaries tool is selected, I need to click on the green “Typeset” button next to it. This doesn’t make any changes to the PDF. I need to select the PDFLaTeX tool again and once again click on the green “Typeset” button. The updated PDF now contains all the glossaries, but the log file has the warning:

```
pdfTeX warning (dest): name{glo:elite} has been referenced but does not exist, replaced by
```

This is because `\gls{elite}` has only been used in the glossary (within the description of `elitism`). The process needs to be repeated from step 2 above.

(If you’re also creating a bibliography and index, all these steps can be a nuisance, so you may want to consider using an automated system, such as `arara` or `latexmk`. See, for example, [Building Your Document](#).)

Once the document is complete, the main glossary and list of abbreviations at the end of the first page appear as follows:

Glossary

attaché person with special diplomatic responsibilities. 1

cul-de-sac passage or street closed at one end. 1

élite select group or class. 1

elitism advocacy of dominance by an **élite**. 1

Acronyms

EEPROM electrically erasable programmable read-only memory. 1

LED light-emitting diode. 1

and the list of symbols is on the second page:

Symbols

ångström non-SI unit of length. [1](#)

ohm unit of electrical resistance. [1](#)

The default glossary style is to use a description environment with the entry names in the optional argument of `\item` (which is why they appear bold). The number following the description is the page number where the entry was referenced. (It's displayed in red because it's a hyperlink.)

The glossaries package provides 100 different styles (most of them just minor variations on a theme). The style may be set using

```
\setglossarystyle{style-name}
```

(before `\printglossaries`) or through the `style` key in the optional argument of `\printglossary`. You can view samples of all the predefined styles at the [glossaries styles gallery](#). The styles can be modified, but simple adjustments can be made with the `glossaries-extra` package through the use of categories.

For example, try out the above `glossaries-extra` example with the following added to the preamble:

```
\glsssetcategoryattribute{general}{glossname}{firstuc}
```

or

```
\glsssetcategoryattribute{abbreviation}{glossdesc}{firstuc}
```

Installing Perl

Perl usually comes with Unix-like operating systems, but if you are a Windows user you may need to install it. There are various options including [Strawberry Perl](#) and [ActivePerl](#). I rarely use Windows, but Strawberry Perl seems popular and easy to install.

Remember that you don't need to have Perl installed if you're not using xindy (or any other Perl scripts, such as `latexmk`).

Using makeglossaries

If you use a [command prompt](#) or [terminal](#) and your document is called `myDoc.tex`, then you need to do:

```
pdflatex myDoc
makeglossaries myDoc
pdflatex myDoc
```

(or `latex` or `xelatex` as appropriate).

If you use [arara](#), then you need to add the following comments to your document:

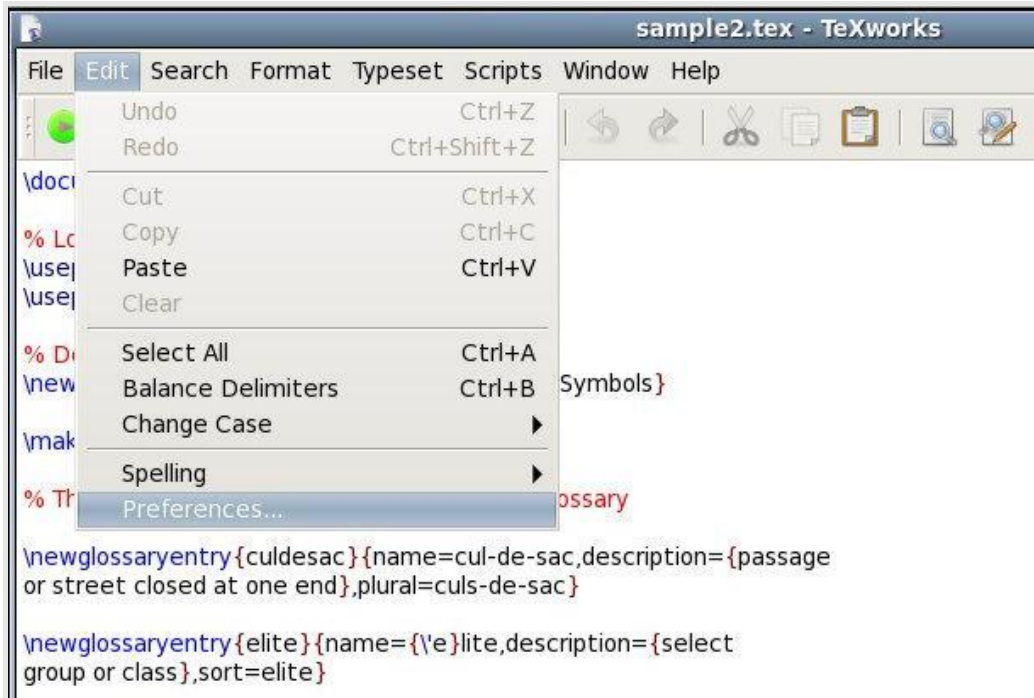
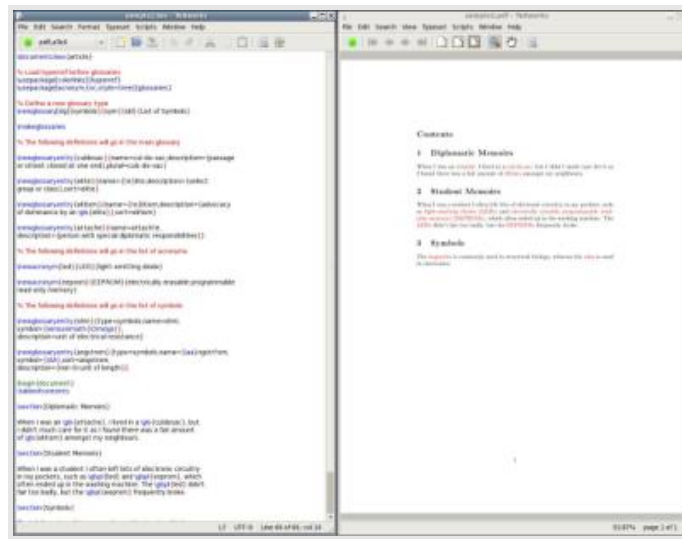
```
% arara: pdflatex
% arara: makeglossaries
% arara: pdflatex
```

If you are using MikTeX, the `makeglossaries` script will probably be located in `C:\Program Files\MikTeX 2.7\scripts\glossaries`. If you are using a Unix style system, it should be in the same location as `makeindex` (or there may be a symbolic link to it from that location).

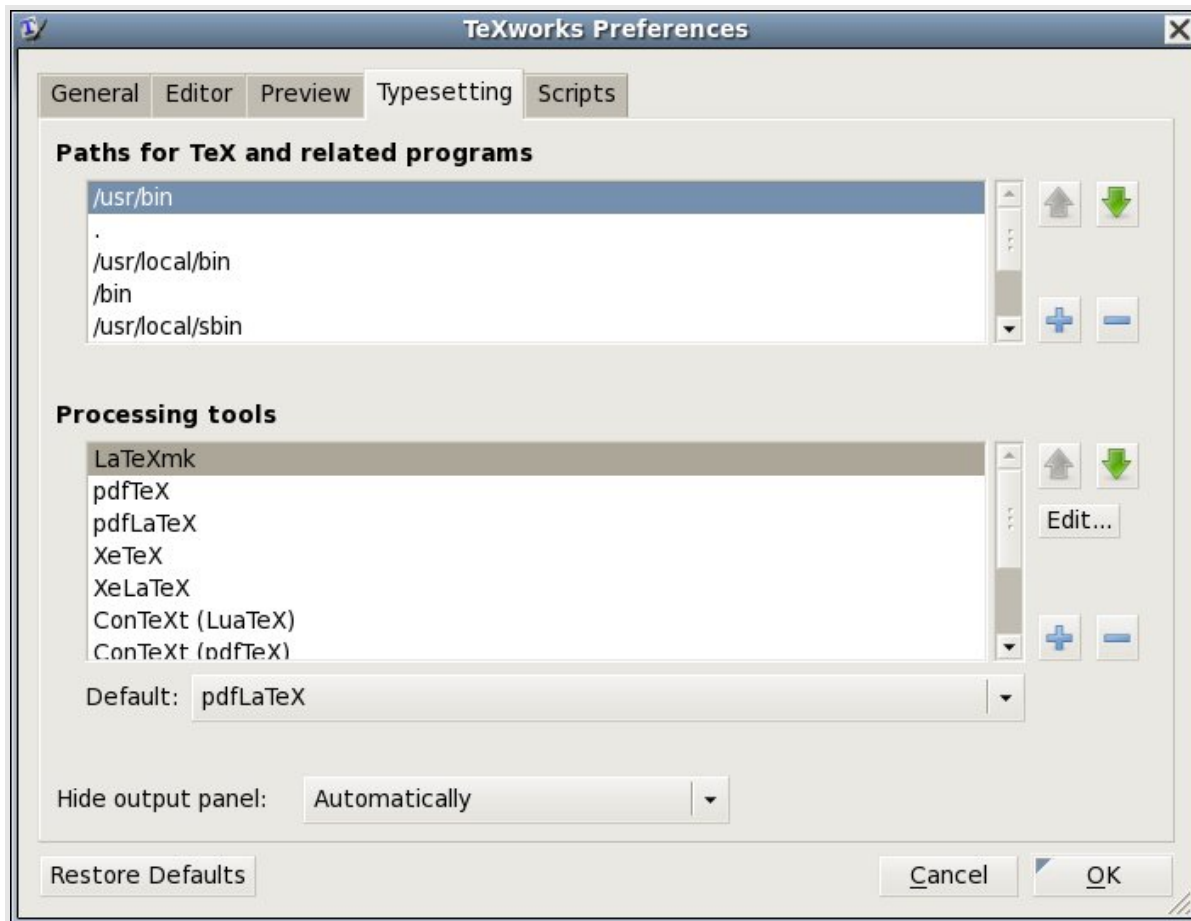
Note that `makeglossaries`, `makeindex` and `xindy` are all command line applications, but most front-ends provide a way to run them from a menu item or button press. Here, I will demonstrate using [TeXWorks](#), which is a cross-platform front-end. (At the end of this article, there is advice for [TeXnicCenter](#) and [WinEdt](#) users. If you use something else, you'll have to consult your front-end's manual to find out how to do it or search the relevant forum.)

The image here shows the example listed above in [TeXWorks](#) (click on it to view a larger version). The first `latex` run doesn't produce any glossaries (and the table of contents is also empty).

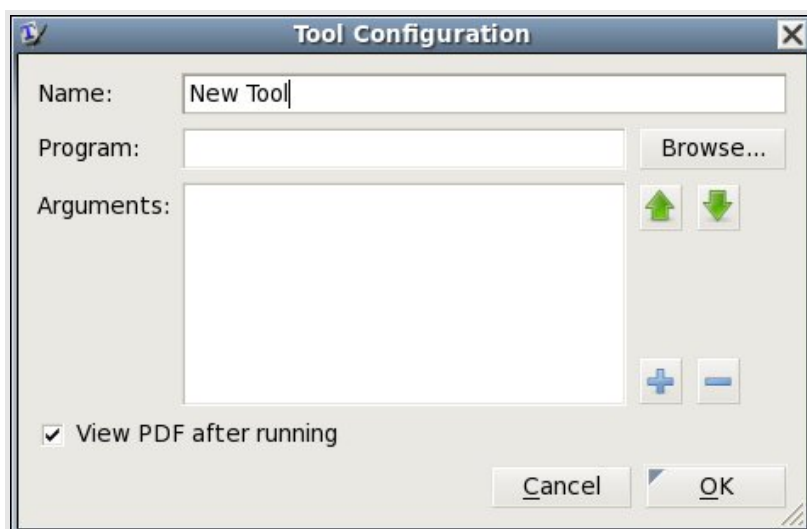
Next, you need to add `makeglossaries` to the list of applications that can be run in `TeXWorks`, so select `Edit → Preferences...`



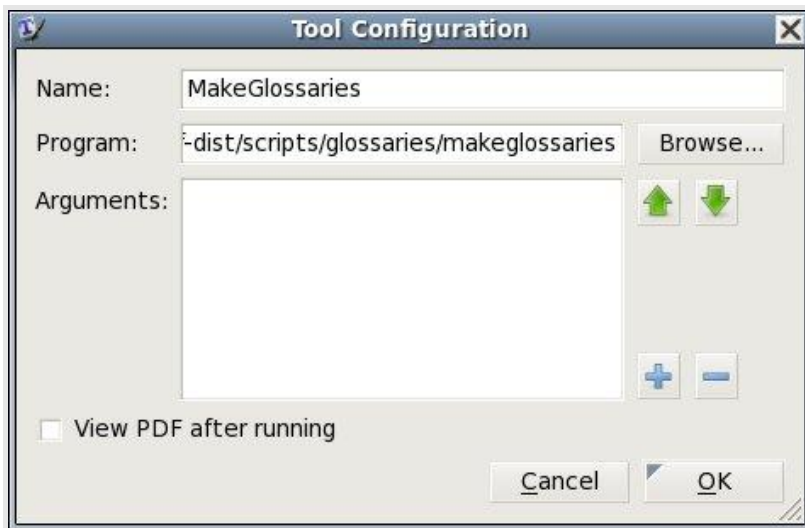
This will open the Preferences dialog box. You need to make sure the Typesetting tab is selected:



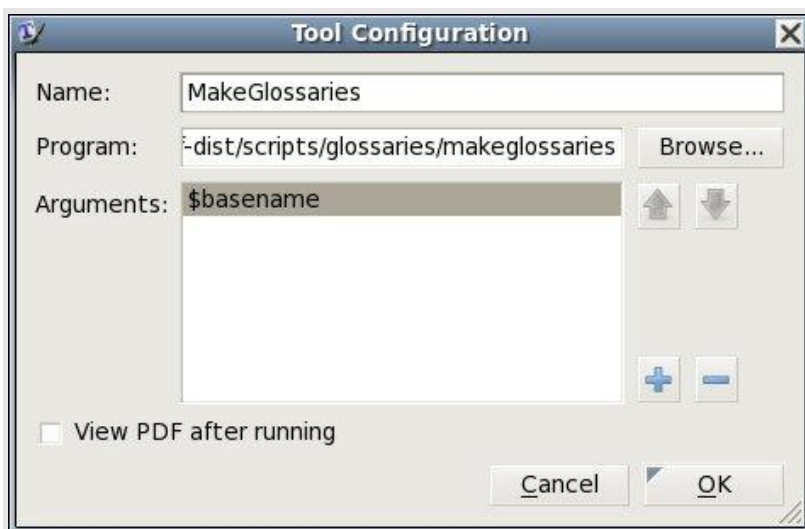
Click on the lower plus (+) symbol next to the "Processing tools" list. This will display the "Tool Configuration" window:



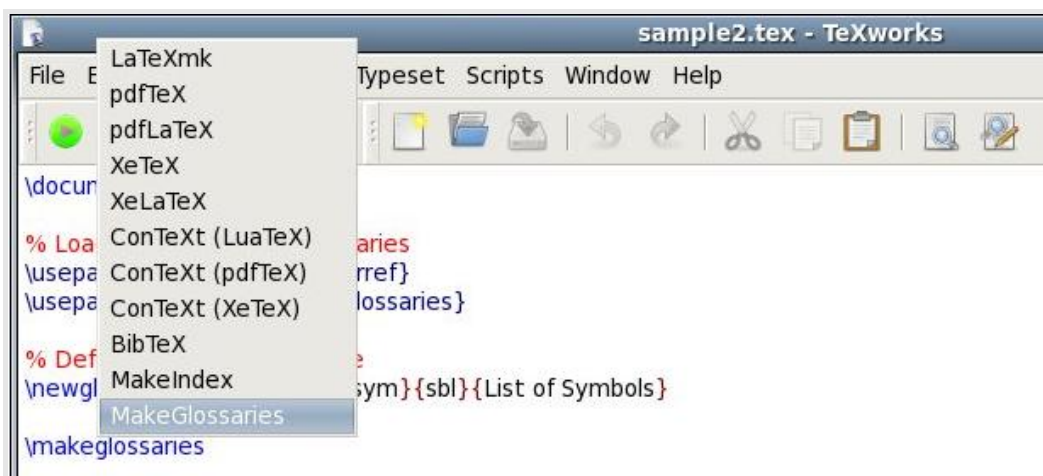
Enter the name of the tool (MakeGlossaries) and use the Browse button to select the makeglossaries script (makeglossaries.bat for Windows users):



(You don't need the "View PDF after running" box selected as running makeglossaries doesn't modify the PDF file.) Next click on the plus (+) symbol and type \$basename:



Click on "OK" and you should now see MakeGlossaries present in the drop-down tool list:



Select "MakeGlossaries" and click on the green run button. You won't see anything different in the PDF yet. Use the drop-down box to select pdfLaTeX again and click on the green run button. Your document should now be updated and the glossaries should be present. (Click on image to view larger version.)

Note that the glossary (page 1), list of acronyms (page 1) and list of symbols (page 2) haven't appeared in the table of contents. Another run (click the green button) is required.

The numbers after each glossary entry indicate the page number on which that entry was used. The numbers are red

because the hyperref package has been used, and they link to the corresponding page. You can use the glossaries package option `nonumberlist` to suppress the number list.

The list of symbols doesn't show the corresponding symbol for each entry. This is because the default style ignores the symbol. Try changing the package options to:

```
\usepackage[acronym,toc,style=tree]{glossaries}
```

See the section "Glossary Styles" in the manual for a list of predefined styles.

In general, you need to perform at least three steps to ensure the glossaries are up-to-date:

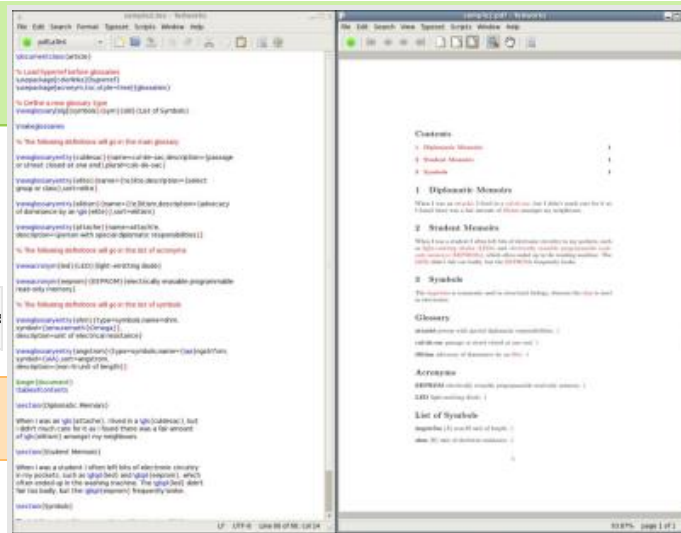
1. Run (pdf)latex
2. Run makeglossaries (or MakeGlossariesGUI described below)
3. Run (pdf)latex

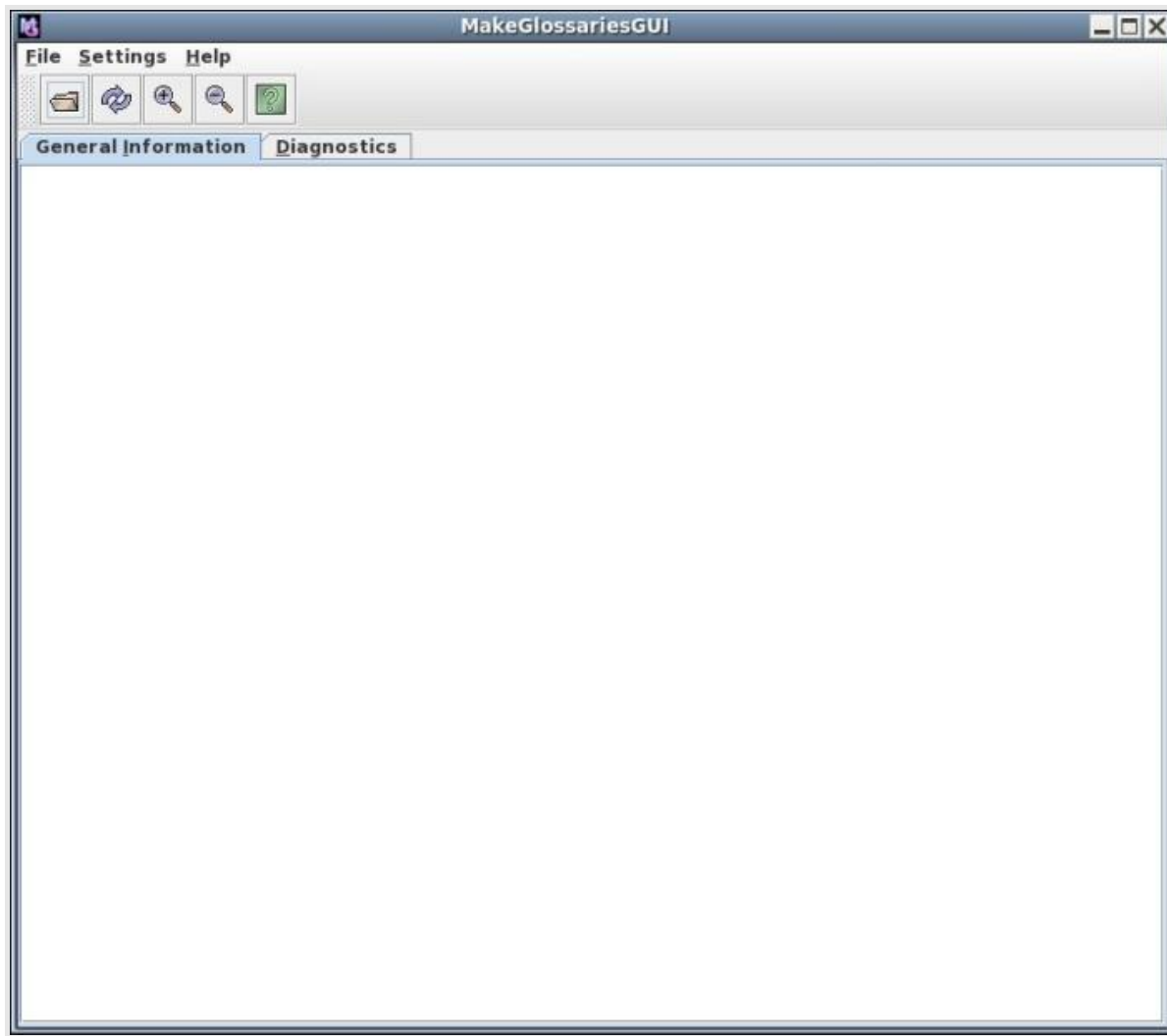
Sometimes (as above) step 3 needs to be repeated. If your glossaries contain cross-references, you may need five steps:

1. Run (pdf)latex
2. Run makeglossaries (or MakeGlossariesGUI described below)
3. Run (pdf)latex
4. Run makeglossaries (or MakeGlossariesGUI described below)
5. Run (pdf)latex

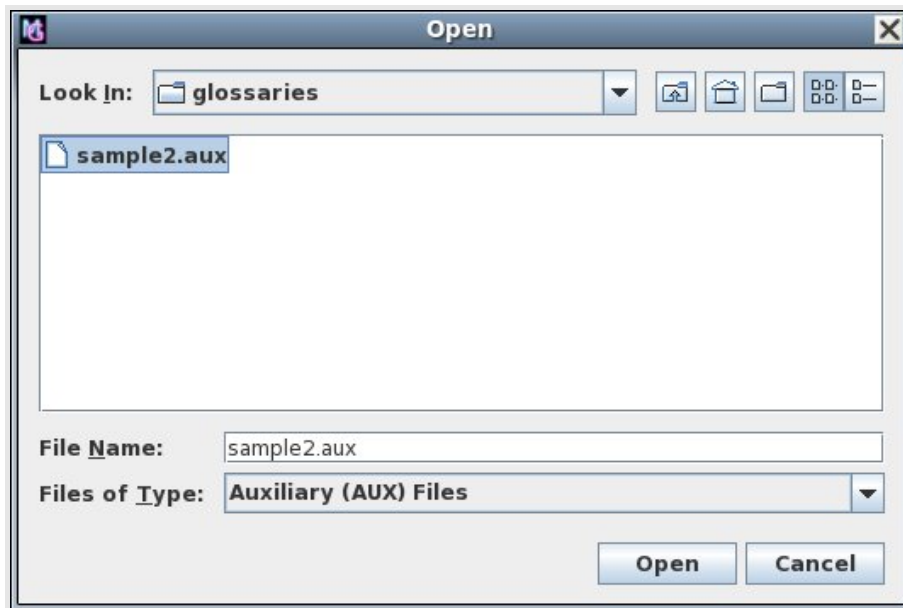
MakeGlossariesGUI

There is a Java alternative to makeglossaries called [MakeGlossariesGUI](#). This doesn't come with MikTeX or TeXLive, but can be [downloaded from CTAN](#). (Make sure you first have an up-to-date version of the [Java Runtime Environment](#) installed.) Windows users should download and run [makeglossariesgui-setup.exe](#). Users of other operating systems need to download [makeglossariesgui.zip](#) and follow the installation instructions in the README file. When you run MakeGlossariesGUI for the first time, you will be prompted for the paths to makeindex and xindy. (If you don't want to use xindy, you can leave that box blank. Similarly, if you only intend to use xindy, you can leave the makeindex box blank.) After that you should see the following window:

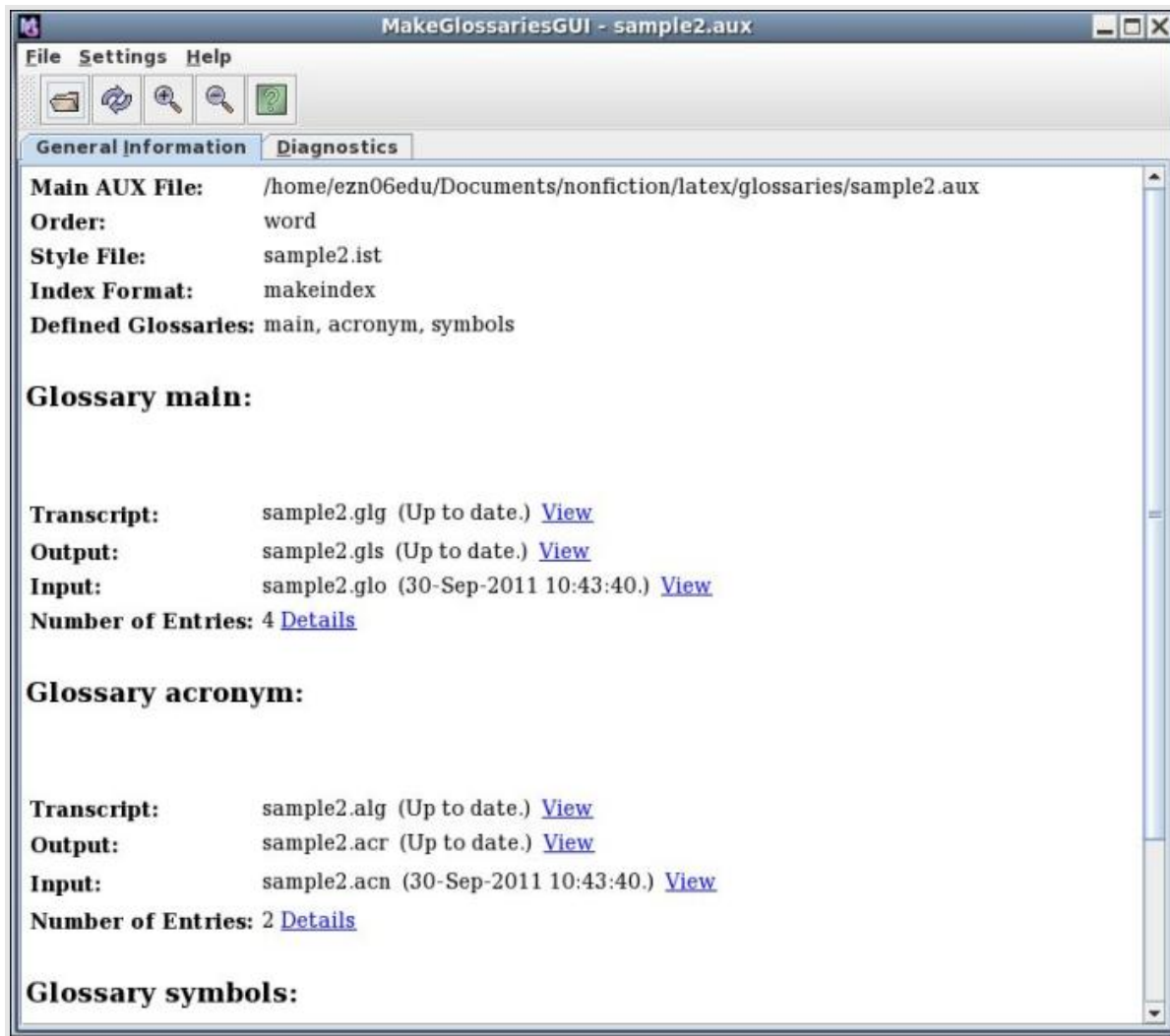




Use the open button or the File → Open menu item to load the auxiliary file created when you latexed your document. For example, my document was called "sample2.tex", so I need to load "sample2.aux":



Once you load the file, the application will automatically call makeindex or xindy for each defined glossary. The glossaries information will be displayed in the window:



If any problems occur, you can check the "Diagnostics" tab for help. For example, suppose I forgot to add `\makeglossaries`, then I'll get an error message saying "Can't determine indexer" and the diagnostics tab will display the message:

Your document doesn't seem to have used \makeglossaries.

If you modify your document, run latex as usual and then reload the auxiliary file in MakeGlossariesGUI using the reload button or the File → Reload menu item.

Using makeindex explicitly

If you don't have Perl (or Java and MakeGlossariesGUI) installed you can run `makeindex` explicitly. Let's suppose that your document is called `myDoc.tex`. If you are using `makeindex` (that is, you haven't used the `xindy` package option) then you need to use `makeindex` for each defined glossary. This includes the list of acronyms if you have used the `acronym` package option. The above example document has three glossaries: the main one, the list of acronyms and the list of symbols. The main glossary is created using:

```
makeindex -s myDoc.ist -t myDoc.glg -o myDoc.gls myDoc.glo
```

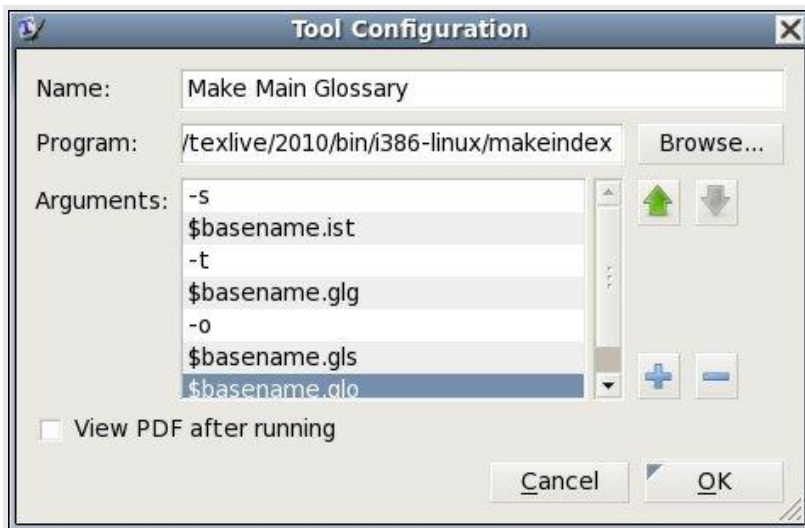
The list of acronyms is created with:

```
makeindex -s myDoc.ist -t myDoc.alg -o myDoc.acr myDoc.acn
```

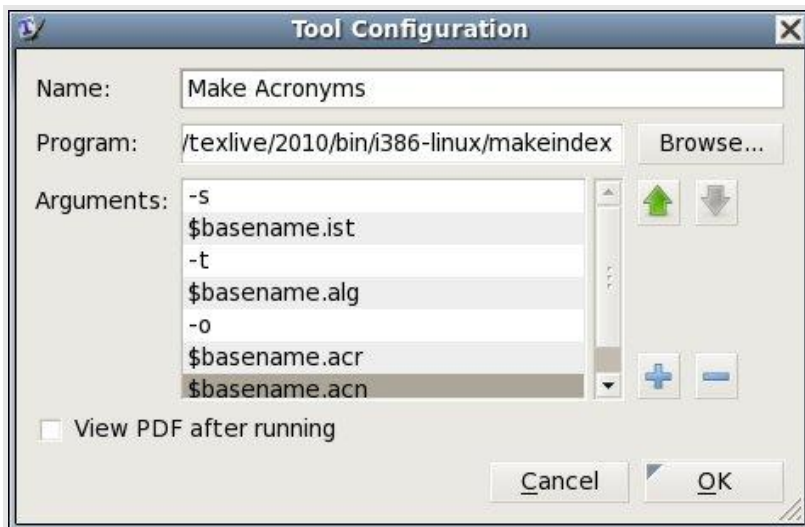
The list of symbols is created with:

```
makeindex -s myDoc.ist -t myDoc.slg -o myDoc.sym myDoc.sbl
```

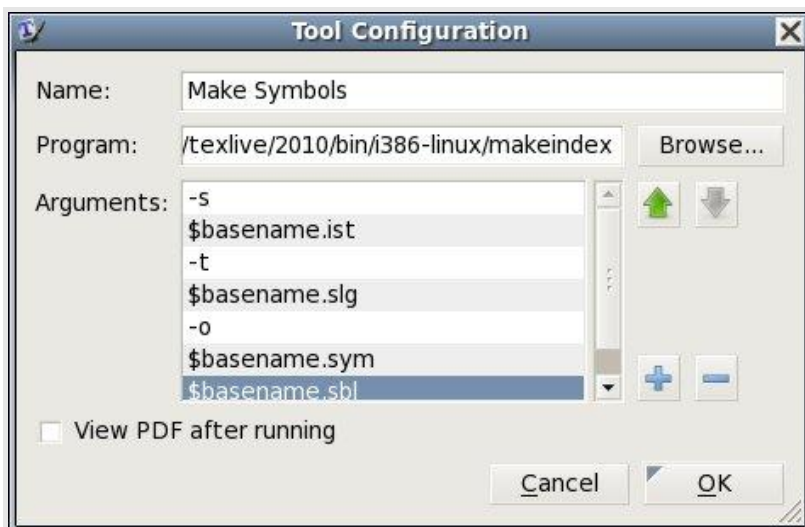
This means that you need to set up three separate tools in your editor. For TeXWorks, you need to go to the "Edit Preferences" window and create a new tool called, say "make main glossary", select the path to `makeindex` and add the appropriate arguments:



Similarly for the list of acronyms:



and for the list of symbols:



Now, everytime you modify your document, you need to:

1. Select the pdfLaTeX tool and click run button
2. Switch to the Make Main Glossary tool and click run button
3. Switch to the Make Acronyms tool and click run button
4. Switch to the Make Symbols tool and click run button
5. Switch to the pdfLaTeX tool and click run button

Depending on your document, you may need to return to step 2.

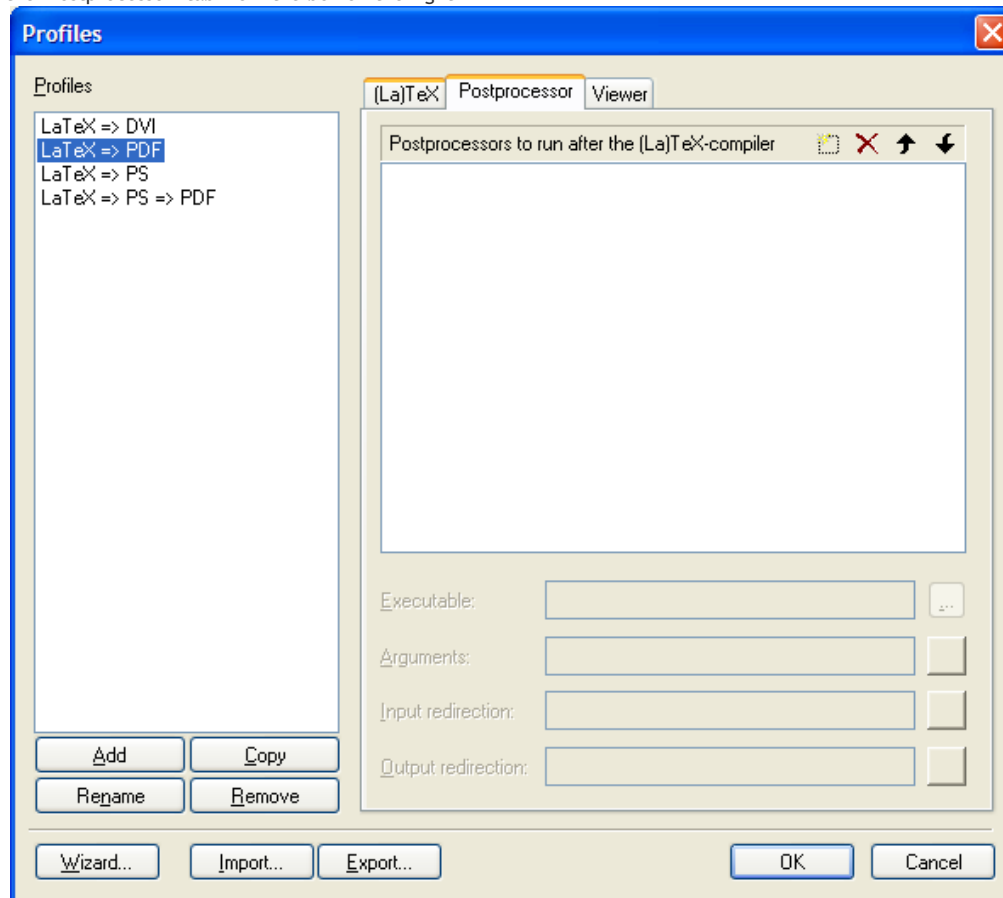
You may find it simpler to just use the automake option:

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

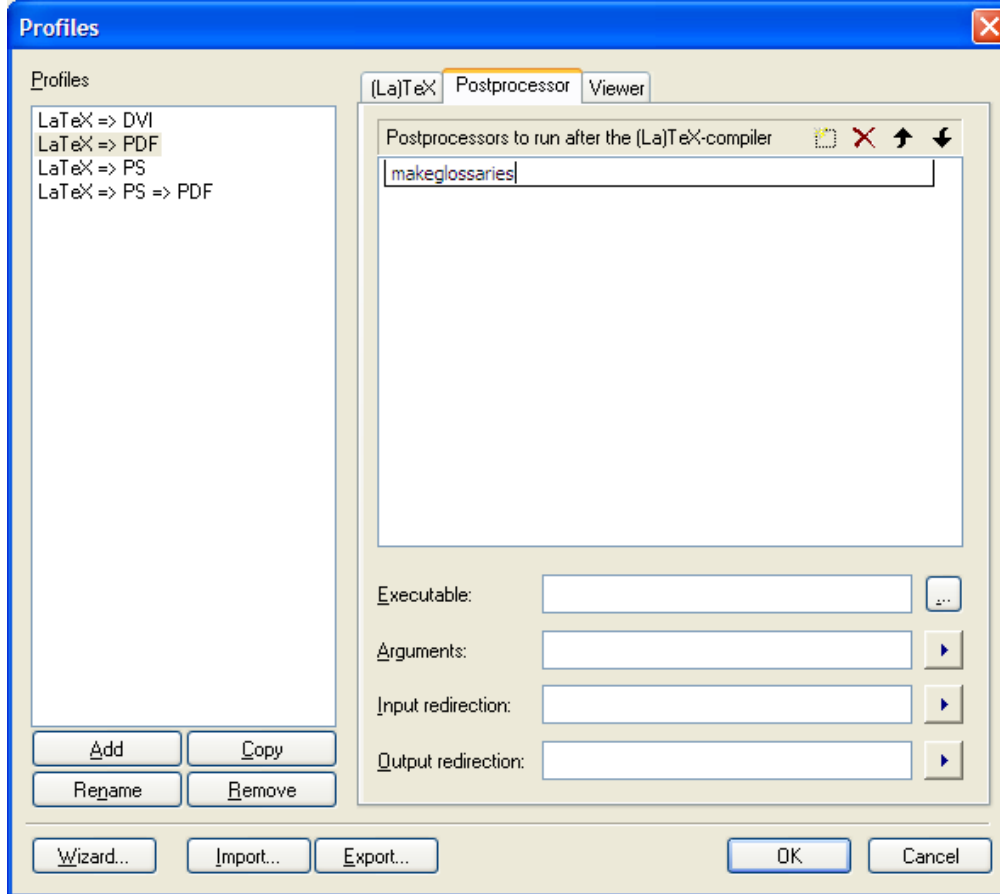
TeXnicCenter

You can add makeglossaries to the build profile of TeXnicCenter as follows:

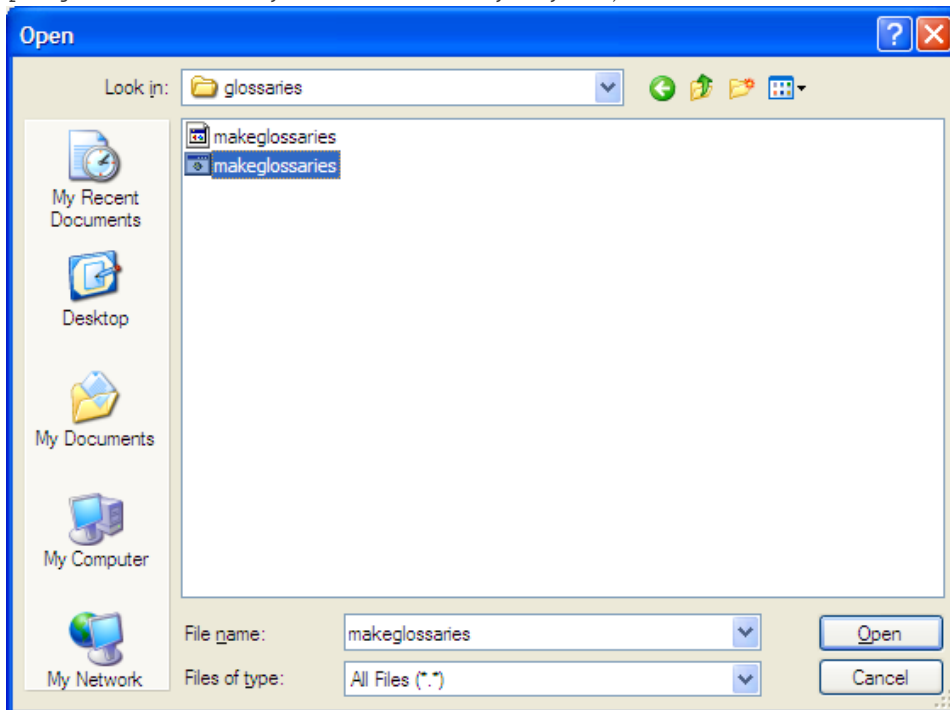
1. Install Perl (you will need to quit TeXnicCenter and restart it after installing Perl)
2. Select the menu item Build → Output Profiles...
3. Select the desired output profile from the box on the left (e.g. LaTeX=>PDF)
4. Select the "Postprocessor" tab from the box on the right



5. Click on the "New (insert)" icon. (This looks like a dashed rectangle to the left of the red cross.)
6. Type in makeglossaries

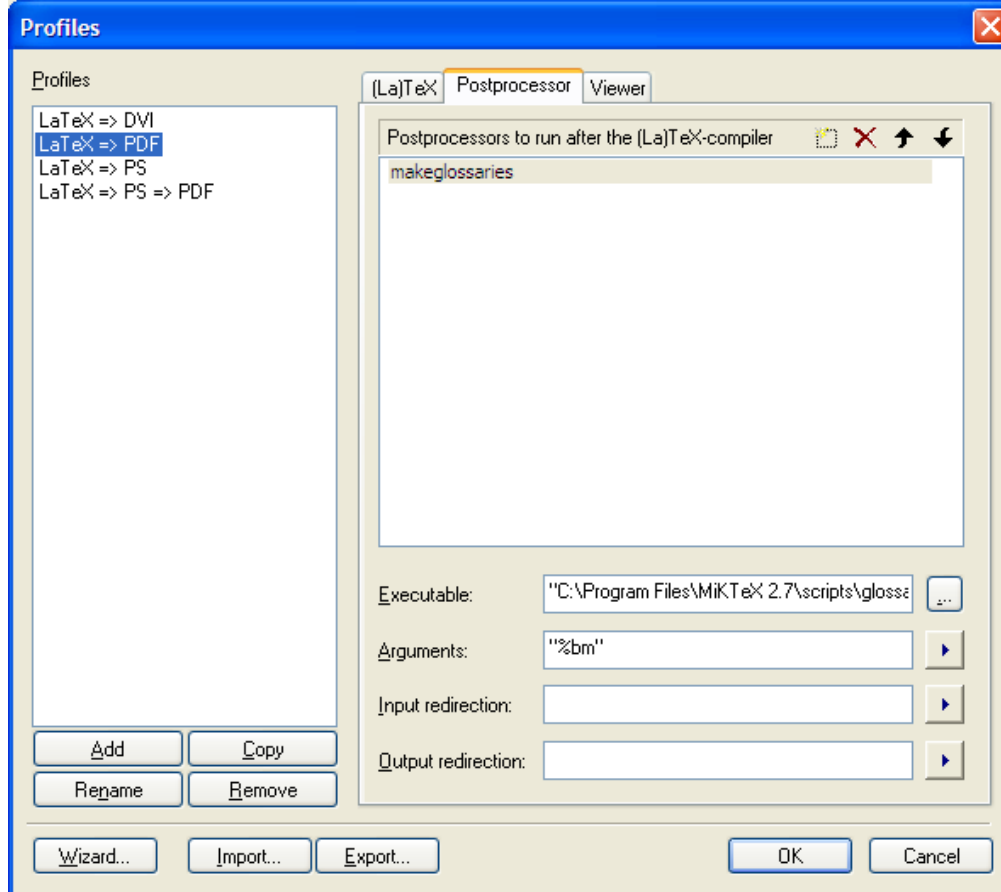


7. Type in the full path name of makeglossaries.bat into the executable box or use the ellipsis button on the right to browse the filing system. (On my Windows partition, the file can be found in the folder C:\Program Files\MiKTeX 2.7\scripts\glossaries but it may be somewhere else on your system.)



 The full path name must be surrounded by double quotes if it contains spaces.

8. Type "%bm" in the Arguments box



9. Repeat for each output profile as desired.
10. Click on okay

WinEdt

There is a [thread](#) on comp.text.tex that discusses how to use makeglossaries with WinEdt.

Comments

foxtrott

+12  

Really nice.. Tank you very much

Sunday 22 March 2009, 11:16

↳ Milagros

0  

Here's some atoadidnil Lewisian symbols:
 \boxright Lewis's stronger 'would' counterfactual
 \diamondright Lewis's stronger 'might' counterfactual
 \diamonddot Lewis's inner necessity
 \boxright should work already) Note that these also requires nt .

Tuesday 21 February 2012, 08:42

↳ Korosh

0  

I really like this idea - it would be very usefl to so many people. I wish I could do the same but my system really discourages listing/naming particular items and I figure they would really hit the roof if I used pictures.

Tuesday 12 June 2012, 15:55

zroutik

+2  

Dear Nicola, dear Latex users,

is there a command, please, to show the glossary entry in text (as with gls command), point it to the Glossary (when created), but do not create a new page number and a new back-referencin g link? Page numbers and back-referencin g links (in the Glossary list) would be created and pointed only for the first use of gls or the appropriate command.

Thank you very much for listening,
 Zroutik

Friday 05 February 2010, 13:45

zroutik(note on previous)

+2  

links in the text pointing to the Glossary list are very good. But many page numbers back-referencing from the Glossary list to the text are a nice mess, IMO.

Thank you,
Zroutik

Friday 05 February 2010, 13:49

spkmn

-1  

Hi... first, I love the package. I have a problem that I'm looking for advice on. I have a glossary entry that contains a reference to another glossary entry.

```
\newglossaryentr y{Hi}{description={This is an \gls{entry}},name={Hi}}
\newglossaryentr y{There}{description={There is a whole world out there.},name={Hi}}
```

Inside of a table cell, I want to use `\gls{Hi}`.

When it prints, I get "This is an ``gls --There"" inside of the table...

This entry appears fine in the `\printglossaries` and if I use it outside of a table.

Thanks
James

Monday 29 March 2010, 18:53



spkmn

-1  

sorry.... it should be

```
\newglossaryentr y{Hi}{description={This is an \gls{There}},name={Hi}}
\newglossaryentr y{There}{description={There is a whole world out there.},name={Hi}}
```

Thanks
James

Monday 29 March 2010, 18:55

nlct

-1  

Hi,

it's better to post requests for help in the forum as you'll get a quicker response and more people are likely to see it. It's also helpful to provide a <http://theoval.cmp.uea.ac.uk/~nlct/latex/minexample/> rel="nofollow" target="_blank">minimal working example that illustrates the problem.

Regards
Nicola Talbot

Sunday 11 April 2010, 13:52



nlct

-1  

Sorry, that link got a bit mangled.

Sunday 11 April 2010, 13:55

horacv

-1  

Hi,
i am having a couple of problems with the definition of several glossaries. In particular, i have two separate input files. Where do i indicate this to the `\newglossary` command?
i defined it this way:

```
\include{./acr/acr.tex}
\include{./acr/not.tex}
\newglossary{notacion}{gls}{glo}{Notaci\ 'on}
```

inside `acr.tex` i defined each entry, this way:

```
\newglossaryentr y{rz}{
type=\acronymtype,
```

and, inside `not.tex`, exactly the same way, except i didn't defined the type.
The output pdf_lates produces the same list (the one in `acr.tex`) two times. Any ideas what i am doing wrong?
thanks for the help

Monday 26 April 2010, 17:12

If you have problems with the glossaries package, please post your query to the forum rather than as a comment here. Other people are more likely to search the forum than the comments if they have a similar problem.

Monday 24 May 2010, 12:30

Please fix your picture for the TeXnicCenter setup. In MiKTeX 2.9 on Windows 7 64 bit:

- use C:\Program Files\MiKTeX 2.9\miktex\bin\x64\makeglossaries.exe

- use "%tm" NOT "%bm"

I have perl setup but I am not sure it is needed anymore since it is a .exe file.

Thursday 18 April 2013, 00:56

Please login to post comments or replies.



Share

Latest Forum Posts

Re: Metafile to EPS converter: should it work from within LY

01/11/2016 15:21, scottkosty

Ah OK, I see. This is a bit a pity, since I strongly believe that what LyX does with LaTeX is exactly what "Windows"-people want/need: something "easy"/"user-friendly"/"working-out-of-the-box"/"easy-GUI"/WYSIWYM. Yes it is too bad we do not have more developers on W...

Re: Use TikZ with LyX

01/11/2016 14:34, jonjoexx

Thank you so much for posting this I am new to Lyx and spent two days trying to get tikz to work....

Re: Metafile to EPS converter: should it work from within LY

01/11/2016 09:49, Stef Pillaert

- There is only one LyX developer who uses Windows. Most use Linux. The developer on Windows does what he can and dedicates a lot of time, but it is hard when he is the only one. Ah OK, I see. This is a bit a pity, since I strongly believe that what LyX does with LaTeX is exactly what "Windows"-people wan...

Re: Metafile to EPS converter: should it work from within LY

01/11/2016 00:07, scottkosty

OK, I did that for both emf and wmf to eps, now it works (though it takes quite a while the first time; also seems slow the first time I generate a pdf). But OK, it seems fine now. Thanks a lot for your time!!! Great! Sorry it took so long. Thanks for your patience. (Just out of curiosity: did I do something wrong...

Re: Android

31/10/2016 19:16, Stefan Kottwitz

It is good to see Lyx in the cloud. Yes! lyx-cloud.jpg (Click image to expand) On my Blackberry phone. Stefan...

Re: Metafile to EPS converter: should it work from within LY

31/10/2016 17:24, Stef Pillaert

OK, I did that for both emf and wmf to eps, now it works (though it takes quite a while the first time; also seems slow the first time I generate a pdf). But OK, it seems fine now. Thanks a lot for your time!!! (Just out of curiosity: did I do something wrong while installing? Or does this happen on every Windows-machi...

Re: Metafile to EPS converter: should it work from within LY

31/10/2016 14:51, scottkosty

"magick

i

o" OK, copy that command to the command for the converter that is not working. Then restart LyX. Does it work now?...

Re: customizing screen font size

31/10/2016 14:50, scottkosty

In Tools > Preferences > Screen Fonts you can change the Zoom %....

Re: hyperlinks in tikz

31/10/2016 06:55, omerangel

Looks great. Thanks!...

customizing screen font size

30/10/2016 20:22, jalea148

My vision is poor and I would like everything larger - especially the fonts....

