
FOREWORD

This book explores a great number of concepts, methods, technologies, and tools—in one word resources—which apply to various domains of typesetting. These resources have been developed and are used by the members of a very special community of people, which is also a community of very special people: the T_EX community. To understand the motivation that led these special people to develop and use these resources, I believe it is necessary to make a short flash-back. Since it is true that the past (uniquely?) determines the present and the future, I decided to divide this foreword into three parts: *The Past*, *The Present*, and *The Future*.

At this point, I am asking the readers to excuse my tendency of sometimes becoming autobiographic. This is very hard to avoid when talking about people and events important for one's life. And after all, avoiding it could mean betraying the subject I would like to talk about. . .

The Past

Back in the 1980s, when I started working on my Ph.D. thesis, people in my Department at the time (the Math Department, University of Lille, Northern France) were using a piece of software called "ChiWriter." This DOS program produced a very ugly low-resolution output of text and mathematical formulas. Others preferred to use IBM's Selectric II typewriter machines, spending hours and hours switching balls between Roman, Italic, and Symbol characters. Then came the day when the department finally bought a Macintosh Plus (with 1 MB of RAM and a 20 MB external hard drive!) and we installed *Textures* (a Macintosh implementation of T_EX) on it. That day, my thesis advisor gave me a photocopy of the *T_EXbook*, which I spent the whole night reading.

The last appendix chapter of that book was called "Joining the T_EX community" and talked about TUG (the T_EX Users Group), *TUGboat* (the newsletter of TUG) and so on. But the reader must realize that at that time things were quite different from today: computers were of course unfriendly, expensive, and slow, but the main difference was that there was as yet no Internet. Without the Internet, distances were more real than today, and for people like me who had not yet traveled to the States, places such as

“Stanford” or “Princeton” were infinitely far away and seemed to exist only for the privileged few. This is probably hard to understand today, but at that time, imagining the “ \TeX community” for me was like seeing a Star Trek episode or an old Hollywood movie: it was about people knowing and communicating with each other and acting together, but in a totally different place, time, and context—there could *de facto* be no interaction between them and myself.

That was in 1986, and then came the day when, during a stay at the Freie Universität Berlin, two things happened: I met and became friend with Klaus Thull (one of the European \TeX veterans) and I opened my first *TUGboat*. By a coincidence so strong that one would be tempted to consider it as paranormal, the first *TUGboat* page I read, was exactly page 22 of volume 9 (1), namely the one containing Silvio Levy’s examples of Kazantzaki’s text typeset in Silvio’s Computer Modern Greek. Here is a translation of that text, reminiscent of the storm in Beethoven’s sixth symphony:

“At this moment I understand how heavy the mystery of confession is. Until now no one knows how I spent my two years at Mount Athos. My friends think I went there to see Byzantine icons, or because of a secret longing to live a bygone era. And now, look, I feel embarrassed to speak.

How shall I put it? I remember a late afternoon in the spring, when a storm overtook me as I was coming down Mount Taygetos, near Pentavli. The whirlwind was so fierce I fell flat on the ground so I wouldn’t be blown off the mountain. Lightning encircled me from everywhere and I closed my eyes to keep from being blinded and waited, face down, on the bare earth. The whole towering mountain shook and two fir trees next to me snapped in the middle and crashed to the ground. I felt the thunderbolt’s brimstone in the air, and suddenly the deluge broke, the wind died down, and thick warm drops of rain struck the trees and soil. It pelted the thyme, oregano, and sage, and they shook off their odors and scented the whole earth.”

Goethe (and Beethoven) wanted to communicate “von Herzen zu Herzen”; well this is exactly what happened to me: altogether, the marvelous inebriating contents of this text which I had not read before, its appearance (which at that time I also found marvelous), and its context were quite a shock. That same day, I was able to communicate with Silvio (at that time still at Princeton) through e-mail. A few days later, Klaus and I had written our first joint *TUGboat* paper and submitted it to Barbara Beeton, again through e-mail. Suddenly, there were no frontiers anymore: the \TeX community was quite real and a new world opened in front of me. It is obvious that without traveling to Freie Universität Berlin, without Klaus, without e-mail, without *TUGboat*, none of these would happen.

In the summer of 1990, just a month after I defended my Ph.D. thesis, Tereza (who later became my wife) and I went to the \TeX Users Group meeting in Cork, Ireland, and we had the chance to meet there all those mythical people who made \TeX , the pioneers of the \TeX community—except Donald Knuth himself, whom I met two years later, in Stockholm, in the pure Bergmanian atmosphere of the late Roswitha Graham’s house. The occasion was the ceremony where Donald Knuth was conferred

an honorary doctor's degree at the Kungl Tekniska Högskolan. Roswitha cashed in on that opportunity and organized a small but very interesting Nordic TUG meeting.

In the late 1980s and early 1990s many wonderful things happened (to name only one: the fall of the Berlin wall while Klaus spent the whole night cycling from East to West Berlin and back). At the same time, using communication tools such as mailing lists and ftp, the T_EX community was able to communicate more and more and became wider and more powerful.

But who were these people and where did they come from? The twenty-first century reader should realize that in the 1980s and early 1990s, when Linux was in the mind of its creator and GNU software was not widely known, public domain software did not have the same degree of popularity and reputation as it has today. On the other hand, computers and commercial software were horribly expensive. The psychology of computer users was different as well: there was a tremendous psychological gap between "users" and "programmers"; especially, Macintosh and Windows users would be shocked if they had to type something that even vaguely looked like programming code, and writing T_EX was indeed "programming," even if learning T_EX was far more pleasant than learning, for example, Fortran IV or 8086 Assembler—not to mention the frightening task of implementing T_EX on different platforms, which was, at that time, sometimes still unavoidable for people who simply wanted to use T_EX for their documents. In France, in the early 1980s, there were Ph.D.s written on the process of implementing T_EX on specific platforms.

It is not surprising that most members of the T_EX community were students or scientists from computer science, mathematics, or physics departments. Because they had a reason to use T_EX (writing their reports and publications), and because they had the means to communicate with each other, many of them contributed to T_EX by writing code, and surprisingly enough, the T_EX code that they wrote was very often not connected to the subject of their studies and research. Some projects were linguistic (extending T_EX's capabilities to other languages and scripts), others typographical (facing the challenges of book typesetting), others artistic, ludic, or educational. In fact, what happened was, on a smaller scale, the same phenomenon as with Web pages some years later: students and scientists suddenly had the possibility to include their private life and hobbies in their work context, and to share them with the community. The human dimension of T_EX (and later of the Web) was flexible enough to allow input from various areas of human activities and interests. *TUGboat* was a wonderful mirror of that activity.

There were also the human needs of creativity and commitment: many T_EX users wrote some code for their own needs, realized then that such code could be useful to others, extended it and wrapped it into a package with documentation and examples, and finally committed themselves to supporting it. By doing that, others became interested and communicated with them to express gratitude and suggestions for further development, which in turn resulted in reinforcing that commitment even more, and so on. Years before the widespread use of the Internet, the T_EX community was already

what we now call a *virtual community*, providing a positive and creative identity to people.

That identity was—and still is—one of the most charming aspects of T_EX.

The Present

In the years that followed, the emergence of the Web brought big changes to the T_EX community and to the perception of T_EX by computer users in general. Thanks to HTML, it is quite natural today for everybody to be able to read and write “code.” On the other hand, Adobe’s PDF file format has bridged the gap between T_EX output and electronic documents (and there is indeed a version of T_EX producing PDF output directly). DVI was defined as a “device independent” and “typographically correct” file format: it was abstract enough to be usable on any platform and at the same time precise enough to be able to describe a printed page without loss of information. This was, more or less, also the case for the PDF format, which has the enormous advantage of being self-contained in the sense that it contains all resources (images, fonts, etc.) necessary for displaying and printing the document.

Finally, thanks to Linux and GNU, public domain software is nowadays very well-reputed, and, quite naturally, T_EX is still part of every public domain operating system. That is why it gained popularity among computer gurus who used it to prepare their documents with other tools.

For every new T_EX user, the contact with the T_EX community (which has been such a big deal for me) has become instantaneous, since nowadays almost everybody is connected to the Web. T_EX code can be distributed to the whole community—and this includes people in places unimaginable ten years ago—in a few minutes or hours. Even better, collaborative development tools such as `sourceforge.net` allow people to work simultaneously on an arbitrary number of different versions of the same software, however extensive and complicated this software may be.

The Web was very profitable for T_EX for a number of reasons. Besides providing the T_EX community with the means to be a true virtual community, it also made the principle of the dual nature of a document (source code versus compiled result) to become completely natural: when you write HTML code and preview it in your browser, you see two different representations of the same document. In other words, the “WYSIWYG” principle (which in the 1980s was quite an annoyance to T_EX) has, at last, lost its supremacy.

Also, thanks to the Web and to political changes, there are no frontiers anymore, and standards such as Unicode have emerged to allow communication in all languages. T_EX has always been a pioneer in multilingual typesetting, a feature that becomes more and more important today. As we will see in a while, a successor to T_EX is one of the few (if not the only) software packages nowadays allowing true multilingual typesetting.

But are all things really well in the best of all possible worlds?

Talking of free software, let us return to one of the biggest achievements in the public domain, namely the Linux operating system, developed by hundreds of people

all around the world. The obvious question to ask is: can T_EX be compared to Linux? Unfortunately *not*, for several reasons.

First of all, is the absence of a Linus Torvalds for T_EX: in fact, the author of T_EX, Donald Knuth, one of the biggest computer scientists of the twentieth century and indeed a fabulous person with interests far beyond computer science, unfortunately decided to stop working on T_EX once a certain number of goals were achieved. This happened in 1992, when version 3 of T_EX was released. New versions after that were just bug fix releases. There are some small groups of people working on specific T_EX-related projects (such as the L^AT_EX group, the Ω group, the $\mathcal{N}\mathcal{T}\mathcal{S}$ group, etc.) and some institutions maintaining specific T_EX packages (such as the $\mathcal{A}\mathcal{M}\mathcal{S}$). But outside of these, there is no coordination of the individual programming efforts.

Secondly, the goal to be reached in further developing T_EX is not quite clear. T_EX is a program dedicated to *typography*, a craft that very few people actually have studied, some people have learned by themselves—mainly by actually making books—and most people are generally unaware of. To continue our comparison with Linux, the latter is an operating system and hence deals with the global use of the computer: it is easy to imagine improvements, and if you lack imagination, you can always look into commercial operating systems to get ideas. T_EX is the *only* piece of software dedicated to typography, and it does a *very* good job. Some people even believe that T_EX *is* already perfect and hence there is no need for further improvement. But what *is* the ultimate goal of T_EX, its *raison d'être*?

For years now, pessimists have been predicting T_EX's extinction, but T_EX is still alive and kicking! Maybe the most important reason for that is that T_EX bridges the gap between the cultural heritage of the precomputer era and us today. Typography is both a craft and an art 500 years old, and Donald Knuth actually learned it and encoded his knowledge to T_EX so that T_EX is a "typographer-in-your-machine." Using just standard L^AT_EX, people unaware of typography can produce decent documents by including in their text some markup reminiscent of XML. With a little more effort, and using a little more than standard L^AT_EX, people aware of typography can produce brilliant documents. This degree of proficiency at attaining the sublime is cruelly missing from contemporary commercial software where the goal is not really commitment to our cultural heritage. T_EX is a craftsman's tool like in the good old days: using such a tool, a novice can produce decent results and a master can make works of art. And, as always with Donald Knuth, a work of art in the context of T_EX is both beautiful typesetting and efficient programming.

This book presents some of the achievements of the T_EX community in the last two decades. For reasons inherent to the T_EX users community, the tools presented are of various degrees of quality, efficiency and compatibility. There are so many tools (or packages, in L^AT_EX parlance) available from the Comprehensive T_EX Archive Network that there are strong chances you will find a package for any of your potential needs.

But how efficient will that package be, or how compatible with other packages written by other authors? This is an important question because improvements or resolutions of conflicts require a good knowledge of \LaTeX . Often, there is a high level of support by the author of the package. But what happens when the author is hard to reach, or even unknown? Others in the \TeX community may help you, but, as always in the public domain, there is no guarantee that you will get the help you need precisely when you need it.

This situation may seem frightening to people who expect absolute efficiency and immediate compatibility from software they use. There is a working scheme that is better fit to \TeX and \LaTeX , namely that of small groups of people sharing the same computer resources and being assisted by a “system administrator” (or “guru”). The “guru” is supposed to know \TeX and \LaTeX sufficiently well and to have the necessary time and energy to solve problems for the rest of the group, which can then smoothly use the software. Unfortunately, this organizational scheme does not fit individual personal computer users, who have to be simultaneously users and administrators.

So, how does one deal with problems in \LaTeX packages? Well, experience shows that if you are a convinced $\text{\LaTeX}/\text{\TeX}$ user, then you always manage to get by the problems, either by searching in literature (and books such as this one are very important for that very reason) by diving into the code and trying to “make it work,” or, finally, by contacting other members in the community, even if the developers of the package are unreachable. A combination of these three methods actually works best. What is important is to realize that you are extremely lucky to be able to do all three: you have valuable books (such as this one and others), you can indeed dive into the code since it is open and freely distributed, and you can indeed contact others since there is a virtual—and furthermore friendly and united—community. Commercial software does not offer these opportunities.

The reader may have noticed that this book often mentions Ω and Λ . Where do these mysterious names come from and how do they fit in the “ \TeX and friends” context?

Ω , one of the major current \TeX projects, is an effort by two people (John Plaice and myself) to develop a successor to \TeX . It started two years after Donald Knuth’s decision to freeze \TeX . The philosophy of Ω is to take \TeX as a starting point and to progressively add techniques and tools allowing the resolution of specific typesetting problems one at a time. The first major goal was to achieve typesetting in all languages of the world in the most natural and efficient way. In particular, one of the tasks that Ω seeks to accomplish is Unicode compliance (as explained in the book, Unicode is a standard 21-bit encoding for information interchange).

But Ω has other goals as well and is in fact an open platform for enhancements and additions to \TeX . The name Ω has been chosen because traditionally the last letter of the Greek alphabet stands for ultimacy, “the ultimate tool,” and also probably because 50% of Ω ’s development team is Greek. Finally, because choosing a Greek letter as the

invariable and nontranslatable name and logo of a program is an additional argument for using the Unicode encoding (just as the fact of lowering the letter ‘E’ in the T_EX logo was a very clever way to show the absolute need of using T_EX to typeset even its own name).

Contrarily to Ω , which is existing, and quite extensive software, Λ is just a nickname, a kind of parody of the L^AT_EX name: In fact, the “La” in L^AT_EX comes from “Lamport”, as in Leslie Lamport, the author of pre-1992 L^AT_EX. The word “Lambda” also starts with “La”, but has no relationship whatsoever with “Lamport” and is a Greek letter just like “Omega.” Λ stands (as explained in this book) for the current L^AT_EX (an achievement of the L^AT_EX team, headed by Frank Mittelbach) when used in conjunction with the Ω engine.

It is quite probable that future versions of L^AT_EX (for instance, version 3) will either be entirely written for Ω or at least have parts dedicated to Ω , in which case the Λ nickname will be useless. Also, due to the fact that the greatest part of Ω resources has not yet been released publicly, and that the Ω team still has to take a certain number of important global decisions, some information on Ω contained in this book may undergo minor changes in the future. In particular, there is (at the time this text is being written in March 2002) still no standard user-level L^AT_EX interface for Ω .

Nevertheless the basics of Ω will not change, and this book has the merit of being the first one to describe some of the very fundamental aspects of Ω , like Ω Translation Processes, Ω Virtual Property Lists, and so on and to illustrate them by examples.

The Future

The “future of T_EX” (including the question of whether there is a future for it at all) has been a popular discussion subject for years in the T_EX community. In fact, T_EX is the sum of a big variety of different things, and for each one of them one can more or less predict its destiny, but one can hardly do this for the sum of them.

For example, T_EX is both a programming language and a program (a “compiler” for that language): one could imagine that the program survives (for example as a typesetting or “rendering” engine inside a bigger system, and rumors circulate that this is already the case in Adobe InDesign); on the other hand, one could imagine Ω or some other successor to T_EX becoming more and more different from T_EX but—for reasons of upward compatibility—keeping the same programming language for input.

Besides being a programming language and a program, T_EX is also a popular notation for mathematical formulas: mathematicians worldwide use T_EX notation when writing formulas in, for example, e-mail messages: $x^2 + y^2 < 1$ with or without dollars is a natural choice for expressing the formula $x^2 + y^2 < 1$ in a text-only context. For writing mathematical formulas, T_EX is exhaustive, clear, unambiguous, and short enough—all of the qualities of a good notation.

In recent years, the computer industry has become more and more involved in typesetting engine projects: the context in which source code of some kind has to produce more or less rigid formatted output becomes more and more important. After the first

enthusiastic years of explosion of the Web, people realized that HTML (even combined with CCS) was definitely *not* sufficient for formatting documents. XML provided the necessary standard for structuring documents in an arbitrarily fine way, but still there was no “standard” way to *represent* an XML document. In October 2001, a new standard filled that gap: XSL-FO. The tools provided by XSL-FO for formatting documents are a quite serious challenge, and a new generation of XSL-FO-compliant typesetting engines is slowly emerging.

More generally, the current trend is to use XML as the basis of every kind of file format. For example, the SVG standard is, in some sense, an “XML-ized version of PostScript.” One could very well imagine all file formats involved in \TeX becoming XML-compliant: the input file could be pure XML “processing instructions” for including code in the \TeX language) the DVI file format could be replaced by SVG, the font metrics could be expressed in XML, illustrations could be in SVG instead of EPS, and so on. In that case, \TeX (or Ω , or some other successor to \TeX) would simply transform one XML document into another one. The fact that XML document transformation is nowadays an increasingly popular and important concept is by no means a coincidence.

Another area where Ω can be applied to revolutionize the electronic document is that of adaptive documents. A research project in that area deals with *vario-documents*, namely documents that contain a big number of page descriptions and display the right one according to context parameters, just as HTML browsers reflow text when their display window is resized. Only here each page description of the document has been compiled in advance by a “super- Ω ,” always with the same high typesetting quality standards.

Yet another area of drastic improvement of Ω 's capabilities would be an on-the-fly interaction between typesetting and dynamic fonts. Already, in *Vector \TeX* (a commercial \TeX for Windows platform), Dimitri Vulis has included `METAFONT` capabilities into \TeX . By using more modern font formats, such as OpenType, one could obtain a dialog between the font and \TeX 's typesetting engine so that each one instructs the other on constraints and context parameters and so that the final result is optimal for both.

There is also the more global, operating system-oriented point of view: Ω could very well become a server, and arbitrary client applications could send requests with text extracts and macros or parameters and receive in return small parts of page descriptions.

All of these “mutation” scenarios could be compared with the common skeleton of many science-fiction stories, where humans mutate to become less and less organic. Usually sci-fi authors want to express the fact that despite and beyond the changes of the human body (including an artificial brain), a core of *humanity* will always emerge as a fundamental quality of mankind. This is exactly the case for \TeX : I am convinced that however drastically \TeX (and its successors) will change in the future, its fundamental quality, which is the love of one man—and not just any man!—for good typography and good programming will always prevail and will always be the ultimate guarantee for the survival of this magnificent tool.

If this book succeeds in transmitting the fundamentally human quality of $\text{T}_{\text{E}}\text{X}$ and its successors, due to the love, sweat and tears of Don Knuth and the hundreds of members of the active $\text{T}_{\text{E}}\text{X}$ community, then it will have reached its goal. I sincerely hope it does.

Yannis Haralambous
Brest, France
March, 2002