LaTeX Workshop

TeXnical Specials

In this, the last part of our LaTeX workshop, we will be looking at a few special features, giving you advice on the structure of complex documents, and examining PDF and HTML data export facilities.

BY HEIKE JURZIK AND STEVEN GOODWIN

LaTeX is geared to producing long documents, including books with 1,000 (or more) pages. Storing such a manuscript in a single file is obviously inefficient, so LaTeX supports three commands that allow you to store each chapter (or section [1]) of an extremely long document in separate files. These files, named with the special *.tex* suffix, are then loaded and parsed by a central document using the commands *\input*, *\include* and *\includeonly*.

The simplest of these three is *\input{file}*, which will insert the contents of the file *file.tex* at the location of the command. The file itself (being only part of a document) should not include a preamble, a *\begin{document}* or *\end{document}* command of its own.

\include{file} provides similar functionality to *\input*; that is, the insertion of a specified file at the current location in the text. However, it really comes into its own when used in conjunction with the *\includeonly* command.

As time goes on, and your book increases in size, the number of *\include* and *\input* will grow. Processing the entire book, just to check on a couple of changes, is an inefficient use of time.

The *\includeonly{filelist}* command comes to rescue here, and indicates the files that should only be included when processing the file. Those marked with *\input* will be merged into your file, regardless; those with *\include* will not.

So, if your 20 chapter book is nearly finished, and you only want to polish up a single chapter, a simple *\includeonly{chapter17}* command will ensure that LaTeX only adds the modifications for chapter 17. The other chapters will be ignored, although they will still appear in the table of contents. Even the cross references to passages in the chapters you have excluded will be resolved correctly, thanks to the magic of *\includeonly*.

It makes sense to use a combination of *\include{file}* and *\includeonly{filelist}* if you only need to re-format a few sections, as this avoids the need to modify or comment out individual *\include{}* commands in the document. Simply add an appropriate entry to the header (see Figure 1).

LaTeX manages this feat of magic by deriving information on chapter, page, and footnote numbers from your file when you first parse your document with *latex*. This generates an *.aux* file for each document section, which also includes a count of the number of paragraphs, sections and figures (amongst many other things).

One issue that might prevent you from taking this approach is the fact that LaTeX needs to parse all your external documents if any of the component parts you are adding (or modifying) change the enumeration scheme. In this case, you should use the standard comment symbol (a percentage sign, %), at the start of the *\includeonly{}* line; LaTeX will then parse all documents specified by *\include{}* commands and build or modify the *.aux* files.

Far and Away

Today, many documents are not only printed, but also published electronically on the web. You can use the *latex2html* program which, as the name suggests, translates LaTeX documents to HTML format, complete with multiple links.

If this program is not installed on your machine, you will find it at *ftp://ftp.dante.de/*. When run, the *latex2html file.tex* command will create a new directory called *file* which will store the generated HTML (as *index.html* and *file.html*) along with a short "About this document" page, including the fact it was created with *latex2html*. Any images embedded in the document need to be copied into the *file* directory.

There are several command line options available to influence the behavior of

Figure 1: Using \includeonly{filelist} to modify external files

latex2html. One that we find particularly useful is the *-split [number]* parameter, which defines the nesting depth. By default, the tool creates separate HTML pages down to the eighth level:

- 0 – whole document
- 1 – *\part*
- 2 – *\chapter*
- 3 – *\section*
- 4 – *\subsection*
- 5 – *\subsubsection*
- 6 – *\paragraph*
- 7 – *\subparagraph*
- 8 – *\subsubparagraph*

Thus, calling

```
latex2html -split 0 file.tex
```

will create a single HTML file from your LaTeX document (see Figure 2).

If your main aim is to export a file to HTML, you can utilize a few of LaTeX's special features by using the *html* package and creating portions of your text in either a *latexonly*, or *htmlonly*, environment. This allows you to specify alternative HTML code at certain points, which is especially useful for the graphical elements of your document.

```
\usepackage{html}  % in the ➥
preamble
[...]
\begin{latexonly}
\begin{figure}[H]
\centerline{\psfig{figure=➥
screen1.eps}}
\end{figure}
\end{latexonly}

\begin{htmlonly}
\begin{rawhtml}
<img src="screen1.jpg">
\end{rawhtml}
\end{htmlonly}
```

latex2html will not process anything in the *latexonly* environment, in the same way that *htmlonly* blocks will be ignored for normal LaTeX operations. You can even use HTML tags in a *rawhtml* environment, within *htmlonly*. In this case, *latex2html* will not try to understand the resulting TeX code, and copy it blindly into the HTML document.

latex2html can process complex structures, such as cross references, tables, or pictures with great ease. However, the converter has a few issues – most notably with mathematical formulae. Representing these in HTML is difficult because they are not rendered correctly by most browsers. As a solution, *latex2html* converts them to bitmaps which are then embedded into the HTML. Ultimately, working with HTML gives you, the author, very little influence on the external appearance of your work because of vagrancies within the user's browser, so we might wish to pursue a different format.

Top Gun

PDF ("Portable Document Format") is an operating system independent standard created by Adobe to facilitate document exchange, where the layout of the PDF file is independent of the output medium. Linux has two main programs that can display PDF documents; Xpdf [3] and the Acrobat Reader [4].

Following the introduction of version 3.0 in 1996, PDF offers a whole range of additional functions, such as embedded links to internal and external resources. LaTeX can also facilitate the creation of PDF documents.

There are two approaches to creating PDF documents:

- Either, create a LaTeX document, then, after calling *latex*, use *dvips* to create a postscript file, and use *ps2pdf* to create a PDF file
- Or, create a LaTeX document with PDF extensions in place, and call *pdflatex* instead of *latex*.

We shall look at the second method, by showing what changes need to be made to your LaTeX files to create PDF documents directly from *.tex* files. Firstly, you need to include some additional packages to the preamble of your document:

```
\usepackage[pdftex,dvips,xdvi]➥
{graphicx}
\usepackage[backref]{hyperref}
\usepackage{times}
\usepackage{thumbpdf}
```

You may be familiar with *graphicx* from Part 1 of our workshop [2] – the parameters in square brackets specify *pdftex*, *dvips*, and *xdvi* as output drivers. The *hyperref* package extends the functionality of LaTeX cross references, which in turn can be converted to hyperlinks. Since *hyperref* overwrites some LaTeX commands it is often included as the final package, to make sure that nothing can overwrite *it*. The *backref* option in *hyperref* creates a so-called back reference to the text passage, and also adds links into the bibliography.

Additionally, we shall be utilizing a different font – *times*. By default, LaTeX uses fonts that have been optimized for productions in print. However, these fonts are more difficult to read on-

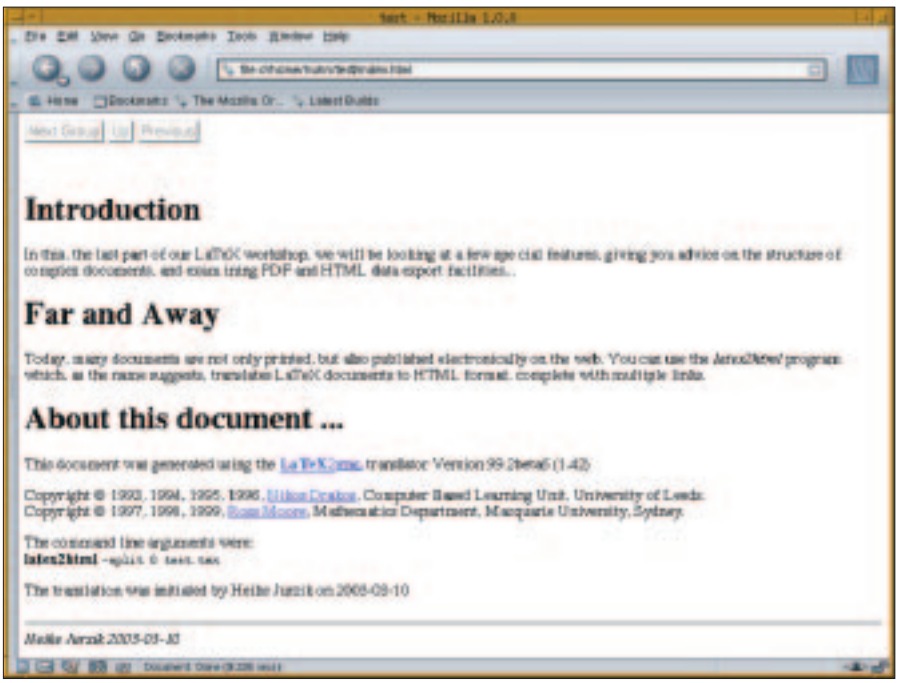

Figure 2: latex2html at work

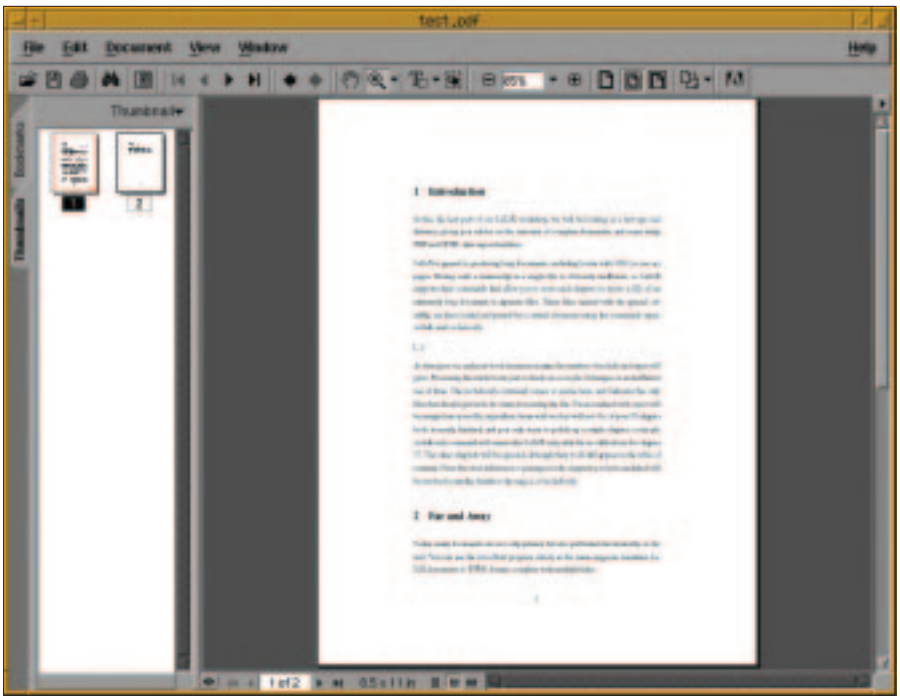

Figure 3: Thumbnails facilitate navigation in Acrobat Reader

screen, and so we must choose a more suitable one.

The *thumbpdf* package creates miniature pictograms (or thumbnails) for the pages of your document, which make for easier navigation through a document. Acrobat Reader supports this feature as seen in Figure 3.

If you need to embed images, ensure that they are available in a pixel-orientated format (such as *.png* or *.jpg*), because *pdflatex* cannot handle PostScript images. *pdflatex* does, however, supports the vector-based PDF format, should you need it.

In order to use thumbnails, you need to make at least two calls to LaTeX, and one to *thumbpdf*. The first call to LaTeX will generate the images that the *thumbpdf* program will incorporate into the final document. The following steps demonstrate the commands required for a TeX file called *test.tex*:

```
pdflatex test
thumbpdf test
pdflatex test
```

If you use *\tableofcontents* to create a table of contents, this will automatically be hyperlinked in Acrobat Reader: you can then click on a line in the contents to jump to the first page of the appropriate chapter, or section. Cross references (*\label{X}* and *\ref{X}*) are linked in a similar fashion.

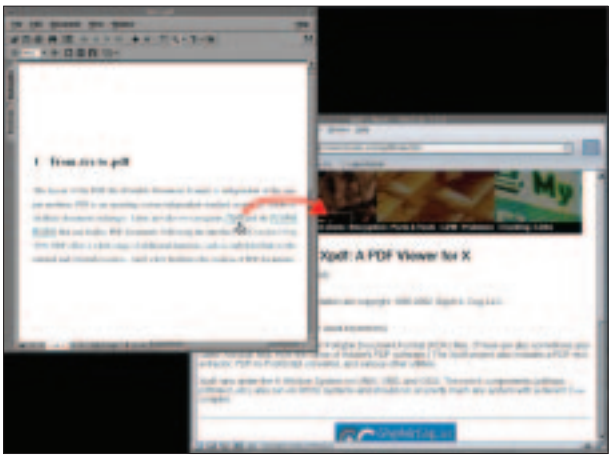

Figure 4: Out of the PDF, into the browser

Out of Africa

Such labels are fine for linking internally to other parts of the same document. But in the world of modern electronic communications, this is no longer enough: we want to link to web sites for updated information, or to sites with more involving discussions than our work includes. Such connections are called external links, and can be referenced by our *.tex* file with the *\href* environment:

```
\href{http://www.foolabs.com/xpdf/home.html}{XML}
```

After running the *pdflatex* program, you will be able to click on those links in Acrobat Reader and go directly to the web site in question. In order for this to work, however, you must configure the reader, and specify a web browser. To configure the weblink preferences go to *Edit/Preferences/Weblink...* From here you can choose the web browser application. Click on browse and add your least unfavorite browser to the selection, for example */usr/bin/mozilla*. If hunting through the filesystem to find where your browser is hidden is not your idea of fun, a quick trip to the command line will help:

```
which mozilla
```

This will return the full path to Mozilla.

Once configured in this way, clicking on any external link in the Acrobat Reader will start Mozilla and take you to the appropriate web page (see Figure 4).

Days of Thunder

As we've seen, there's a lot more to writing a document than just writing. The extra work involved in splitting the document into chapters will more than pay for itself by the reduced processing that LaTeX has to do. And our increased audience, after converting into HTML or PDF, will more than reward our effort.

The extra commands we have to add might not be big – but they are clever – so there's no reason not to make good use of them. ■

INFO

[1] Heike Jurzik: "Tidying up Documents" – LaTeX-Workshop: Part 2, Linux Magazine, Issue 28, p54

[2] Heike Jurzik: "Making up with LaTeX" – LaTeX-Workshop: Part 1, Linux Magazine, Issue 26, p46

[3] *http://www.foolabs.com/xpdf/home.html*

[4] *http://www.adobe.com/products/acrobat/readstep.html*