Word Processors: Stupid and Inefficient

Allin Cottrell

April 2001: German translation available.
June 2003: Spanish translation available.
April 2004: French translation available.

Contents

1 The claim

The word processor is a stupid and grossly inefficient tool for preparing text for communication with others. That is the claim I shall defend below. It will probably strike you as bizarre at first sight. If I am against word processors, what do I propose: that we write in longhand, or use a mechanical typewriter? No. While there are things to be said in favor of these modes of text preparation I take it for granted that most readers of this essay will do most of their writing using a computer, as I do. My claim is that there are much better ways of preparing text, using a computer, than the word processor.

The wording of my claim is intended to be provocative, but let me be clear: when I say *word processors* are stupid I am not saying that *you*, if you are a user of a word processor, are stupid. I am castigating a technology, but one that is assiduously promoted by the major software vendors, and that has become a *de facto* standard of sorts. Unless you happen to have been in the right place at the right time, you are likely unaware of the existence of alternatives. The alternatives are not promoted by the major vendors, for good reason: as we shall see, they are available for free.

Let's begin by working back from the end product. Text that is designed to communicate ideas and information to others is disseminated in two main ways:

1. As ``hard copy'', that is, in the form of traditional printed documents.
2. By digital means: electronic mail, web pages, documents designed to be viewable on screen.

There is some overlap here. For instance, a document that is intended for printing may be distributed in digital form, in the hope that the recipient has the means to print the file in question. But let us consider these two modes of dissemination in turn.

2 Printed documents

You want to type a document at your computer keyboard, and have it appear in nicely printed form at your computer's printer. Naturally you don't want this to happen in real time (the material appearing at the printer as you type). You want to type the document first and ``save'' it in digital form on some storage medium. You want to be able to retrieve the document and edit it at will, and to send it to the printer when the time is right. Surely a word processor-such as the market leader, Microsoft Word-is the ``natural'' way to do all this? Well, it's one way, but not the best. Why not?

2.1 Composition versus typesetting

Preparing printable text using a word processor effectively forces you to conflate two tasks that are *conceptually* distinct and that, to ensure that people's time is used most effectively and that the final communication is most effective, ought also to be kept *practically* distinct. The two tasks are

1. The composition of the text itself. By this I mean the actual choice of words to express one's ideas, and the logical structuring of the text. The latter may take various forms depending on the nature of the document. It includes matters such as the division of the text into paragraphs, sections or chapters, the choice of whether certain material will appear as footnotes or in the main text, the adding of special emphasis to certain portions of the text, the representation of some pieces of text as block quotations rather than as the author's own words, and so on.

2. The typesetting of the document. This refers to matters such as the choice of the font family in which the text is to be printed, and the way in which structural elements will be visually represented. Should section headings be in bold face or small capitals? Should they be flush left or centered? Should the text be justified or not? Should the notes appear at the foot of the page or at the end? Should the text be set in one column or two? And so on.

The author of a text should, at least in the first instance, concentrate entirely on the first of these sets of tasks. That is the author's business. Adam Smith famously pointed out the great benefits that flow from the division of labor. Composition and logical structuring of text is the author's specific contribution to the production of a printed text. Typesetting is the typesetter's business. This division of labour was of course fulfilled in the traditional production of books and articles in the pre-computer age. The author wrote, and indicated to the publisher the logical structure of the text by means of various annotations. The typesetter translated the author's text into a printed document, implementing the author's logical design in a concrete typographical design. One only has to imagine, say, Jane Austen wondering in what font to put the chapter headings of *Pride and Prejudice* to see how ridiculous the notion is. Jane Austen was a great writer; she was not a typesetter.

You may be thinking this is beside the point. Jane Austen's writing was publishable; professional typesetters were interested in laying it out and printing it. You and I are not so lucky; if we want a printed article we will have to do it ourselves (and besides, we want it done much faster than via traditional typesetting). Well, yes and no. We will in a sense have to do it ourselves (on our own computers), but we have a lot of help at our disposal. In particular we have a professional-quality typesetting program available. *This program (or set of programs) will in effect do for us, for free and in a few seconds or fractions of a second, the job that traditional typesetters did for Shakespeare, Jane Austen, Sir Walter Scott and all the rest*. We just have to supply the program with a suitably marked-up text, as the traditional author did.

I am suggesting, therefore, that should be two distinct ``moments'' in the production of a printed text using a computer. First one types one's text and gets its logical structure right, indicating this structure in the text via simple annotations. This is accomplished using a *text editor*, a piece of software not to be confused with a word processor. (I will explain this distinction more fully below.) Then one ``hands over'' one's text to a typesetting program, which in a very short time returns beautifully typeset copy.

2.2 The evils of WYSIWYG

These two jobs are rolled into one with the modern WYSIWYG (``What You See Is What You Get'') word processor. You type your text, and *as you go* the text is given, on the computer screen, a concrete typographical representation which supposedly corresponds closely to what you will see when you send the document to the printer (although for various reasons it does not always do so). In effect, the text is continuously typeset as you key it in. At first sight this may seem to be a great convenience; on closer inspection it is a curse. There are three aspects to this.

1. The author is distracted from the proper business of composing text, in favor of making typographical choices in relation to which she may have no expertise (``fiddling with fonts and margins'' when she should be concentrating on content).

2. The typesetting algorithm employed by WYSIWYG word processor sacrifices quality to the speed required for the setting and resetting of the user's input in real time. The final product is greatly inferior to that of a real typesetting program.

3. The user of a word processor is under a strong temptation to lose sight of the logical structure of the text and to conflate this with superficial typographical elements.

The first two points above should be self-explanatory. Let me expand on the third. (Its importance depends on the sort of document under consideration.)

2.3 Document structure

Take for instance a section heading. So far as the logical structure of a document is concerned, all that matters is that a particular piece of text should be ``marked'' somehow as a section heading. One might for instance type `\section{Text of heading}`. How section headings will be implemented typographically in the printed version is a separate question. When you're using a word processor, though, what you see is (all!) you get. You are forced to decide on a specific typographical appearance for your heading as you create it.

Suppose you decide you'd like your headings in boldface, and slightly larger than the rest of the text. How are you going to achieve this appearance? There's more than one way to do it, but for most people the most obvious and intuitive way (given the whole WYSIWYG context) is to type the text of the heading, highlight it, click the boldface icon, pull down the little box of point sizes for the type, and select a larger size. The heading is now bold and large.

Great! But what says it's a heading? There's nothing in your document that logically identifies this little bit of text as a section heading. Suppose at some later date you decide that you'd actually prefer to have the headings in small caps, or numbered with roman numerals, or centered, or whatever. What you'd like to say is ``Please make such-and-such a change in the setting of all section headings.'' But if you've applied formatting as described above, you'll have to go through your entire document and alter each heading manually.

Now there *is* a way of specifying the structural status of bits of text in (for instance) Microsoft Word. You *can*, if you are careful, achieve effects such as changing the appearance of all section headings with one command. But few users of Word exploit this consistently, and that is not surprising: the WYSIWYG approach does not encourage concern with structure. You can easily-all too easily-``fake'' structure with low-level formatting commands. When typing one's text using a text editor, on the other hand, the need to indicate structure is immediately apparent.

2.4 Text editors

OK, it's probably time to explain what a text editor is, and how it differs from a word processor. A modern text editor looks a bit like a word processor. It has the usual apparatus of pull-down menus and/or clickable icons for functions like opening and saving files, searching and replacing, checking

spelling, and so on. But it has no typesetting functionality. The text you type appears on screen in a clear visual representation, but with no pretense at representing the final printed appearance of the document.

When you save your document, it is saved in the form of *plain text*, which in the US context usually means in ``ASCII'' (the American Standard Code for Information Interchange). ASCII is composed of 128 characters (this is sometimes referred to as a ``7-bit'' character set, since it requires 7 binary digits for its encoding: 2 to the seventh power = 128). It includes the numerals 0 through 9, the roman alphabet in both upper and lower case, the standard punctuation marks, and a number of special characters. ASCII is the lowest common denominator of textual communication in digital form. An ASCII message will be ``understandable'' by any computer in the world. If you send such a message, you can be sure that the recipient will see precisely what you typed.

By contrast, when you save a file from a word processor, the file contains various ``control'' characters, outside of the ASCII range. These characters represent the formatting that you applied (e.g. boldface or italics) plus various sorts of internal ``business'' relating to the mechanics of the word processor. They are not universally ``understandable''. To make sense of them, you need a copy of the word processor with which the document was created (or some suitable conversion filter). If you open a word processor file in a text editor, you will see (besides the text, or bits of it) a lot of ``funny looking stuff'': this is the binary formatting code.

Since a text editor does not insert any binary formatting codes, if you want to represent features such as italics you have to do this via *mark-up*. That is, you type in an annotation (using nothing but ASCII), which will tell the *typesetter* to put the specified text into italics. For example, for the LaTeX typesetter (more on this below) you would type `\textit{stuff you want in italics}`. Actually, if you are using a text editor which is designed to cooperate with LaTeX you would not have to type this yourself. You'd type some kind of shortcut sequence, select from a menu, or click an icon, and the appropriate annotation would be inserted for you; the mechanics of typing an ASCII document suitable for feeding to LaTeX are not much different from typing in a modern word processor.

2.5 The virtues of ASCII

The approach of composing your text in plain ASCII using a text editor, then typesetting it with a separate program, has several ``incidental'' virtues.

1. Portability: as mentioned above, *anybody*, using any computer platform, will be able to read your marked-up text, even if they don't have the means to view or print the typeset version. By contrast your Snazz 9.0 word processor file can be completely incomprehensible to a recipient who doesn't have the same brand and version of word processor as you-unless he or she is quite knowledgeable about computers and is able to extract the actual text from the binary ``garbage''. And this applies to *you* over time, as much as to you and a correspondent at one time. You may well have difficulty reading Snazz 8.0 files using Snazz 9.0, or vice versa, but you'll never have any trouble reading old ASCII files.
2. Compactness: an ASCII file represents your *ideas*, and not a lot of word processor ``business''. For small documents in particular, word processor files can be as much as 10 times as large as a corresponding ASCII file containing the same relevant information.
3. Security: the ``text editor to typesetter'' approach virtually guarantees that you will never have any problem of corruption of your documents (unless you suffer a hard disk crash or some comparable calamity). The source text will always be there, even if the typesetter fails for some reason. If you regularly use a word processor and have *not* had a problem with file corruption then you're very lucky!

(Further reading: Sam Steingold's page "No Proprietary Binary Data Formats".)

2.6 The typesetter

By this time I owe you a bit more detail on the typesetter part of the strategy I'm advocating. I won't go into technical details here, but will try to say enough to give you some idea of what I'm talking about.

The basic typesetting program that I have in mind is called TeX, and was written by Donald Knuth of Stanford University. TeX is available for free (via downloading from many Internet sites), in formats suitable for just about every computer platform. (You can if you wish purchase a CDROM with a complete set of TeX files for a very modest price.) Knuth started work on TeX in 1977; in 1990 he announced that he no longer intended to develop the program-not because of lack of interest, but rather because by this time the program was essentially perfected. It is as bug-free as any computer program can be, and it does a superb job of typesetting just about any material, from simple text to the higher mathematics.

I referred above to LaTeX. If TeX is the basic typesetting engine, LaTeX is a large set of macros, initially developed by Leslie Lamport in the 1980s and now maintained by an international group of experts. These macros make life a lot easier for the average user of the system. LaTeX is still under active development, as new capabilities and packages are built on top of the underlying typesetter. Various ``add-ons'' for TeX are also under development, such as a system which allows you to make PDF (Adobe's ``Portable Document Format'') files directly from your ASCII source files. (I say ``under development'' but by this I just mean that they are continuously being improved. The programs are already very stable and full-featured.)

As mentioned above, you indicate the desired structure and formatting of your document to LaTeX in the form of a set of annotations. There are many books (and web-based guides) that give the details of these annotations, and I will not go into them here. The common annotations are simple and easily remembered, besides which LaTeX-friendly text editors (of which there are many) offer you a helping hand.

One very attractive feature of LaTeX is the ability to change the typeset appearance of your text drastically and *consistently* with just a few commands. The overall appearance is controlled by

1. The ``document class'' that you choose (e.g. report, letter, article, book).
2. The ``packages'' or style files that you decide to load.

You can, for instance, completely change the font family (consistently across text, section headings, footnotes and all) and/or the sizes of the fonts used, by altering just one or two parameters in the ``preamble'' of your ASCII source file. Similarly, you can put everything into two-column format, or rotate it from portrait to landscape. It may be possible to accomplish something similar using a word processor, but generally it's much less convenient and you are far more likely to mess up and introduce unintended inconsistencies of formatting.

You can get as complex as you care to, typesetting with LaTeX. You can choose a ``hands off'' approach: just specify a document class and leave the rest up to the default macros. Generally this produces good results, the typesetting being of much higher quality than any word processor. (Naturally, things like numbering of chapters, sections and footnotes, cross-references and so on, are all taken care of automatically.) Or you can take a more ``interventionist'' approach, loading various packages (or even writing your own) to control various aspects of the typography. If this is your inclination, you can produce truly beautiful and individual output.

2.7 Putting it together

Let me give you just a brief idea of how this all works. If you have a good TeX setup it's like this: You type your text into a TeX-aware editor. You can type the required annotations directly or have

the editor insert them via menus or buttons. When you reach a point where you'd like to take a look at the typeset version you make a menu choice or click a button in the editor to invoke the typesetter. Another menu item or button will open a previewer in which you see the text as it will appear at the printer. And generally this is true ``WYSIWYG''-the previewer will show a highly accurate representation of the printed output. You can zoom in or out, page around, and so on. You send the output to the printer with another menu choice or button, or go back to editing.

At some later point in the process you want to preview the updated file. Click the typesetter button again. This time you don't have to invoke the previewer again: if you've left it running in the background it will now automatically display the updated typeset version. When you're done with an editing session you can delete the typeset version of the file to conserve disk space. You just need to save the ASCII source file; the typeset version can easily be recreated whenever you need it.

3 Digital dissemination

The previous section was mostly angled towards producing good-looking typeset output at the printer. Some other considerations arise when you're preparing a document with digital transmission in mind (email, web pages and so on).

Take email first. Typically if people wish to send a short, ad hoc, message they type that message directly into an email client program, whether it be a ``traditional'' text-based client such as Pine or a GUI (Graphical User Interface) program such as Netscape or Eudora. In that case the message probably goes out in the form of ASCII (or perhaps in HTML, i.e. HyperText Mark-up Language, the language of Web pages, which is itself mostly composed of ASCII). But what if you want to send a longer piece of text that you have already prepared independently of your email client program?

For this purpose it is increasingly common to ``attach'' a document in a word processor format. How does the alternative strategy work in this case?

Well, we have to distinguish between two situations: Is the text in question relatively short and uncomplicated (a memo, a letter, minutes of a meeting, a listing of agenda, a schedule for a visit) or is it more complex (an academic paper-perhaps with a lot of mathematics, a report with illustrations, a book manuscript)? The ``ASCII plus typesetter'' approach leads to different suggestions in these two cases.

3.1 Simple documents

With simple documents, we have to ask: Do we really need the typesetting, the font information and all that? Is it not more efficient, more in the interest of effective and economical communication, just to post plain ASCII text, with the minimal formatting that ASCII allows? This both conserves communications bandwidth (remember that word processor files can be *much* bigger than ASCII files containing the same actual text) and ensures that nobody will be frozen out of the communication effort because they happen not to be running Snazz 9.0. You can attach an ASCII file, created in a text editor, in the same way that you'd attach a word processor file, or you can simply paste it into the body of your email (since it's nothing but plain text). Since TeX source files are nothing but ASCII-and if we're talking about a simple document there won't be too many annotations, and those pretty self-explanatory-they can be treated in the same way.

3.2 Complex documents

Longer and more complicated documents may well be easier to read in typeset form. Math may be hard to convey in ASCII and of course complex diagrams and images are out altogether. So what about TeX, in this context? I have argued that word processor files can be problematic, because your correspondent might not have Snazz 9.0 like you do. But doesn't this cut both ways? Even if you're fired up enough about TeX to give it a try, how many of your correspondents have a TeX

installation? This is a reasonable query, but it is answerable. If you want your correspondent to be able to see a typeset version of your file, and she doesn't have a TeX installation, you have these options:

1. Convert the TeX source file to HTML. There are good conversion programs for this purpose. (HTML and TeX actually have a strong family resemblance, in that they both involve logical mark-up, so inter-conversion can be accomplished with a high degree of fidelity.[1]) Then your correspondent can read your text using a web browser.
2. Does your correspondent have access to a Postscript printer? In an academic or business environment this is quite likely. In that case you could send a fully typeset version of your document in the form of a postscript file, which she can just send to the printer. And/or she can view it on screen if she installs the ``ghostview'' program (free for downloading from the Internet).
3. Does your correspondent have the ``Acroread'' reader for Adobe PDF files installed? (Again, it's a free download.) If so, you can send a PDF version of your typeset document.

In discussing the options for transmitting text via email, we've already hit on the issue of preparing text for web pages. You have the option of writing HTML directly. If you don't want to do that, you can write HTML indirectly using a suitable GUI editor, Netscape Communicator for example. Sure, you can also produce HTML using MS Word (incidentally, horrible HTML, full of extraneous tags that make it awkward to edit using any other application). If you're in TeX mode it's easy to convert your documents into (clean, standard-compliant) HTML.

4 Qualification

I have attempted to make a strong pitch for the ``ascii plus typesetter'' alternative to word processors. I will admit, however, that there are *some* sorts of documents for which a WYSIWYG word processor is indeed the natural tool. I'm thinking of short, ad hoc, documents which have a high ratio of formatting ``business'' to textual content: flyers, posters, party invitations and the like. You could do these in TeX, but it would not be efficient. The standard LaTeX document classes (report, article, etc.) would be of little use to you. And while LaTeX is very smart at handling automatically the range of fonts that you're likely to want in a formal text, it's not geared toward the sort of ``mixing and matching'' of jolly fonts that you might want in a casual production. Logical structure is not really an issue: you're interested in ``raw formatting''. You want to know, for instance, If I put *that* line into a 36-point font, will that push my last line onto the next page, which I don't want? WYSIWYG is your man.

If most of your word-processing work is of this kind, you probably stopped reading a long time ago. If most of your text preparation work involves the production of relatively formal documents, this qualification doesn't affect the essentials of my case.

5 Rant, rant

It may not have escaped your notice that I'm a bit worked up about this theme. Yes, I am. The point is that it's not just a matter of an academic debate between alternative modes of text preparation. It's a set of scales in which the might and wealth of the major software vendors is all on one side. To be blunt, we're looking at a situation in which MS Word is poised to become, for much of the world, *the* standard for the preparation of documents using computers. But Word is a standard that has little to commend it other than the fact that it *is* (or aspires to be) a standard.

It's a bit like QWERTY. Do you know that story? Why the standard arrangement of keys on typewriter keyboards (and by extension computer keyboards) has QWERTYUIOP along the top line? That was not the original arrangement of typewriter keys. It was designed for a purpose, namely to *slow typists down*. The problem was that the expertise of the early typists quickly outran the

capabilities of the early mechanical typewriters: a fast typist could jam the keys, hitting them faster than they could return after striking the ribbon. QWERTY distributed the keys so they couldn't go so fast. This is clearly a crazy arrangement for the keys on an electronic keyboard, but it's too late to change: QWERTY is standard, and all attempts to rationalize the keyboard have failed in the face of that reality.

Similarly, I'm arguing that MS Word has no *right* to be a standard for document preparation, since it's clearly less efficient (for most purposes) than readily available alternatives. I'm hoping that it's not too late in this case, that there's still the opportunity of saying No to Word. Actually, in a sense Word is worse than QWERTY: it's not a real standard, but rather an escalator. The Microsoft ``standard'' for the binary representation of document formatting is something that is variable at the whim of Microsoft Corporation. The MS Word quasi-monopoly piggybacks off the Microsoft Windows quasi-monopoly (an issue which I will not get into here). And so long as they are not hard-pressed by commercial rivals, Microsoft has no particular interest in establishing any sort of long-term standard for the binary representation of formatting. On the contrary, they have a strong interest in forcing you to ``upgrade'' Word at regular intervals. Oh dear, Word N.0 won't read the document your colleague just sent you, prepared using N+1? Well, you'd better update then, hadn't you? Even if there are no features in N+1, that were not present in N, that are of any real value to you.[2]

6 References

If you've come with me this far, you might be interested in more details about good text editors, the TeX typesetting system and so on.

The best place to start for info about TeX and friends is probably the TUG homepage (TUG is the TeX Users Group). This will provide all the links you might need; one of the main ones is to the Comprehensive TeX Archive Network (CTAN) sites, from which you can download complete TeX systems for just about all computer platforms. Such systems include the actual typesetter, a large collection of macros, a previewer, and software for generating printable files.

TeX packages (free ones at any rate) do not generally include the text editor that you'll also need (unless you already have one that you like). There are many choices, but my personal favorite for working with TeX files is Emacs, along with the AUC TeX package. The latter makes Emacs very TeX-friendly: it will highlight TeX syntax so you can see any errors in your mark-up at a glance, and it also offers a wide range of TeX-related commands on convenient menus.

In case you're interested, here's a screen shot of an TeX editing session using Emacs (PNG, 40678 bytes).

Footnotes:

[1] The binary coding used by word processors is a quite different animal, so inter-conversion between TeX and word processor formats is not easy. In addition, since TeX is a superior typesetting engine, it is in principle impossible to convert a TeX document to, say, Word without loss of information.

[2] For what it is worth, in my opinion as somebody who used Word for several years before switching to TeX, and who has a keen interest in typesetting, no worthwhile features have been introduced into MS Word for Windows since version 2.0 of circa 1990.

File translated from TEX by TTH, version 1.93.
On 29 Jun 1999, 14:47.