Managing Large Projects with PreTEX: A Preprocessor for TEX

Robert L. Kruse
PreTEX, Inc.
2891 Oxford Street
Halifax, Nova Scotia, B3L 2V9
Canada
bob@pretex.com

Abstract

PreTEX is a preprocessor for TEX that supplies an author with many tools to simplify the writing and management of larger (book-length) projects. This paper concentrates on PreTEX's use of secondary input files and conditional typesetting in managing large projects. The user may insert location tags within a file which can then be used by PreTEX to include parts of one file within another, in any order determined by the user. Parts of a file may also be selectively typeset according to the status of various conditions.

Introduction

Let us consider two projects large enough to challenge most TEX systems.

The first is a large science or math textbook at the high school or elementary college level. Such a book is often over 1000 pages long, requires four-color printing, and contains many hundreds of full-color photographs or other complicated images. It has a large index, bibliography, and cross references throughout. Answers to some of the exercises appear at the back of the book. There may be many supplements, such as a separate solutions manual, a study guide, a lab manual, a bank of test questions, or transparency masters. All of these contain many references that need to be keyed to the textbook itself. There may be a separate book of instructor's notes keyed to the text, or the book may be simultaneously published in a special instructor's edition with the notes in the margins or on extra, unnumbered pages. Some of these many supplements may be published on paper, some on the Internet, some in PDF or other format on CD-ROM with interactive links.

Our second example project is the documentation library for a large software system, one that may contain hundreds of thousands or millions of lines of computer code, written in several different programming languages and distributed over many files. To avoid errors and inconsistencies, it is important to keep all the documentation in the same files with the code. Such a system will have several kinds of documentation: informal introductory guides for the user, reference volumes for more sophisticated users, precise specifications for each part of the program, in-house documentation and other comments from programmers, and bug/modification reports and history.

PreTEX is a preprocessor for TEX (consisting of about 15,000 lines of C code), designed explicitly to facilitate the work of authors and editors in writing and production of large projects such as these. PreTEX's features, moreover, remain equally useful for more ordinary book-length projects or even for small typesetting projects such as research papers.

By exploiting context dependency, PreTEX supplies much of the routine markup required for high-quality typesetting in TEX, simplifies the notation for mathematics, supplies user-friendly error diagnostics, uses its own tables of information to resolve many ambiguities in typesetting, and, by recognizing some of the syntax of various programming languages, provides powerful tools for typesetting computer-program listings.

Some of these features were discussed for a preliminary version of the PreTEX software in Kruse (1988). Since that article was published, the PreTEX software has been substantially revised, extended, and used intensively in a commercial environment. It is now ready for initial public release.

This paper discusses only a fraction of the tools provided by PreTEX. Here, we concentrate on PreTEX's file-management facilities that are especially valuable to authors of large projects. For a discussion of PreTEX's implementation of hypertext links in PDF, see Mailhot (1999).

Independent chapter processing

High-resolution scans of color photographs or other complicated graphics require considerable space in computer storage. If a book contains hundreds of such images, its output files can become prohibitively large, often several gigabytes in PostScript. It therefore becomes imperative to divide such a project into smaller files that can be processed independently.

In fact, PreTEX allows a project to be divided into conveniently small, chapter-length files which are processed independently at *every* stage, including the final production of PostScript or PDF files. At the same time, PreTEX integrates the cross references, the index and contents entries, and the bibliographic citations from *all* the input files comprising the entire project. Hence, while the author works on the files for one chapter, cross references to other chapters will still resolve properly. Page numbers, chapter numbers, and other elements that number consecutively throughout the book are automatically updated for each chapter file.

To accomplish these goals, PreTEX maintains a whole directory of auxiliary files in place of the single auxiliary file used by LaTeX. Indeed, for each chapter there are several auxiliary files that are accessed by PreTEX, by TEX, by BibTEX, and by PreTEX's enhanced equivalent of MakeIndex. Under normal circumstances, the user never needs to look at any of these auxiliary files directly. Since there are separate auxiliary files for each chapter, the system modifies only those corresponding to the chapter currently being developed.

For some purposes, PreTEX must string all these files together in the correct order; to do so, PreTEX uses a short file containing a master list of all chapter files. A sample of this master file list is:

```
title
contents
preface
part1
chapter1
chapter2
part2
chapter3
appendix1
appendix2
index
```

The blank line specifies that the page numbering is not continuous from `preface` to `chapter1`; otherwise each file begins after its predecessor. Contents, index, cross-reference, and bibliographic entries, however, are merged for all the files.

Secondary input files

One of PreTEX's most powerful features is its ability, while processing one file, to include portions of other files, arranged in any desired order, with parts skipped under the control of Boolean (that is, true-false) variables. This feature is like TEX's `\input` command, except that `\input` includes the entire file, whereas PreTEX may select only portions.

First, we need some notation.

PreTEX Command Syntax. As a preprocessor, PreTEX has its own extensive command language, as well as responding to certain TEX commands. It recognizes several different environments and processes its input differently according to the environment. In the mathematics, verbatim, and computer-program environments, the characters '`<`' and '`>`' are processed as usual, but in text (where they would normally not appear) '`<`' and '`>`' are used to delineate commands to the PreTEX preprocessor. Such commands take forms such as

```
<:command_name option1 option2>
```

where the number of options and their syntax vary with the command.

To access a secondary input file, we need only write an instruction such as

```
<:read file filename from tag1 to tag2>
```

at the place where we wish to include part of the secondary file `filename`. The location tags, denoted `<tag1:>` and `<tag2:>`, are placed immediately before and after the part of the secondary file that will be read. Any number of location tags may be placed in a file (at any place where the file is in text environment); these tags are used only to control file reading and will not appear in the output after processing by PreTEX.

Conditional Typesetting. When the same material is processed for different purposes by PreTEX, it is often convenient to use TEX macros both to control how an element is typeset and, depending on the purpose, to determine if the element is included at all. In a textbook, for example, the author may wish to place solutions to exercises immediately adjacent to the exercises themselves, so that, if the exercise is modified, its solution can easily be modified to match. When we are processing the textbook itself, these solutions must not be included in the output, but when typesetting the solutions manual, they must appear.

We can accomplish this goal by instructing PreTEX to create a new Boolean variable, which we might call `solutions`, and to use this variable to

control the typesetting of material between a pair of control sequences, which we might call `\solution` and `\endsolution`. With this instruction, PreTEX will recognize two new commands:

`<:solutions on>` and `<:solutions off>`

Depending on the setting, PreTEX will include or delete material between

`\solution` and `\endsolution`

For the textbook itself, we specify

`<:solutions off>`

(likely in a style file, so it need be done only once for the entire book), in which case all solutions will be deleted. When processing the solutions manual, we specify

`<:solutions on>`

so that all solutions will appear immediately after their exercises.

With these tools, typesetting a separate solutions manual is trivial, and it will be guaranteed to be kept current with any revisions to exercises in the text. All we need to do is to place location tags before and after each group of exercises in the text's chapter files. Then the file for the solutions manual will contain almost nothing except for the command `<:solutions on>` followed by a long series of `<:read file ...>` commands to include each group of exercises and their solutions from the various chapter files.

Some authors, on the other hand, prefer to place all exercises and their solutions into separate files, one or more exercise files per chapter. This organization will work just as well. Now, with location tags before and after each group of exercises, `<:read file ...>` commands in the main chapter files will include the exercises where desired (along with their solutions, which will be deleted, provided `solutions` is `off`).

To place answers in the back of the book is just as easy. We now introduce a second Boolean variable, say `answers`, include the commands `<:solutions off>` and `<:answers on>`, and use the same `<:read file ...>` commands to typeset, at the book's end, only the selected answers.

Production of other supplements for a large book follows much the same plan. The material for these supplements can either be placed in the main files with typesetting controlled by Boolean variables, or placed in separate files used to produce the supplements, with `<:read file ...>` commands as appropriate to include material from the main file or other supplements. In any case, the names of the files for all the supplements should normally be included (in separate groups) in the master file list for PreTEX. In this way, cross references may be made from any of the supplements to any other or to the main text. Similarly, in PDF, hypertext links allow the user to jump directly from locations in the text to appropriate locations in any supplement, or the reverse.

Program code and StripTEX

Now let us turn to our second motivating example, a documentation library for a large software system, where the code and documentation are distributed over many different files, generally with differentauthors.

By treating each file of program code and documentation as a secondary file, PreTEX can be used to combine any desired extracts of the documentation or the program code in any desired order. The identical source files can in this way be used to construct, for example, informal user guides, user reference manuals, programmer's reference manuals, or other documentation, either for in-house use or external distribution.

In PreTEX's different environments, the same symbols will be processed in different ways. PreTEX provides special facilities for typesetting computer programs, understanding enough of the program syntax to adjust spacing and choose special symbols. For example, curly braces `{ ... }` (delineating a group in TEX) become printed symbols in C or C++, enclosing code segments, whereas in Pascal, they delineate comments that will be typeset as text, not as computer code. PreTEX provides environments for many common programming languages (Java, C++, C, Pascal, Ada, Fortran, Basic, Cobol, among others). By treating program lines as tab-controlled lines (as illustrated in *The TEXbook*, page 234), tab stops can be used to achieve proper alignment, or other TEX markup can be inserted into programs.

For these program files, PreTEX provides a further utility, called StripTEX, that removes all the text surrounding program code, as well as any TEX markup in the program code, yielding output that can be submitted directly to a compiler. StripTEX recognizes all the same commands as PreTEX, including reading from secondary files. Hence, the order of subprograms within files and their accompanying documentation need not have any connection with the order expected by the compiler; StripTEX can be used to extract subprograms from arbitrary files and arrange them in whatever order is needed by the compiler. The same file(s), containing both documentation and

code, can be processed through PreTEX to obtain a well-formatted, documented program listing, or through StripTEX to obtain the program in exactly the form needed by the compiler.

In this way, PreTEX and StripTEX provide all the functionality and capabilities of Knuth's `WEB` system. PreTEX replaces `WEAVE`, and StripTEX replaces `TANGLE`. The PreTEX–StripTEX system, moreover, brings several advantages over `WEB`:

- The user has unlimited flexibility in the ordering of subprograms and documentation and their placement in various files.
- The same software manages programs in any number of computer languages, including several for which `WEB` is not available. For software systems using several languages, the identical software generates all the files needed for the various compilers.
- The user has no need to learn a new command language: StripTEX shares the identical syntax of PreTEX, which is an easy extension of TEX.

Bibliographic entries

For large projects with many references, maintaining the bibliography can require considerable work. BibTEX provides excellent facilities for maintaining a bibliographic database; PreTEX uses BibTEX without change. Any LaTEX user of BibTEX will immediately be familiar with the PreTEX commands `<:cite ...>`, `<:nocite ...>`, `<:bibliography ...>`, and `<:bib_style ...>`.

As a preprocessor, PreTEX can automate and streamline the use of BibTEX. LaTEX, for example, requires a four-pass process:

1. Run LaTEX to collect the citation keys.
2. Run BibTEX to assemble the corresponding references from the database(s).
3. Run LaTEX to associate the citation keys with their references.
4. Run LaTEX to resolve the citations.

PreTEX accomplishes the same results with only two passes and with no special attention from the user. On its first pass, PreTEX assembles the citation keys and automatically invokes BibTEX, after which PreTEX immediately associates the citations with their references. On its second pass, PreTEX resolves the citations. Whenever the document is processed again, PreTEX automatically detects if the citations have been changed, and PreTEX invokes BibTEX only when necessary to keep the bibliography up to date.

As our final example, consider a symposium or conference proceedings in which the individual chapters are written by different authors who may not be in touch with one another. Each chapter will then be written and processed independently; if desired, each chapter may have its own cross references, index, or contents. The editor may then later merge all these resources for the entire volume. The editor may supply a bibliographic database for use by all authors, accessed as needed by BibTEX from within PreTEX. In addition to these global citations, PreTEX allows bibliographic citations local to each chapter, so each author may use both the global database and, independently, use local citations from other bibliographic databases. To enable local citations, PreTEX includes a second set of citation commands with '`l`' prefixed to the name of each, such as `<:lcite>`, `<:lbibliography>`, and the like.

Preview

This paper touches on only a few of the tools PreTEX provides to authors to simplify the writing, management, and typesetting of large projects. Together with its preprocessor, the full PreTEX software package contains the StripTEX processor, an auxiliary program for the construction of the index, another for the table of contents, and a large TEX macro package of about the same size and capability as LaTEX, but with a somewhat different design philosophy.

The PreTEX software has been under development and revision for several years, and at the same time it has been used intensively by PreTEX, Inc., for the commercial typesetting of many textbooks in mathematics, engineering, and science. The software is now ready for initial external testing and application, with public release to follow in due course. Before its general public release, however, some work remains to be completed, especially the writing of user guides, reference manuals, and other documentation necessary for the effective application of the PreTEX tools.

Bibliography

Kruse, Robert L. "PreTEX: Tools for Typesetting Technical Books." *TEXniques* No. 7: *Conference Proceedings of the Ninth Annual Meeting, Montreal, August 1988*. Ed. Christina Thiele, pp. 219–226. TEX Users Group. 1988.

Mailhot, Paul A. "Implementing Dynamic Cross-Referencing and PDF with PreTEX." 1999. (See elsewhere in these Proceedings.)