CSTUG, Charles University, Prague, March 1996 Questions and Answers with Prof. Donald E. Knuth

Karel Horák:
[Introductory remarks in Czech, then English.]

I'm very glad to have such a happy occasion to introduce you, Professor Knuth, to our audience, who are mostly members of CSTUG, the Czech/Slovak TEX User Group, but also some academicians from Prague because this session is organized by CSTUG and the Mathematics Faculty of Charles University. We are very happy to have you here, and I would be happy, on behalf of Charles University, to give you a special medal. [wide applause]

DEK: [surprised] Thank you very much.

Prof. Ivan Netulka: Professor Knuth, dear colleagues, dear friends, ladies and gentlemen. I feel really very much honored having the opportunity to greet Professor Donald Knuth, as well as most of you here sitting in this guildhall, on behalf of the Dean of the Faculty of Mathematics and Physics of Charles University, Professor S.G. Sedwa.

As far as I know, Professor Knuth has come to Prague for the first time. Despite this fact, he has been known here, not only among *all* mathematicians, *all* computer scientists, but also many physicists, and even to people having nothing to do with our subjects. People here are fully aware of the significance of Donald Knuth's [...] treatise, *The Art of Computer Programming*. Many of us have had the opportunity to be pleased by reading the charming booklet devoted to *Surreal Numbers*. We know — and here I am going to fall [stumbled] — that Donald Knuth's favorite way to describe computer science is to say that it is the study of algorithms. We share his opinion that the study of algorithms has opened up a fertile vein of interesting new mathematical problems and that it provides a stimulus for many areas of mathematics which have been suffering from a lack of new ideas.

My personal experience — the personal experience of a mathematician — says that, for every mathematician, there exists a personality who [has] brought an extraordinarily great service to his field. Here we have a rare case where, in that statement, the order of the quantifiers may be reversed, maybe: There exists a personality who [has] brought a great service, an extraordinarily great service, to every mathematician. Here is my one-line proof: Donald Knuth — TEX.

Professor Knuth, in acknowledgement of your achievements in computer science, in mathematics, as well as in computerized typography, which has given the whole of the community an excellent tool for presenting scientific results, the Faculty of Mathematics and Physics of Charles University [has] decided that you be awarded the Faculty's Memorial Medal. I am happy to make that presentation now. [wide prolonged applause]

Figure 1: The Seal of Charles University

DEK: Well, this is a quite beautiful medal; I hope you can come and look at it. "Universitas Carolina Pragensis" — so we all speak Latin; maybe I should speak Latin today. [laughter]

I don't know much about the Czech language, but I've tried to learn some of it. On many doors this week I see the word "Sem". [laughter] And then as I came up to this lecture hall today, there were many other signs that said "TEX". [laughter] So I thought we could have an especially powerful version of TEX [writes 'SemTEX' on the blackboard; more laughter] but perhaps it's dangerous; I don't know....

This morning I have no prepared lecture, but I want to say just what you want to hear, so I want to answer your questions. This is a tradition that I maintained in California: The very last session of every class that I taught at Stanford was devoted to questions and answers. I told the students they didn't have to come to that class if they didn't want to, but if they came I would answer any question that they hoped to have answered when they signed up for the class. I actually borrowed this tradition from Professor [Richard] Feynmann at Caltech. And I decided I would do it in my classes, too; it's a wonderful idea that I recommend to all professors — to have open-ended question and answer sessions.

I've recently made some home pages on the World Wide Web that you can get via
`http://www-cs-faculty.stanford.edu/~knuth`
and there on those pages I have the answers to all frequently asked questions. But today, you can ask me the *un*frequently asked questions. [laughter] By the way, I'll tell you one more joke and then we'll get started. Do you know what the home page is of OJ — O.J. Simpson — in the United States? It's "http colon slash slash slash backslash slash escape". [laughter]

Now, please ask me questions. [pause]
Well, if there are no further questions, ... [laughter]. You may ask in Czech, and then someone will translate.

?: Maybe a question to start [with]. I learned TEX carefully, and I had a problem when someone asked me to take the integral with tilde accent. I found that maybe there isn't one with TEX because you can't specify an italic correction to boxes.

DEK: The italic correction is ... With each character there's a limited amount of information that goes in the data structure for each character, and so we have [drawing on blackboard] the height, the depth, the width, and the italic correction. But those are the only numbers that are allowed, and in mathematics mode, the italic correction is used in a different way from outside of mathematics. In mathematics mode, the italic correction is actually used for subscripts; it's the amount by which you would bring the subscript to the left — otherwise, it would typeset "P sub n" (P_n) like this: $P\ _n$.

The italic correction on the integral sign might even be another case because the large operators use the italic correction to cover the spacing between the lower limit and the upper limit. Anyway, there's only one number in there. If you want a special construction that demands many more numbers, the only way I know is to make a special macro for that. I would carry the information somewhere up in the TEX level, not in the inside, not with the character. You would have to build a structure that has this information in it. I don't know how general a solution you need, but certainly if you said the ... I can't even remember the name now ... my goodness, how do you get the ... like the same mechanism by which someone would take an equal sign and then put something over it, like this. It's defined in plain TEX by a macro ...

?: It's something like `\mathord` and upper limits ... [he means `\buildrel`]

DEK: I would build it up out of the primitives, but if you had different integral signs, you would probably have to allow the person who specified the font to ...

?: I have a solution, but it is not a TEX solution: I used METAFONT to produce special characters, which have the [...]

DEK: Yes, using METAFONT would be the ideal way to get the correct artistic effect, but then everyone else has to get your METAFONT code and compile your font. Just by a combination of boxes and glue, you should be able to position the characters that you have. You could just make a `\vbox` [drawing on blackboard] or a `\vcenter` of something or other, and then you build the `\hbox` of ... with a kern and then a tilde or so on. Otherwise, I don't know any simple way of doing exactly that balancing because it's complicated by the visual proportions of the spacing with integral signs — it gets really complicated to handle *all* cases.

My general philosophy with TEX was to try to have a system that covers 99% of all cases easily [laughter]; and I knew there would always be a residual number. But I felt that this residual would only be needed by the people who really care about their papers, and then if they're only spending 1% of the time on this, then they would enjoy feeling that they had contributed something special by adding their little signature, their special character to it. So, I didn't try to do everything automatically. I still believe that it's worthwhile thinking about how to do more automatically, but I don't believe you ever get all the way there.

Karel Horák: I would be very interested in your way of thinking — when you started thinking about making TEX and the typesetting system — when you realized that you also needed to produce some letters, to have not only TEX but also METAFONT. Because — I don't know too much about all types [typefaces] of digital typography — but I think there weren't very many types which you *could* use with TEX, so probably you started thinking about METAFONT, about something like that, from the first?

DEK: Exactly.

I have to erase this beautiful calligraphy [laughter]. It's too late now; well, whoever did it can do it again later, but I need the board. It's gorgeous, although this should really be a different "A". [laughter]

Let's go back to April 1977. [writes on board] I sat down at a computer terminal and started writing a memorandum to myself about what I thought would be a good language for typesetting. And in May 1977, I began working on fonts. This was going to be my sabbatical year, where I would do no teaching through the end of 1977, and the beginning of 1978. I thought that I would write a typesetting system just for myself and my secretary. [laughter] I had no idea that I would ever be seeing TEX on, for example, the tram signs in Brno [laughter] or by the churches of the city, and so on. It was just for my own purposes, and I had one year to do it. And I thought it would be easy. So, in May of 1977, I went to Xerox PARC, the place where the ideas of mouses and windows and interfaces and so on were being worked on, and I knew that they were playing with splines for letterforms. I saw Butler Lampson at a computer terminal, and he was adjusting splines around the edges of letters that he had magnified; so I thought, "good, I'll make an arrangement to work at Xerox PARC during my sabbatical year, and use their cameras and make the type."

I knew from the beginning that I wanted the type to be captured in a purely mathematical form; I wanted to have something that would adapt to technology as it kept changing, so that I would have a permanent mathematical description of the letters. Unfortunately, Xerox said, "Yes, you're welcome to use our equipment, but then we will own the designs, they will be the property of Xerox." I didn't want any of this work to be proprietary; I didn't want people to have to pay to use it. ... A mathematical formula is just numbers — why shouldn't everybody own these numbers?

So instead, I worked only at Stanford, at the Artificial Intelligence Laboratory, with the very primitive equipment there. We did have television cameras, and my publisher, Addison-Wesley, was very helpful — they sent me the original press-printed proofs of my book, from which *The Art of Computer Programming* had been made. The process in the 60s that I wanted to emulate was interesting: They would first print with metal type, Monotype, onto good paper, one copy. They made one copy with the metal, then they photographed that copy and printed from the photograph. They gave me that original copy from which they had made the original photographs. So I could try putting the TV camera on that, and go from the TV camera to a computer screen to copy the letters. At that time, we could connect our display terminals to television and movies on television; people were looking at the titles of movies, and capturing the frames from the movies and then making type. They would keep waiting for more episodes of *Star Trek* or something so that we would have the whole alphabet; eventually we would get a title with the letter "x" in it. That's how we were trying to get type by means of television at the time.

I thought it would be easy, but immediately I noticed that if I turned the brightness control a very little bit, the letters would get much thicker. There was a tremendous variation, so that what I would see on my TV screen had absolutely no consistency between a letter that I did on Monday and a letter that I did on Tuesday, the following day. One letter would be fat and one letter would be thin, but it would be the same letter because the brightness sensitivity was extremely crude. This is still true now: If you look at a scanner and you change the threshold between black and white, a small change in the threshold changes the character of the letter drastically. So I couldn't use TV.

For the next attempt, my wife made photographs of the pages and then we took our projector at home and projected them down a long hallway. On the wall I would try to copy what the letters were. But at that point I realized that the people who had designed these typefaces actually had ideas in their mind when they were doing the design. There was some logic behind the letters. For example, you have the letter 'm', you have the letter 'n', you have an 'i' and an 'l', and I noticed that the 'm' was 15 units, the 'n' was 10 units, and the 'i' was 5 units. Aha! A pattern! The 'l' was 5 units, the 'f' was 5 units, the 'fi' ligature was 10 units. So, if you cut off the tops of these letters, you would see an exact rhythm of 5 units between stems. Great — there were regularities in the design! That's when it occurred to me that maybe I shouldn't just try to copy the letterforms, but I should somehow try to capture the intelligence, the logic, behind those letterforms. And then I could do my bold font with the same logic as the regular font.

The truth therefore is that in May 1977 I didn't know what to do about fonts; June 1977 is when I started to have the idea of METAFONT.

I spent the summer of 1977 in China, and I left my students in California; I told them to implement TEX while I was gone. [laughter] I thought it would be very easy; I would come home and they would have TEX working, and then I could do the fonts. But when I got back, I realized that I had given them an impossible task. They actually had gotten enough of TEX running to typeset one character on one page, and it was a heroic achievement, because my specifications were very vague. I thought the specifications were precise, but nobody understands how imprecise a specification is until they try to explain it to a computer. And write the program.

When I was not in China — in June, the first part of July, and September, October, November — I spent most of my time making fonts. And I had to, because there was no existing way to get a font that would be the same on different equipment. Plenty of good fonts existed, but they were designed specifically for each manufacturer's device. There was no font that would go to two devices. And the people at Xerox PARC — primarily John Warnock — were still developing their ideas; they eventually founded Adobe Systems about 1980 or so. Now, with the help of many great designers, they have many beautiful fonts. But that came later, about two or three years after I had an urgent need for device-independent type.

My lecture to the American Math Society was scheduled for January 1978. The transcript of the lecture that I gave, the Gibbs Lecture to the Society, shows the work that I did with fonts in 1977. It was a much longer task than I ever believed possible. I thought it would be simple to make something that looked good — it was maybe six years before I had anything that I really was satisfied with.

So, the first big ideas were to get fonts that would be machine independent and work on many different computers, including future ones that had not been invented, by having everything defined in mathematics. The second idea was to try to record the intelligence of the design. I was not simply copying a shape, I also would specify that if part of the shape changes, the other should change in a logical way. My goal was to understand the designer's intention, and not just copy the outcome of the intention.

Well, I didn't have TEX running until May of 1978 — I didn't have TEX — I drew the fonts first. For the article, "Mathematical Typography", my talk to the American Math Society,[1] I made individual letters about 4cm high and I pasted each one on a big sheet of paper and took a photograph of that.

That's a long answer. I hope I answered the question.

Karel Horák: I have another question about this system; it is, when you started to learn typography, you had some knowledge before, or you started in the process, learning more and more? Because my experience with *The TEXbook*, and [that of] others also, is that there is very much about typography. You can learn a lot about typography, much more than some people who are doing typesetting on the professional level, using those windows mouse systems. They never can learn from the books which are supplied with those systems.

DEK: Thank you. So, what was my background before 1977? When I was in secondary school — like gymnasium — I had a part-time job setting type (so-called) on what was known as a mimeograph machine. I'm not sure what would be the equivalent here. On a mimeograph you had a sort of blue gelatinous material. The typewriter typed into it and it made a hole. I would also use a light table, and special pens, and try to make music or designs on the mimeograph stencil. I had a summer job where I would type, and then I would use my stylus to inscribe pictures on the gel. So I knew a little bit about typography. This was not fine printing, of course; it was very amateurish, but at least it gave me some idea that there was a process of printing that I could understand. After making the stencils, I would run the machine, and cut the paper, and so on. I was doing this as a student.

Later, my father had a printing press in the basement of our house, and he did work for the schools of Milwaukee; this was to save money from going to the professional places. He would work for some architects that were friends of ours, to make their specification documents. Also in the schools, there would be a program for a concert, or graduation ceremony, something like that, printing tickets for football games ... he would do this in our basement. He started with a mimeograph machine, then he upgraded to something called a VariTyper, which was marvelous, because it had proportional spacing — some letters were wider than others; the fonts were terrible, but we had this machine, and I learned how to use it.

Still later, I started writing books, *The Art of Computer Programming*. So, by 1977, I had been proofreading thousands of pages of galley proofs. I certainly was looking at type. And you might say I was getting ink in my blood.

But I also knew that engineers often make the mistake of not looking at the traditions of the past. They think that they'll start everything over from scratch, and I knew that that was terrible. So actually, right during April and May of 1977, when I was thinking about starting my sabbatical year of typesetting, I took a trip with the Stanford Library Associates, a group of book lovers from Stanford. We visited places in Sacramento, California, where people had special printing presses. We stopped at a typographic museum, which had a page from a Gutenberg Bible, and so on. Everywhere we went on this tour, I looked intently at all the letters that I saw. And I saw people's collections of what they felt was the finest printing.

At Stanford Library there is a wonderful collection of typographic materials donated by a man named Gunst, who spent a lifetime collecting fine printing. As soon as I got back from the library trip, I knew about the Gunst collection, so I spent May and June reading the works of Goudy and Zapf and everything I could find, back through history. First of all, it was fascinating, it was wonderful, but I also wanted to make sure that I could capture as well as possible the knowledge of past generations in computer form.

The general idea I had at that time was the following. At the beginning, when I was young, we had computers that could deal only with numbers. Then we had computers that knew about numbers *and* capital letters, uppercase letters. So this greatly increased our ability to express ourselves. Even in Volume 1 of *The Art of Computer Programming* when I designed my `MIX` computer, I never expected that computers could do lowercase letters. [laughter] The Pascal language was developed approximately 1968, 1969; Pascal originally used only uppercase letters, and parentheses, commas, digits, altogether 64 characters.

Next, in the early 1970s, we had lowercase letters as well, and computers could make documents that looked almost like a typewriter. And then along came software like the *eqn* system of UNIX, which would make documents that approached printing.

[1] *Bull. (N.S.) Amer. Math. Soc.* **1** (1979), pages 337–372; republished in *TEX and METAFONT: New Directions in Typesetting*, Bedford, MA: Digital Press, 1979.

You probably know that *troff* and the *eqn* system for mathematics were developed at Bell Labs. This was an extension of a program that began at MIT in 1959 or 1960, and it developed through a sequence of about five levels of improvement, finally to *eqn* in 1975.

So I knew that it was possible, all of a sudden, to get better and better documents from computers, looking almost like real books. When contemplating TEX I said, "Oh! Now it's time to go all the way. Let's not try to *approach* the best books, let's march all the way to the end — let's do it!" So my goal was to have a system that would make the best books that had ever been made, except, of course, when handmade additions of gold leaf and such things are added. [laughter] Why not? It was time to seek the standard for the solution to all the problems, to obtain the very best, and not just to approach better and better the real thing. That's why I read all the other works that I could, so that I would not miss any of the ideas. While reading every book I could find in the Gunst collection, to see what they could tell me about typesetting and about letterforms, I tried to say, "Well, how does that apply, how could I teach that to a computer?" Of course, I didn't succeed in everything, but I tried to find the powerful primitives that would support most of the ideas that have grown up over hundreds of years.

Now, of course, we have many more years of experience, so we can see how it is possible to go through even many more subtle refinements that I couldn't possibly have foreseen in 1980. Well, my project took more than one year, and I had more than one user at the end. The subsequent evolution is described in my paper called "The errors of TEX", and the complete story after 1978 is told in that paper.[2]

In 1980, I was fortunate to meet many of the world leaders in typography. They could teach me, could fill in many of the gaps in my knowledge. Artisans and craftsmen usually don't write down what they know. They just do it. And so you can't find everything in books; I had to learn from a different kind of people. And with respect to type, the interesting thing is that there were two levels: There was the type *designer*, who would draw, and then there was the *punchcutter*, who would cut the punches. And the type designer would sometimes write a book, but the punchcutter would not write a book. I learned about optical illusions — what our eye thinks is there is not what's really on the page. And so the punchcutter would not actually follow the drawings perfectly, but the punchcutter would distort the drawings in such a way that after the printing process was done and after you looked at the letter at the right size, what you saw was what the designer drew. But the punchcutter knew the tricks of making the right distortions.

Some of these tricks are not necessary any more on our laser printers. Some of them were only for the old kind of type. But other tricks were important, to avoid blots of ink on the page and things like that. After I had done my first work on METAFONT, I brought Richard Southall to Stanford; he had been working at Reading University with the people who essentially are the punchcutters. He gave me the extra knowledge that I needed to know. For example, when stems are supposed to look exactly the same, some of them are a little bit thinner, like the inside of a 'p' — you don't want it to be quite as thick, you want it to be a little thinner; then, after you have the rest of the letter there, the lightened stem will look like it was correct. Richard taught me that kind of requirement. I learned similar things from Matthew Carter, Hermann Zapf, Chuck Bigelow, Gerard Unger, and others.

But we had very primitive equipment in those days, so that the fonts that we could actually generate at low resolution did not look professional. They were just cheap approximations of the fine type. Stanford could not afford an expensive typesetting machine that would realize our designs at the time. Now I'm so happy that we have machines like the LaserJet 4, which make my type look the way I always wanted it to look, on an inexpensive machine.

?: Now that PostScript is becoming so widely used, do you think it is a good replacement for METAFONT — I mean, good enough? Right now, we can use TEX and PostScript ...

DEK: The question is, is PostScript a good enough replacement for METAFONT?

I believe that the available PostScript fonts are quite excellent quality, even though they don't use all of the refinements in METAFONT. They capture the artwork of top-quality designs. The multiple master fonts have only two or three parameters, while Computer Modern has more than sixty parameters; even with only two or three it's still quite good. The Myriad and Minion fonts are excellent.

I'm working now with people at Adobe, so that we can more easily substitute their multiple master fonts for the fonts of public-domain TEX documents. The goal is to make the PDF files

[2] *Software — Practice and Experience* **19** (1989), pages 607–681. Reprinted with additional material in *Literate Programming* (CSLI Publications, Stanford, 1992, and Cambridge University Press), pages 243–339.

smaller. The Acrobat system has PDF files which are much larger — they're ten times as big as dvi files, but if you didn't have to download the fonts, they would only be three times as large as the dvi files. PDF formats allow us search commands and quite good electronic documents. So I'm trying to make it easier to substitute the multiple master fonts. They still aren't quite general enough. I certainly like the quality there.

Adobe's font artists, like Carol Twombly and Robert Slimbach, are great; I was just an amateur. My designs as they now appear are good enough for me to use in my own books without embarrassment, but I wouldn't mind using the other ones. Yes, I like very much the fonts that other designers are doing.

Asking an artist to become enough of a mathematician to understand how to write a font with 60 parameters is too much. Computer scientists understand parameters, the rest of the world doesn't. Most people didn't even know the word 'parameters' until five years ago — it's still a mysterious word. To a computer person, the most natural thing when you're automating something is to try to show how you would change your program according to different specifications. But this is not a natural concept to most people. Most people like to work from a given set of specifications and then answer that design problem. They don't want to give an answer to all possible design specifications that they might be given and explain how they would vary their solution to each specification. To a computer scientist, on the other hand, it's easy to understand this kind of correspondence between variation of parameters and variation of programs.

In the back?

Láďa Lhotka: I have a problem for you. [question about structured programming]

DEK: I was talking with Tony Hoare, who was editor of a series of books for Oxford University Press. I had a discussion with him in approximately ... 1980; I'm trying to remember the exact time, maybe 1979, yes, 1979, perhaps when I visited Newcastle? I don't recall exactly the date now. He said to me that I should publish my program for TEX.[3]

As I was writing TEX I was using for the second time in my life ideas called "structured programming", which were revolutionizing the way computer programming was done in the middle 70s. I was teaching classes and I was aware that people were using structured programming, but I hadn't written a large computer program since 1971. In 1976 I wrote my first structured program; it was fairly good sized — maybe, I don't know, 50,000 lines of code, something like that. (That's another story I can tell you about sometime.) This gave me some experience with writing a program that was fairly easy to read. Then when I started writing TEX in this period (I began the implementation of TEX in October of 1977, and I finished it in May 78), it was consciously done with structured programming ideas.

Professor Hoare was looking for examples of fairly good-sized programs that people could read. Well, this was frightening. This was a very scary thing, for a professor of computer science to show someone a large program. At best, a professor might publish very small routines as examples of how to write a program. And we could polish those until ... well, every example in the literature about such programs had bugs in it. Tony Hoare was a great pioneer for proving the correctness of programs. But if you looked at the details ... I discovered from reading some of the articles, you know, I could find three bugs in a program that was proved correct. [laughter] These were *small* programs. Now, he says, take my *large* program and reveal it to the world, with all its compromises. Of course, I developed TEX so that it would try to continue a history of hundreds of years of different ideas. There had to be compromises. So I was frightened with the idea that I would actually be expected to show someone my program. But then I also realized how much need there was for examples of good-sized programs, that could be considered as reasonable models, not just small programs.

I had learned from a Belgian man (I had met him a few years earlier, someone from Liège), and he had a system — it's explained in my paper on literate programming.[4] He sent me a report, which was 150 pages long, about his system — it was inspired by "The spirit in the machine". His 150-page report was very philosophical for the first 99 pages, and on page 100 he started with an example. That example was the key to me for this idea of thinking of a program as hypertext, as we would now say it. He proposed a way of taking a complicated program and breaking it into small parts. Then, to understand the complicated whole, what you

[3] "I looked up the record when I returned home and found that my memory was gravely flawed. Hoare had heard rumors about my work and he wrote to Stanford suggesting that I keep publication in mind. I replied to his letter on 16 November 1977 — much earlier than I remembered." – D. Knuth

[4] Pierre Arnoul de Marneffe, *Holon Programming*. Univ. de Liège, Service d'Informatique (December, 1973).

needed is just to understand the small parts, and to understand the relationship between those parts and their neighbors.

In February of 1979, I developed a system called `DOC` and `UNDOC` ... something like the `WEB` system that came later. `DOC` was like `WEAVE` and `UNDOC` was like `TANGLE`, essentially. I played with `DOC` and `UNDOC` and did a mock-up with a small part of TEX. I didn't use `DOC` for my own implementation but I took the inner part called *getchar*, which is a fairly complicated part of TEX's input routine, and I converted it to `DOC`. This gave me a little 20-page program that would show the *getchar* part of TEX written in `DOC`. And I showed that to Tony Hoare and to several other people, especially Luis Trabb Pardo, and got some feedback from them on the ideas and the format.

Then we had a student at Stanford whose name was Zabala — actually he's from Spain and he has two names — but we call him Iñaki; Ignacio is his name. He took the entire TEX that I'd written in a language called *SAIL* (Stanford Artificial Intelligence Language), and he converted it to Pascal in this `DOC` format. TEX-in-Pascal was distributed around the world by 1981, I think. Then in 1982 or 1981, when I was writing TEX82, I was able to use his experience and all the feedback he had from users, and I made the system that became `WEB`. There was a period of two weeks when we were trying different names for `DOC` and `UNDOC`, and the winners were `TANGLE` and `WEAVE`. At that time, we had about 25 people in our group that would meet every Friday. And we would play around with a whole bunch of ideas and this was the reason for most of the success of TEX and METAFONT.

Another program I wrote at this time was called *Blaise*, because it was a preprocessor to Pascal. [laughter]

Petr Olšák: I have two questions.

What is your opinion of LATEX, as an extension of TEX at the macro level? I think that TEX was made for the plain TEX philosophy, which means that the user has read the *The TEXbook* ... [laughter] while LATEX is done with macros, and takes plain TEX as its base. And the second question: Why is TEX not widely implemented and used in commercial places. They use only mouse and WYSIWYG-oriented programs.

DEK: The first question was, what do I think about LATEX?

I always wanted to have many different macro packages oriented to different classes of users, and LATEX is certainly the finest example of these macro packages. There were many others in the early days. But Leslie Lamport had the greatest vision as to how to do this. There's also AMS-TEX, and the mathematicians used Max Diaz's macros — I think it might have been called MaxTEX or something — in the early days before we had LATEX. Mike Spivak and Leslie Lamport provided very important feedback to me on how I could improve TEX to support such packages. I didn't want to ... I like the idea of a macro system that can adapt to special applications. I myself don't use LATEX because I don't have time to read the manual. [laughter] LATEX has more features than I need myself, in the way I do things. Also, of course, I understand TEX well enough that it's easier for me not to use high-level constructions beyond my control.

But for many people it's a simpler system, and it automates many of the things that people feel naturally ought to be automated. For me, the things that it automates are largely things that I consider are a small percentage of my total work. It doesn't bother me that I hand tune my bibliography, but it bothers other people a lot. I can understand why a lot of people prefer their way of working.

Also, when you're writing in a system like LATEX you can more easily follow a discipline that makes it possible for other programs to find the structure of your document. If you work in plain TEX, you can be completely unstructured in your approach and you can defeat any possible process that would try to automatically extract bibliographic entries and such things from your document. If you restrict yourself to some kind of a basic structure, then other processes become possible. So that's quite valuable. It allows translation into other structures, languages and so on.

But I use TEX for so many different purposes where it would be much harder to provide canned routines. LATEX is at a higher level; it's not easy to bend it to brand-new applications. Very often I find that, for the kind of things that I want to do, I wake up in the morning and I think of a project ... or my wife comes to me and says, "Don, can you make the following for me?" So I create ten lines of TEX macros and all of a sudden I have a new language specifically for that kind of a document. A lot of my electronic documents don't look like they have any markup whatsoever.

Now, your second question, why isn't TEX used more in commercial publication? In fact, I was quite pleasantly surprised to see how many commercial publishers in the Czech Republic are using TEX. Thursday night, I saw three or four Czech-English dictionaries that were done with TEX, and you know it's being used for the new Czech encyclopedia. And Petr Sojka showed me an avant garde novel that had been typeset with TEX with some nice tricks of its own very innovative page layout. In America, it's used heavily in legal publications, and behind the scenes in lots of large projects.

I never intended to have a system that would be universal and used by everybody. I always wanted to write a system that would be used for just the finest books. [laughter] Just the ones where the people had a more difficult than ordinary task, or they wanted to go the extra mile to have excellent typography. I never expected that it would compete with systems that are for the masses.

I'm not a competitive person, in fact. It made me very happy to think that I was making a system that would be primarily for mathematics. As far as I knew, there wasn't anybody in the world who would feel offended if I made it easier to typeset mathematics. Printers considered this to be "penalty copy", and it was something that they did only grudgingly. They charged a penalty for doing this extra horrible work, to do mathematics. I never expected that I would be replacing systems that are used in a newspaper office or anything like that. It turned out that after I got going, we found we could make improvements ... in one experiment we re-typeset two pages of *Time* magazine, to show how much better it would be if they had a good line-breaking algorithm. But I never expected when I began that such magazines would ever use what I was doing because, well, it was a billion-dollar industry and I didn't want to put anyone out of work or anything.

So it was very disturbing to me in the early 80s when I found there was one man who was very unhappy that I invented TEX, because he had worked hard to develop a mathematical typesetting system that he was selling to people, and he was losing customers. So he wrote to the National Science Foundation in America, saying, "I'm a taxpayer and you're using my tax money to put me out of business." This made me very unhappy. I thought everything I was doing was for everybody's good. And here was a person I'd obviously hurt. But I also thought that I still should make TEX available to everyone, even though it had been developed with some help from the government. I don't think the government should give money only to things that are purely academic and not useful.

Yes?

?: I have a question about the usage of your typographic programs in commercial institutions like DTP studios and so on. I'd like to ask about using parts of the TEX source. You made clear that the programmers were free to incorporate parts of the TEX source into their own programs. There are some remarkable examples of this, do you know.

DEK: That question came up also last summer when I had a question and answer session at the TUG meeting in Florida.[5] I thought it would be fairly common to have special versions of TEX. I designed TEX so that it has many hooks inside; you can write extensions and then have a much more powerful TEX system readily adapted.

I guess I was thinking that every publishing house using TEX would have an in-house programmer who would develop a special version of TEX if they wanted to do an edition of the Bible, if they wanted to do an Arabic-to-Chinese dictionary or something. If they were doing an encyclopedia, they could have their own version of TEX that would be used for this application.

A macro language is Turing-complete — it can do anything — but it's certainly silly to try to do everything in a high-level language when it's so easy to do it at the lower level. Therefore I built in hooks to TEX and I implemented parts of TEX as demonstrations of these hooks, so that a person who read the code could see how to extend TEX to other things. We anticipated certain kinds of things for chemistry or for making changebars that would be done in the machine language for special applications.

Certainly, if I were a publishing house, if I were in the publishing business myself, I would have probably had ten different versions of TEX by now for ten different complicated projects that had come in. They would all look almost the same as TEX, but no one else would have this program — they wouldn't need it, they're not doing exactly the book that my publishing house was doing.

That was what I thought would occur. And certainly, there was a point in the middle 80s when there were more than a thousand people in the world that knew the TEX program, that knew the intricacies of the TEX program quite well. They had read it, and they would have been able to make any of these extensions if they wanted. Now I would say that the number of people with a working knowledge of TEX's innards is probably less than a thousand, more than a hundred. It hasn't developed to the extent that I expected.

One of the most extensive such revisions is what I saw earlier this week in Brno — a student

[5] *TUGboat* **17**(1) (1996), pages 7–22.

whose name is Thanh,[6] I think, who has a system almost done that outputs PDF format instead of `dvi` format. If you specify a certain flag saying `\PDFon`, then the output actually comes out as a file that an Acrobat reader can immediately read. I also expected that people would go directly to PostScript; that hasn't happened yet as far as I know.

No one has done a special edition of the Bible using TEX in the way I expected. There were some extensions in Iceland; I don't remember if they did it at the higher level — I think they did it mostly at the macro level, or maybe entirely.

Anyway, I made it possible to do very complicated things. When you have a special application, I was always expecting that you would want to have a specially tuned program there because that's where it's easiest to do these powerful things.

?: I want to ask which features of TEX were in the first version — for example, line-breaking, hyphenation, and macro processing — if all these things were in the first version?

DEK: The very first version was designed in April 1977. I did have macros and the algorithm for line-breaking. It wasn't as well developed; I didn't have all the bells and whistles like `\parshape` at that time, but from the very beginning, from 1977 on, I knew I would treat the paragraph as a whole, not just line by line. The hyphenation algorithm I had in those days was not the one that we use now; it was based on removing prefixes and suffixes — it was a very peculiar method, but it seemed to catch about 80% of the hyphens. I worked on that just by looking at the dictionary: I would say, the word starts with "anti", then put a hyphen after the "i"; and similarly at the end of the word. Or if you have a certain combination of letters in between, in the middle, there were natural breaks. I liked this better than the *troff* method, which had been published. The hyphenation algorithm is described in the old TEX manual, which you can find in libraries.[7]

Now, you said the line-breaking, hyphenation, macros, ... I developed the macro language in the following way. I took a look at Volume 2 of *The Art of Computer Programming* and I chose representative parts of it. I made a mock-up of about five pages of that book, and said, "How would I like that to look in a computer file?" And that was the whole source of the design.

I stayed up late one night and created TEX. I went through Volume 2 and fantasized about natural-looking instructions — here I'll say "backslash algorithm", and then I'll say "algorithm i", and then I'll say "algstep", you know. This gave me a little file that represented the way I wanted the input to look for *The Art of Computer Programming*. The file also included some mathematical formulas. It was based on the idea of *eqn*; the *troff* language had shown me a way to represent mathematics that secretaries could learn easily. And that was the design. Then I had to implement a macro language to support those features.

The macro language developed during 1978, primarily with the influence of Terry Winograd. Terry was writing a book on linguistics, a book on English grammar. He wanted to push macros much harder than I did, and so I added `\xdef` and fancier parameters for him.

The hyphenation algorithm we have now was Frank Liang's Ph.D. research. He worked with me on the original hyphenation method, and his experience led him to discover a much better way, which can adapt to all languages — I mean, to all western languages, which are the ones that use hyphens.

As far as the spacing in mathematics is concerned, I chose three standards of excellence of mathematical typesetting. One was Addison-Wesley books, in particular *The Art of Computer Programming*. The people at Addison-Wesley, especially Hans Wolf, their main compositor, had developed a style that I had always liked best in my textbooks in college. Secondly, I took *Acta Mathematica*, from 1910 approximately; this was a journal in Sweden ... Mittag-Leffler was the editor, and his wife was very rich, and they had the highest budget for making quality mathematics printing. So the typography was especially good in *Acta Mathematica.* And the third source was a copy of *Indagationes*, the Dutch journal. There's a long fine tradition of quality printing in the Netherlands, and I selected an issue from 1950 or thereabouts, where again I thought that the mathematics was particularly well done.

I took these three sources of excellence and I looked at all the mathematics formulas closely. I measured them, using the TV cameras at Stanford, to find out how far they dropped the subscripts and raised the superscripts, what styles of type they used, how they balanced fractions, and everything. I made detailed measurements, and I asked myself, "What is the smallest number of rules that I need to do what they were doing?" I learned that I could

[6] Han The Thanh; see Petr Sojka, Han The Thanh and Jiří Zlatuška, "The Joy of TEX2PDF — Acrobatics with an alternative to DVI format", *TUGboat* **17**(2) (1996).

[7] *TEX and METAFONT: New Directions in Typesetting* (cited in footnote 1).

boil it down into a recursive construction that uses only seven types of objects in the formulas.

I'm glad to say that three years ago, *Acta Mathematica* adopted TEX. And so the circle has closed. Addison-Wesley has certainly adopted TEX, and I'm not sure about the Dutch yet — I'm going to visit them next week. [laughter] But anyway, I hope to continue the good old traditions of quality.

I have to call on people who haven't spoken. George...

Jiří Veselý: I have a question. You are asked every time carefully regarding all suggestions and things like that for improvements. Once I was asked about the possibility to make a list of all hyphenated words in the book. I was not able to find in your book a way to do this. I would like to know something about your philosophy what to include and what not to include. What would be in that special package, and what would be in TEX?

DEK: The question is, what is the philosophy that I use to try to say what should be a basic part of TEX and what should be harder to do or special, or something like that. Of course, these decisions are all arbitrary. I think it was important, though, that the decisions were all made by one person, even though I'm not ... I certainly make a lot of mistakes. I tried the best to get input from many sources, but finally I take central responsibility to keep some unity. Whenever you have a committee of people designing a system, everyone in the committee has to feel proud that they have contributed something to the final language. But then you have a much less unified result because it reflects certain things that were there to please each person. I wanted to please as many people as I could but keep unity. So for many years we had a weekly meeting for about two hours every Friday noon, and we had visitors from all over the world who would drop in. We would hear their comments and then we would try to incorporate the ideas that we heard during that time.

Now you ask specifically about why don't we have an easy way to list all the hyphenations that were made in the document. It sounds like a very nice suggestion, which I don't recall anyone raising during those weekly meetings. The words that actually get hyphenated, the decision to do that is made during the *hpack* routine, which is part of the line-breaking algorithm. But the fact that a hyphenation is performed by *hpack* doesn't mean that it's going to appear in the final document, because you could discard the box in which this hyphenation was done.

It's very easy in TEX to typeset something several times and then choose only one of those for the actual output. So, to get a definitive representative of the hyphenation, you'd have to catch it in the output routine, where the discretionary had appeared. This would be easy to do now in a module specially written for TEX. I would say that right now, in fact, you could get almost exactly what you want by writing a filter that says to TEX "Turn on all of the tracing options that cause it to print, to give the page contents." Then a little filter program would take the trace information through a UNIX pipe and it would give you the hyphenated words. It would take an afternoon to write this program, maybe two afternoons ... and a morning. [laughter] You could get that now, but it was not something that I can recall I ever debated whether or not I should do at the time we were having these weekly discussions on TEX.

My paper on "The errors of TEX" has the complete record of all the changes that were made since 1979, with dates, and with references to the code, exactly where each change appears. And so you can see the way the evolution was taking place. Often the changes would occur as I was writing *The TEXbook* and realizing that some things were very hard for me to explain. I would change the language so it would be easier to explain how to use it. This was when we were having our most extensive meeting with users and other people in the group as sources of ideas; the part of the language I was writing about was the part that was changing at the moment.

During 1978, I myself was typesetting Volume 2, and this led naturally to improvements as I was doing the keyboarding. In fact, improvements occurred almost at a steady rate for about 500 pages: Every four pages I would get another idea how to make TEX a little better. But the number of ways to improve any complicated system is endless, and it's axiomatic that you never have a system that cannot be improved. So finally, I knew that the best thing I could do would be to make no more improvements — this would be better than a system that was improving all the time.

In fact, let me explain. As I was first developing TEX at the Stanford Artificial Intelligence Laboratory, we had an operating system called WAITS, which I think is the best that the world has ever seen. Four system programmers were working full time making improvements to this operating system. And every day that operating system was getting better and better. And every day it was breaking down and unusable.

In fact I wrote the first draft of *The TEXbook* entirely during downtime. I would take my tablet of paper to the Artificial Intelligence Laboratory in the morning and I would compute as long as I could. Then the machine would crash, and I would write another chapter. Then the machine would come up and I could type a little bit and get a little more done. Then, another hang up; time to write another chapter. Our operating system was always getting better, but I couldn't get much computing done.

Then the money ran out; three of the programmers went to Lawrence Livermore Laboratory and worked on a new operating system there. We had only one man left to maintain the system, not to make any more improvements. And it was wonderful! [laughter] That year, I could be about as productive as anyone in the world.

So I knew that eventually I would have to get to the point where TEX would not anymore improve. It would be steady and reliable, and people would understand the warts it had ... the things that it doesn't do.

I still believe it's best to have a system that is not a moving target. After a certain point, we need something that is stable, not changing at all. Of course, if there's some catastrophic scenario that we don't want ever to happen, I still change TEX to avoid potential disasters. But I don't put in nice ideas any more.

Of course, there are other people working on extensions to TEX that will be useful for another generation. And they will also be well advised at a certain point to say "Now we will stop, and not change our system any more." Then there will be a chance for another group later.

Karel Horák: I'd like to ask about the idea of the italic font in mathematics. I never saw other textbooks that use different fonts for italics in text and in mathematics, so I'm asking if it's your own idea or if it comes also from these three sources?

DEK: That's right. I didn't find in any of the other books the idea of having a text italic and a math italic. I wanted the math italic to look as beautiful as possible, and I started with that. But then I found that the text italic was not as good, so I had METAFONT and it was easy to get text italic that would look better. If I made the text italic good, then the math would not position the subscripts and the superscripts as well.

It's partly because of what I explained before — TEX has only four numbers to go with every character. Printers, in fact, in the old days, had only three numbers; they didn't have the italic correction. So they couldn't do even that much automatically; the better printers adjusted mathematical spacing by hand. But italic now, the italic fonts of today by all the font designers are much better than they used to be. We've seen a great improvement in italic typography during the last ten, fifteen years. In fact, if you read older books you'll sometimes say, "How could anybody read this italic?", or "Why did they accept such peculiar spacing?", but it was based on the constraints of metal type. The whole idea of italic correction was not in any other book, but it was necessary for me to get the spacing that I wanted.

When I showed type designers mathematical formulas, they could never understand why mathematicians want italic type in their formulas. It seems you're combining a roman 2 with an italic x. And they said, "Wouldn't the positioning be so much simpler if you had a regular, non-sloped font in mathematics?" And I think it was Jan van Krimpen, who worked with a Nobel Prize physicist in the Netherlands, in Haarlem — what was his name?[8] I think he was the second person to receive the Nobel Prize in physics; he died in the 20s — anyway he and van Krimpen were going to develop a new font for mathematics in the Netherlands, and it wasn't going to have italics for mathematics. It was going to be unified between the Greek letters and other symbols that mathematicians wanted. But the project stopped because the physicist died; van Krimpen finished only the Greek, which became fairly well used.

Several other font designers have visited Stanford. When they looked at mathematics, they said, "Well, why don't you use a non-sloping font?" Hermann Zapf made a proposal to the American Mathematical Society that we would create a new typeface for mathematics which would include the Fraktur alphabet, and Greek, and script, and special characters, as well as ordinary letters. One key idea was that it would not have sloped characters, so that x would be somehow straight up and down. Then it should be easier to do the positioning, the balancing. Hermann created a series of designs, and we had a large committee of mathematicians studying the designs and commenting on them and tuning them.

This font, however, proved to be too radical a change for mathematicians. I've seen mathematicians actually writing their documents where they will write an x slanted twice as much — I mean, they make it look very italic; it looks like a mathematical

[8] It was H.A. Lorentz. See John Dreyfus, *The Work of Jan van Krimpen* (London: Sylvan Press, Museum House, 1952), page 28.

letter to them. So after 300 years of seeing italic math in print, it's something that we feel is right. There are maybe two dozen books printed, well, maybe more, maybe a hundred, printed with the AMS Euler font, but most mathematicians think it's too different.

I find now that the Euler Fraktur font is used by almost everyone. In Brno, I saw Euler Roman used as a text font for a beautiful book, a translation of Durer's *Apocalyse* in Czech. I also saw it a few days ago in some class notes. Once, when I was in Norway, I noticed that everyone's workstation was labeled with the workstation's name in AMS Euler, because people liked it. It's a beautiful font, but it hasn't been used as the typeface for mathematics in a large number of books.

Karel Horák: If there are no other questions, I would thank Professor Knuth very much for this session. [wide prolonged applause]

DEK: Thank you all for excellent questions.

Karel Horák: [Closing comments in Czech.]