General Computer Science I (320101) Fall 2014
TeX/LaTeX Tutorial

Michael Kohlhase
Jacobs University Bremen

November 24, 2014

Abstract

This document is a tutorial for the use of TeX/LaTeX in GenCS; it has two parts: The first is a brief general introduction to the concepts of the "documents-as-programs" paradigm and the TeX typesetting system from the perspective of Computer Science. The second is a sequence of simple (but increasingly difficult) typesetting problems designed to practice the art of typesetting beautiful documents with LaTeX

Contents

1 Programming Documents

Idea: Even though documents should be thought of as sequences of characters with markup (and images, formulae, tables, etc.), we can also think of them as *programs that produce such characters with markup*. In some situations, this is profitable, e.g. when the documents have parts that can be computed from the rest, e.g. a table of contents, the section numberings, or indices. In such situations, the author does not need to type in the computable document fragments, but can just represent them by a command. A conversion program interprets such a "document program" (usually text interspersed with commands), executes all the commands, and outputs a document (without commands), which can then be read. The main advantage of the "documents as programs" paradigm is that the computed document fragments can never get out of sync with the rest of the document, which eases the maintenance burden over the document life-cycle.

There are various implementations of this idea, in this section we present the TEX/LATEX system, in which the `pdflatex` program is used to transform documents with macros into `PDF`. Systems like PHP do similar things for the Web.

The TEX Typesetting System

▷ **Definition 1.1** Typesetting is the process of creating the visual appearance of a document by assembling glyphs (visual representations of characters; also called types) on pages.

▷ Since Gutenberg's time (to ca. 1975), typesetting was done by assembling movable types (special metal positives of single letters) into lines and later into pages, which were inked and the printed; or using negatives to form cast-metal positives for printing.

▷ **Definition 1.2** TEX is a typesetting program designed by Donald Knuth in 1978. It combines movable types (character boxes) with macro programming.

▷ **Definition 1.3** The `pdftex` program reads a file of text marked up with TEX macros and outputs `PDF`.

▷ **Example 1.4 (Hello World in TEX)** `pdftex` typesets the following TEX file

```
Hello, World \bye
```

The command sequence \bye stops `pdftex` and is not shown in the output.

Note that the "document program"

```
Hello, World \bye
```

the `pdftex` interprets all characters as "self-inserting characters", i.e the character "a" is essentially a command that inserts a character "a" into the `PDF` (in the right font and size).

We have already seen one document program command used by TEX above, and there are many more. Most of them insert special characters into the document or change the formatting. But TEX goes much further, it allows the author to define commands as well. This makes the TEX format self-extensible, and into a very expressive special purpose programming language for documents.

TEX Macros for Programming Documents

- ▷ TEX uses command sequences (words starting with "\"; also called macros) for special effects.
- ▷ **Example 1.5** `\bye` stops the formatter, `\alpha` prints α, `\int` prints $\int$,...
- ▷ Users can also define TEX macros as abbreviations via `\def`
- ▷ **Example 1.6** `\def\tdm{Text and Digital Media}` defines the macro `\tdm`. `We love the USC ''\tdm''!` expands to "We love the USC "Text and Digital Media"!
- ▷ TEX macros can have arguments specify with `#1`, `#2`...: delimit with `{` and `}`
- ▷ **Example 1.7** with the macro `\def\tnwhat#1{Text and \textbf{#1}}` `\tnwhat{Beer}` expands to "Text and **Beer**"

TEX was invented by a mathematician, so it is not a surprise that it is the most capable tool for typesetting formulae — an art that only a select few professional typesetters (humans who put lead into rows) could do.

Mathematical Formulae in TEX

- ▷ **Definition 1.8** TEX has a math mode for formulae delimited with `$` (inline math) or `\[` and `\]` (display math)
- ▷ **Example 1.9** Some TEX commands can be used everywhere: e.g. the Greek letters, `\alpha` prints α, `\beta` prints β,...
- ▷ **Example 1.10** Many TEX commands only make sense in math mode: e.g. superscripts with `^`, e.g. `x^3` gives x^3, subscripts with `_`, e.g. `x_{ij}` gives x_{ij}, `\int` prints $\int$, `\frac{1}{2}` prints $\frac{1}{2}$,...
- ▷ **Example 1.11** `$\int_0^\infty f(\theta) d\theta$` expands to $\int_0^\infty f(\theta)d\theta$
- ▷ **Example 1.12** Use macros in math mode as well: `\def\frac#1#2{#1\over #2}`

  Then `\[1+\frac{2}{2+\frac{3}{3+\ldots}}\]` expands to

$$1+\frac{2}{2+\frac{3}{3+\ldots}}$$

One of the things that TEX is useful for is to automate numbering of sections, subsections, footnotes, etc. For that TEX offers some basic data structures. Here we introduce counters, and show how we can make simple sectioning macros from them.

TEX Counters

- ▷ TEX uses special macros as counters, `\newcount`, allocates a counter, `\advance` alters it, and `\the` references it.
- ▷ **Example 1.13** We define a sectioning macros

```
\newcount\seccount % allocate a new counter for sections
\newcount\subseccount % allocate a new counter subsections
\seccount0\subseccount0 % initialise both with 0
\def\section#1{ % begin macro definition
\advance\seccount by 1 % step the counter
\subseccount0 % reset the subsection counter
\textbf{\Large\the\seccount. #1} % section number and title
} % end macro definition
\def\subsection#1{\advance\subseccount by 1
\textbf{\large\the\seccount.\the\subseccount. #1}}
```

Anyone who is experienced in programming realizes that TEX is not a modern programming language. But of course, it was conceived in 1978, the age of COBOL, and a lot has happened in programming language design since then. But even if it is relatively inconvenient and ugly code, it gets the job done.

We will now present a couple of internal macros that build up to more document automation that shows the advantages of programming documents: a serial letter macro.

TEX Conditionals

- ▷ TEX provides some conditionals for your use:
  e.g. `\ifx` compares two macros, `\ifnum` compares two number, and `\ifmmode` tells you if you are in math mode.
  `\if`⟪cond⟫`...\else...\fi` uses it.
- ▷ TEX uses special macros for user-defined conditionals, `\newif\if`⟪cond⟫, allocates a conditional, ⟪cond⟫`true` and ⟪cond⟫`false` alter it,

Programming a Chain Letter

- ▷ **Example 1.14 (A Parametric Reminder)**

```
\def\reminder#1#2{\hfill Bremen, \today\par\bigskip
\noindent Dear #1,\par\medskip\noindent
please be sure that you will not forget to come to the lecture
today. We are planning big things.\par\medskip\noindent
Sincerely,\par\bigskip\noindent #2\newpage}
```

- ▷ **Example 1.15 (Programming a Serial Letter)**
  We can use arbitrary characters to delineate arguments in macro definitions.

```
\def\sletter#1,#2;{\def\first{#1}\def\second{#2}\def\empty{}
\ifx\first\empty\else\reminder{#1}{Thomas \& Michael}
\ifx\second\empty\else\sletter#2,;\fi\fi}
```

```
\def\serialletter#1{\sletter #1;}
```

Also nothing prevents us from using recursion.

▷ **Example 1.16 (Making a Serial Letter)**

```
\serialletter{Mati, Anca, Isabel, Calin}
```

Our serial letter example shows that with a bit of programming effort the self-extensibility of TEX can be used to automate various document-oriented tasks, or style the documents for a given situation. Naturally, this brought forth a vibrant community that started swapping and re-using TEX programs.

TEX Macro Packages

▷ Idea: Separate out common macro definitions into a separate file and include that via \input. (So we can reuse them over multiple documents)

▷ Actually: many people have already done that.

▷ The AMS (American Mathematical Society) supplies AMSTEX: TEX macros that make it more convenient to write Math (e.g. the \frac macro)

▷ Till Tantau supplies tikz (TEX ist kein Zeichenprogram): TEX macros that allow you to draw images.

▷ Leslie Lamport supplies LATEX, a set of TEX packages and classes. pdflatex is pdftex with the LATEX package macros pre-loaded.

▷ The bibTEX package handles bibliographic references.

The most widely used macro package for TEX is LATEX, there are tens of thousands of macro packages that use the basic LATEX infrastructure. LATEX is the standard for high-end document formatting for scientific/technical documents nowadays. We now show a typical document as model for your own documents.

The Anatomy of a LATEX Document

▷ **Example 1.17** A LATEX file: main.tex

```
\documentclass{article} % use the article class (Journal Article)
\title{Anatomy of a {\LaTeX} Document} % specify the title,
\author{Michael Kohlhase\\Jacobs University Bremen} % author,
\date{\today} % and date
\begin{document} % start the document
\maketitle % make the title
\tableofcontents % make the table of contents
\section{Introduction}\label{sec:intro}
This is really easy, just start writing,
\section{Main Part}\label{sec:main}
```

```
We refer the reader to~\cite{Lamport:ladps94} for details.
But there should be at least one formula:
\[1+\frac{2}{2+\frac{3}{3+\ldots}}\]
\section{Conclusion}\label{concl:intro}
As we already said in Section~\ref{sec:intro} on
p. \pageref{sec:intro} this was not so bad was it?
\bibliographystyle{alpha}
\bibliography{example}
\end{document}
```

▷ Format it with `pdflatex main` (generates `main.aux` for references)

and the bibTEX database used in it

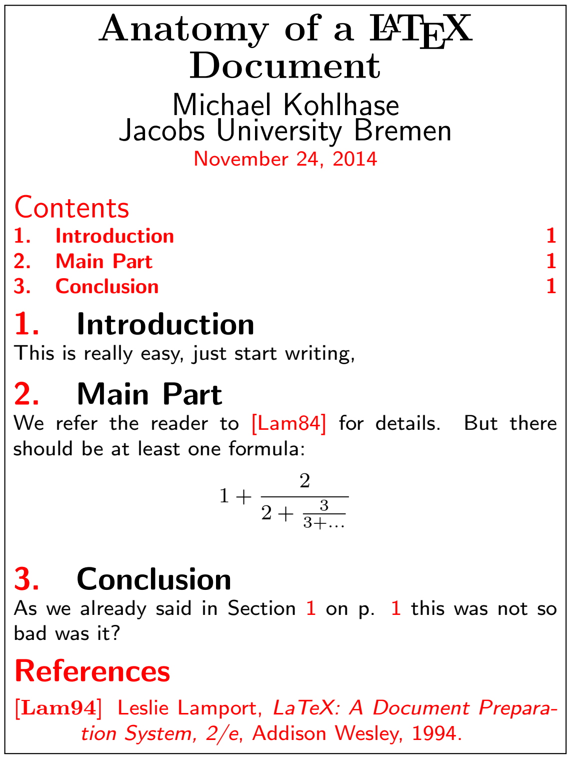
▷ **Example 1.18** a bibTEX file `example.bib`

```
@BOOK{Lamport:ladps94,
  title = {LaTeX: A Document Preparation System, 2/e},
  publisher = {Addison Wesley},
  year = {1994},
  author = {Leslie Lamport}}
```

▷ Generate bibliography with `bibtex main`(it knows about `example.bib` from `main.aux`)

▷ run `pdflatex` twice (to get all the cross-references right)

The Result (generated parts in red)



Anatomy of a LaTeX Document

Michael Kohlhase
Jacobs University Bremen
November 24, 2014

Contents

1. Introduction

This is really easy, just start writing,

2. Main Part

We refer the reader to [Lam84] for details. But there should be at least one formula:

$$1 + \frac{2}{2 + \frac{3}{3+\ldots}}$$

3. Conclusion

As we already said in Section 1 on p. 1 this was not so bad was it?

References

[Lam94] Leslie Lamport, *LaTeX: A Document Preparation System, 2/e*, Addison Wesley, 1994.

2 Learning LaTeX by Example

The best way of learning LaTeX is to "program" a set of example documents. The problems below provide you with a set of problems that gradually introduce the salient features of LaTeX and should get you going for most of the documents you will need initially.

Solutions to these problems are available at

`http://kwarc.info/teaching/GenCS1/latex-tutorial-with-solutions.pdf`.

But you should try them alone first to maximize learning.

There are good TeX, LaTeX, and bibTeX tutorials on the Web which you should use for solving these problems, but also consult [?, ?] and (if you want to drink from the source and know the gory details) [?] (als known as the TeX bible). The course instructor and the TAs will be happy to help you and get you unstuck, when necessary. But you should try to solve them by yourself first to make progress.

But before you can start, you will need a LaTeX installation on your computer, so that you can format your documents and practice. For `UNIX`-based systems (e.g. `linux` and `MacOSX`), the TeXLive distribution is currently the best (see `http://www.tug.org/texlive/` for details and installation instructions). For `Windows`, you should use MikTeX (see `http://miktex.org`).

You should not expect to be able to get through all the problems in the tutorial itself, indeed, if you manage the first six or seven, then you are on a very good track. The remaining ones are for self-study in the next weeks. The introduce the finer points of TeX/LaTeX.

2.1 LaTeX Basics

Problem 2.1 (Hello World in LaTeX)
Write a "hello world" document in LaTeX, i.e. a document that only contains the two words "Hello World".

Problem 2.2 (A LaTeX with Title)
Write a document with a title, the date of today, and yourself as an author (with Jacobs University as the affiliation) It should look like this:

The Evolution of Abstract Nonsense

General Computer Scientist
Jacobs University Bremen
`gc.scientist@jacobs-university.de`

23. July 2011

Problem 2.3 (A LaTeX Document with Sections and Table of Content)
Extend the document from ?prob.doctitle? with a couple of sections and subsections of your choice via the `\section` macro for sections and (correspondingly) `\subsection` for subsections.

Cross-reference various of the sections using the `\label` and `\ref` macros.

Hint: When you use the `hyperref` package (use `\usepackage{hyperref}` at the very end of the preamble), then the references become hyper-references (clickable in the PDF). Try this on your document!

Problem 2.4 (Complex Tables)

Write the GenCS Grading Table on the right using the `tabular` environment. Note that the first column in this table is left aligned, the second one centered,

Component	%	Comment
Monday Quizzes	30	to make you study continuously
Homeworks	20	practice
Midterm Exam	20	to see if you excel at CS
Final Exam	30	to prove that

and the third one is 4 cm long and allows multi-line content. Note furthermore, that there is a double line after the first row.

The `tabular` environment takes a format string in the first argument. Here `|` makes a table cell border, `l` and `c` specify left/centered alignment, and `p{4cm}` a paragraph box 4 centimeters wide. `&` separates columns, `\\` makes a new table row, and `\hline` a horizontal cell border.

Problem 2.5 (Creating a Bibliography)
Extend the paper from ?prob.docsections? with three references: your Bachelor's thesis, your first journal article, and your first book (make them up if you have not written those). You should use the bibTEX program for this.

Hint: It is generally a good idea to start a bibTEX database of the scientific papers and books you have read early, so that you can cite them in your papers later.

Hint: There is a relatively new successor to bibTEX called `biblatex`, you may want to eventually have a look at that.

2.2 LATEX Math

Problem 2.6 (Simple Math Formulae)

The solutions of the quadratic equation $ax^2+bx+c=0$ are $\dfrac{-b\pm\sqrt{b^2-4ac}}{2a}$.

Write this in LATEX

Problem 2.7 (A more complex Math Formula)

The Taylor series of $\sqrt{1+x}$ about $x=0$ converges for $|x|\leq 1$ and is given by

$$\sqrt{1+x}=\sum_{n=0}^{\infty}\frac{(-1)^n 2n!}{(1-2n)(n!)^2(4^n)}x^n=1+\frac{1}{2}x-\frac{1}{8}x^2+\frac{1}{16}x^3-\frac{5}{128}x^4+\ldots$$

Write this in LATEX, but note that the last multi-equation is in "display style" (i.e. centered and with bigger fonts).

Problem 2.8 (Matrices)
Write the following multiplication of 2×2 matrices in LATEX:

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}=\begin{pmatrix} a_{11}b_{11}+a_{12}b_{21} & a_{11}b_{21}+a_{12}b_{22} \\ a_{21}b_{12}+a_{22}b_{21} & a_{21}b_{21}+a_{12}b_{22} \end{pmatrix}$$

Problem 2.9 (Displayed Equations)
Write the formula from ?prob.math-display? as an equation array using the `eqnarray` environment and reference the second equation in the text, so that it looks like

$$\sqrt{1+x} = \sum_{n=0}^{\infty}\frac{(-1)^n 2n!}{(1-2n)(n!)^2(4^n)}x^n \quad (1)$$

$$= 1+\frac{1}{2}x-\frac{1}{8}x^2+\frac{1}{16}x^3-\frac{5}{128}x^4+\ldots \quad (2)$$

Now we reference the partial equations: the first with (1) and the second with (2). Note that there is a variant `eqnarray*` that does not make the equation numbers.

2.3 LaTeX Macros

Problem 2.10 (Matrix Macros)
You can make TeX macros to make your life easier.

1. Write a macro `\ttmatrix` that takes four arguments and writes a 2×2 matrix: for example `\ttmatrix{a}{b}{c}{d}` prints $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$.
2. Write a macro `\gttmatrix` that takes a single argument variable and prints a generic 2×2 matrix: for example `\gttmatrix{a}` prints $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$.
3. With these macros write the matrix multiplication from ?prob.math-matrices? more succinctly.