Glossaries, Nomenclature, Lists of Symbols and Acronyms

(66 votes, average: 3.88 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 acronyms 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 it has a large manual that some beginners find daunting. This article is an introductory guide to help you get started.

What do I need installed?

The latest version of the glossaries package (at the time of writing this) is version 1.19 and is available on CTAN. In addition, you also need to have the following packages installed:

- ifthen
- xkeyval at least version 2.5f (2006/11/18)
- xfor
- amsgen (part of amstex)

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 are all distributed with MikTeX and TeX Live.

In addition to the above LaTeX packages, you also need to have an indexing application installed to collate and sort the entries. The glossaries package is configured to work with makeindex or xindy. Makeindex is distributed with MikTeX, TeX Live and teTeX, so it should be readily available on your system. Xindy is distributed with TeX Live, so you might not have it installed if you use a different TeX distribution.

There is a common misconception that you must have Perl installed to use the glossaries package. There is a Perl script called makeglossaries that acts as a convenient interface to makeindex or xindy, but it is not essential. This will be discussed in more detail below.

Getting Started

The first thing that you need to do in your document is declare 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 installed if you want to use this option.

If you are also using the hyperref package, you must load the glossaries package after the hyperref package:

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

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

If you want your glossaries, list of symbols and list of acronyms to appear in the document's table of contents, you also need to use the toc package option:

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

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 characters that are not active in the label.

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 acronyms.

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 they must be defined before they are used. Alternatively, you can put all your definitions in a separate file and input it using `\input{`*`filename`*`}` or `\loadglsentries{`*`filename`*`}`

Referencing Terms

Once a term has been defined using `\newglossaryentry`, it can be referenced 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.

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}
```

When I build this document (run it through LaTeX) I get the following warnings:

```
Package glossaries Warning: \makeglossaries hasn't been used,
the glossaries will not be updated.

Package glossaries Warning: No \printglossary or \printglossaries found.
This document will not have a glossary.
```

These warnings can be ignored for the moment.

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.*

It might be worth trying out this simple example to check that you have the required packages installed. (Check Installing things on a (La)TeX system if you're not sure how to install packages.)

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}
```

If your term has more than one plural, use the `plural` key for the most commonly used plural together with `\glspl{label}` and use `\glslink{label}{alternative plural}` for the other plural.

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 may not 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 needs to 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. 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}
```

Accents and non-Latin characters

Makeindex is hard-coded for English, 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 can just do:

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

or if you are using the inputenc package, you can just do:

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

Care must be taken 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}
```

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}
```

I can reference the name as usual:

```
\gls{ohm}
```

or I can reference the symbol using:

```
\glssymbol{ohm}
```

Acronyms

When you define a term, you can specify that it should appear differently on first use. You can use this facility to define acronyms. For example:

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

There is a shortcut command that is equivalent to the above:

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

> 🕮 The glossaries package provides other acronym formats, such as making the long form appear in a footnote on first use. These are covered in the section "Acronyms" in the manual.

You can then use the acronym in the same way as any other glossary term. For example:

```
\gls{led}
```

If this is the first time you have used this entry, it will appear as

> *light-emitting diode (LED)*

otherwise it will appear as

> *LED*

If you define the acronym using `\newacronym` rather than explicitly using `\newglossaryentry`, you can access the long and short forms using:

```
\acrlong{label}
```

and

```
\acrshort{label}
```

For example:

```
\acrlong{led}
```

will produce

> *light-emitting diode*

and

```
\acrshort{led}
```

will produce

> *LED*

If you feel that is far too much typing, you can enable the shortcuts using the `shortcuts` package option:

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

You can now use `\acs` instead of `\acrshort` and `\acl` instead of `\acrlong`.

> 🕮 Other acronym related commands are described in the manual. In addition to the section called "Acronyms", there is also a section that covers resetting and unsetting entries.

Multiple Glossaries

You may have as many glossaries, lists of symbols or lists of acronyms as you like in your document. The default is to just have one glossary. You can add a list of acronyms using the `acronym` package option:

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

Any acronyms defined using `\newacronym` will automatically be added to the list of acronyms rather than the main glossary if the `acronym` package option is used. If you want an additional glossary, for example a list of symbols, 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 in the next section.) 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. This is only needed if you use the makeglossaries Perl script. You don't need to worry about it if you don't have Perl installed.

> 🕮 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 symbols. I can define a glossary called "symbols" as follows:

```
\newglossary{symbols}{sym}{sbl}{List of Symbols}
```

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

If the `type` key is omitted, the main glossary is assumed.

Using makeindex or xindy

So far I have only covered how to define a term, symbol or acronym 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 acronyms 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` which is described in the section "Displaying a glossary" in the manual.

Here is an example document:

```
\documentclass{article}

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

% Define a new glossary type
\newglossary[slg]{symbols}{sym}{sbl}{List of Symbols}

\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={\ensuremath{\Omega}},
description=unit of electrical resistance}

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

If we run this document through pdflatex, we get the following warnings (amongst others):

```
pdfTeX warning (dest): name{glo:ohm} has been referenced but do
es not exist, replaced by a fixed one

pdfTeX warning (dest): name{glo:angstrom} has been referenced but does not exis
t, replaced by a fixed one

pdfTeX warning (dest): name{glo:eeprom} has been referenced but does not exist,
 replaced by a fixed one

pdfTeX warning (dest): name{glo:led} has been referenced but does not exist, re
placed by a fixed one

pdfTeX warning (dest): name{glo:elitism} has been referenced but does not exist
, replaced by a fixed one

pdfTeX warning (dest): name{glo:culdesac} has been referenced but does not exis
t, replaced by a fixed one

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

These warnings are due to the fact that I have used the hyperref package, which creates hyperlinks from the terms to their definition in the relevant glossary, list of acronyms or list of symbols, but the targets don't exist yet. If you try this example and view the resulting PDF/DVI file, you will see that the glossaries haven't appeared yet. You may also notice that the terms generated by `\gls` and `\glspl` have all appeared in red. This is the default colour used by the hyperref package for internal links when used with the `colorlinks` option.

The next step is to use an indexing application to create the glossaries.

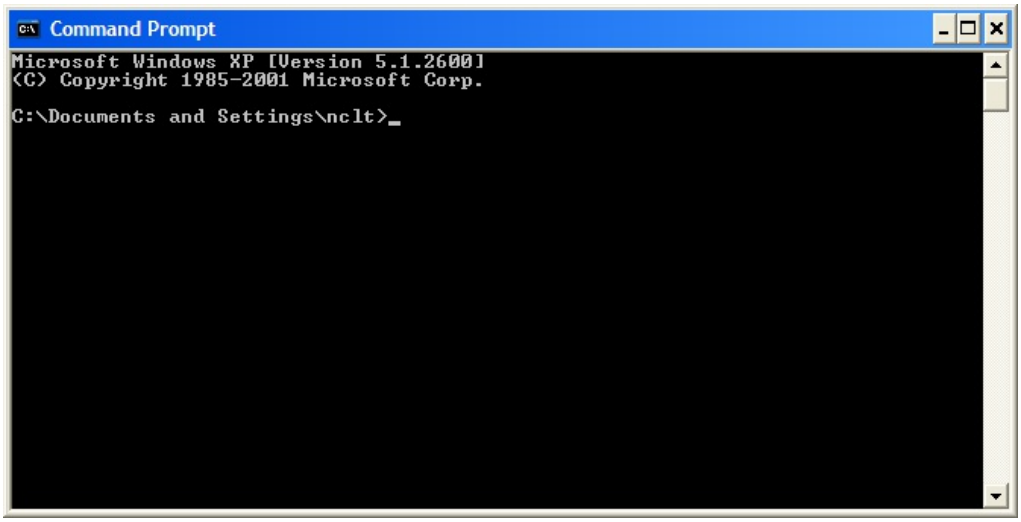
Using makeglossaries

The easiest way of creating the glossaries is to use the makeglossaries Perl script that comes with the glossaries package. If you are using MikTeX, this 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).

> ⚠ In order to use makeglossaries, you must have Perl installed. Don't be put off by this. There are a number of useful TeX related Perl scripts (such as epstopdf) so it really helps to have Perl, but if you are set against it for some reason, there are instructions below to help you access makeindex or xindy explicitly.

Makeglossaries, makeindex and xindy are all command line applications. There are several ways to access that type of application. I will first describe how to do this using the MSDOS command prompt. (Unix style users are more likely to be familiar with using a terminal, so I shall assume they can work out what to do from the MSDOS command prompt instructions.)

Windows users can access the command prompt from the Start menu. It is usually in the Accessories folder. It should produce something that looks like:



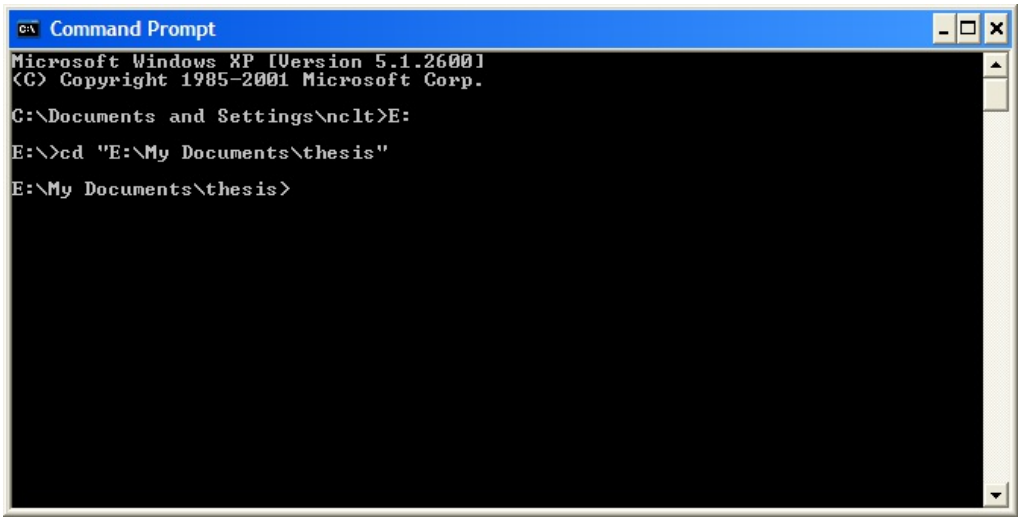
Next, you need to be in the same folder as your document. Let's suppose that my document is contained in the folder `E:\My Documents\thesis`. There are two things to note about this path name: it uses the E drive rather than the default C drive and the name contains a space. So I need to switch to the E drive by typing

```
E:
```

and then I need to specify the path by typing

```
cd "E:\My Documents\thesis"
```

Note that I have used double quotes around the path name because of the space character. These two steps are illustrated below:

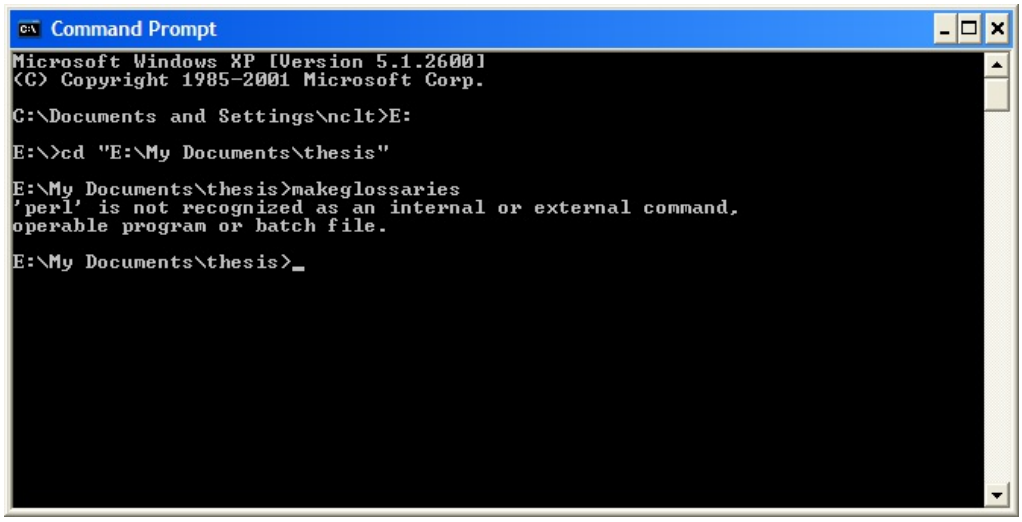


Next, I'm going to test if makeglossaries is working. To do this I'm just going to type

```
makeglossaries
```

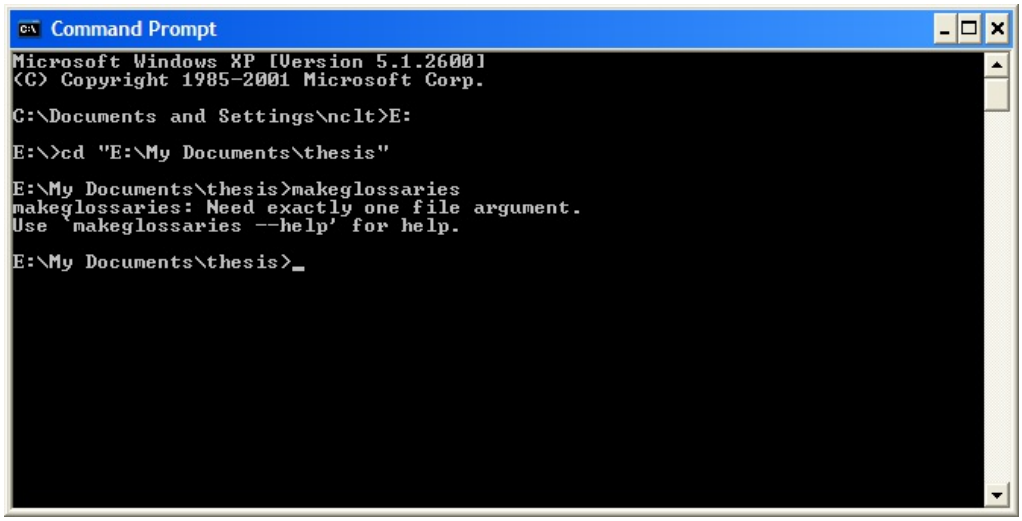
This can produce one of two responses:

```
'perl' is not recognised as an internal or external command,
operable program or batch file.
```



This means that you don't have Perl installed. If you don't want to install Perl, see Using makeindex or xindy explicitly below. If you want to install Perl, the easiest method is to download Active Perl, run the downloaded executable and follow the instructions. You will have to exit the command prompt (type `exit` or click on the close icon) and restart it.

```
makeglossaries: Need exactly one file argument.
Use `makeglossaries --help' for help.
```



This means that Perl and makeglossaries have been successfully installed.

Once Perl and makeglossaries have been successfully installed you can now create your glossaries, lists of acronyms and symbols for your document. Let's suppose the document is called `myDoc.tex`, then you need to type the following line into the command prompt:

```
makeglossaries myDoc
```

Note that the file extension hasn't been used.

This should produce the following:

```
added glossary type 'acronym' (alg,acr,acn)
added glossary type 'symbols' (glg,sym,sbl)
makeindex  -s "myDoc.ist" -t "myDoc.alg" -o "myDoc.acr" "myDoc.acn"
This is makeindex, version 2.15 [20-Nov-2007] (kpathsea + Thai support).
Scanning style file ./myDoc.ist.........................done (27 attributes redefined, 1 i
Scanning input file myDoc.acn....done (4 entries accepted, 0 rejected).
Sorting entries....done (10 comparisons).
Generating output file myDoc.acr....done (9 lines written, 0 warnings).
Output written in myDoc.acr.
Transcript written in myDoc.alg.
makeindex  -s "myDoc.ist" -t "myDoc.glg" -o "myDoc.sym" "myDoc.sbl"
This is makeindex, version 2.15 [20-Nov-2007] (kpathsea + Thai support).
Scanning style file ./myDoc.ist.........................done (27 attributes redefined, 1 i
Scanning input file myDoc.sbl....done (2 entries accepted, 0 rejected).
Sorting entries....done (2 comparisons).
Generating output file myDoc.sym....done (9 lines written, 0 warnings).
Output written in myDoc.sym.
Transcript written in myDoc.glg.
makeindex  -s "myDoc.ist" -t "myDoc.glg" -o "myDoc.gls" "myDoc.glo"
This is makeindex, version 2.15 [20-Nov-2007] (kpathsea + Thai support).
Scanning style file ./myDoc.ist.........................done (27 attributes redefined, 1 i
```

```
Scanning input file myDoc.glo....done (3 entries accepted, 0 rejected).
Sorting entries....done (4 comparisons).
Generating output file myDoc.gls....done (12 lines written, 0 warnings).
Output written in myDoc.gls.
Transcript written in myDoc.glg.
```

This means that the glossary, list of acronyms and list of symbols have been successfully created. You now need another LaTeX run to incorporate them into the document. If you view your document, it should now contain the glossary, list of acronyms and list of symbols.

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.

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.

Using makeindex and xindy explicitly

If you don't have Perl installed you can run makeindex or xindy explicitly. Again, let's suppose that your document is called `myDoc.tex`. If you are using makeindex (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 by typing the following line into the command prompt:

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

The list of acronyms is created by typing the following line into the command prompt:

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

The list of symbols is created by typing the following line into the command prompt:

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

Note that if your file name contains spaces (a bad idea), you must use double quotes around each file name. For example:

```
makeindex -s "my Doc.ist" -t "my Doc.glg" -o "my Doc.gls" "my Doc.glo"
```

If you have decided to use xindy instead of makeindex (remember to use the `xindy` package option) then the main glossary is created by typing the following into the command prompt:

```
xindy -L english -I xindy -M myDoc -t myDoc.glg -o myDoc.gls myDoc.glo
```

The list of acronyms is created by typing the following line into the command prompt:

```
xindy -L english -I xindy -M myDoc -t myDoc.alg -o myDoc.acr myDoc.acn
```

The list of symbols is created by typing the following line into the command prompt:

```
xindy -L english -I xindy -M myDoc -t myDoc.slg -o myDoc.sym myDoc.sbl
```

If you use makeindex when you should have used xindy, you will get an error message that looks something like:

```
Scanning input file myDoc.glo...done (0 entries accepted, 4 rejected).
Nothing written in myDoc.gls.
```

If you use xindy when you should have used makeindex, you will get an error message that looks something like:

```
ERROR: EVAL: variable |gLOSSARYENTRY{ATTACH'E?gLOSSARYENTRYFIELD{ATTACHE}{gLSNAMEFONT{ATTACH
```

All things considered, it's much easier to dispense with all of the above and just type

```
makeglossaries myDoc
```

How many times do I need to build/LaTeX the document or use the indexing application?

If you tried the example in the previous section, and followed all the instructions: build/LaTeX the document, run makeglossaries (or makeindex) and build/LaTeX the document, you may have noticed that the document still isn't complete and is still generating the warning message:

```
pdfTeX warning (dest): name{glo:elite} has been referenced but d
oes not exist, replaced by a fixed one
```

This warning has resulted from the fact that one of the glossary entries (`elite`) has only been referenced in the glossary (in the description of `elitism`). This means that the first LaTeX run didn't encounter any reference to `elite` so it didn't write the relevant information to the external glossary file which means that it didn't end up in the glossary. Now that the glossary is present, we need another makeglossaries (or makeindex) run to add `elite` to the glossary. This in turn means that another LaTeX run is required to update the glossary section (and to add the glossary section headers to the table of contents). Once this has been done, the document should now be complete.

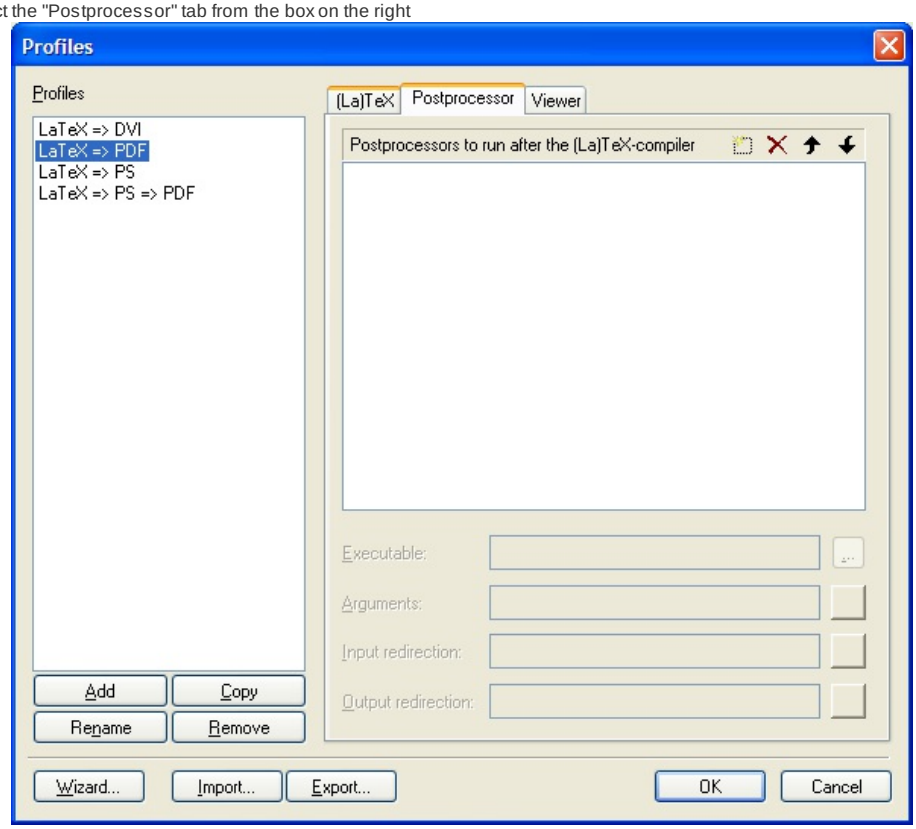
Incorporating makeglossaries into editors

Some editors, such as TeXnicCenter, allow you to specify external applications to run as a post-processor. My knowledge of front-ends is fairly limited (I tend to only use vi and make), but here are a few pointers:

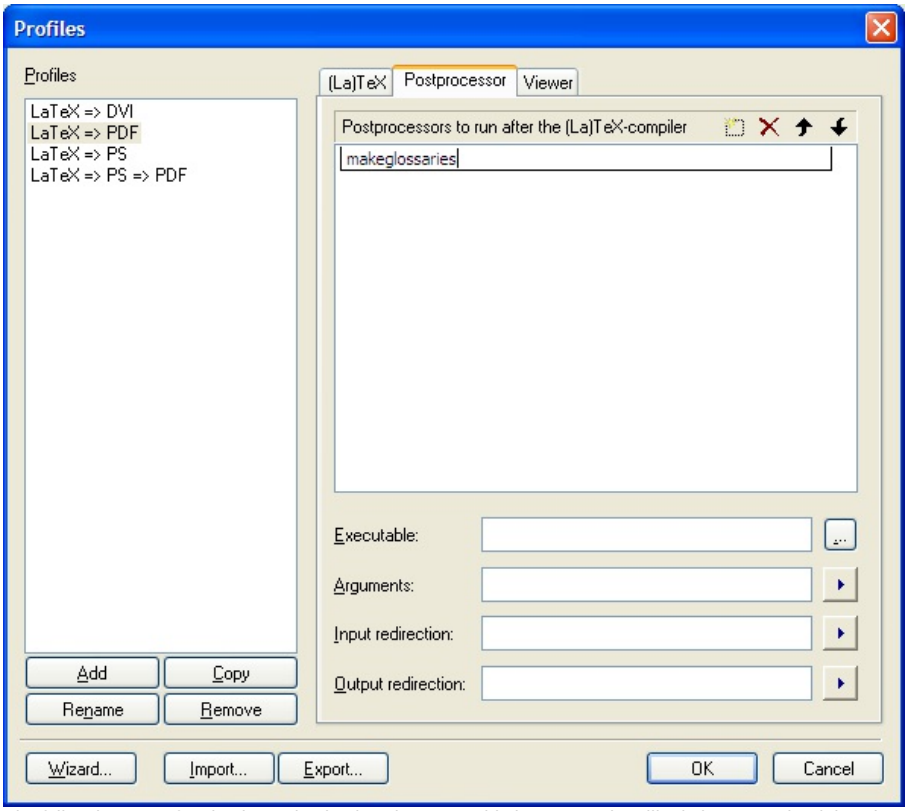
TeXnicCenter

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

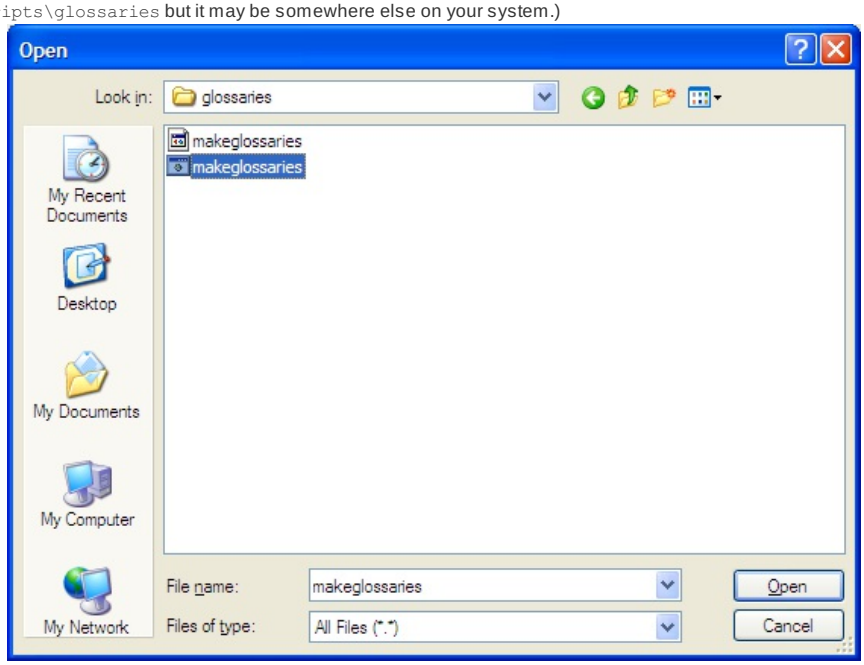
1. Install Perl (you will need to quit TeXnicCenter and restart it after installing Perl)
2. Select the menu item Build->Output Profiles...
3. Select the desired output profile from the box on the left (e.g. LaTeX=>PDF)
4. Select the "Postprocessor" tab from the box on the right



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

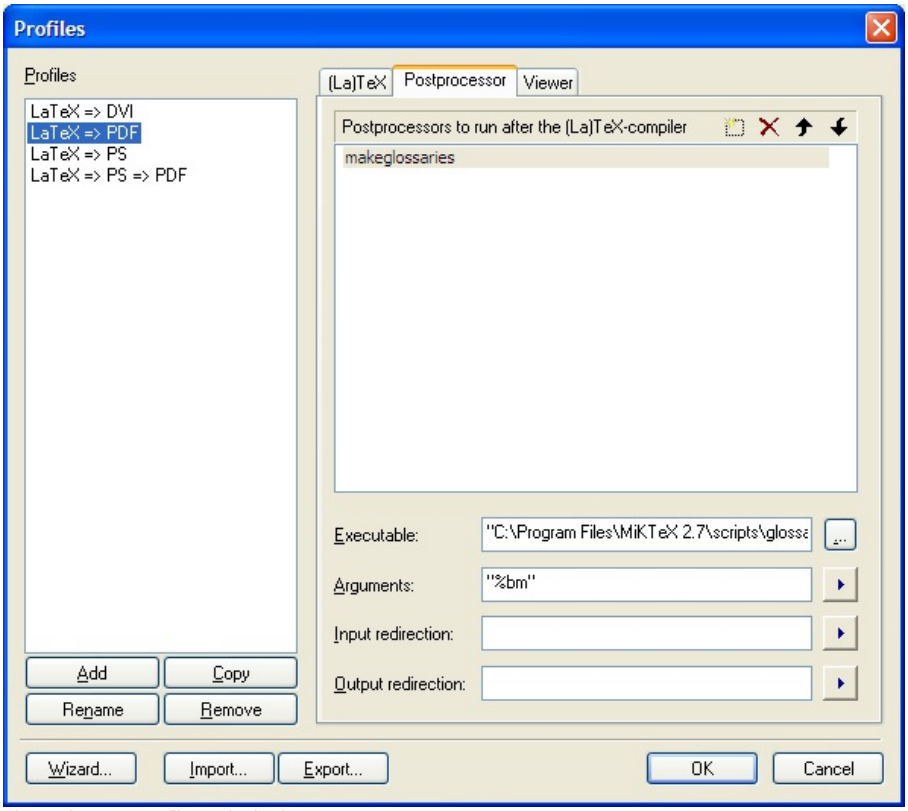


7. Type in the full path name of makeglossaries.bat into the executable box or use the ellipsis button on the right to browse the filing system. (On my Windows partition, the file can be found in the folder `C:\Program Files\MiKTeX 2.7\scripts\glossaries` but it may be somewhere else on your system.)



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 do use makeglossaries with WinEdt.

Other Editors

If you use another editor, I suggest you search the editor's documentation to find how to use makeindex. It's possible that you may be able to infer from that how to use makeglossaries. (Perhaps users who have already worked out how to do it for their preferred editor might like to add instructions as a comment to this article.)

Comments

foxtrott +5

Really nice.. Tank you very much

Sunday 22 March 2009, 11:16

Please login to post comments or replies.