THE COMPUTER SCIENCE OF TEX AND LATEX; COMPUTER SCIENCE COURSE 594, FALL 2004

VICTOR EIJKHOUT
DEPARTMENT OF COMPUTER SCIENCE,
UNIVERSITY OF TENNESSEE, KNOXVILLE TN 37996

CS 594 – Practical Matters.

aim of this course This course will teach you the fundamentals of TEX and LATEX use. However, you will pick up more than just a few practical skills. We will see all sorts of mathematical and computer science topics that are (sometimes maybe only marginally) related to TEX and LATEX. So, TEX/LATEX is here really just an excuse for teaching cool cs subjects.

homework and exams There will be regular homework and occasional pop quizzes. Homework needs to be done in LATEX: mail both source and pdf output to `cs594tex`.

final project The final exam will be in the form of a project that can be done individually or with 2 or 3 people. You can find the project descriptions at the end of each chapter. If you have an idea for a project, feel free to talk to me about it.

course materials The main handout can be bought from Graphic Creations (1809 Lake Avenue, 865-522-6221). It is very much recommended that you buy the *Guide To LATEX* (fourth edition) by Kopka and Daly. (Order from Amazon, Ecampus, Bookpool, et cetera; the university bookstore has a small number of copies.) Certain reference books have been put on reserve in the library (2nd floor).

lecture notes Further material can be downloaded from `http://www.cs.utk.edu/~eijkhout/594-LaTeX`. Files with `tutorial` in the name are written out lecture notes; you can download and print them. Files with `slides` in the name are slides; there should be no need for printing them.

classes We meet every tuesday and thursday 2:10–3:25pm. No class on sep 2, oct 14 (fall break), nov 25 (thanksgiving).

Victor Eijkhout
eijkhout@cs.utk.edu
329 Claxton, daily 11–12 and by appointment

Lan Lin
lin@cs.utk.edu
110E Claxton, mon/tue/wed 3:30–5:30

Enjoy!

Contents

Chapter 1

TEX and LATEX

In this chapter we will learn

- The use of LATEX for document preparation,
- LATEX style file programming,
- TEX programming.

Handouts and further reading for this chapter

For LATEX use the ‘Not so short introduction to LATEX’ by Oetiker *et al.* For further reading and future reference, it is highly recommended that you get ‘Guide to LATEX’ by Kopka and Daly [11]. The original reference is the book by Lamport [12]. While it is a fine book, it has not kept up with developments around LATEX, such as contributed graphics and other packages. A book that does discuss extensions to LATEX in great detail is the ‘LATEX Companion’ by Mittelbach *et al.* [13].

For the TEX system itself, consult ‘TEX by Topic’. The original reference is the book by Knuth [8], and the ultimate reference is the published source [7].

Projects for this chapter.

Project 1.1. TEX has a syntax that can to a large extent be altered, dynamically or statically. This has had the effect that macro packages typically use a syntax that is somewhere between ad hoc and plain unsystematic. Explore how it would be possible to set up a macro package with object-oriented syntax and design. It would probably be a good idea to read [1, 2, 3], and to look at macro package such as Lollipop and ConTeXt.

Project 1.2. The web site `http://wwww.cookingforengineers.com` uses a table-like layout for notating recipes. Design an easy syntax for inputting these diagrams, and write a LATEX package that implements them.

Chapter 2

Parsing

The programming language part of TeX is rather unusual. In this chapter we will learn the basics of language theory and parsing, and apply this to parsing TeX and LaTeX. Although TeXcan not be treated like other programming languages, it is interesting to see how far we can get with existing tools.

Handouts and further reading for this chapter

The theory of languages and automata is discussed in any number of books, such as the Hopcroft and Ulman one. For a discussion that is more specific to compilers, see the compilers book by Aho and Ulman or Aho, Seti, and Ulman.

The tutorials on *lex* and *yacc* should suffice you for most applications. The O'Reilly book by Levine, Mason, and Brown is probably the best reference on *lex* and *yacc*. A copy of it is on reserve in the library, *QA76.76.U84M37*.

The definitive reference on hashing is Knuth's volume 3 of The Art of Computer Programming [10], section 6.4. This is on reserve, *QA76.5.K57*.

Projects for this chapter.

Project 2.1. Use the *lex* and *yacc* programs you have written for LaTeX to write a full LaTeX-to-HTML translator.

Project 2.2. A number of projects involve parsers or code generators for (parts of) TeX or LaTeX.

formulas Reinhold Heckmann and Reinhard Wilhelm. 1997. A functional description of TeX's formula layout. Journal of Functional Programming 7(5):451-485. Available online at `http://rw4.cs.uni-sb.de/users/heckmann/doc.html`. For software, see `ftp://ftp.cs.uni-sb.de/formulae/`.

also formulas Preview-LaTeX (`http://preview-latex.sourceforge.net/`) displays formulas right in the emacs edit buffer.

math on web pages see `http://www.forkosh.com/mimetex.html`.

LaTeX to HTML HeVeA, TtH, TeX4ht and LaTeX2HTML.

front end for LaTeX `http://www.lyx.org/` Ages ago there was 'griff'. Scientific Word still exists but is commercial.

reimplementation of TeX TeX in Python: `http://www.pytex.org/`

Investigate these parsers: what is their basic approach and theoretical power, what are they good at, what are they bad at.

Project 2.3. Develop the full theory of the compound NFA that does lexical analysis.

- This automaton basically parses the whole file, rather than small chunks; every once in a while it will report that it has recognized an identifier, number, special symbol &c. This means that the definition of the output alphabet has to be expanded. Analyze the structure of this output language.
- As explained, the returning ϵ-transition only has to be taken if a maximal string is recognized. Formalyze this notion.
- The automaton may need look-ahead of one or more tokens. Formalize this notion and give equivalence proofs.

Project 2.4. Do an experimental study of hashing. Implement open and closed hash table algorithms, and do a search on the web for hash functions. Compare to storing the names in a tree. Find some documents to hash: the source of some programs, books. Construct some input data sets to trip up the various algorithms. Measure statistics and do timings.

Chapter 3

Breaking things into pieces

The line breaking algorithm of TEX is interesting, in that it produces an aesthetically optimal solution in very little time.

Handouts and further reading for this chapter

If you still have the book ‘Introduction to Algorithms’ by Cormen *et al.*, you can find a discussion of Dynamic Programming and NP-completeness there. The books by Bellman are the standard works in this field. Bellman’s ‘Applied Dynamic Programming’ [**?**] has been put on reserve, *QA264.B353*. The TEX line breaking algorithm is described in an article by Knuth and Plass [9], reprinted in [5].

Issues in page breaking are discussed in Plass’ thesis [14].

Projects for this chapter.

Project 3.1. What is the paragraph breaking algorithm of OpenOffice? Replace by TEX's algorithm.

Project 3.2. Write a paragraph breaking algorithm that prevents 'rivers' in the text.

Project 3.3. TEX's line breaking algorithm is not just for good looks. Many aesthetic decisions in typography actually influence readability of the document. Read 'Digital Typography' by Rubinstein [15] and find an issue to investigate.

Project 3.4. Many page layout parameters (Rubinstein [15] and the references therein) have an influence on legibility. Has typographic design deteriorated in that sense now that authors publish their own works? Do a study, using various books in the library.

Project 3.5. The following sentence

> Only the fool would take trouble to verify that this sentence was composed of ten a's, three b's, four c's, four d's, forty-six e's, sixteen f's, four g's, thirteen h's, fifteen i's, two k's, nine l's, four m's, twenty-five n's, twenty-four o's. five p's, sixteen r's, forty-one s's, thirty-seven t's, ten u's, eight v's, eight w's, four x's, eleven y's, twenty-seven commas, twenty-three apostrophes, seven hyphens and, last but not least, a single !

is called a pangram. (There are other pangrams. Google for the combination of 'pangram' and 'Lee Sallows' for this particular type.) Given a beginning of the sentence ('Only the fool. . . '), solve the rest of the sentence by dynamic programming.

Chapter 4

Fonts

Knuth wrote a font program, Metafont, to go with TeX. The font descriptions involve some interesting mathematics.

Handouts and further reading for this chapter

Bezier curves and raster graphics are standard topics in computer graphics. The book by Foley and Van Dam (section 11.2 about Splines) has been placed on reserve, *T385.C587*.

Digital typography is a very wide area, spanning from perception psychology and physiology to the electronics and mathematics of display equipment. The book by Rubinstein [15] is a good introduction. This book has been placed on reserve, *Z253.3.R8*.

The relation between Bezier curves and aesthetics is explicitly discussed in `http://www.webreference.com/dlab/9902/examples-ii.html`.

Projects for this chapter.

Project 4.1. Bezier curves can also be used for graphics problems such as enlarging a bitmat or projecting text on a waving flag. Consult `http://www.tinaja.com/cubic01.asp` and report on the techniques used.

Project 4.2. (very mathematical) Explain elliptical integrals and how they are used to compute the length of a Bezier curve. Explain approximations. Same resource as the previous project.

Project 4.3. (very mathematical) Study the theory of NURBS (Non-Uniform Rational B-Splines); what are their capabilities and limitations? Start at `http://devworld.apple.com/dev/techsupport/develop/issue25/schneider.html` for an introduction.

Project 4.4. Investigate perception issues in font design or display technology. Start by browsing through Rubinstein's book.

Chapter 5

TEX's macro language

The programming language of TEX is rather idiosyncratic. One notable feature is the difference between expanded and executed commands. The expansion mechanism is very powerful: it is in fact possible to implement lambda calculus in it.

Handouts and further reading for this chapter

The inspiration for this chapter was the article about lists by Alan Jeffrey [4].

Chapter 6

Character encoding

This chapter is about how to interpret the characters in an input file – no there ain't such a thing as a plain text file – and how the printed characters are encoded in a font.

Handouts and further reading for this chapter

There is very little printed material on this topic. A good introduction is `http://www.joelonsoftware.com/articles/Unicode.html`; after that, `http://www.cs.tut.fi/~jkorpela/chars.html` is a good tutorial for general issues, and `http://en.wikipedia.org/wiki/Unicode` for Unicode.

For the technical details on Unicode consult `http://www.unicode.org/`. An introduction to ISO 8859: `http://www.wordiq.com/definition/ISO_8859`.

Projects for this chapter.

Project 6.1. What is the problem with ‘Han unification’? (section **??**) Discuss history, philology, politics, and whatever may be appropriate.

Project 6.2. How do characters get into a file in the first place? Discuss keyboard scan codes and such. How do modifier keys work? How can an OS switch between different keyboard layouts? What do APIs for different input methods look like?

Project 6.3. Dig into history (find the archives of `alt.folklore.computers`!) and write a history of character encoding, focusing on the pre-ascii years. Describe design decisions made in various prehistoric computer architectures. Discuss.

Chapter 7

Software engineering

In the course of writing TEX and Metafont, Knuth developed some interesting ideas about software engineering. For instance, there is the Web system for ‘literate programming’. We will take a look at that, as well as at the general idea of markup.

Handouts and further reading for this chapter

Knuth wrote a history of the TEX project in [6], reprinted in ‘Literate Programming’, which is on reserve in the library, *QA76.6.K644 1992*.

For software engineering, consult the following journals:

- Software practice and experience
- Journal of systems and software
- ACM Transactions on Software Engineering and Methodology
- IEEE Transactions on Reliability

Some of these the library has available online.

Projects for this chapter.

Project 7.1. Do a literature study of code/documentation development. Here are some places to start:

POD Plain Old Documentation; used for Perl. `http://www.perl.com/pub/a/tchrist/litprog.html`

JavaDoc `http://java.sun.com/j2se/javadoc/`

Doxygen `http://www.stack.nl/˜dimitri/doxygen/`

Fitnesse `http://fitnesse.org/`

Leo `http://webpages.charter.net/edreamleo/front.html`

What schools of thought are there about developing medium size codes such as TEX? How does Knuth's philosophy relate to the others?

Project 7.2. Compare the TeX ”way” to MS Word, PageMaker, FrameMaker, Lout, Griff, previewLaTeX.

Project 7.3. TEX has been criticized for its arcane programming language. Would a more traditional programming language work for the purpose of producing text output? Compare TEX to other systems, in particular lout, `http://www.pytex.org/`, ant `http://www-mgi.informatik.rwth-aachen.de/˜blume/Download.html` and write an essay on the possible approaches. Design a system of your own.

Project 7.4. TEX and HTML were designed primarily with output in mind. Later systems (XML, DocBook) were designed so that output would be possible, but to formalize the structure of a document better. However, XML is impossible to write by hand. What would be a way out? Give your thoughts about a better markup system, conversion between one tool and another, et cetera.

Project 7.5. Several improvements on TEX and LATEX have been developed or are under development. Investigate NTS, LaTeX3, Context, Lollipop TEX describe their methodologies, and evaluate relative merits.

Project 7.6. Knuth has pretty liberal ideas about publishing software; somewhat against the spirit of the times. Report on software patents, the difference between patents and copyright, the state of affairs in the world. Read `http://swpat.ffii.org/gasnu/knuth/index.en.html`

Project 7.7. Knuth devised the ‘torture test’ approach to program correctness. Report on various schools of thought on this topic. Where does Knuth's approach stand?

Bibliography

[1] V. Eijkhout. An indentation scheme. *TUGboat*, 11:613–616.

[2] V. Eijkhout. A paragraph skip scheme. *TUGboat*, 11:616–619.

[3] V. Eijkhout and A. Lenstra. The document style designer as a separate entity. *TUGboat*, 12:31–34, 1991.

[4] A. Jeffrey. Lists in TEX's mouth. *TUGboat*, 11:237–245, 1990.

[5] D.E. Knuth. *Digital Typography*.

[6] D.E. Knuth. The errors of TEX. *Software Practice and Experience*, 19:pages = 607–681.

[7] D.E. Knuth. *TEX: the Program*. Addison-Wesley, 1986.

[8] D.E. Knuth. *The TEX book*. Addison-Wesley, reprinted with corrections 1989.

[9] D.E. Knuth and M.F. Plass. Breaking paragraphs into lines. *Software practice and experience*, 11:1119–1184, 1981.

[10] Donald E. Knuth. *The Art of Computer Programming, Volume 3, Sorting and Searching*. Addison Wesley Longman, 1998. Second edition.

[11] Helmut Kopka and Patrick W. Daly. *A Guide to LATEX*. Addison-Wesley, first published 1992.

[12] L. Lamport. *LATEX, a Document Preparation System*. Addison-Wesley, 1986.

[13] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, and Chris Rowley. *The LATEX Companion, 2nd edition*. Addison-Wesley, 2004.

[14] Michael F. Plass. *Optimal Pagination Techniques for Automatic Typesetting Systems*. PhD thesis, 1981. also Xerox technical report ISL-81-1.

[15] Richard Rubinstein. *Digital Typography*. Addison-Wesley, 1988.