Authors New To TEX Publish a TextBook With a Publisher New to TEX

Samuel E. Rhoads
Information and Computer Science, Honolulu Community College, 874 Dillingham Boulevard, Honolulu, HI 97817
(808) 845-9277
Internet: sam@uhccux.uhcc.hawaii.edu

Abstract

This paper decribes our adventure of writing a texbook using TEX and LATEX, and in working with a publisher, William C. Brown, Inc., who had not worked successfully with TEX in the past. The paper discusses the learning process we went through in learning TEX, in working with a publisher new to TEX, and in writing a textbook. It is hoped that by sharing our experiences, other authors and publishers will realize how easy producing a high-quality book can be, and perhaps some of the mistakes we made can be avoided.

Introduction

In April of 1987, I received a phone call from Mike Gearen who teaches computer science at Punahou School in Honolulu. He had seen a teacher's guide for the Advanced Placement Computer Science course that I'd written, and he wanted to know what programming-in-Pascal textbook I used in my classes. He had been unable to find a book he liked and thought maybe I'd found one I liked. We met a few days later and discussed writing our own book. I'd heard of TEX and thought that using TEX would make the writing of a book easier. Thus began our experiences with TEX and with publishing a textbook.

It's an understatement to say that the four years since that meeting have been exciting. We have seen many frustrating times, but the rewarding and exciting times far outweigh the frustrating ones. I still get excited everytime I see a beautiful page of print coming out of my laser printer.

We purchased our first copies of TEX and LATEX from Addison-Wesley in the summer of 1987, and started writing. The start-up time in learning TEX and LATEX was much shorter than I'd feared. In what now seems like no time at all, we were preparing pages of the book. In fact, we had enough done by the end of the year that we used our laser printer output as a textbook in our classes in the spring semester of 1988. (Although we used LATEX for the preparation of the book, we always have thought of it as using TEX. Unless the distinction is important, for the rest of this article, I'll say "TEX" instead of saying "TEX and LATEX.")

We sent a prospectus and a couple of chapters (done in TEX, of course) to several publishers in late '87 and waited to hear from them. Two of the publishers liked our prospectus, and offered us contracts. In both cases we made it very clear that we wanted to use TEX to prepare the manuscript. Neither publisher had had a successful experience with TEX before. One publisher had had no experience at all with TEX, and the other had only had one, unsuccessful attempt. (More on the unsuccessful experience later.) Both publishers initially discussed what they called "electronic submission of manuscripts." To them, this meant sending the manuscript to them on a disk so it would not have to be typed in again. By this time we were so impressed with the appearance of the book that we were brave enough to insist that the book be done in TEX as a condition of signing the contract. After thinking about the two publishers, and trying to decide which offer was better, we decided to go with William C. Brown Publishers, Inc., and signed the contract on February 3, 1988.

By this time a rough version of the majority of the book had already been completed, and, as mentioned above, we were using it as a textbook in our first course in programming during the spring semester, 1988. We were still working on the later chapters and developing our TEX skills. Rough versions of these chapters were completed in time to hand out to our classes by the time they were needed.

TEX and LATEX Skills

As our work on the manuscript progressed, we decided it would be convenient to define some macros. Our first macro was designed to simplify the printing of ⟨*block*⟩. We saw that we could accomplish this by typing: `$\langle${\it block}$\rangle$` everytime we needed it, but soon realized that there was an easier way. Looking back, it seems like a minor victory but we were pleased when we learned that we could define the LATEX macro:

```
\newcommand{\block}
{$\langle${\it block}$\rangle$}}
```

and from then on whenever we wanted to see ⟨*block*⟩ in our book, we only needed to type `\block`.

As would be guessed, this quickly led to many other macros. We busied ourselves defining macros that were intended to make our typing easier. We decided to collect these macros in a file, `macros.tex`, and to input the file at the beginning of each `.tex` file. It was easy to get carried away with this. In looking at `macros.tex`, I see a macro: `\newcommand{\real}{{\tt real}}`. Even though we had defined this macro, we seldom used it; we just typed `{\tt real}` rather than `\real`.

Fonts. In our book we spend a great deal of time discussing algorithms. We stress the development of an algorithmn prior to the writing of the code. We decided that we should use a special font to represent an algorithm, and we started looking through *Computer Modern Typefaces* for a font to use. We wanted a typeface that reminded the reader of handwriting — algorithms probably should be handwritten rather than typed — and we found `cmff10` and `cmfi10`. Knuth [*Computer Modern Typefaces*, page 28] calls these fonts Computer Modern Funny Font and *Computer Modern Funny Italic*, respectively, and says that `cmfi10` is not quite as "hilarious" as `cmff10`. I'm not sure whether Knuth expected people to actually use these two fonts, or was just defining them for fun. In any case, we don't find `cmfi10` funny at all; we like it. We tried `cmff10`, but it proved too hard to read. I wonder if any other use for either `cmfi10` or `cmff10` has been found by other authors.

The definition for this font — we call it `\al` for "algorithmic" — was placed in `macros.tex`. Thenceforth, to produce: *Store true in Found*, we just typed: `{\al store True in Found}`.

While this worked fine for short examples of a step or two, when we wanted a complete algorithm to appear, we encountered a harder problem: We wanted an algorithm to fit completely on a page and not to be broken where LATEX decided to break the page. We wanted the algorithm to be printed in the algorithmic font we'd chosen. We also wanted to be able to indent the examples of algorithms and code that we presented in the book, and we wanted the indentation to be uniform. This was one of the hardest LATEX problems we had to solve. As with many problem solutions, I'm not sure our solution was the best solution — in fact, I'm sure it isn't — but it works and we've used it since. We played with different ideas for quite awhile, and finally developed the following two LATEX definitions:

```
\newcommand{\balg}
{
\par
\al
\begin{minipage}{\textwidth}
\begin{tabbing}
123\=123\=123\=123\=123\=123\=123\=\kill
\mbox{\ }\\
}

\newcommand{\ealg}
{\mbox{\ }\\
\end{tabbing}
\end{minipage}
\rm
\par
}
```

Then whenever we wanted to write a complete algorithm, we just coded:

```
\balg
START OF Algorithm
.
.
.
END OF Algorithm
\ealg
```

to produce:

START OF Algorithm

.

.

.

END Of Algorithm

The `minipage` environment forces LATEX to keep the entire algorithm on the same page; this sometimes caused ugly page breaks that we had to fix by hand. The `tabbing` environment allowed us to indent our algorithms by just typing `\+` whenever we wanted to move to the right one level of indentation, and `\-` when we wanted to move

to the left one level. The `\mbox{\ }\\` produces an empty line. The `\al` changes to the algorithmic font, and the `\rm` changes back to roman.

The appendix contains an example of how we typed an algorithm for a function that returns the greatest common divisor of two positive integers, and the algorithm as it appeared in our book.

Figures and Pictures. We made great use of LaTeX's figure and picture capability. We defined figures and pictures to be things that floated. Figures and pictures always had captions; indeed every picture was in a figure. As an example of how these figure/pictures were coded, I've included a very simple one in the appendix.

Both of us have decent math backgrounds so the geometry and algebra that were required to get the coordinates of the rectangles, lines, vectors, ovals and circles correct wasn't hard. I imagine that for less mathematically trained writers this would prove intimidating, and thus I would expect other authors to turn to more powerful aids. Indeed, as easy as we found it, it would be nice to know better ways to get the pictures of syntax diagrams and trees drawn.

One figure in particular needs to be discussed. Early in the book we give a skeleton of a complete Pascal program. It was supposed to look like this:

> `program` ⟨*programname*⟩(⟨*filelist*⟩);
> ⟨*block*⟩
> .
>
> **Figure 4.1** The Form of a Pascal Program

Instead, it looked like this:

> `program` ⟨*programname*⟩(⟨*filelist*⟩);
> ⟨*block*⟩
>
> **Figure 4.1** The Form of a Pascal Program

We had decided to put all examples of algorithms and all figures in a different color (blue). We had also agreed to have the final pages of the book done on a high-resolution printer by ArborText, Inc., in Ann Arbor, Michigan. In order to do the second color, the publisher had to "cut out" some of the final copy and print it in blue. (I'm not sure how this is done, so I'm being deliberately vague.) When Wm. C. Brown printed it, they must have thought the period was a flyspeck, or something, and they left it off. That wouldn't have been so bad except on the next page of the book, we wrote, "Did you see the period after the ⟨*block*⟩ in figure 4.1? It's easy to miss, but it's necessary." It sure was easy to miss, it wasn't there.

Wm. C. Brown must be given credit. When they learned of the missing period, they had someone go through the warehouse and put a little dot in every one of several thousand books.

Other Design Issues

Illustrations. We have always been taken with Duane Bibby's illustrations in *The TEXbook* and decided to find an artist to draw illustrations of a computer programmer, a user, and a personified computer for our book. (We considered trying to find Duane Bibby himself, but couldn't muster up the nerve.) After considerable searching, we finally contacted a local caricaturist, Katie Ralston, and had her draw the illustrations. (Wm. C. Brown calls these illustrations "cartoons.") I mention the illustrations since they were the only things in the book not done by TEX. The publisher had to insert them into the final pages prepared by ArborText. We did, of course, leave room for them in the proper places.

Style Files. The design staff at Wm. C. Brown wanted a few changes made in the design of the book. In order to implement these changes, we had to modify the style files. We didn't want to change the `.sty` files themselves, so we made copies of the `book.sty`, `bk10.sty`, `bk11.sty`, and `bk12.sty` files, giving them different names, and made the modifications to those files. Rather than describe all these changes, I'll just describe one, as an example.

We used the `book` style with 10-point type. In this mode, LaTeX causes the running heads to be printed in ten-point, uppercase italics. The design people wanted nine-point upper- and lowercase italics. We searched through our renamed style files until we found the following definitions:

```
\def\@evenhead{\rm\thepage
    \hfil\sl\leftmark}
\def\@oddhead{\hbox{}\sl
    \rightmark\hfil\rm\thepage}
```

We modified these two definitions to:

```
\def\@evenhead{\rm\thepage
    \hfil\small\sl\leftmark}
\def\@oddhead {\hbox{}\small\sl
    \rightmark\hfil\normalsize\rm\thepage}
```

The `\small` that we added simply changed the running head to nine point. The `\normalsize`

changed it back to ten point before printing the page number on odd-numbered pages.

That took care of changing the size of the letters. Now we had to see if we could have it print the head in upper- and lowercase rather than all uppercase. Again searching through our renamed copy of `book.sty`, we found this definition:

```
\def\chaptermark##1{\markboth
    {\uppercase
    {\ifnum\c@secnumdepth\m@ne\@chapapp\
    \thechapter. \ \fi ##1}}{}}
```

We finally figured out that the `\uppercase` was turning the name of the chapter into all uppercase letter. (Actually, it was easy figuring out what it was doing; the hard job was finding it.) By changing the definition to:

```
\def\chaptermark##1{\markboth
    {{\ifnum \c@secnumdepth\m@ne\@chapapp\
    \thechapter. \ \fi ##1}}{}}
```

that is, by just removing the `\uppercase`, the running head was in upper and lower case.

We made several other changes to the style files. We changed the margins so that odd and even pages would print with margins that would make it easy to cut the book down to a $9 \times 7\frac{1}{2}$ format. We changed the definition that printed the caption in figures to print the word **Figure** in bold face. All of these changes took time to figure out what to do, but none were particularly hard to figure out.

Comments From the Publisher

I asked the woman who copyedited our book to send me some feedback concerning our use of TEX to print the book, explaining that I was writing this article. She forwarded my request to Wm. C. Brown's electronic text coordinator, the person responsible for working with authors "preparing manuscripts on disk." Here are her comments regarding Wm. C. Brown's experiences with TEX:

> WCB's initial experience with TEX was not a successful one. About three years ago, Kendall-Hunt had an author working on TEX who produced a math book. Manufacturing was persuaded to purchase the TEX program, as it seemed to be becoming the software of choice for those who wished to produce texts containing math and other types of equations. After several unsuccessful attempts in-house to tailor the files to be compatible with our typesetting system, manufacturing resorted to going to an outside preparer (an engineer) who produced camera-ready pages from laser output.
>
> Approximately a year later, WCB received a computer programming book that was done in LATEX. Outside suppliers using TEX (and its various versions) had been busy writing software to make the TEX program compatible with traditional typesetting systems. We were able to find an outside vendor who produced high-resolution, paged output from a typesetter.
>
> The authors worked with in-house staff on questions of design, layout, and typography. They were cooperative in making adjustments wherever possible to achieve a pleasing format for the text.
>
> The disadvantages of this were that control of the project went out of house. Also, at that time, the choices of typefaces were limited.
>
> The advantages, however, outweighed these disadvantages. Namely:
>
> 1. The authors were willing to make whatever changes were needed because of copyediting, so WCB had no involvement in time or personnel in the updating process. The responsibility for getting "perfect" disks to the vendor was the authors'.
> 2. The vendor worked directly with the authors and was able to solve any start-up problems.
> 3. There was no keystroking required in-house.
> 4. All formatting integrity was maintained.
> 5. WCB received paged output in approximately three weeks — much less time than "traditional" output.
> 6. All graphics were in place on the page except for photos and acquired cartoons, minimizing hand keylining time.
> 7. Corrections were minimal.
> 8. The cost of the project was much lower than if it had been handled traditionally.
>
> The project was such a success that we will continue to consider sending any TEX project to an outside vendor that is capable of sending the files to high-resolution output. There are at least two projects that will be handled that way this year. Progress continues to be made in expanding the capabilities in regards to typefaces and output possibilities.
>
> TEX is also offering various forms of its own product to attract and expand its

> market potential. We continue to monitor these products as there is interest in our math area about TEX products. Many of our math authors and math ancillary authors are using TEX. We may want to test the possibility of having manufacturing using the software in typesetting.
>
> The one drawback may be that TEX does not give the same high quality that may be expected in upper-level textbooks, and it is difficult to integrate it with traditional typesetting modes. Also, one must be careful regarding vendor claims to be able to print pages for nominal charges. There are hidden traps in these offers.

Response. While I don't intend to discuss each of these comments, there are a couple of things I want to say: The "computer programming book" she refers to is, of course, our book. The "vendor" who produced the high-resolution paged output was ArborText, Inc.

It's clear that doing a book this way causes things to be done differently than they have been done in the past. In our case, the copy editor made changes on the hard copy we sent her, and then sent these changes back to us to make. In a few instances, we didn't want to make the change. I suspect that this might have caused some problems.

She has the impression that TEX is only good for math and "math ancillary" textbooks. I hope, and suspect, that the future of TEX will prove it to be the best choice for books of all types.

She indicates that the choices of typefaces were limited. That may have been true, and it may still be true; I'm not sure. In any case, I would have chosen Computer Modern if I had a choice, so I'm glad that there weren't others available.

I don't know what she means by "All formatting integrity was maintained."

It seems clear that TEX made the project both faster and cheaper for Wm. C. Brown. The final sentence is puzzling, though. I don't know whether she is refering to the first experience or the second, and I don't know of any hidden traps. The process of sending the disks to ArborText and having them send high-resolution hard copy to Wm. C. Brown seemed to go very smoothly.

I am concerned about the "drawback" regarding quality. Here I must take the blame. I'm convinced that TEX is capable of producing textbooks of the highest quality. If there was a limitation perceived by Wm. C. Brown, the limitation was in my ability, not in TEX's. I will have to try to make that clear to them.

Conclusion

The experience of writing a textbook was quite a challenge. It was the first major book either of us had written, and we didn't know much about the process. By doing it in TEX, I suspect there are many things we never had to learn.

I have a friend who wrote another textbook at the same time we were writing ours. His book was published by a different publisher, and his publisher re-typed the entire book — even though he had sent them disks. He complains to this day about typographical errors that he finds in the book that were not there in the version he sent them. He had a chance to proofread his book, but he was not able to find them all. Our experience was quite different from his; once we finished a page, we knew what the final version would look like, and typographical errors could not creep in through the typesetting process. I wouldn't think of trading places with him.

A good part of the fun was in learning TEX; another was in reading *The TEXbook*. The interchange between us and the various editors at Wm. C. Brown was a learning experience that we won't soon forget.

After all is said and done, and even though there were times that we got frustrated, it's obvious to us that TEX is the way to go. Had we not had TEX to use to prepare the prospectus and the preliminary versions that our students used, I doubt that it would ever have gotten finished. Indeed, without the beautiful prospectus, it might never have been accepted for publication.

I'm convinced that any author can learn enough TEX and/or LATEX to write their articles, books and papers; it's not necessary to become a TEXpert to develop beautiful results. A fast computer — I'm now using a Sun workstation — and a previewer make life easier. I cannot imagine writing without TEX.

Bibliography

Knuth, Donald E. *The TEXbook*. Reading, Mass.: Addison-Wesley, 1986

Knuth, Donald E. *Computer Modern Typefaces*. Reading, Mass.: Addison-Wesley, 1986.

Rhoads, Samuel E., and Michael V. Gearen. *Disciplined Programming Using Pascal*. Dubuque, Iowa: William C. BrownPublishers, 1990.

Appendix

This is how we typed the algorithm for GCD:

```
\balg
START OF GCD$($First, Second$)$\\
Precondition: First and Second are positive integers\\
\mbox{\ }\\
store the remainder of First $\div$ Second in Remainder\\
loop while Remainder $>$ 0\+\\
  store Second in First\\
  store Remainder in Second\\
  store the remainder of First $\div$ Second in Remainder\-\\
return the value stored in Second\\
\mbox{\ }\\
END OF GCD
\ealg
```

And this is what the algorithm looked like in the textbook:

START OF GCD(First, Second)
Precondition: First and Second are positive integers

store the remainder of First ÷ Second in Remainder
loop while Remainder > 0
    store Second in First
    store Remainder in Second
    store the remainder of First ÷ Second in Remainder
return the value stored in Second

END OF GCD

The following is an example of a simple figure/picture done in LaTeX. It draws a syntax diagram of the Pascal `while statement`. This example is included just to show, in general, how we constructed such diagrams. Most of the figure/pictures were much more complicated.

```
% whstate.tex   the while statement

\begin{figure}[htb]
  \begin{picture}(352,64)
  \put(  0, 48){{\it while statement:}}
  \put(  8, 24){\vector(1,0){24}}
  \put( 56, 24){\oval(48,16)}
  \put( 56, 24){\makebox(0,0){{\tt while}}}
  \put( 80, 24){\vector(1,0){24}}
  \put(104, 16){\framebox(88,16){{\it Boolean expression}}}
  \put(192, 24){\vector(1,0){24}}
  \put(232, 24){\oval(32,16)}
  \put(232, 24){\makebox(0,0){{\tt do}}}
  \put(248, 24){\vector(1,0){24}}
  \put(272, 16){\framebox(48,16){{\it statement}}}
  \put(320, 24){\vector(1,0){32}}
  \end{picture}
\caption{\label{whstate}}
\end{figure}
```