Chapter 4: Moving Information Around

LaTeX often has to move information from one place to another. For example, the information contained in a table of contents comes from the sectioning commands that are scattered throughout the input file. Similarly, the command that generates a cross-reference to an equation must get the equation number from the `equation` environment, which may occur several sections later.

LaTeX requires two passes over the input to move information around: one pass to find the information and a second pass to put it into the text—it occasionally even requires a third pass. To compile a table of contents, for example, one pass determines the titles and starting pages of all the sections, and a second pass puts this information into the table of contents. But instead of making two passes every time it is run, LaTeX reads your input file only once and saves the cross-referencing information in special files for use the next time.

For example: if `sample2e.tex` had a command to produce a table of contents LaTeX would write the necessary information into the file `sample2e.toc`; then it would use the information in that file to produce the table of contents in the typeset document. After that, the `sample2e.toc` file would be re-written to a newer version (this version will be used to produce the table of contents the next time LaTeX is run) on the tex file. But notice that this "newer" version will actually be a version from a previous execution, meaning that LaTeX's cross-referencing information is always "old". This will be noticeable mainly when you are first writing the document—for example, a newly added section won't be listed in the table of contents—but running it again on the same input will correct any errors[1].

1 Cross-References

One reason for numbering things like figures and equations is to refer the reader to them, as in: "See Figure 3 for more details". You don't want the character '3' to actually be written in the input file because adding another figure might make the figure become "Figure 4", and require changing the '3' to a '4' everywhere it appears. Instead, you can assign a *key* of your choice to the figure and refer to it by that key. The **key is assigned a number** by the `\label` command, and the **number is printed** by the `\ref` command.

[1] An error in your input file could produce an error in one of the special cross-referencing files. The error in the cross-referencing file will not manifest itself until that file is read, the next time you run LaTeX. Section 8.1 explains how to recognize such an error.

In the example below:

- the `\label{example}` binds the subsection number to the key `example`
- the `\label{eq:euler}` command binds the equation number to the key `eq:euler`
- `\ref{eq:euler}` and `\ref{example}` can be then used to reference the equation and the subsection

1.1 Example

Euler's equation

$$e^{i\pi} + 1 = 0 \tag{1}$$

combines the five most important numbers in mathematics in a single equation.

Equation 1 in subsection 1.1 is Euler's Famous result.

1.2 Keys, Labels, Page References, Captions

1.2.1 Keys

A key can consist of any sequence of letters, digits, or punctuation characters (Section 2.1). Upper- and lowercase letters are different, so "gnu" and "Gnu" would be distinct keys. In addition to sectioning commands, the following environments also generate numbers that can be assigned to keys with a `\label` command: `equation`, `eqnarray`, `figure`, `table`, `enumerate` (this assigns the current item's number), and any theorem-like environment defined with the `\newtheorem` command of Section 3.4.3.

Using keys for cross-referencing saves you from keeping track of the actual numbers, but it requires you to remember the keys.

Possibly incorrect or outdated: You can produce a list of the keys by running LaTeX on the input file`lablst.tex`. (You probably do this by typing "latex lablst" ; check your Local Guide to be sure.) LaTeX will then ask you to type in the name of the input file whose keys you want listed, as well as the name of the document class specified by that file's `\documentclass` command.

One method that worked: open the .aux file and look there[2].

1.2.2 Labels

When a `\label` command appears in ordinary text, it uses the number of the current sectional unit as key. A `\label` command appearing inside a numbered environment assigns the environment's number to the key.

[2]https://tex.stackexchange.com/questions/525394/get-a-list-of-all-labels-in-a-tex-document

- To assign the number of a sectional unit to a key put the `\label` command in the argument of the sectioning command. (Technically, you can put the `\label` command anywhere within the unit, except within a command argument or environment in which it would assign some other number.)
- To refer to an equation in an `eqnarray` environment, put the `\label` command anywhere between the `\\` or `\begin{eqnarray}` that begins the equation and the `\\` or `\end{eqnarray}` that ends it
- The position of the `\label` command in a figure or table **must** go after the `\caption` command or in its argument
- A `\label` can appear in the argument of a sectioning or `\caption` command, but in no other moving argument

More detail: The `\label` command writes an entry on the aux file; this entry contains the key, the current `\ref` value, and the number of the current page.

When this aux file entry is read by the `\begin{document}` command[3], the `\ref` value and page number are associated with the *key*. Then, a `\ref{key}` or `\pageref{key}` command will produce the associated `\ref` value or page number, respectively.

The `\label` command is fragile[4], but it can be used in the argument of a sectioning or `\caption` command.

1.2.3 Page References

LaTeX maintains a current `\ref` value, which is set with the `\refstepcounter` declaration (Section C.8.4). (This declaration is issued by the sectioning commands, by numbered environments like `equation`, and by an `\item` command in an enumerate environment.)

The `\pageref` command produces the page number where its corresponding `\label` command appears. (A `\ref` or `\pageref` command generates only a number, so if you want to produce something like "page 42", you have to type the word 'page'.)

The numbers generated by `\ref` and `\pageref` were assigned to their corresponding keys the previous time you ran LaTeX on your document. Thus, the printed output will be incorrect if any of these numbers have changed. LaTeX will warn you if this may have happened, in which case you should run it again on the input file to make sure the cross-references are correct. (This warning will occur if any number assigned to a key by a `\label` command has changed, even if that number is not referenced.)

Each `\ref` or `\pageref` referring to an unknown key produces a warning message; **such messages appear the first time you process any file containing these commands**.

[3] the next time LaTeX is run on the same input file

[4] https://tex.stackexchange.com/questions/4736/what-is-the-difference-between-fragile-and-robust-commands-when-and-why-do-we-n

1.2.4 Captions

A `\caption` command within its `figure` or `table` environment acts like a sectioning command within the document. Just as a document has multiple sections, a figure or table can have multiple captions.

2 Splitting Your Input

A large document requires a lot of input. Rather than putting the whole input in a single large file, you may find it more convenient to split the input into several smaller files. Regardless of how many separate files you use, there is one that is the root file; it is the one whose name you type when you run LaTeX.

The `\input` command provides the simplest way to split your input into several files. For example, the command `\input{gnu}` in the root file causes LaTeX to insert the contents of a secondary file `gnu.tex` right at the current spot in your manuscript. The input files are not changed, and the secondary file `gnu.tex` may also contain an `\input` command calling another file that may have its own `\input` commands, and so on.

Besides splitting your input into conveniently sized chunks, the `\input` command also makes it easy to use the same input in different documents. While text is seldom recycled in this way, you might want to reuse declarations. You can keep a file containing declarations that are used in all your documents, such as the definitions of commands and environments for your own logical structures (Section 3.4). You can even begin your root file with an `\input` command and put the `\documentclass` command in your declarations file.

Continue reading on page 73 to learn about how to use the `\include` command, instead of `\input`, to only run LaTeX on part of the document (although it might be more trouble than it's worth).

3 Making an Index or Glossary

There are two steps in making an index or glossary: gathering the information that goes in it, and generating the LaTeX input to produce it. Section 3.1 describes the first step. Section 3.2 describes the second step if you don't use the *MakeIndex* program (which is the easiest way to do this and is separately described in Appendix A.)

3.1 Compiling the entries

Compiling an index or glossary is not easy but LaTeX can help by writing the necessary information onto a special file. If the root file is named `myfile.tex`, index information will be written on a file called `myfile.idx`.

LaTeX makes an `idx` file if the preamble contains a `\makeindex` command; and the information on the file is written by `\index` commands. If there is no `\makeindex` command, the `\index` command does nothing.

Example: to index an instance of the word "gnu" in the phrase "The gnu drinks water from the river", you would type `The gnu\index{gnu} drinks water from the river`. If this word were in page 42, the command `\index{gnu}` causes LaTeX to write `\indexentry{gnu}{42}` on the `idx` file. (It's best to put the `\index` command next to the word it references, with no space between them; this keeps the page number from being off by one if the word ends or begins a page.)

Example

```
When in the Course of
\index{human events)%
\index{events, human)%
human events, it becomes necessary for one people to dissolve the political bands ...
```

In this example, the % ends a line without producing any space in the output (but you can't split a command name across two lines).

The procedure for making a glossary is completely analogous: there is a `\makeglossary` command to make a `glo` file which has `\glossaryentry` entries created by a `\glossary` command.

The `\index` and `\glossary` commands are fragile[5]. Moreover, an `\index` or `\glossary` command should not appear in the argument of any other command if its own arguments contains any of LaTeX ten special characters.

3.2 Producing an Index or Glossary by Yourself

If you don't use the the *MakeIndex* program, you can use the `theindex` environment to produce an index in two-column format. In this scenario:

- Each main index entry is begun by an `\item` command
- A subentry is begun
- A subentry is begun with `\subitem`, and a subsubentry is begun with `\subsubitem`
- Blank lines between entries are ignored
- An extra vertical space is produced by the `\indexspace` command, which is usually put before the first entry that starts with a new letter

[5] https://tex.stackexchange.com/questions/4736/what-is-the-difference-between-fragile-and-robust-commands-when-and-why-do-we-n

Example

```
\item gnats 13, 97
\item gnus 24, 37, 233
\subitem bad, 39, 236
\subsubitem very, 235
\subitem good, 38, 234
\indexspace
\item harmadillo 99, 144
```

There is no environment expressly for glossaries. (However, the description environment of Section 2.2.4 may be useful.)