Making a Book from Contributed Papers: Print and Web Versions

Harry Payne
Space Telescope Science Institute
3700 San Martin Drive
Baltimore, MD 21218, USA
payne@stsci.edu
http://www.stsci.edu/~payne/

Abstract

This paper describes a set of tools used for several conference proceedings projects. Processing of a run-file for the individual components is managed by the UNIX make utility, and performed with perl and sed scripts. One advantage to this scheme is that references to other papers in the same volume may be supplied with a page number in a straightforward way. Another is that the entire book may then be processed with latex2html to produce a Web version from the same set of input files. The tools will be made freely available via the Internet.

Introduction

In my field (astronomy), it is common practice for conference proceedings to be generated from a set of contributed papers, each of which is a stand-alone LaTeX document. Editors are expected to not only edit the individual contributions but also combine them into a book, with a table of contents and proper page numbers. But publishers generally provide little more than a style file and instructions for the individual authors.

In this paper I will discuss my experience with turning LaTeX manuscripts into a book and into an on-line volume on the Web. I will describe some tools I have written for this purpose, after giving some general comments that may save you some work if you are faced with a similar task. But first, I will describe how I got started on this, and the choices and constraints we faced.

My job is in astronomical software, and in 1994 my institute hosted the fourth annual meeting in a series devoted to Astronomical Data Analysis Software and Systems (ADASS). I volunteered to be a proceedings editor, at least in part thinking that I could enjoy the meeting since my work would all come later (unlike TUG, we go to the meeting first, and then write our papers). The proceedings of the first three meetings were already published, so we knew what our book had to look like. The Astronomical Society of the Pacific (ASP) provided instructions to the authors. Previous editors used perl scripts to kept track of the number of pages in each contribution, so that each paper could be printed with the right page numbers, and a table of contents and index could be generated. But I chose to depend on LaTeX itself as much as possible.

The ASP kindly allowed the editors of the 1993 volume to translate the proceedings into HTML for access over the Web. The Web was new in 1993, and although this first Web volume was attractive, I wanted to do things differently. The first version of Nikos Drakos' LaTeX2HTML had appeared in the meantime. I decided that I wanted to do the entire book as a single LaTeX document and then feed the whole thing to LaTeX2HTML.

I settled on a scheme to edit each contribution as a stand-alone LaTeX document, but to add information for use in the book, such as index and author index entries, and cross references between contributions (fairly common since the authors have already heard one another's talks). The contributions are preprocessed into chapters, and a skeleton document pulls in each chapter to make a book. Front and back matter are created by LaTeX in a straightforward way. The UNIX make utility keeps track of all the pieces, updating the book if any of the papers is changed.

The on-line version of the book is produced from the same files used to produce the printed volume, processed by LaTeX2HTML. The output from LaTeX2HTML is modified with the help of some perl scripts to obtain the final product. Reprints of the papers and a searchable index are provided.

I have used this scheme on about a half dozen books, and other editors have used it on a few more. The editors of the most recent volume in the ADASS series[1] have made some improvements that I will mention.

---

[1] D. Mehringer, R. Plante, and D. Roberts, University of Illinois.

Editing the Book

As I was editing my first book with this scheme, I made notes to myself every time I went through all of the contributions to fix something. Here are some examples:

- "Took out multiply named labels."
  The instructions to authors said to label your first figure with `\label{fig-1}`. When I put all of the contributions together into one document, `fig-1` was used many times, of course.

- "Added author index information."
  I cloned the `\index` macros to provide a way to compile an author index.

- "i.e. and e.g. are followed by a comma, and are not italicized." "Em dashes don't have spaces around them." "No bold, italics, or Greek symbols in titles or section headings."
  And many other substitutions as well.

- "Work on tables."
  The editors decided that each table would have two `\hline`s, the headers, one `\hline`, the body, and two final `\hline`s.

- "Work on references."
  Describing the new *TUGboat* macros, Robin Fairbairns writes "Bibliographic citations give much grief to the editorial team" (1996, p. 286). This is a mild statement of the situation faced by editors in technical fields. North American and European astronomers use different citation formats, and BibTeX is rarely used.

The common thread connecting these items is that if we had made these decisions and properly instructed our authors before they submitted their papers, we could have saved ourselves a lot of work. Time spent providing your authors with the information they can use will repay itself many times over.

In this spirit, the editors of the most recent volume in the ADASS series came up with a way to add cross references to other papers in the same volume. Authors were instructed to label their papers with the identifier given in the meeting program, and to use this identifier to refer to other papers. In the absence of such instructions, inserting these cross references requires searching for all variations of "this meeting," "this conference," "these proceedings," "this volume," etc. These editors also required authors to make their own author index entries. Instructions for all recent ADASS volumes describe how to refer to Web resources in ways that turn into live links in the on-line version.

You cannot give authors enough information to make their own entries in a real index — making an index is an iterative process that depends on seeing all of the entries. We provide a `\keyword` macro for authors to provide a half dozen keywords or so, just as a starting point for the editors. Our astronomical software series now has enough volumes that we can ask authors to use previous volumes as a guide, but authors must be free to create.

Once the papers and their associated figures have been collected, you can finish editing them only after deciding the trade-off between uniformity and the author's own voice. Making all of the tables look the same was a worthwhile effort. On the other hand, we foolishly once replaced all British spellings with their American equivalents. In the case of authors who are not native speakers of English, you will have to decide when your voice is preferable to the author's — when unidiomatic phrasing is more confusing than colorful.

Finally, if PostScript figures accompany the papers, plan on spending time fiddling with the figures to make sure that the individual papers all print on your system.

Making the Book

Once you have finished editing the papers, you can create the structure for making the book. You need to convert each paper from a stand-alone document into a chapter in the book and create a skeleton document for pulling all of the pieces together.

Depending on the tools you have available, you may not have to wait until you have final versions of the papers. I work with the UNIX operating system, and the scheme I use depends on the `make` utility program. Programmers use `make` to manage software projects. You tell it which components depend on which others and specify rules for building the dependent pieces from the independent ones. A program might depend on a number of files containing source code. After editing any of the source code files, the programmer runs `make` to recompile only the changed files and then build an updated version of the program.

Each paper's `.tex` file is preprocessed to become a chapter in a file with the same name but using the `.ltx` extension. In the `make` configuration file (Makefile), I list all of the `.ltx` files I will need

```
PAPERS =  \
   accomazzia.ltx agafonovm.ltx \
   alexova.ltx antunesa.ltx \
   ballesterp.ltx ...
```

and provide a rule that accomplishes the preprocessing. Below is an example using `sed`, the UNIX stream

Harry Payne

editor, just because all of the pieces are visible; you could use perl or a compiled program instead.

```
.tex.ltx:
    sed "s/\\\\documentstyle/{% &/" $< |\
    sed "s/\\\\begin{document}/% &/" |\
    sed "s/\\\\nofiles/% &/" |\
    sed "s/\\\\title/\\\\chapter/" |\
    sed "s/\\\\begin{abs/\\\\label{$*}&/"|\
    sed "s/\\\\end{document}/}% &/" > $@
```

To translate, this is a rule for converting a `.tex` file into a `.ltx` file. Each line performs a substitution, commenting out `\documentstyle`, `\nofiles`, and `\begin` and `\end{document}` commands, replacing `\title` with `\chapter`, inserting a label containing the name of the file ahead of the abstract, and putting braces around the entire paper to limit the damage when authors define their own macros. "$<," "$@," and "$*" are `make` macros for the source file, target file, and filename root, respectively.

For the skeleton document, I found it convenient to define a `\paper` macro which, in addition to an `\input`, gives the list of author names as they should appear in the table of contents, as well as the title and list of authors as they should appear in the headers:

```
\paper{accomazzia}{A. Accomazzi, G.
    Eichhorn, M. J.  Kurtz, C. S. Grant
    and  S. S. Murray}{Accomazzi, Eichhorn,
    Kurtz, Grant and Murray}{Mirroring the
    ADS Bibliographic Databases}
```

I like each `\paper` command to be on a single long line, to make it easier to rearrange their order. The ADASS papers alternate author names and affiliations, making it tricky for a program to sort them out. I confess to generally using cut and paste to build these commands, but a perl script can give you a head start. Finish up the book skeleton with a table of contents, and front and back matter of your choice, such as a preface, list of participants, conference photograph, and colophon.

The Makefile says that the book's `.dvi` file depends on the chapter files and the index file

```
book.dvi: $(PAPERS) book.ind
```

As well, the Makefile has rules for obtaining `.dvi` and index files:

```
.tex.dvi:
        $(LATEX) $*

.tex.idx:
        $(LATEX) $*

.idx.ind:
        makeindex $<
```

With these pieces in place, the `make book.dvi` command will create all of the chapter files, run LaTeX to make the `.idx` file, run makeindex to generate the `.ind` file, and then run LaTeX again to make the `.dvi` file. If you go back and edit the papers, you can update the book just by running this command again — only the papers you edited will be processed into chapters again.

A style file for the book defines the `\paper` macro, and redefines the `\chapter` and table of contents macros to match the format required by the publisher. A number of details are also addressed: page format, macros for references to earlier volumes in the series, the author index, and proper numbering of figures, equations, appendices, etc. I wrote macros that add the publisher's copyright notice to the first page of every paper, which I use to produce a version of the book that is sliced up into reprints for the on-line version.

There are Makefile dependencies to control the production of the final printed copy to be sent to the publisher. Our publisher reduces the camera-ready pages we send, so I let LaTeX work with pages and fonts that are the same size as the book, and then ask the dvips program to enlarge the pages to the size requested by the publisher.

In theory, once you have finished editing and produced a final printout, you can send it to the publisher and turn your attention to the on-line version — after all, it is desirable that printed and on-line versions be identical. In practice, I strongly suggest that you wait. At this point, you will be tired of looking at the text, but the first look at the on-line version will make it all fresh again. I guarantee that you will find things to change in the printed version.

Making the On-line Version

Running LaTeX2HTML on a large book does impose some requirements on the hardware you use. Because the program holds the entire book in memory, you will need a large amount of virtual memory — for a 500-page book, I have seen LaTeX2HTML require 750 MB of memory. Aside from this, there is nothing too unusual about the way I use LaTeX2HTML. But I do manipulate its output to produce the final product.

Neither the papers nor the book skeleton file needs to be modified to produce the on-line version. But the Makefile does contain a rule for creating a new skeleton file from the original. The LaTeX2HTML program runs a separate task to expand `\input` instructions; instead of modifying this task I create a

new skeleton file, where all of the `\paper` commands are demoted to ordinary `\input` commands.

LaTeX2HTML allows for customization using perl style files (LaTeX2HTML itself is written in perl). My customizations are either routines to translate new macros into HTML or routines which change the default behavior of LaTeX2HTML. For example, my simple macro for referring to the fifth volume of the ADASS series

```
\def\adassv{in Astronomical Data Analysis
    Software and Systems V, ASP
    Conf.~Ser., Vol.~101, eds.
    G.~H. Jacoby \& J. Barnes
    (San Francisco: ASP)}
```

is implemented with a routine that simply inserts the same text, translated into HTML, into the output stream:

```
sub do_cmd_adassv {
  local($_) = @_;
  join('',"in Astronomical Data Analysis
    Software and Systems V, ASP
    Conf. Ser., Vol. 101, eds.
    G. H. Jacoby &amp; J. Barnes
    (San Francisco, ASP)$_");}
```

The changes to the default behavior are primarily in the navigation panels, where I wanted a particular format, links to an index and reprint files, and a copyright notice. Other changes suppress the index, which I make in a different way, and make sure that chapters are not numbered.

The Makefile has a rule for running LaTeX2HTML on my new skeleton file to produce HTML and image files. Most ADASS papers are short, so I let LaTeX2HTML put each paper into its own HTML file. I also select options which keep footnotes with each paper (rather than breaking them out into separate files) and which add section numbers, an HTML title for the book, and an address at the bottom of every page. Using a Sun SPARCstation 5, running LaTeX2HTML on a 500-page book with 120 PostScript figures and a lot of equations takes the better part of an afternoon. If you have to do it again, it will go much faster if you can re-use the image files. After LaTeX2HTML has done its job, you need to examine the output with a Web browser, looking for problems that might force you to do the conversion again: broken image files or images that are out of sequence.

When you are satisfied with the output from LaTeX2HTML, you are ready to make the final product. I like to rename the output HTML files for the papers, restoring the original name. To do this, I combine information in the `.aux` file from LaTeX

and the `labels.pl` file from LaTeX2HTML to create a file containing three columns: the new file name, the original file name, and the page number in the printed version:

```
accomazzia.html node99.html  395
agafonovm.html  node14.html  58
albrechtm.html  node91.html  363
albrechtr.html  node62.html  248
```

I then run a script which renames the files and updates all of the links. The page numbers are used for making the table of contents and index (so you can determine a citation to the book from the on-line version). The table of contents produced by LaTeX2HTML will not contain the author names, so I replace it by running a script on the `.toc` file from LaTeX. While working on the table of contents, I put all of the front and back matter together under a "Preface" heading. I generally make a version of the book cover by hand and use either this cover or the table of contents as the entry page for the on-line book.

One of the advantages of the on-line version of the book is the ability to do full-text searching. There are many options for indexing software, and many are free, but your choice will depend on your Web server platform. I have used WAIS, a somewhat dated search technology, since I already had a WAIS server available. Every page in the on-line version has a link to an index page from which the user can perform a search. Although this mechanism provides a powerful way to find what you are looking for, I also translate the author and subject indices from the book into HTML, using perl scripts and information from the `.ind` file produced by LaTeX.

We published the on-line version of the book with the permission of the publisher. As a courtesy, we put the publisher's copyright notice on each HTML page (making the notice into a link taking users to the publisher's home page) and each reprint. I also added instructions for obtaining a printed copy of the book by way of the publisher's website. Except for the searchable index, all of the links in the on-line version are relative. This makes it easy to create mirror sites.

You can spend as much time as you like on projects like JavaScript commands to identify people in the conference photograph just by moving the mouse. However, you can realistically expect to produce an on-line volume with a few days of effort, once the printed volume is ready.

Harry Payne

Trying it Yourself

You can find the files you need to try out this scheme at `ftp://ftp.stsci.edu` in `/pub/software/tex/bookstuff/`.

The most recent on-line volumes produced this way can be found at `http://www.stsci.edu` in `/stsci/meetings/lisa3/` and `/stsci/meetings/adassVII/`.

References

[1] Fairbairns, Robin. "The New (LaTeX 2ε) *TUGboat* Macros." *TUGboat* 17,3 (1996), pp. 282–288.

TUG '99 musings

Mr. Fine, with the eye of the sleuth
has discovered why TeX is uncouth,
    that its catcodes are many
    when there shouldn't be any
Has anyone told Donald Knuth?

Mr. Flynn is now running on La-
TeX, and finds, when the tread starts to fray,
    he can patch up the rules
    with his vulcanize tools
pump them up, and back on his way

Though some of you authors may fret
about how every page should be set
    and think what you see
    and what it must be
Mr. Bazargan says what you get

           —Pierre MacKay

Ode to a special

Oh what a tangled web we get
When first we find `\def` and `\futurelet`.
We turn to eTeX, we turn to Omega,
They both look good, we start to get eager.
But no, whats this, a misplaced `\omit`?
Did we miss a brace, does `\vbox` still not fit?
Help me fix my macros, bracket heroes all!
But Pierre's font is calling, Lamport's left the hall
Barbara says use `\downcase`, Downes just strokes his beard.
Erik sells me 4TeX, pretends he never heard.
David's hacking tables, Frank is fixing floats,
Ogawa's talking slowly, Kacvinsky wants his oats.
Will Kaveh help me out? no, his dhoti's dirty,
Nelson's feeding awks, Mimi's feeling flirty.
Flynn says use a rubber, Wendy needs a hug,
Kath has got the answer, no it's just another mug.
Young Ross is such a hoot, he says use xypic
Why not turn to Sojka, he's sure to know the trick.
Anita, a Hoover, what use to me's a dam?
Send me out with Kiren, we'll both go on the lam.
Irina claims 'for us in Mir is no problem',
TRIUMF uses `\mathcode` but `\hspace`s just one em.
TrueTeX does it both ways, and you can trust the Blue Sky
but hyperref the backend, oh why oh why oh why?

ActiveTeX, PassiveTeX, what a great to do,
Can there be yet some way through?
MathML has pointies, XSL can claim its templates
ExerQuiz is **so** cute—so phooey on you Billy Gates.
           —Sebastian Rahtz