LaTeX source from word processors

Bart Childs

Abstract

Hennings' CTAN survey is a good starting point when considering projects implied by the title of this article. I found it a fair view of most related packages. He suggests having one of two goals: converting the *document structure* or converting the *appearance*. My goal is neither of these. I want to produce LaTeX source that is accurate in content, clean, and therefore maintainable.

This is in keeping with Knuth's original goals in producing TeX: graphic excellence and a document convenient for archiving. Structure and appearance are important. I believe clean LaTeX is more likely to have this intrinsic result (not use of word processing systems). My current conversion system is a hybrid based on the use of the Open Office `Writer` package, its `Writer2LaTeX` application, and macros for the Emacs editor written in Elisp. The test cases for this system are books: 1) on rotordynamics, 2) a C++ programming text, 3) a memoir on a friend's life including significant text fragments in the Czech language, and 4) a novel that includes three love triangles. Even the worst case with significant mathematics formatting done in WordPerfect is tractable; *I did not say easy*. The lack of (intelligent?) use of word processors causes many of the problems. I estimate that a 300-page novel written in a reasonable dialect of Word, WordPerfect, or `Writer` could be converted to LaTeX in an hour or two.

1 Genesis

My primary formatting system has been TeX-based for more than thirty years. Throughout this time I have had occasional need to import small parts of documents done in word processors into my TeX-based documents. I have accomplished that in a number of ways: from keyboarding small projects to somewhat automatic conversion depending upon what was available. I used some of the earlier systems discussed by Hennings [2].

Several years ago, two colleagues were writing a text on "programming" and became aware that they would have significant advantages if they could convert the half of the book that was completed to LaTeX and take some instruction on how to complete the rest in LaTeX.

I sketched the process and created a small set of Emacs Elisp macros to do that conversion. We agreed to the generalities with plans to make a formal agreement upon the return of the senior author from a summer-long trip. Much of the LaTeX work was to be done by the junior author, naturally. The health of the junior author suddenly deteriorated and my conversion project was cancelled.

I continued to be intrigued by the concept. I learned more Elisp, added macros, and a number of *open* packages that seemed to offer promise as a means of getting much of the conversion done in an automatic manner. I never felt that a mostly automatic conversion was realistic for projects involving significant mathematics content. I expected to pursue a "PhD with a screwdriver" approach. I was willing to do this based on working from the Word `.rtf` (Rich Text File format), total extraction of text without formatting, *and/or* a mostly automatic conversion that needed tweaking — my pipe dream.

A few years after retirement, a friend and colleague in the college of engineering asked me for help finding someone to keyboard a new text he was writing based on a few dozen of his research papers — and related studies. The topic of the text is rotordynamics — from small pumps and turbines to large ones as in the main engine of the space shuttle. I resurrected my plan and we agreed on the plan of work.

The draft source of this rotordynamics text is being done in WordPerfect, the formatter the author has used for many years. Most of the text is being adapted from the author's contributions in the subject. The current version is approximately 400 pages in length with another 25% to be added. The lists of contents, figures and tables will likely occupy 18 pages. There are hundreds of equations with one of them being a full page.

2 The process evolves

I started this conversion using the process I had prototyped for the programming text. The rotordynamics text was quite a different document because of the large fraction of displayed equations. The displayed equations and figures in the rotordynamics text require approximately the same fraction of space required by figures, programs, and code fragments in the programming text. Most (maybe all) of the code fragments, programs, and figures in the programming text were restricted from floating. There had to be some "manual floats".

I did some small portions of the rotordynamics book as manual conversions for test cases. Some of the equations were manually entered because conversion of mathematics among word processing systems was generally accepted to be non-existent (I think that is improving). The manual process was based on: a) having a `.pdf` of the document, b) editing the `.rtf` file, c) editing a text file exported from a word processor (with some encoding), and/or d) a form of LaTeX exported from one of several systems. I was delivering LaTeX source faster than I could have keyboarded it from good copy. Still, it was unsatisfactory because it was mostly a manual process.

The source documents were done in WordPerfect on a PC and I was doing LaTeX on a Mac. There are good TeX and Emacs systems for the Mac using Mac OS X. Some Emacs systems were not acceptable to me because my system uses function keys.

I continued to strive for big improvements because keyboarding mathematics would be slow. A significant improvement came by changing the format in which sources were delivered to me. The source was 1) edited to remove the graphics from the WordPerfect source, and 2) exported in `.rtf` form, with 3) the graphics elements put in a `.zip` file. The version of WordPerfect being used would create `.rtf` files hundreds of times bigger than needed if the graphics was included in the export to the `.rtf`. Removing the graphics was no loss because it — like the mathematics — was not being exported.

I would take the `.rtf` from WordPerfect, import it to OOo `Writer`, and save it! This apparently lost nothing but gave a smaller file and therefore my system was faster in using it. I also noticed that `Writer`'s export of *text with encoding* was different from the other systems I had used. Further, the export could be done in Unicode which was compatible with Emacs.

Apparently there was significant appreciation of Unicode in the WordPerfect export process. The export of the mathematics from WordPerfect was not converted but many symbols, Greek letters, etc. were now viewable on the screen. Most (LA)TeX users should be able to glean the proper content from a printed `.pdf` of the WordPerfect. Now, the Emacs macros could do much more. At this time, my benefactor had other obligations and so I had time to work on the macros and test the system using the modified process.

I continued to learn more Elisp.

3 Keeping the mind busy

My benefactor's diversions lasted a bit longer than planned. I read more about Unicode and realized how provincial some of us are here in the *English-only USA*.

A college buddy of mine is a Czech immigrant and was corresponding with a publisher in the Czech Republic about his memoir. When he wrote to the publishers and sent it by email, the formatting was lost. I suggested learning a bit of LaTeX, converting it to `.pdf`, and emailing that. He had sent me a draft of the book so I could create some examples. The published version [1] was done while I was creating this system. Of course I was naïve and would still have been so had I not read Horak's note [3].

But while waiting, I thought I could polish my Emacs macros to handle his Czech problems. It was fairly easy and with the improvements in the `Writer` export process, it was really easy. I mention this project because it shows evidence of *real problems* with similar projects. That will be discussed later.

In the abstract I mentioned a novel about three love triangles. That project was technically trivial but also contains the same *real problems* with conversion of word processor sources.

4 Real problems

There are several sources of problems that impeded progress in these projects. Some of these sources could be avoided by "user learning" while others resulted from differences in the design and implementation of the systems they used. The authors had several kinds of problems that automatic conversion did not handle:

1. Inconsistent use of functionality.
2. Wrong use of functionality.
3. Not using available functionality.
4. Oops. Operator, operand placement. Misunderstandings. Mysticism about style files.

This quote is in section 1.2 of the ***`Writer2LaTeX`*** *User's Manual* [4]:

> You can use LaTeX as a typesetting engine for your OOo documents: `Writer2LaTeX` can be configured to create a LaTeX document with as much formatting as possible preserved. Note that the resulting LaTeX source will be readable, but not very clean. ... You will find that `Writer2LaTeX` uses the principle *garbage in — garbage out*!

Each of the above examples of *garbage in — garbage out* was present in at least two of the test cases cited. *Garbage in — garbage out* may be a bit strong of a description for these but the message is clear. For example, in the Czech memoir it was certainly appropriate to attempt to show correct accents — Horak [3] would be proud. It overwhelmed the author's limits of skill with the systems he was using.

Each of the authors has a doctorate and has taught at major universities. They are consistent users of computers but obviously are not the most persistent readers of the formatter manuals. Maybe the manuals are poor, non-existent, or not convenient? Maybe the easy-to-use graphics interfaces overwhelmed the authors? Maybe these interfaces do not encourage users like these to seek the information they need? Maybe they just do not care?

4.1 Inconsistent use of functionality

The author of the memoir that used many Czech words, phrases, and sentences is to be saluted for attempting to make that text look proper to a Czech reader. There are five special items in this sentence

> On my next visit to Prague, he joined Vláďa and me, along with our wives, for lunch at a French restaurant in *Obecní dům* (Municipal House).

The nickname Vláďa has an accent over the letter "a" and an accent often called a *caron* modifying the letter "d". The accented "i" in the first italicized word is a dotless "i". Finally, the second italicized word has an accent that almost appears to be the degrees (as in temperature) symbol. Although it was not the author's intention, the distances these accents were raised or kerned differed in most cases. (I do not claim my caron here is perfect.)

4.2 Misuse of functionality

In the rotordynamics book there were many instances of using different Greek characters as the same: the phi and varphi, ϕ and φ, as well as others. Since this document was constructed using papers written years ago, this is easily understood.

The author of the novel containing three love triangles suffered a similar problem. The author did not like the double prime (") for the opening and closing quotes. When he wrote the first part he selected special graphics characters for the quotes. When he wrote the other two parts, the smart quotes were automatic for him. He did not recall why; it may have been a new revision of his formatter.

4.3 Not using available functionality

In two of the test cases the authors used itemized lists. The exported form yielded consecutive lists of one item. This did not bother the bulleted lists but would have been an error with enumerated and description lists.

In many cases the authors did not use styles, so chapter and section beginnings show the formatting but no LaTeX commands. This is not a total loss, because I convert the section numbers into labels that would aid if we were trying to resolve differences in my output with the older version.

4.4 Oops?

These examples can be difficult. A glaring example is that WordPerfect's mathematics operators may follow the operand in some cases. In LaTeX the operator is always first! I did not find a general rule as to when to expect this. My Emacs macros for adjusting this are interactive to enable the user (me) to minimize such problems.

A really big *oops* worth repeating is the lack of using styles, which caused inconsistencies. I had to handle some of these manually.

5 Typical Emacs macros

The first versions of these macros were developed when I was using an export that was usually designated *text with encoding.* This export would discard all (or nearly all) formatting, such as emphases. The improvements in `Writer2LaTeX` have led to a reduced need of this kind of detailed editing. Still, the concepts in the design of these macros are applicable in the current system of conversion as well as keyboarding original documents.

This list contains three cases where it is more efficient to use *text with encoding* exports than the converted exports, assuming the goal of clean LaTeX. These came from the rotordynamics text, the programming text, and the User's Manual. These are:

Tables Tables are exported with all formatting on every cell. The usual (LaTeX) procedure is to give default formatting in a template and exceptional formatting when needed in a cell.

Mathematics Text is often used for explanatory purposes in equations.

Programs as well as verbatim text need special handling.

Portions of some documents are easier to convert by exporting as *text with encoding* and then inserting the formatting by editing. Two examples are mathematics that does not convert and formatted code fragments in a processor where font changes are done manually rather than using a package like `listings`.

The macros were implemented using the mouse (or similarly functioning device) to point or highlight in conjunction with function keys. In Emacs one can also highlight a region of text by setting the mark and moving the point. The function keys can also be modified by use of `shift`, `control`, and `alt`.

5.1 Applying fonts to text

In this paragraph there are single words and a three-word sequence that are emphasized by changing fonts. The default font is changed to *italics* or `typewriter`. Source exported as *text with encoding* will have formatting removed. A similar situation occurs when text is inserted into mathematics code.

The user can highlight a phrase or click within the single word. Then the user presses the appropriate function key for the formatting command to be inserted with grouping of the appropriate text. If the user has clicked within a word, then the extent of the word is determined by whitespace delimiters. Clicking on whitespace is a special form of this — the commands are inserted and the cursor placed on the right brace for user input.

Instead of highlighting a region, the user can use the Emacs form of setting the mark and moving the cursor to the other end of the region. I implemented these functions for **bold**, *italic*, sans serif, and `typewriter` fonts. I did not insert the italic correction but easily could have paying attention to the following character. I did not because in many cases it is just not needed, and besides, the user should have some responsibilities. The same functions are reused for simple grouping and the `\text{}` commands which were used mostly in math modes.

5.2 Inline mathematics

Inline mathematics is common in the rotordynamics text. Most of the resulting mathematics is usually a fraction of a line in length.

The implementation is like the font changes in the previous subsection. A significant difference is that the export processes handling WordPerfect mathematics yields significant artifacts of excessive white space and formatting trash. This almost always includes many of the grave characters — this must be an *escape character* for the internal form of WordPerfect mathematics.

I have not had a reasonable test case with Word mathematics, yet. There are small examples of mathematics in the programming text.

5.3 Display mathematics

The concepts in the previous subsection are applicable. However, there are several forms of display mathematics. These forms were used in the rotordynamics text:

1. `\[...\]`, the standard for display equations without numbers.
2. `\begin{equation*}...\end{equation*}`, just an alias for the former, or *vice versa*.
3. `\begin{equation}...\end{equation}`, which numbers the equations and should have an accompanying `\label`.
4. `\begin{equation}\begin{split}...` `\end{split}\end{equation}`, which numbers a collection of equations and should have an accompanying `\label`.

Chapter 8 of Frank Mittelbach et al.'s *LaTeX Companion* has some seventy pages of excellent details of advanced mathematics formatting.

I implemented these four display math choices using one function key and prompting the user for which of the above forms was desired. I developed similar choice macros for presenting fractions and matrices which made conversions faster and most importantly more consistent. The most important facet of this conversion is that with a little care the totality of the mathematics was converted correctly and hours of detailed, laborious proofreading was avoided.

5.4 Programs, code fragments, verbatim text

Programs should be formatted by language sensitive packages like `listings`. The package `fancyvrb` requires some study but gives great results. Both packages come with inline commands whose use is aided by adaptation of the above font changing and inline mathematics concepts.

5.5 Other macros — fix-up

There were several other macros that aided the conversion. I consider these to be "fix-up" in nature. These include:

- `\caption`s in the rotordynamics text often contain inline mathematics. The use of the LaTeX delimiters (`\( \)`) is not allowed; they must be converted to the TeX toggle (`$`).
- Interactive aid to standardizing presentation of fixed-point and floating-point numbers.
- Locating multicharacter super/subscripts that were likely exported incorrectly (needing grouping).
- Locating likely problems due to insertion of inadvertent whitespace.
- Locating unescaped TeX control characters.
- Macros to aid the insertion of labels and their references.

6 Current system

The current system has been improved greatly with the release of OOo `Writer2LaTeX` version beta 1.2. Despite its being labeled a beta release, I have not found any problems to date.

I find these observations about this new release interesting: 1) the user's guide is 10% shorter and 2) the output files are 3–5% shorter than with version 1.0. The LaTeX output is cleaner, as most of the reduction in the size is the elimination of needless formatting like: 1) most paragraphs were inside grouping braces and a declaration that I used English and 2) `{\textquotedblright}` for a simple (”). A cursory look at the user's guide indicates some removal of redundancy. There is a lack of the completeness that is characteristic of the documentation of releases from the TeX communities.

I plan to work with OOo and continue to make this product better. I believe it to be the best hope I know of, especially in the *open* domain.

The following quote is from one of the pages on its web site (`http://writer2latex.sourceforge.net/index3.html`):

> You will never get a result that looks *identical* to the original, in fact that's the whole point: LaTeX is in general a superior typesetting engine compared to Writer. For example LaTeX produces much better results for formulas, it has an excellent paragraph and page breaking mechanism, it uses ligatures etc. On the other hand Writer has a few features that LaTeX does not support well. If the layout of your document depends on text flowing around pictures or linked text boxes, you will never get good results with Writer2LaTeX.
>
> According to TeX's author Donald E. Knuth, TeX is a *typesetting system intended for the creation of beautiful books - and especially for books that contain a lot of mathematics* (quoted from "The TeX book"). **Writer2LaTeX** will aim to produce excellent result for this kind of documents; including of course shorter texts with a book-like layout.

This quotation is fair but I think it makes my point “go ahead and inhale”. Show the logos (TeX and LaTeX) correctly, use the correct dashes and spacing, use the proper quotes, ...

6.1 Examples of other problems

I present an annotated list of a few other problems I addressed in the macros. These are based on two of the test cases: the rotordynamics text and the programming text. I think it is fair to classify most of these as “not very clean LaTeX”.

Export of spacing. The export of chapter 5 of the rotordynamics text has 47 occurrences of (`}`), a space preceding a right brace. The majority of those are in constructs like `\textit{word }` while most of the rest are weird constructs like `\textit{ }` and `\textbf{\textit{\ \ }}`.

The first may be sloppy keyboarding by the author. The second seems to be intentional spacing, why not (`\ `)? The last is likely a hacked indentation kludge?

Inline mathematics. Some inline mathematics is converted to italics. That is troublesome to me because it should really remain as unconverted mathematics. Then too, that may be the fault of the author.

Export of structure. The structure of the chapter and lists range from inconsistent to missing. This is likely the authors' fault as the use of styles seems to be the cause.

7 `Writer` and friends

In spite of these remarks I salute OOo. I believe that the `Writer` package and `Writer2LaTeX` application have made a great contribution to the goal of converting many documents into a form for better presentation and archival, namely (LA)TeX. That may not have been the intent. The intent may have been to enable a good `Writer` user to simply use LaTeX as an output device?

The LaTeX code output in version beta 1.2 is improved, but not clean. The *`Writer2LaTeX` User's Manual* is 45 pages in length. The exported LaTeX (with the `clean` option) source averages about fourteen occurrences of `\mdseries` and twelve occurrences of `\textstyleSourceText` per page. Each paragraph is grouped with `\mdseries` as the start. The latter is effectively an alias for `\texttt` and used in tables.

8 Conclusions

Reasonable document interchange and archival quality is now possible for a wide range of systems. I believe that (LA)TeX is the most reasonable basis for many archival systems.

The advances by OOo and its `Writer` system are impressive and appreciated. I hope that its *open* status and development will continue. Note: I have addressed only a small part of a large project — OOo.

A point made in a number of venues is the problem of TeX systems not having a native graphical input process. Lyx and OOo are touted as solutions — along with several others. The authors of the three test cases I have used show that the graphical interfaces are not a solution to the problems — in my humble opinion. All the authors are highly educated and familiar with the problems of getting people to learn at the college level. Still, each has shown the results from casual learning about their systems. The effective use of styles, consistent use of symbols and special functions, document structure, etc., were lacking in each of their documents.

The first line of a LaTeX document requires a statement of the class of the document. There is a finite number of them. It does not seem to enter the stream of consciousness for many that if they learned how to type “Mary had a little lamb” on a machine that there should be at least a small change in the start of a letter to a sweetheart, a grocery list, or any other class of documents.

In a moment of frustration I lamented “Users avoid using LaTeX because you have to learn how to do some things while users of Word believe if it takes any non-obvious effort to do something, *it should not be done!*”

I raised the question earlier about why educated users of computers seem to get so little from user’s guides and manuals. Maybe the manuals are poor, non-existent, or not convenient? Maybe the easy-to-use graphics interfaces overwhelmed the authors? Maybe these interfaces do not encourage users like these to seek the information they need? Maybe they just do not care?

Was the intent in creating `Writer2LaTeX` to give the user “LaTeX as an improved output device”? I think that poses a bigger challenge, “How do you teach a `Writer` user to write for LaTeX?”

9 Questions

I did not intend this as a FAQ but thought it might be a good way to end the present paper.

LL LaTeX Do any of the test cases use LaTeX beyond Leslie Lamport’s book?

Answer *No* for memoir and book on the three love triangles. *Yes* for the science and engineering texts. Packages used: `float`, `lscape`, `makeidx`, `fancyvrb`, `graphicx`, `array`, `amsmath`, `amssymb`, `sidecap`, `wrapfig`, and `caption`. These were probably not all necessary, but useful.

Word test case? What do you want for a Word test?

Answer A one-pager, like Norman Naugle’s *An Elementary Sum*. Then, many others would help. I hope it would also convert to `Writer` and back too.

How long? How long did it take you to type Norman’s note?

Answer An hour or so. The next question might be, why didn’t you just do it in Word? Well, probably that would have taken seven or eight hours — and fortunately I do not have Word in my house.

References

[1] Charles Ota Heller. *Prague, My Long Journey Home*. Abbott Press, December 2011.

[2] Wilfried Hennings. Converters from PC Textprocessors to LaTeX – Overview, June 2012. `http://tug.org/utilities/texconv/pctotex.html`.

[3] Karel Horák. Those obscure accents. . . . *TUGboat*, 29(1):42–44, 2007.

[4] Henrik Just. User’s manual for `Writer2LaTeX`, March 2012. `http://sourceforge.net/projects/writer2latex`.

◇ Bart Childs
Texas A&M University
College Station, TeXas 77843-3112
USA
`bart (at) tamu dot edu`
`http://faculty.cse.tamu.edu/bart/`