TUG'95
Questions and Answers with
Prof. Donald E. Knuth

Don Knuth with past (and present) TUG presidents [Malcolm Clark absent]. (l–r) Pierre MacKay, Bart Childs, Don Knuth, Nelson Beebe, Christina Thiele and Michel Goossens (Photo courtesy Luzia Dietsche).

Session called to order by Barbara Beeton.[1]

Barbara: I've had the pleasure of knowing Don for quite a long time. I'd like to start off with the first question . . .[2] the obvious question, *other* than what the T-shirt means: How's volume 4 doing? [laughter]

DEK: Thank you very much, Barbara. You said that I'm the reason that most of you are here. I think that Barbara is just as much a part of the reason, as me or anyone, about why we're here. She's done wonderful work for us all these years.

You know, the reason I came to this meeting is that, after the 10th TUG conference, I promised I would come to the 16th because that was the most important number for a computer scientist. Sixteen is not only a power of two, it's two to a power of two to the two, and 16 is two to the two to the two to the two to the minus-infinity. [laughter] So, it's about as binary a number as you can get until 65,536. Numbers are important to me. So this is a momentous meeting for the whole TEX project.

[1] This Q&A session took place at the TUG'95 Annual Meeting in St. Petersburg, Florida, on 25 July 1995 (9–10:30 a.m.) –Ch.

[2] Transcription notes: (a) . . . means a bit of a pause in the speaking; (b) [. . .] means text missing because unclear or unintelligible.

I looked up what was I doing exactly 16 years ago today. And I found out that, of all things, I was working with Barbara Beeton, who had come to Stanford for 2 or 3 weeks for the American Math Society. They were showing me the problems they were having with making the index to *Math Reviews*. So, on July 25th, 1979, Barbara and I were trying to figure out how to do the index to *Math Reviews*. This led to more powerful facilities for leaders and things like that, because the index to *Math Reviews* has occasions where you have lots of problems because you have to run the dots in a certain way, based on how many references there are. It was very interesting because I found two bugs in TEX that day — they were numbers 414 and 415 in the history of the development of TEX. Something to do with an error message in case you get to an end-of-file in the middle of something else. Anyways, it was that kind of error. That was sixteen years ago today.

Barbara said: "Why do I have this T-shirt on?" The T-shirt says:

$$x^n + y^n = z^n \quad \ldots \textit{NOT!}$$

That's a mathematical formula which I could show you how to do in TEX, if you're interested ... [laughter]

I'm wearing this T-shirt because I had a thrill a month ago. It's continually exciting for me to see the uses that people are making of TEX all over the world. Very exciting. One of the most important somehow to me was last month when I went to the library and saw Andrew Wiles's solution to Fermat's Last Theorem. I think a lot of you know that it was front-page news.[3] For a while, there was some doubt whether there was maybe still a gap in his proof, and then it was fixed up. So, this is the most famous by far of all problems in mathematics. Just as people can remember where they were when they heard about Kennedy being assassinated, I know mathematicians can all remember where they were when they first heard that Fermat's Theorem was solved. The paper came out in the *Annals of Mathematics* last month;[4] it arrived in our library and I saw it sitting there, and I looked at it and it was just wonderful for me because it was in TEX and it looked gorgeous! [laughter] This to me was the ... you know, it was so ... I mean, I almost felt like I had helped to solve the Theorem myself! [more laughter] So now, I'm also very glad to find out that the people who were responsible are here this week. In the back row, we have the editor of *Annals of Mathematics*[5] and also Geraldine Pecht, who was the typesetter. So, it's a thrill to me. Let's give them a hand. [applause]

But I didn't want to talk about anything prepared, I wanted to answer questions.

So, Barbara asked the first question: "What about Volume 4 of *The Art of Computer Programming*?" Now, I usually only answer that question on special occasions. [laughter] These days I'm a full-time writer and I'm working very hard on *The Art of Computer Programming*. We have, uh ... let me just see if I can find a scratch page to work on ... [went to overhead with live Emacs screen on computer]. This is just to remind me about what to talk about ...

Now, it used to be that we used ACP as the abbreviation for *The Art of Computer Programming*. But someone else suggested that it should be called TAOCP. So now this is the new abbreviation for it: *The Art of Computer Programming*. This is my life's work, this is what I started working on in 1962, and I think I have about 20 years of work to go on it yet, after which I'll be 77 years old. So you see why I retired early — in order to be able to work very hard on this.

Bart Childs: Should that be TAOCP? [there was a typo on the screen, which showed TOACP]

DEK: It is. You want me to enlarge this font? I only have three or four fonts ... Oh! [sees typo]

Bart: Does that mean I get a check for five dollars and twelve cents? [laughter]

DEK: Not for you!! [laughter] ... You know I did that on purpose just to see if anybody was looking. [laughter] Alright ... Then there's this *other* book I'm working on, called *The Art of Computer Programming* [laughter] ... OK, so ... Boy, am I nervous. [laughter]

So, in order to finish this project, I have to work very hard, because computer science people keep discovering new things. Originally, my idea was that I was going to be able to summarize *all* of the good stuff in computer science, but now I have to say that I just have to work very hard in order to summarize all the *classical* good stuff in computer science. I'm working especially to get all the history correct and to lay the right foundation for the specialized things. But I can't go up to the frontiers in everything as I could have in the 60s, when I began the project. I worked on TEX for about 10 years total, I guess, and I'm hoping that those 10 years

[3] Wiles proved that $x^n + y^n = z^n$ is impossible when $n > 2$ and $xyz > 0$.

[4] Andrew Wiles, "Modular elliptic curves and Fermat's Last Theorem," *Annals of Mathematics* **142** (1995), 443–551.

[5] Maureen Schupsky, Managing Editor, Princeton University Press.

actually will save me about 6 or 7 years of the time I would have had to put into *The Art of Computer Programming* because I can now do my other work more efficiently.

The Art of Computer Programming is sort of what I view as the thing that I'm most uniquely able to do in my life. I'm feeling very healthy now and happy, and I feel that what I'm accomplishing every week is about as much as I've ever been able to do in my life in a week. So, I hope I can keep it up for a while. But I know that it takes a lot of time. That's why I'm retired and I'm working full-time on this.

I spent last year building infrastructure for the project, which meant making large computer files of what's in my house. So I have thousands and thousands of items that I've indexed and put into place so that I know how to find things.

Right now, my current project is to finish answering mail about *The Art of Computer Programming* that came in since I was working on TEX. [laughter] You know that if anybody found errors in *The TEXbook*, I answered the mail eventually and paid for the errors and so on. Well, people also get a reward for finding errors in *The Art of Computer Programming*. But the fact is, the last time I wrote a check for that was July of 1981. [laughter] In August of 1981, my secretary started issuing a form letter, typeset with TEX, saying "I will get back to you soon." [laughter] I started putting these letters into a little pile. Then the pile got to be a bigger pile, and it got mixed with all the other preprints I was receiving, until the pile grew to 260 inches high! Now, convert that to centimeters ... well, anyways, it's a lot! [laughter] It was about seven to eight meters of material. I went through all that and I am now answering those letters. Actually, the number of letters that I hadn't answered was less than 500 — something like 450 letters — and I'm now answering those letters and hoping that the checks will reach the people at the addresses from where they sent me their comments.

I'll show you the errata because I'm working on it now. Here's an example [Figure 1]. Just so you can see what the Index is about. This is p. 77 of the errata to Volume 1.[6] This is 8pt type being enlarged a lot. I wanted to show you one of the things I'm working on right now ... For all the authors that I cite in *The Art of Computer Programming*, when they have a non-Western name, I'm building a big database of the names in their native script, for example, Chinese or Japanese. (I haven't put in

von Neumann, John [= Margittai Neumann János], 18, 225, 456.
Wadler, Philip Lee, 594.
Wall, Hubert Stanley, 481.
Wallis, John, product, 50, 112, 480.
Wang, Hao (王浩), 382–384.
Wang, Paul Shyh-Horng (王士弘), 436, 631.
Watson, Dan Caldwell, 248.
Wedderburn, Joseph Henry Maclagan, 583.
Wegbreit, Eliot Ben, 603.
Weierstrass, Karl Theodor Wilhelm, 381.
Wiles, Andrew John, 465.
Wilf, Herbert Saul, 92, 483.
Windley, Peter F., 518.
Wise, David Stephen, 420, 434, 595.
Wiseman, Neil Ernest, 420.
Yao, Andrew Chi-chih (姚期智), 538.
Young Tanner, Rosalind Cecilia Hildegard, 75.
Zave, Derek Alan, 90, 603.
Zeilberger, Doron, 64.

Figure 1: Excerpt from the end of `err1.dvi`

the Indian names yet, but I'm working with people in India to get that solved.) ... Right now, I have most up-to-date stuff on the Chinese part of it ... I have bitmap fonts for all the UNICODE characters — especially the Chinese characters — and I now have a pretty good database of these things, hopefully. So, by the time UNICODE software is ready and available, I'll be ready to use it and I'll be able to typeset the various names properly.

I have some interesting Emacs macros that help me with the UNICODE characters even though I don't have any software yet for UNICODE. I can type in four hexadecimal digits in Emacs and then say `M-X unic` and so magically it will convert that into the Chinese character — the bitmap of it — which then can be put into the document. There's Andy Yao's name in Chinese. I just have 24-by-24 bitmaps of all these characters, but it's enough for proofreading purposes. I have it set up so that it's very easy for me to get the characters into my file. The Emacs macro sends it to a little program that looks it up in a file, finds the bitmap, and inserts it into TEX format. Between angle brackets, the stuff's sent to PostScript. I'm planning that at the end of the year, I'll announce this errata list, which will be finished by that time. Right now, it's about 180 pages and I'm still building it while I'm answering these letters.

Then we're going to issue Volume 4 in fascicles, about 128 pages at a time. The idea is to do that about twice a year for the next 10 years. My steady state, I figure, is going to be about 256 pages a year of output. We're going to have three or four fascicles in hand before we actually start this publication. The first four fascicles — one of them will

[6] See Knuth's Web pages at `http://www-cs-faculty.stanford.edu/~knuth/taocp.html` –Ch.

be this larger one [the update errata to Volumes 1, 2 and 3]; the other three — the second one will be the design of a computer called MMIX[7] which is replacing the MIX computer. MMIX is a RISC machine, very much like the computers that we're all converging to these days. It's a 64-bit RISC computer which I might even *own* one day — I don't know, if somebody's going to build it. I've got the experts in the field helping me design it. Dick Sites, who was the architect of the Alpha chip, is one of my students and has promised to work on all the final steps of it. Also John Hennessy, who designed the MIPS chip. And some of the people from SPARC. So MMIX is going to be a nice clean RISC computer in order to make experiments on algorithms, to see how much the different cache management schemes work with different coding algorithms and so on.

The second fascicle will be to replace MIX by MMIX and I'm hoping that, eventually, every time I have a MIX program in *The Art of Computer Programming*, it will be replaced by an MMIX program. I'm not going to do that until I finish Volumes 4 and 5. But I'm hoping that a lot of other people will have done that work already by the time I get there. People have already promised me that they're going to have a C-compiler up for MMIX next year, and we're trying to get operating systems written for it.

The other two fascicles — one of them will be the first part of Volume 4, which talks about bit-fiddling. These are a collection of techniques that are mostly in folklore about efficient methods for computers where you're using the logical operations of a machine — the exclusive 'or', the 'and' and 'not', as well as 'plus', 'minus', 'times', and 'divide' — to gain great efficiency. I've got that material pretty much written already. In fact, it was what I had drafted just before I started working on TEX — that was 1977 when I wrote the first draft about bit-fiddling.

And then I get into the study of brute force enumeration methods. The subject of Volume 4 is combinatorial algorithms, and this means the methods that have been developed to deal with problems where you have zillions of cases; all kinds of ideas have come up as to how to speed up, by many orders of magnitude, the obvious methods for dealing with cases of combinatorial importance. I begin the chapter by talking about bit-fiddling, and the next part by talking about fast methods for listing all permutations, and listing all subsets of a set, and things like that. A vast literature about such things exists. Surprisingly, more people have written papers about generating permutations than about sorting. Sorting is the idea of putting into order; generating permutations is about putting into disorder. More people have explained how to unorder things than to order things. [laughter] Most of those papers are fairly repetitious and trivial, though, and not as interesting as the sorting papers, so the difficulty for me is mostly to survey this literature and put it all together. Most of the people writing on it were unaware that other people were working on the same thing.

OK. Well, that's more than enough of an answer to your question, I hope, on the state of Volume 4. Every day I should be able to finish about a page or so, and I think I've been going at about that rate for a while now.

While I've got this on the screen ... Let's try going forward a couple of pages ... I really want to look at this equation here [Figure 2]. I used to have Fermat's Last Theorem as a research problem at the beginning of the book. [laughter] In the errata it says now: "Prove that when n is an integer greater than 4, the equation $w^n + x^n + y^n = z^n$ has no solution in positive integers." So, now I've just added another variable to the equation [laughter] and we've got another good research problem. And it turns out that for $n = 4$, there are infinitely many solutions. The proof of Fermat's Last Theorem caused a personal crisis for me, but I've now resolved it in this way. [laughter]

I want to look at p. 61 as an example of new material [Figure 3]. This is about some very nice constructions that come out of studying trees. I put this up as an example of an illustration — I'm doing all the illustrations for the book in MetaPost, and this is one of the ones that happened to be around here. I love MetaPost, and John [Hobby], in the next talk this morning, will show you his system. It handles technical illustrations much better than anything else. The great beauty of MetaPost from my point of view is that if I have to modify any of my illustrations later on, maybe a year later, I can look at it and see from the MetaPost code exactly what I had in mind when I made the original. MetaPost is a declarative language where you state the characteristics that you want your illustration to have, and then it draws the diagram. I've got 5 or 6 examples in the errata where I've either redrawn an old figure or, as in this case, made a new one. Many, many other examples that I've done with MetaPost when I did the *Stanford GraphBase* book convinced me that it's really the answer for technical illustrations. There's nothing else I think

[7] MMIX = 2009, in roman numerals.

Page xi replacement for exercise 3 25 Mar 1995

3. [*34*] Leonhard Euler conjectured in 1772 that the equation $w^4 + x^4 + y^4 = z^4$ has no solution in positive integers, but Noam Elkies proved in 1977 that infinitely many solutions exist [see *Math. Comp.* **51** (1988), 825–835]. Find all integer solutions such that $0 \le w \le x \le y < z < 10^6$.

4. [*M50*] Prove that when n is an integer, $n > 4$, the equation $w^n + x^n + y^n = z^n$ has no solution in positive integers w, x, y, z.

Figure 2: Excerpt from the beginning of `err2.dvi`

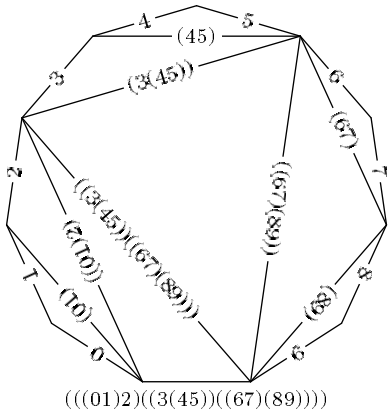

Figure 3: Excerpt from `err1.ps`, showing MetaPost illustration

will ever be able to be much better. It's not the answer for the kind of illustrations that people do when they're doing advertisements or something like that, but when you're writing a techical book, you have — the speaker yesterday made the same point with PSTricks — to produce a diagram that has a certain mathematical property to it. A Super MacDraw-type of program doesn't very easily give you this ability, while MetaPost does it, and very easily — all the technical things are correct, according to the mathematics. It's kind of scandalous that the world's calculus books, up until recent years, have never had a good picture of a cardioid. Yesterday, you saw a cardioid on the screen. Nobody ever knew what a cardioid looked like because it was done by some graphic artist who was trying to imitate another thing, but they never had a real one shown.

OK — I've talked too much about this. Ask another question!

Robert McGaffey: Do you think the RSA algorithm will ever be broken?

DEK: Do I think the RSA algorithm will ever be broken? Now, the RSA algorithm is the Rivest–Shamir–Adleman scheme for encryption. People have just factored the key of the original cipher that was put up by R, S and A in 1977. That was a 140-digit number, which was a very weak version of the thing. But Rivest et al. said: "Here's our secret message. Can you decode it?" By the time someone decoded it last year, Ron [Rivest] had lost the original answer ... forgotten what it was. But it turned out that, after decoding, the encoded message was: "The secret word is squeamish ossifrage." That was the solution. People, in order to break this cipher, had to factor a 140-digit number, and it was done with many thousands of hours of computer time last year. Now the thing is, though, if you go from a 140-digit number to a 141-digit number, already the problem gets much larger. So if you go to a 300-digit number, it would, as far as we know — all of the computers running now in the universe would not be able to do it. But there might still be advances in factoring, so Rivest himself predicts that a 300-digit encoding would last at least for about 30 years. A 500-digit number for a hundred years — he would rather confidently predict that's true.

We doubt if anyone's going to discover a magic way to factor numbers. The only problem is that it's illegal to use the full power of RSA. I mean, the government wants to be sure that it can read secrets if necessary, because it doesn't want the Mafia to have the secrets. So we now have a peculiar situation where it's against the law to compute a

certain mathematical equation, a mathematical formula. Well, I don't like confrontational issues. [laughter] I don't live in a secret way ... I mean, I'm not a secretive kind of guy. I spent a year of my life working in cryptanalysis and I've met a lot of wonderful people in that community, but I knew that the life wasn't for me. I like to be a college professor and tell what I know. So, I'm not a good consultant on that kind of question. But it is possible to send secure information assuming that nobody can factor numbers. That's either a blessing or a threat, depending on your point of view. For me, I can see it from both sides.

In the back?

Michael Sofka: Is $P = NP$, and if not, how far do you think we are from the proof?

DEK: $P = NP$ is the most famous unsolved problem in computer science, analogous to Fermat's Last Theorem, although the $P = NP$ problem has only been around for about 30 years, 25 maybe. In the context of combinatorial algorithms, it says: Are we going to be able to solve problems that would require going through 2^n cases? Can we actually do those in n^{10}, or something like that, if we knew the best method? If $P = NP$, the answer would be 'yes', with some polynomial: we could reduce all these exponential problems to polynomial problems. If not, then it says 'no', we'll never be able to reduce them.

No, I have a feeling that someone might resolve the problem in the worst possible way, which is the following. Somebody will prove that P is equal to NP because there are only finitely many obstructions to it *not* being equal to NP. [laughter] The result would be that there is some polynomial such that we could solve all these problems in polynomial time. However, we won't know what the polynomial is, we just know that it exists. So maybe this will be n to the trillionth or something like that — but it'll be a polynomial. But we'll never be able to figure it out because it would probably take too long to find out what the polynomial is. But it does exist. Which means that the whole question $P = NP$ was the wrong question! [laughter] It might go that way. You see, even if you have something that takes 2^n steps and you compare it to something that takes n^{100}, then at least you can solve the 2^n one for n up to 20 or 30. But the n^{100} you can't even do for $n = 2$. So the degree of that polynomial is very important. There are so many algorithms out there, the task of showing that no polynomial ones exist is going to be very hard. Still, I really thought with Fermat's Theorem, it was a similar kind of thing, where it was more important to have the problem than to solve it. Therefore, my real feeling about Wiles's Theorem is that he did a marvellous wonderful piece of work, but I wish he'd solved something else! [laughter]

A lot of people think that as soon as a problem is shown to be in this class NP, they shouldn't work on it, because it means that there's probably no polynomial way to solve the problem. But before we studied NP, we had *unsolvable* problems, problems for which there didn't exist any algorithms at all — no matter how long you worked, you would never solve the problem. To tell whether a Turing machine ever stops: this is an unsolvable problem by any algorithm, no matter how long you give yourself. So, people would stop working on a problem as soon as it was proved to be unsolvable in general. But that was a bad strategy, because almost every problem we ever solve is a special case of some unsolvable problem.

Take calculus, for example. The problem of taking a formula, a function of n, and saying: "Is the limit as n goes to infinity equal to zero or not?" That's an unsolvable problem. But its unsolvability doesn't imply that we shouldn't study calculus. I mean, limits of lots of functions we've been able to decide do go to zero, and therefore people were able to develop calculus. But it's an unsolvable problem. I mean, you could define f of n — it only takes a few lines to make a formula that is equal to zero if a given Turing machine is stopped at time n, and it's equal to 1 if the Turing machine is still going at time n. And so the limit is equal to zero if and only if that Turing machine stops. It's unsolvable.

A similar thing happens with NP. That is, we have lots of special cases of problems that are NP-hard that we can solve efficiently; just knowing that something is NP doesn't mean that it's a good idea to give up on it or to stop trying to get good heuristic methods for it.

Questions about TEX?! [laughter] Yes?

Raman: One of the nice thing about TEX is that it gives the authors the flexibility to define macros that sort of encode semantics, so if you're writing a paper about permutations you can define a thing called `permute` and then write your contents using that. I rely heavily on this in my system ASTER. What I wanted to know —

DEK: Excuse me, you're going to be talking later on ... ?

Raman: Yes, I'll be talking tomorrow afternoon.

DEK: So your system uses the TEX source as part of the semantics of the presentation of the document, while the author is thinking mostly of the convenience of writing.

Raman: Yes. So, what happens, in fact, is that everything turns into an object. And so, the more semantics there are in the mark-up, the better it is. Now, in normal documents, there is this tension between wanting to write things using a base level of mark-up where an author just writes `x backslash over y` whenever he wants 'x horizontal rule y' versus an author defining, say, `\inference` as a macro that takes two things and then puts x over y. In the latter case, I win; in the earlier case I lose. My gut feeling is that, if you look at a large book, it's like a large computer program, and, sort of, in order to preserve your sanity, it seems to make more sense to read it that way.

DEK: Can people hear what the question is? He's saying that if you're blind or handicapped, you can look at the TEX source of a document and if it's properly done, it can even be better than if you had the hardcopy, because the document could have been written with a very logical mark-up scheme. In typography we try to reveal the structure by typographic means. But, in fact, we know even more of the structure when we're making our source files. So, an author with that in mind would maybe prepare the source files to have more information than just what you're going to see on the page afterwards.

When you're writing, you have an audience in mind. If you look at any book about how to write, or any course that deals with writing, the number one rule they have is: Keep your reader in mind. If an author realizes that he's writing something for hypertext, then he'll be planning his exposition so that it takes best advantage of hypertext. When I wrote my first paper in a foreign language it was published in Canada, where they spell 'color' with a 'u'. [laughter] My second paper in a foreign language was Norwegian, and so I when I was talking about variables for 'left' and 'right', I would say 'h' and 'v' instead of 'l' and 'r'. I mean, you plan as you're writing, you plan for the reader. These are very simple examples. So, if you expect that somebody is looking at your TEX source or that somebody will be able to click on part of your document and therefore it'll highlight something that is logically related, you might approach the whole process of exposition in a different way and you'll be able to reach more readers. So I try to make my ... I can show you the macro files developing for *The Art of Computer Programming* in the new style, but ...

Raman: I'd love to see that, because the other thing I'd like to do is run the system off the ...

DEK: [more screen manoeuvers; see Figure 4.] Let's just take a look at that file. It's still under construction, ok? [laughter] I input various macro files: the "`epsf`" is to get the MetaPost figures; "`rotate`" allows PostScript to do rotation; "`picmac`" is my subset of LATEX picture mode; "`unicode`" is that thing that I told you about for getting Chinese names. [goes through the macro on the screen — not transcribed] ... I type `6709` and then I say `M-X-unic` and magically this appears on my screen — that's my temporary solution to the UNICODE problem. OK, now. Let's take a look at some formatting ... [more pointing out of various codes for formatting]

These are my composition macros [Figure 5] ... some of the hyphenation exceptions I've put in; equation numbers are going into oldstyle numerals; star is for a starred section, something that's advanced; slug means at the end of a proof — I might have to change it so it's not so black, because people see so many overfull boxes, they don't like to see the black slug anymore. [laughter] I'm redefining backslash dot — I make it a colon so I can use `\.` for typewriter type in the middle of math mode. There's an important notation due to Iverson, where you can put any formula in square brackets and that evaluates the formula to zero or 1 — you can use that in the middle of equations, it's very useful. Here are macros for saying A.D. or B.C. Here's something for special emphasis I use, like a hint — I'm not using this to indicate italics, I say `\it` for that kind of emphasis. My `\em#1:` is a special format that I often use for a hint or a note or something like that [walking through the macros, with comments on some of the more interesting ones] ... These are Eulerian numbers ... with angle brackets instead of the parentheses of binomial coefficients.

Raman: Do you have a macro called `\euler` there?

DEK: Yeah, these are Eulerian numbers here, yeah. ... Capital `\Euler` is where it uses two delimiters and has the right amount of negative space between them so that you have two angle brackets next to each other; similarly, for binomial coefficients, the `\Choose` puts in two parentheses. Here's `\smallsum` for a small summation sign — I don't remember where I used it. `\phihat` — this is a special symbol — the letter ϕ has to have the hat put just right, because it's a common thing that arises when you're studying Fibonacci numbers. ... Then I have here the format for an algorithm [...]; a whole bunch of macros for typesetting assembly code; underlining text in the comments ... [remaining examples not included]

```
% Macros for The Art of Computer Programming

% (STILL UNDER CONSTRUCTION!
%  I started with manmac.tex and am letting this evolve)

\input epsf
\input rotate
\input picmac
\input unicode

\catcode`@=11 % borrow the private macros of PLAIN (with care)

\font\ninerm=cmr9
  .
  .
  .
```

Figure 4: The beginning of `acpmac.tex`

Right now, my TEX file has some structure of that form. Let's take a look at the file itself:[8] This is where I'm working day-by-day to put in my new corrections — unfortunately, we have a real narrow screen here. I have four kinds of errata: One is called an 'amendment', which is something new, that we didn't have before; one is called a 'bug', something that has to be fixed; one is called a 'plan', which is something where I'll work out the details later but I want to note down in the file that it's in my mind to make the change; and then there's something called an 'improvement', which is kind of trivial, but still I thought of it and I want to use it when we go to the final book.

For example [Figure 6], here's a quotation by Turing I kind of like from 1945, which is before computers were even invented, but he'd been thinking about it for a long time: "Up to a point, it's better to let the snags be there than to spend such time in design that there are none. How many decades would this course take?" That quotation is an amendment to Volume 1.

Look, here are the amendments for Volume 2, p. 2 [Figure 7], and I made this change on July 13; you can see these are the things I'm working on right now. Here's a new exercise I put in, here's an example where I changed a comma to a semi-colon. The way I use this is that trivial improvements don't usually get listed in the hardcopy unless you work extra hard. There's a special way of getting all these improvements to come out, but usually they only appear in the file.

[8] This macro file `acpmac.tex` is downloadable from the web page mentioned earlier.

So that's the kind of thing I'm doing. This improvement, by the way, is dated 1981. For 20 years, I've had copies of *The Art of Computer Programming* sitting in my office and I kept putting notes in the margins, marking things that I want to improve. That's now all in these files.

Other questions?

Silvio Levy: How come you don't use LATEX? [laughter]

DEK: How come I don't use LATEX? [laughter] I'm scared of large systems! [louder laughter] Bart?

Bart: Your paper, "The Errors of TEX," was great. Have you ever thought of one about "Mistakes of TEX"?

DEK: "The *Mistakes* of TEX"?? [laughter]

Bart: I mean, I guess I'm kind of thinking of the changes you made when you went to TEX3. The 7-bit/8-bit and things there that might be thought of as mistakes. Are there any other things you can think of in that line?

DEK: In that paper, I think I listed everything that I would consider a mistake. I think I would have noted that [reference to 7-bit/8-bit], but of course, I wrote that paper before TEX3 came out. I've promised to put out a sequel to that paper when everything has cooled down and you know, when the last error in TEX has been found. [laughter] So the present state is this. [File manipulations on screen bringing up the file `errorlog.tex` (which is in the CTAN archives under `system/knuth/errata`); see Figure 8.] The last change was on March 19. Well, no, that was the date the bug was reported. So, here's Peter Breitenlohner — he's here today — "Avoid spurious reference counts in format files".

```
  .
  .
  .
% Composition macros

\hyphenation{logical Mac-Mahon hyper-geo-metric hyper-geo-met-rics Ber-noulli
 Greg-ory dis-trib-uted}

{\obeyspaces\gdef {\ }}
\def\hang{\hangindent\parindent}
\def\dash---{\thinspace---\hskip.16667em\relax}
\def\eq(#1){{\rm({\oldsty#1})}}
\let\EQNO=\eqno \def\eqno(#1){\EQNO\hbox{\eq(#1)}}
\def\star{\llap{*}}
\def\slug{\hbox{\kern1.5pt\vrule width2.5pt height6pt depth1.5pt\kern1.5pt}}
\let\:=\. % preserve a way to get the dot accent
\def\.#1{\leavevmode\hbox{\tt#1}}
\def\[#1]{[\hbox{$\mskip1mu\thickmuskip=\thinmuskip#1\mskip1mu$}]} % Iverson
\def\bigi[#1]{\bigl[\begingroup\mskip1mu\thickmuskip=\thinmuskip
          #1\mskip1mu\endgroup\bigr]} % big Iverson brackets
\def\AD{{\adbcfont A.D.}}
\def\BC{{\adbcfont B.C.}}
\def\og#1{\leavevmode\vtop{\baselineskip\z@skip \lineskip-.2ex
  \lineskiplimit\z@ \ialign{##\cr\relax#1\cr
    \hidewidth\kern.3em\sh@ft{40}‘\hidewidth\cr}\kern-1ex}} % ogonek
\def\em#1:{{\it#1:\/}} % \em Hint: or \em Caution: or \em Reference: etc
   .
   .
   .
\def\euler{\atopwithdelims<>}
\def\Euler#1#2{\mathchoice{\biggl<\mskip-7mu{#1\euler#2}\mskip-7mu\biggr>}%
 {\left<\!{#1\euler#2}\!\right>}{}{}}
\def\Choose#1#2{\mathchoice{\biggl(\mskip-7mu{#1\chooser#2}\mskip-7mu\biggr)}%
 {\left(\!{#1\choose#2}\!\right)}{}{}}
\def\smsum{\mathop{\vcenter{\hbox{\tenrm\char6}}}} % small summation sign
\def\phihat{{\mkern5mu\hat{\vrule width0pt height1.2ex\smash{\mkern-5mu\phi}}}}
\def\umod{\nonscript\mskip-\medmuskip\mkern5mu% least remainder (underline mod)
  \mathbin{\underline{\rm mod}}\penalty900\mkern5mu\nonscript\mskip-\medmuskip}
   .
   .
   .
```

Figure 5: Excerpt from `acpmac.tex`

```
\amendpage 1.189 insert quotation before the exercises (95.07.13)
{\quoteformat
\vskip-3pt
Up to a point it is better to let the snags [bugs] be there
than to spend such time in design that there are none
(how many decades would this course take?).
\author A. M. TURING, Proposals for ACE (1945)
% p18, quoted in Comp J 20(1977)273
}
\endchange
```

Figure 6: One of the errata in `err1.tex`

```
\amendpage 2.2 line $-2$ (95.07.13)
digits, and in 1955 \becomes digits. The Ferranti Mark~I computer, first
installed in 1951, had a built-in instruction that put 20 random bits
into the accumulator using a resistance noise generator; this feature had
been recommended by A.~M. Turing. In 1955,
%  "resistance noise generator" -- quoted from Brooker's manual for Mark II
\endchange
```

Figure 7: One of the errata in `err2.tex`

```
  .
  .
  .
* 26 June 1993
R928\>668. Avoid potential future bug (Peter Breitenlohner). @628,637
* 17 December 1993
S929\>881. Boundary character representation shouldn't depend on font
 memory size (Berthold Horn). @549,1323
* 10 March 1994
R930. Huge font parameter number may exceed array bound (CET). @549
* 4 September 1994
F931\>926. Math kerns are explicit (Walter Carlip). @717
R932. Avoid overflow on huge real-to-integer conversion. @625,634
* 19 March 1995
R933. Avoid spurious reference counts in format files (PB). @1335
\relax
\bye
```

Figure 8: End of the file `errorlog.tex`

This was causing some problems ... he found you could break TEX if you kept saving format files and loading them again and saving them again and loading them again several hundred times; you would exceed memory capacity because the reference count could get larger than the total memory size. So that was a bug that we fixed, and he got $327.68 for that — 2^{15} pennies.

What were the other most recent changes? Number 932, "Avoid overflow on huge real-to-integer conversion." Number 931, "Math kerns are explicit" — this was a bug introduced by change 926. Number 930, "Huge font parameter number may exceed array bound," a place where the implementation wasn't totally robust. Number 929, "Boundary character representation shouldn't depend on font memory size" — this was a fairly serious one that was fixed by the major implementors shortly after we put out the previous update in 1993.

These errors — the dates listed here are actually dates when the people found them. I fixed them all in March of this year [1995]. I plan to look again at reported bugs in TEX in 1997, and again in 2000, and then 2004 and 2009, hoping that each time I'll be able to do that in about a day. There are a lot of people out there filtering these reported bugs, and vetting that they really do seem to require my attention. Yesterday, when we were running some programs at Stanford, somebody noticed that Stanford was still using a very old version of TEX and it didn't seem to matter. [laughter] I believe the bugs are starting to taper off. The remaining ones are getting to be scenarios that can cause it to break, but only if you really try hard.

There's one severe bug in the design that will have to remain as a feature, and it has to do with multilingual typesetting. I don't think I put it in the `errorlog` file, but it's noted at the end of another file called `tex82.bug` [Figure 9]. Let's look there ... this file has complete details about every change since 1982. See this one? It's the "absolutely final change to TEX, to be made after my death." The version number changes to `$\pi$`. It's like my last will and testament here. [laughter] So I'll never see that change made. That's when TEX is declared to have no more bugs. Anything that uses that name should be fully compatible with everything else that uses that name.

After that final change, this file lists "possibly good ideas that I won't implement"; ... then come two "design errors that are too late to fix." The most serious one is multilingual. If you're using several

```
-----------
415. The absolutely final change (to be made after my death)
@x module 2
@d banner=='This is TeX, Version 3.14159' {printed when \TeX\ starts}
@y
@d banner=='This is TeX, Version $\pi$' {printed when \TeX\ starts}
@z
When this change is made, the corresponding line should be changed in
Volume B, and also on page 23 of The TeXbook.
My last will and testament for TeX is that no further changes be made
under any circumstances. Improved systems should not be called simply 'TeX';
that name, unqualified, should refer only to the program for which I have
taken personal responsibility.  -- Don Knuth

* Possibly nice ideas that will not be implemented

. classes of marks analogous to classes of insertions
  .
  .
  .

* Design errors that are too late to fix

. additional parameters should be in symbol fonts to govern the space between
  rules and text in \over, \sqrt, etc,

. multilingual typesetting doesn't work properly when the \lccode changes
  within a paragraph

* Bad ideas that will not be implemented

. several people want to be able to remove arbitrary elements of lists,
   but that must never be done because some of those elements (e.g. kerns
   for accents) depend on floating point arithmetic

. if anybody wants letter spacing desperately they should put it in their own
   private version (e.g. generalize the hpack routine) and NOT call it TeX.
```

Figure 9: End of the file `tex82.bug`

languages in the same paragraph, there was some part of the state information that I forgot to save. I forget exactly what it is now,[9] but it was a serious oversight and I should have thought about it, but now it's too late. So that's a glitch that's going to have to remain.

The only other real thing I wished I'd worked harder on was the positioning of square root signs and fraction bars. I don't have enough parameters in there to control the space between the barline and the text. I made the mistake of solving a problem where I needed two parameters by using only one parameter: I got the amount of space by calculating it as a multiple of the thickness of the barline. And I should have had two parameters. Now I find that as I'm writing stuff and I have a square root that doesn't look right, I have to put a hidden strut in the exponent, to give more space there. I wish I'd done that better, but otherwise, considering the amount of inevitable compromise that has to go into any large system, I'm basically happy with the way things converged.

As I read papers typeset with TEX, the main thing that makes me unhappy, besides the way I typeset the square root sign, is the way that people have not updated to the Computer Modern fonts that I put out three or four years ago. We're still seeing the old fonts and I don't know how long it's going to take before people change. Eberhard [Mattes] did make the switch two or three months ago. It's especially evident on lowercase Greek delta.

[9] They all have to use the same `\lccode` table.

I found myself four or five years ago writing a paper and I found myself *not* using the letter delta in the paper. I tried to analyse, "why am I not using the letter delta?" and I realised that I subconsciously hated my letter delta; I didn't like the look of it. But in this paper I really needed the delta, it was the natural letter to use, and so I said, "OK ... I'm gonna take a day and redesign it and make a really beautiful delta." And I think I now have the best delta the human race has ever seen. [laughter] That was ages ago. Now, every time I see a paper using the old one, I cringe.

I also changed a few of the other letters, like some of the calligraphic capitals. I fixed the spacing on the barline on the 'H', and I didn't like the base of the 'T'. I see the NTG uses the old one in their logo ... but I'm hoping everyone will switch over to the real CMR fonts. The `.tfm` files have not changed. Oh, I've also made all the arrows heavier. The arrows were disappearing on xerox all over the place, and now the arrowheads are darker.[10]

Cameron Smith: Is there a date or version number we should look for to make sure ... ?

DEK: Well, if it's '93 or after, it'll be ok.

Silvio: No! Change CM to DM! It's never going to happen unless you do that.

DEK: It's happened in most places by now, but there still are pockets of people who haven't upgraded the old files. If you use `dvips`, all you have to do is delete the `.pk` files and it'll make them all for you.

Silvio: If you've got the new version of the source.

DEK: Yeah, well, the sources are all there.

Silvio: Right, but it's very hard for the public to ...

DEK: Please, figure out a way to solve this, because it's frustrating. Every time I get a letter from someone that has the old delta, I just tell them to tell their computer operators to update, and then they send me back an upgraded paper and say, yes it's ok now. [laughter] As long as people are aware of it. It's not that much of a change. If we get the word around to the distributors and the math journals, it shouldn't take too long to converge to that.

Robin Fairbairns: Can I just make a comment on that? Eberhard Mattes is possibly the author of the version with the largest user base — he has just reissued everything, and a month back he produced an entirely new set of fonts. On the mailing list there is a continual whinging about "we don't want to go to the trouble of updating our font files." I keep saying "You really *do* need to do this." But despite that, people say, "It costs computer time on our PCs."

DEK: They'll get a new PC in five years. [laughter] It'll eventually happen. I'm just hoping that it will happen sooner rather than later.

Silvio: Look, I posted a copy for people in Australia. I can tell you the top priority will not be to update because of a delta. If you issue a new version, with its own number and a new name, and if you make it obligatory, otherwise, ...

DEK: It didn't change that much to make it obligatory.

Jeremy Gibbons: If you change the name, old `.dvi` files won't work when you print them — there'll be lots of missing fonts.

DEK: But it's not that important. I mean, I'm too much of a nit-picker; I'm just telling you it does offend me, but it apparently doesn't offend those other people. [laughter]

Silvio: It offends me, too!

DEK: Oh, OK. Well, I don't want to change the names of my fonts. [pause] Nelson?

Nelson Beebe: Don, the world has changed a lot since 1978 —

DEK: Yes, in fact I put a wonderful quote from Bill Gates that said exactly the same thing at the beginning of my errata list this year ... [Figure 10].

Nelson: Assuming you were 25 years younger and were sitting down to do TEX now, with the market full of word processers and PostScript laser printers and so on — What would you do differently?

DEK: Well, as far as I know, I would still do the same thing, pretty much. So, anything that you don't like, I'd probably still put in! [laughter] It's just the way I do things <laughing>.

Cameron: There's a particular point, related to that, that I wanted to ask you about. You went to a lot of trouble to design a line-breaking paragraphing algorithm that looked over a wide range of possibilities for an optimal set of breaks. But I encountered in *The TEXbook* that computer memories being what they were, it wasn't practical to similarly accumulate several pages of material and look for optimal page breaks. And sort of related to that, there's the difficulty of communications between a line-breaking algorithm and a page-breaking algorithm, where, let's say you're doing a letter and

[10] Knuth has now put a discussion of this issue onto one of his Internet web pages: see `http://www-cs-faculty.stanford.edu/~knuth/cm.html` –Ch.

```
\vfill
{\quoteformat
Things have changed in the past two decades.
\author BILL GATES (1995)
%% CTdbl quote marks
% ~You, too, can start a software firm"
% International Herald Tribune 5 Jan 1995, pages 9 and 11
\bigskip
In addition to the errors listed here,
about half of the occurrences of ‘which’ in volumes one and three
should be changed to ‘that’.
\author DONALD E. KNUTH ({\sl The Art of Computer Programming Errara %
                              et Addenda}, 1981)
\eject
}
```

Figure 10: Another excerpt from `err1.tex`

there's a letterhead on the first page that forces you to have a different page width and you might need to have line breaks change in the middle of paragraphs. Things like that that could have been simplified if there were the ability to defer the cut-off of the page and have better communications between the line-breaking and page-breaking algorithms. Would you redo something like that, now that we have multi-megabyte memory?

DEK: OK, certainly the memory constraints are quite different now. Amazing how much — memory has changed more than anything else. There are also major changes in the way we — well, we've got many more years of experience. We understand these things now. At the time when I was working on TEX — I'm trying to put things in context — many things were experimental, so that we could learn about the territory. In any system design, whenever you go through a new generation, it turns out that you understand the previous generation and you clean up the previous generation, and then you also go into your new experiments, which have to be cleaned up by the next generation. That kind of traditional growth of understanding is the way the world works.

Now a lot of these things about what kind of communication would be useful and so on are becoming clearer. The idea of TEX was — and I think will remain for as long as it survives — to find the smallest number of primitives that would be able to handle the most important things. So that 99% of the work would be done by these primitives, and the other part would be done by tinkering. My attitude on these things is that when I have a job to get done, I don't ever expect to have a system that's going to do 100% of it for me, but I expect it'll do so much of it so that the tinkering is down to noise level. It's no more than a small percentage of the time I've put into the other parts of the job. Noise level for one person is different from another.

For example, I did this book about the Bible[11] where I had a lot of illustrations. I spent 6 or 7 hours on each illustration, preparing it, including color separations, fat-bits editing to clean up joins and various things, some scanning. For me, that was noise level, because I had already spent 40 hours of work writing the chapter that goes with the illustrations — so what's another 6 hours, I mean, it's a small percentage of the job, in some ways.

But if I'm going through a commercial place that's trying to get graphics in and out the door, somebody's paying for their work, and if I have 60 illustrations taking 6 hours each, that's a completely different game. So, my view is that different users will have a different idea as to how much really has to be automated. Many things are relatively easy for an author to spend a little extra time doing, because he's already put much more time into writing the book. But people who work with the author might want them to be automated and part of a fancy system the author doesn't want to take time to learn.

So, if I have a typographic task where I need to do something in 2 or 3 passes, well, I'll just try it a couple of times and run it through the machine and look through the previewer and get it right. A week or two ago, we put out a newsletter for our family. My wife does this every year. We have four grandparents and each of them has an extended family of,

[11] Donald E. Knuth, *3:16 Bible Texts Illuminated* (Madison, Wisconsin: A-R Editions, 1990); reviewed in *TUGboat* 12, #2 (1991), 233–35.

I don't know, about 60 or 80 names. We write to them and say, "Would you like to send us your comments on this year?", and then we collect them all together in a little booklet and send it out to four groups. So I fiddle with that for, I don't know, 6 to 8 hours just to do neat things; like a newsletter editor. ... You spend some time doing all kinds of prettying up, if you have the time to put in. The tools you have change the expectations of what you try to accomplish.

So, I figure the next generation of systems will have a lot more complicated mechanisms to handle the general cases of everything, where I've considered only the cases that I thought were the 99%. There are all kinds of complexities — maybe you want to have the reference point in the middle of a character instead of at the left edge, as you're going left and right, in color and rotated and in many columns, and so on. We now have more understanding of how to design such general mechanisms.

With respect to the memory situation ... I think the page-breaking business is still ... it's not so much memory-bound as maybe — you still want to do 2 passes, but the machines are fast enough for two passes — it isn't that slow anymore. So, I would say this kind of next generation thing is natural for other people to work on, with the understanding that they gained from the first system.

Pierre?

Pierre MacKay: Just going back to the question of upgrading the fonts. I was thinking about something that wouldn't break things. You stick a METAFONT `\special` to identify the font version. Since people upgrade their drivers far more often than they upgrade their fonts, just have the driver recognize that `\special` and say "Tut tut! You shouldn't be using this font!" [laughter]

Fred Bartlett: I think that everything the gentleman over here wants to do with line-breaking and page-breaking could be done fairly simply by writing moderately complex macros and new output routines, *if* there were a way to save the items that get discarded at the end of every page. You want to save the discardable items that get tossed out when TEX calls the output routine. I was wondering why, when you wrote TEX, you threw those away without making it possible to save them at all?

DEK: The output routine can put it into a box — copy it into a box —

Fred: — but it can't save the skip that is thrown out.

DEK: The skip that's thrown out, isn't that the value of one of the parameters that gets passed to the output routine?

Peter Breitenlohner: The penalty is passed, but the skip is thrown out after the page has been printed. And if nothing comes back, it is not thrown out.

DEK: Somewhere in *The TEXbook* it gives a null output routine that's supposed to put everything back together again? And what — that doesn't work?

Peter: It works! Because then the skip is not at the top.

Fred: But you can't ship out a page, you can't save a page to a box and then go back, accumulate a new page and then push it through and save that to a second box, as for left and right-column setting, and then put the two boxes together and have them join smoothly, because then the skip in between will be missing, and you'll have ... you won't have the line skip in between, and it means that if you want to do complex 2-column setting, as for a couple of text books [I've done], then you have a problem with tables and figures and a whole bunch of other junk. It would be nice ... it's something I've wrestled with for, I don't know, 6 or 7 years, and the best I've been able to do is to have TEX warn me when it starts balancing columns, starting on the right-hand column, and because I probably don't have the right skip here. Almost everything else in TEX is parameterized: you can get to it, you can save it, you can inspect it, have TEX do tests — except the discardable items, and discardable skips. I'm just wondering why.

DEK: I guess I didn't think of it. [laughter] The output routine was the most experimental part of TEX. We had no models to go by at all. We had 4 or 5 problems that we knew we had to solve, and we tried to find the smallest number of primitives that would handle those 4 or 5 problems. We got to the point where we could clarify the solutions to the problems. But we *knew* this was experimental. I'm sorry that your problem didn't occur before TEX 3.0 because then I might have been tempted ... [laughter]

Fred: I heard you say you expected more people to extend TEX than have done so.

DEK: Yeah, absolutely. I expected extensions whenever someone had a special-purpose important project, like the *Encyclopedia Britannica*, or making an Arabic–Chinese dictionary, or whatever — a large project. I never expected that one tool would

be able to handle everybody's exotic projects. So I built a lot of hooks into the code so that it should be fairly easy for a computer science graduate to set up a new program for special occasions in a week or so. That was my thought. But I don't think people have done that very much.

It's certainly what I would have done! If I were putting out a Bible or something, if I were a publisher with some project that I wanted to do specially well, then I would want a special typesetting tool for it. Rewriting a typesetting system is fairly easy. [laughter]

I guess people haven't done it because they're afraid they'd break something. I don't think they would have. I think the caution is misplaced. So I tried to show how to do it, by implementing several of the features of TEX as if they *were* added on after, just to show how to use the hooks, as a demo. But that didn't get things going. So, many more people are working with TEX at the macro level. Of course, the big advantage is that then you can share your output with others — you can assume it's going to work on everybody else's systems. But still, I thought special projects would lead to a lot of custom versions of the program. That hasn't happened.

Jeremy: A related question is ... if you can take a vertical box apart into its components, and play with them and reassemble it, one thing you *can't* get is, if you have a box that has been moved left or moved right, when you disassemble the box, you lose that information. [**DEK:** Oh really?] You can get `\lastbox`, and that gives you the box, but it doesn't tell you whether it was shifted. I thought I saw in your list of bad ideas that weren't going to have anything done with them, something to do with taking lists apart. Was that there?

DEK: I'm not sure what that really referred to any more. There is some problem about making sure that no user can access the results of rounding errors, which are different on different machines. So I had to be very careful in TEX to keep it portable — any time you do a glue calculation. Still, I don't think that would happen in shift-left shift-right. One of the changes to TEX not so many years ago was to ... I don't remember. Maybe somebody can ... maybe Peter [Breitenlohner] can recall. I think it was you who suggested it: There was something in the *hlistout* and *vlistout* routines where it looked at the shift amount of the box and ... ?

Peter: It was in leaders. The leaderbox looked for the shift amount, but the shift amount in the data structure was always zero ... TEX took the shift amount, and added it and subtracted it back, or something like that.

DEK: Yeah, so the thing is, I had some code in there that — it wasn't a bug because it could never cause any harm — but I was always adding zero to something. We took it out, so that people wouldn't be confused by it. The amount by which a box is shifted is stored with the box. If it's in a vertical list, that means so much is shifted towards the right and in a horizontal list, it means how much it's shifted up and down. That could never be nonzero in a leaders box. OK, if the shift amount is not restored properly, it might be a bug, something you could report, and in 1997, maybe you can get big money for it. [laughter]

Jeremy: I don't think it's a bug, it's just a problem, something you can't do — taking things apart and reassembling them ... It wasn't a design decision ... a deliberate one?

DEK: Alright, well, the number of possible things like that was too huge to anticipate, so I just am glad that there weren't more, I guess. I'm sorry.

Cameron: A lot of the questions have been of the form "Why did you do X?" but I think maybe part of what the thrust really is, is: If you were doing it again, as some people are trying to do, and as you've suggested that more people *should* be doing, reading from TEX for special needs, are there things that you'd recommend to *those* people? Not so much why did you do it this way 20 years ago, but if someone else were doing it again what would you tell them is most important to think about?

DEK: I just recommend putting extreme care into the design and checking things out and getting a wide number of users to help you with it, to show you the problems that they have and look at as many examples as possible. These are the things people are of course doing already. Dotting the I's and crossing the T's is the name of the game; you have to work very hard over a long period of time. The more you open or extend your vision as to how much you're going to solve, the longer it's going to take to get the whole package to be consistent.

The hardest struggle is the struggle towards convergence, instead of divergence. You need input from a tremendous variety of sources, but you also have to avoid the committee ... you have to have some small number of people who makes the decisions, in some way, so that it converges. Otherwise, you get the big problem of all committee projects — that everybody on the committee has to be proud of one part of the final thing. Then you have lots of

incompatible stuff in there, mostly for political reasons. The hard thing is to do the detailed checking on as many things as you can for consistency and convergence. The steering problem is the hardest thing with TEX.

If you study the paper, "The Errors of TEX," you'll see how this worked.[12] First there was one user — and I took a lot of time to satisfy myself. Then I had 10 users, and a whole new level of difficulties arose. Then I had a hundred users and another level of things happened. I had a thousand users, I had ten thousand — each of those were special phases in the development, important. I couldn't have gone with ten thousand until I'd done it with a thousand. But each time a new wave of changes came along, the idea was to have TEX get better, and not get more diverse as it needed to handle new things. So, when I said I'd still do things pretty much the same way, what I meant is I still think I would have horizontal lists and vertical lists; I still would have boxes and glue, and so on. That basic structure seems to give a lot of mileage from a the small number of concepts, to handle a tremendous variety of typesetting challenges. But I wasn't talking about whether I would do exactly the same with respect to a shift amount here and there. All those fine points are extremely important, but I'd still keep the same basic architecture.

Barbara: It's getting on to refreshment time. So, I would like to thank Don very much for taking time to answer everybody's questions. I will take the prerogative of one last question: Would you be willing to do this again in 2011?! In 16 more years?

DEK: Yeah, that sounds about right. [laughter] Thirty-two isn't quite as nice, but it should be OK. [laughter] Thank you very much. [wide applause][13]

[12] "The Errors of TEX," *Software Practice & Experience* **19** (1989), 607–785; reprinted with corrections and additional material as Chapters 10 and 11 of Knuth's book *Literate Programming*, distributed by Cambridge University Press. See also "Notes on the Errors of TEX," *TUGboat* 10, #4 (1989), 529–31, the keynote address at the 10th Anniversary Meeting, held at Stanford in July, 1989.

[13] As the keyboarding transcriber for this talk — arranged as the audience was just starting to sit down — I have to drag in some names of people who'd had a lot more foresight than I — they had brought mini-cassette recorders along just for the occasion! It was Cal Jackson who generously offered me his tape right after the talk had concluded. Several months later, it was Jeremy Gibbons who casually mentioned in e-mail that he too had taped the talk and would be quite happy to check my transcript. Many of the gaps which I had been unable to decipher were taken care of by Jeremy's super recording. And finally, I must of course thank Prof. Knuth, who graciously took time to read over the edited version to ensure accuracy. –Christina Thiele