David Kastrup

David Kastrup is a frequent contributor to `comp.text.tex` and manifestly a TEXnician (see the answer to Exercise 1.1 in *The TEXbook*).

[Interview completed 21 August 2006.]

Dave Walden, interviewer: Please tell me a bit about yourself personally, outside the world of TEX.

David Kastrup, interviewee: Well, I am afraid I'll have to use the T-word when talking about jobs: for the last few years I tried working on TEX-related tasks, self-employed, but this will have to change soon since it does not pay the bills — partly because of the expectation of people that free software should be written for free, partly because of my preference to focus primarily on things deemed impossible. Unfortunately, there are few tasks not involving clearly possible parts.

I started fiddling with computers at about 13, first working with an Olivetti mini-computer, then later switching to the Cyber 175 in the university computing center. So I actually worked with punch cards as main storage and input media for some time. My own first computer was a Nascom II (Z80-based) which I soldered together myself. I also eventually wrote a CP/M BIOS for it (which became somewhat ubiquitous) and designed, wired and soldered most of its extensions.

For the largest part of my life, I lived in Aachen, Germany, where I eventually got my diploma in electrical engineering. I went to Bochum for a doctorate, but it remains pending completion. For quite some time I have been living on my own with no family to support.

Apart from various computing diversions, my main hobbies are rock climbing and music, with a focus on the Baroque. As part of that I am currently singing alto in some choirs, and I am working on a scheme for employing this voice type in non-classical settings. In the process, I recently added the accordion to my bag of instruments (already containing violin and guitar) to deliver Yiddish, Irish, Russian and German stuff mostly of a folk character.

As a curiosity for your American readers, I never owned a car and do almost all of my transportation by bike (the kind with human propulsion) within a compass of 15 miles.

DW: How and when did you first become involved with TEX and its friends?

DK: I actually don't remember too well. There was some project where porting TEX to some embedded system was considered, but it never got off the ground. That was the

time when I got *TEX: The Program*, in hardcover. Of course, this was still TEX 2.0. LATEX came into play at some later time. I also had to write my own printer driver when the HP Deskjet+ came out (which cost about US$1000 at that time): it was the first "affordable" printer capable of printing graphics as fast as text. I adopted a driver from Nelson Beebe but had to significantly change it to make it work on a computer with 640kB of RAM (with 300 dpi output) *and* keep the printer busy in spite of a rather slow computer. Nowadays I am probably mostly known as the maintainer of AUCTEX, a job I took over from Per Abrahamsen.

DW: Can you tell me and other readers who are not familiar with AUCTEX (`http://www.gnu.org/software/auctex/`) something more about it.

DK: AUCTEX is a TEX/LATEX/ConTEXt/Texinfo editing support package under Emacs (probably the world's greatest editor prototyping platform). AUCTEX is a combination of editing functionality (summarized under a menu called `LaTeX` or similar) and job control (summarized under `Command`). It can run the various TEX processors and utilities and jump to the location of prospective errors in the source. Per was the second maintainer of AUCTEX and got tired of it. Development was pretty much stalling, and so I finally took up his offer of taking over maintenance. AUCTEX's job control is more sophisticated than that of the default modes: it can seamlessly switch between producing PDF and/or DVI files, it deals well with multiple-file documents, it can make use of forward and backward search for DVI (using source specials) and even use pdfsync for PDF. Its text formatting and indentation is pretty "natural", it can understand the outcommented document structures in a `.dtx` file and appropriately format them, too.

It also offers folding away of syntactic structures (like footnotes) in order to better focus on the important parts of the document. And, of course, it can replace math and similar constructs in the source buffer by WYSIWYG renditions of them, by using the magic of preview-latex (a combination of `preview`.sty for extracting images and glue code in Emacs for running LATEX, dvipng, Ghostscript and other utilities and placing the resulting output in the buffer).

DW: I see *many* contributions from you on the `comp.text.tex` discussion group. What are your motivations for participating so much for so long?

DK: Showing off, I guess. TEX is so contorted and idiosyncratic that it gives ample opportunity for impossible things. Apart from that, it annoys me if stuff gets done in unnecessarily complicated or inelegant ways, even if the elegance comes at a price.

DW: Will you please elaborate on your view of TEX, the language.

DK: Let's be blunt: TEX is not a language for quick-and-dirty solutions. There are those things it has been specifically created for, and those tend to be quick-and-easy. That set is limited. Anything else is more in the slow-and-painful department. And sometimes there are shortcuts through the pain. It is like ripping a band-aid off instead of slowly pulling it. Here is an example:

```
\def\sorttext#1{\setbox0\vbox{{\language255\hsize=0pt\hfuzz\maxdimen
   \parfillskip0pt\noindent#1\par}\sortvlist\unpack}\unvbox0 }
\def\sortvlist{{\unskip\unpenalty \setbox0\lastbox
  \ifvoid0\noindent\else\setbox0\hbox{\unhbox0\ }\sortvlist\sortin\fi}}
\def\sortin{\setbox2\lastbox\ifdim\wd2>\wd0{\sortin}\fi\box2\box0}
\def\unpack{{\setbox0\lastbox\ifvoid0\indent\else\unpack\unhbox0\fi}}
```

```
\sorttext{%
The fall (bababadalgharaghtakamminarronnkonnbronntonner%
ronntuonnthunntrovarrhounawnskawntoohoohoordenenthur%
nuk!) of a once wallstrait oldparr is retaled early in bed and later
on life down through all christian minstrelsy. The great fall of the
offwall entailed at such short notice the pftjschute of Finnegan,
erse solid man, that the humptyhillhead of humself prumptly sends
an unquiring one well to the west in quest of his tumptytumtoes:
and their upturnpikepointandplace is at the knock out in the park
where oranges have been laid to rust upon the green since dev%
linsfirst loved livvy.
}
```

[Editor's note: you can see the output of the above `\sorttext` definition at `http://www.tug.org/interviews/interview-files/david-kastrup-example.pdf`.]

Now the above code is dense. Every token has a definite purpose and could not be replaced by something shorter. It is the essence of elegance in TEX programming, and yet it is a total incomprehensible mess, even though it is supposed to solve a simple programming task.

There is an annual obfuscated C programming contest. The same would be rather pointless for TEX: every nontrivial task has only obfuscated solutions, anyway.

TEX is suitable for typesetting *The Art of Computer Programming*. That's about it. Its natural extension language is Pascal, but hardly anybody uses that: I can't think of anybody who has ever invented new whatsits, one of the basic extension mechanisms in TEX.

Instead, macros are used as a substitute for programming. TEX's macro expansion language is the only way to implement conditionals and loops, but the corresponding control variables can't be influenced by macro expansion (TEX's "mouth" in Knuth's terminology). Instead assignments must be executed by the back end (TEX's "stomach"). Stomach and mouth execute at different times and independently from one another. But it is not possible to solve nontrivial programming tasks with either: only the unholy chimera made from both can solve serious problems. ε-TEX gives the mouth a few more teeth and changes some of that, but the changes are not really fundamental: expansion still makes no assignments.

DW: My understanding is that you have been a participant in the exxTeX project (`http://www.extex.org`). What is your role in that, can you tell me something about where it stands, and more generally what are your thoughts on a successor to TEX?

DK: I am not at the current point of time involved in exxTeX. They did consult me while in the initial planning phase as an expert on TEX's internals, in order to derive a good understanding of TEX's existing architecture, and a good understanding how to best modularize it in a manner useful for extensions. We also tried fleshing out what kind of extensions would be desirable, and how to accommodate them. I have not, however, followed the project's progress afterwards.

I have my own ideas about how to make work on a successor to TEX not drown in complexity, and that includes an implementation language which can express the kind of optimization TEX indulges in in a natural manner. Like Emacs Lisp, an implementation language that has evolved with Emacs itself, I think that such a language should coevolve with a typesetting system. In a way, the traveling junkyard of editing solutions that is Emacs has been able to grow much beyond the initial single-person project because of

Emacs Lisp and resulted in a decrease in complexity for an average programmer.

TEX, written in a procedural language, is as complex as a single person of extraordinary capabilities is able to manage. Better modularization, as done in the exxTeX project, can only somewhat reduce the complexity any single extension project will have to deal with.

DW: I presume that the implementation language ideas you just mentioned are the same as those reported in the recently published EuroTEX proceedings. I'll go read that article again.

Since this interview is for the TEX User Group, it occurs to me to ask if TUG or any of the other TEX user groups have relevance to your work and if you see relevance for them regarding the future of TEX?

DK: For me, naturally the most relevant local TEX user group is DANTE in Germany, of which I am also a member. The relevance of this group and others for TEX development is mixed: even though DANTE is the best funded LUG as far as I know, its means are not sufficient for funding long-scale full-time work being done in Germany. DANTE is quite important for organizing conferences and enabling developers to attend them, also for keeping server structures going and so on: in short, they do a lot to maintain a space where hobbyist work will reach its audience and communication is possible. The project funds also are helping to keep work focused (like that of the Latin Modern fonts) and developers willing to finish what it takes to deliver value to the users.

However, the limitations of budgets require working with an efficiency that is not viable in commercial programming projects: with customary project management, programmers are to some degree replaceable, and a project's goals can be finished even with some fluctuation in the personnel. This does not really work for projects funded by the TEX groups: basically, they can only fund persons, not projects, and which projects happen to be funded depend strictly on what people want to do and can be expected to do. If people leave a project, it is usually not viable to replace them.

So in a lot of ways, the user groups do the best they can to increase and support the existing impetus of TEX development, but can't really do much in the way of steering it.

DW: My previous interview was of Alex Simonic and Adriana McCrea who develop and support WinEdt. It sounds like they are at least eking out a living with their shareware package. You indicated in your first answer that making a living in the world of TEX is tenuous. Do you think there is something wrong with the "open source" or "free software" model under which much of the TEX and TEX-related work seems to be done?

DK: Please note that shareware is not free software. The principal problem with free software as a business model is that there really is little in the way of bootstrapping it. Programmers tend to be "mad scientists" to some degree or other, and TEX programming mostly has attraction for the worst of those. This means that you often have people with a bad judgment concerning business requirements and project management and time planning and customer interaction. For proprietary software, this is less of a problem: if you are the only supplier for a marketable product, poor market interaction does not kill your business prospects. In a free software market, however, being the developer of a product gives you just a headstart for marketing your own product, but it does not put anybody else out of the race.

And that makes the transfer of a project into a self-sustaining business with a working marketing department and project management quite a bit harder to do with free software. There are companies like Red Hat and a lot of other solution providers who managed that transition, but it is by no means easy. You need to keep an innovative edge

over your competition throughout, and that is not easy.

On the other hand, making actual progress instead of reinventing the wheel can't be expected to be easy, and the current market dynamics focused on proprietary software waste a lot of energy on duplicate efforts.

DW: Thank you very much for taking the time to participate in this interview. I hope we will have an opportunity to meet in person at some point.

DK: You are welcome.