The Joys of LaTeX: An Entry Guide for the Potentially Obsessive

Kent A. Peacock

Department of Philosophy
University of Lethbridge

October 3, 2016

What Is LaTeX?

- LaTeXis a document formatting system widely used in technical subjects such as logic, math, physics, linguistics, computer science, philosophy of science. . .

What Is LaTeX?

- LaTeXis a document formatting system widely used in technical subjects such as logic, math, physics, linguistics, computer science, philosophy of science. . .
- It has been in use since the 1980s and remains the industry standard; nothing else compares in the quality of output and the huge range of symbols and images that can be produced—if you know how!!

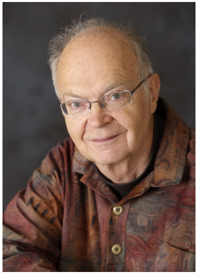
What Is LaTeX?

- LaTeXis a document formatting system widely used in technical subjects such as logic, math, physics, linguistics, computer science, philosophy of science. . .
- It has been in use since the 1980s and remains the industry standard; nothing else compares in the quality of output and the huge range of symbols and images that can be produced—if you know how!!
- LaTeX allows you to format beautiful equations beautifully:

$$R_{\mu\nu} - {}^1\!/_2\, g_{\mu\nu} R = T_{\mu\nu} \tag{1}$$

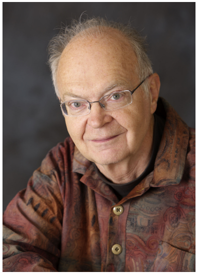
TEX

- LATEX is based on TEX, created by Donald Knuth (1938–) in the 1970s.



TEX

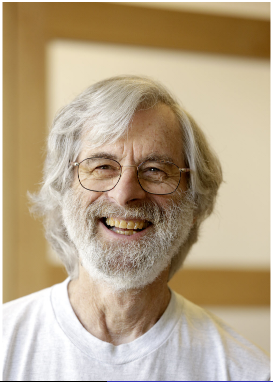
- LATEX is based on TEX, created by Donald Knuth (1938–) in the 1970s.



- N.b.: TEX is pronounced “tech” with a gutteral “ch” as in “Loch”. (It is derived from the Greek $\tau\epsilon\chi\nu\eta$, “techne.”)

LaTeX

- In 1986, Leslie Lamport (1941–) created LaTeX, a system of macros written in TeX designed to simplify document preparation.



TEX *versus* LATEX

- TEX is a powerful language that allows you to command a laser printer to do virtually anything that it is physically possible for it to do.

TeX *versus* LaTeX

- TeX is a powerful language that allows you to command a laser printer to do virtually anything that it is physically possible for it to do.
- Drawback: if all you want to do is write papers in logic etc., it is too much work to construct your documents entirely from scratch.

TEX *versus* LATEX

- TEX is a powerful language that allows you to command a laser printer to do virtually anything that it is physically possible for it to do.
- Drawback: if all you want to do is write papers in logic etc., it is too much work to construct your documents entirely from scratch.
- So Lamport wrote a set of TEX commands (macros) which define standardized ways of setting up papers, books, etc. It has all the power of TEX if you want to use it but greatly speeds up most formatting tasks.

The LaTeX Community

- Important: TeX and LaTeX are open software; unlike proprietary software such as MS Word, the source code is available and anyone is free to modify or add to it (so long as you call your version something different).

The LaTeX Community

- Important: TeX and LaTeX are open software; unlike proprietary software such as MS Word, the source code is available and anyone is free to modify or add to it (so long as you call your version something different).
- Since Lamport published LaTeX in 1986, hundreds of people have improved it and added innumerable special tools ("packages") to increase its usefulness.

The LaTeX Community

- Important: TeX and LaTeX are open software; unlike proprietary software such as MS Word, the source code is available and anyone is free to modify or add to it (so long as you call your version something different).
- Since Lamport published LaTeX in 1986, hundreds of people have improved it and added innumerable special tools (“packages”) to increase its usefulness.
- There are several versions of LaTeX available *via* free download.

LaTeX *vs.* Word

- Common word processors such as MS Word are WYSIWYG—"What you see is what you get."

LaTeX *vs.* Word

- Common word processors such as MS Word are WYSIWYG—“What you see is what you get.”
- LaTeX is a “markup language,” a programming language in which you write formatting commands and text in an *input* file and then process the file to produce your *output*.

LaTeX *vs.* Word

- Common word processors such as MS Word are WYSIWYG—“What you see is what you get.”
- LaTeX is a “markup language,” a programming language in which you write formatting commands and text in an *input* file and then process the file to produce your *output*.
- The input file is written in text (ASCII) characters (in simple terms, the characters on a standard computer keyboard).

LaTeX *vs.* Word

- Common word processors such as MS Word are WYSIWYG—“What you see is what you get.”
- LaTeX is a “markup language,” a programming language in which you write formatting commands and text in an *input* file and then process the file to produce your *output*.
- The input file is written in text (ASCII) characters (in simple terms, the characters on a standard computer keyboard).
- The output file can be in several forms but these days it is most commonly .pdf, which is almost universally used and very stable (unlike Word!!).

Basic Process

- You write an *input file* in ASCII code using a “front end” (such as TeXWorks, which comes bundled with MiKTeX).

Basic Process

- You write an *input file* in ASCII code using a "front end" (such as TeXWorks, which comes bundled with MiKTeX).
  - The "front end" consists of a text editor and other tools (such as spell-checkers), usually designed for work in LaTeX.

Basic Process

- You write an *input file* in ASCII code using a “front end” (such as TeXWorks, which comes bundled with MiKTeX).
  - The “front end” consists of a text editor and other tools (such as spell-checkers), usually designed for work in LaTeX.
  - WinEDT is a particularly good LaTeX front-end.

Basic Process

- You write an *input file* in ASCII code using a "front end" (such as TeXWorks, which comes bundled with MiKTeX).
  - The "front end" consists of a text editor and other tools (such as spell-checkers), usually designed for work in LaTeX.
  - WinEDT is a particularly good LaTeX front-end.
- The file is processed by a compiler which generates an output file with all type, graphics as you commanded; usually in .pdf form.

Basic Document Structure

- Open with `\documentclass` statement.

Basic Document Structure

- Open with `\documentclass` statement.
  - Typical: `\documentclass[12pt]{article}`

Basic Document Structure

- Open with `\documentclass` statement.
    - Typical: `\documentclass[12pt]{article}`
- Next, the *preamble*, which typically loads packages and contains other global instructions.

Basic Document Structure

- Open with `\documentclass` statement.
  - Typical: `\documentclass[12pt]{article}`
- Next, the *preamble*, which typically loads packages and contains other global instructions.
- `\begin{document}`

Basic Document Structure

- Open with `\documentclass` statement.
  - Typical: `\documentclass[12pt]{article}`
- Next, the *preamble*, which typically loads packages and contains other global instructions.
- `\begin{document}`
- The content of your document.

Basic Document Structure

- Open with `\documentclass` statement.
  - Typical: `\documentclass[12pt]{article}`
- Next, the *preamble*, which typically loads packages and contains other global instructions.
- `\begin{document}`
- The content of your document.
- `\end{document}`

Basic Document Structure

- Open with `\documentclass` statement.
  - Typical: `\documentclass[12pt]{article}`
- Next, the *preamble*, which typically loads packages and contains other global instructions.
- `\begin{document}`
- The content of your document.
- `\end{document}`
- You can "store" templates, out-takes, or comments after the `\end{document}` since nothing after that statement is processed.

Entering Text

- Type your content in text characters.

Entering Text

- Type your content in text characters.
- All commands begin with the backslash: \.

Entering Text

- Type your content in text characters.
- All commands begin with the backslash: \.
- Construct accented letters and other special characters with appropriate commands:

Entering Text

- Type your content in text characters.
- All commands begin with the backslash: \.
- Construct accented letters and other special characters with appropriate commands:
    - E.g.: `{'e}` gives é.

Entering Text

- Type your content in text characters.
- All commands begin with the backslash: `\`.
- Construct accented letters and other special characters with appropriate commands:
  - E.g.: `{'e}` gives é.
  - E.g.: `\S` gives § (section indicator).

Entering Text

- Type your content in text characters.
- All commands begin with the backslash: \.
- Construct accented letters and other special characters with appropriate commands:
    - E.g.: `{'e}` gives é.
    - E.g.: `\S` gives § (section indicator).
    - Control characters %, $, &, \ must have a back-slash ('escape character') if you want to use them as a character; e.g., `\&` gives &.

Entering Text

- Type your content in text characters.
- All commands begin with the backslash: `\`.
- Construct accented letters and other special characters with appropriate commands:
  - E.g.: `{'e}` gives é.
  - E.g.: `\S` gives § (section indicator).
  - Control characters %, $, &, \ must have a back-slash ('escape character') if you want to use them as a character; e.g., `\&` gives &.
- Just type: text will wrap automatically; LaTeX interprets an extra line as a paragraph break. But spaces between letters are ignored.

Logical Structure of Your Document

- Use sectioning commands to define structure (the order in which you want to present your story).

Logical Structure of Your Document

- Use sectioning commands to define structure (the order in which you want to present your story).
- You can cut and paste your sections, and LaTeX will automatically number them; you only have to worry about content.

Logical Structure of Your Document

- Use sectioning commands to define structure (the order in which you want to present your story).
- You can cut and paste your sections, and LaTeX will automatically number them; you only have to worry about content.
- Structure can be defined using `\chapter` (in the book class), `\section`, `\subsection`, etc.

Environments

- There are many "environments" which define specific kinds of formatting.

Environments

- There are many “environments” which define specific kinds of formatting.
- Example

  ```
  \begin{center}
    Centred text.
  \end{center}
  ```

  Gives

  Centred text.

Environments

- There are many "environments" which define specific kinds of formatting.
- Example

  ```
  \begin{center}
    Centred text.
  \end{center}
  ```

  Gives

  Centred text.
- There are "list-making environments" that generate bulleted lists, numbered lists, etc.

Indexes, Tables, Tables of Contents

- There are commands to construct tables (though these can be tricky. . .).

Indexes, Tables, Tables of Contents

- There are commands to construct tables (though these can be tricky. . .).
- Tables of contents are constructed automatically in the `book` document class; just put in a new chapter or section, and it is added to the table of contents.

Indexes, Tables, Tables of Contents

- There are commands to construct tables (though these can be tricky. . .).
- Tables of contents are constructed automatically in the `book` document class; just put in a new chapter or section, and it is added to the table of contents.
- There are packages for indexes.

Indexes, Tables, Tables of Contents

- There are commands to construct tables (though these can be tricky. . .).
- Tables of contents are constructed automatically in the `book` document class; just put in a new chapter or section, and it is added to the table of contents.
- There are packages for indexes.
- Footnotes are built into basic LaTeX; you need special packages for end-notes.

Indexes, Tables, Tables of Contents

- There are commands to construct tables (though these can be tricky. . .).
- Tables of contents are constructed automatically in the `book` document class; just put in a new chapter or section, and it is added to the table of contents.
- There are packages for indexes.
- Footnotes are built into basic LaTeX; you need special packages for end-notes.
- The command `\dots` gives ellipsis . . .

Graphics

- The original graphics capability of LaTeX was limited, but many, many packages have been created over the years; I provide links to some in my 'Resources' web page: http://scholar.ulethbridge.ca/kentpeacock/resources.

Graphics

- The original graphics capability of LaTeX was limited, but many, many packages have been created over the years; I provide links to some in my 'Resources' web page: `http://scholar.ulethbridge.ca/kentpeacock/resources`.
- With the `graphicx` package it is quite easy to include illustrations.

The Little Things That Make Your Work Look Professional

- Use accent grave for left quote marks, the apostrophe for right quote marks.

The Little Things That Make Your Work Look Professional

- Use accent grave for left quote marks, the apostrophe for right quote marks.
  - Thus, `` ``quote'' `` will give “quote”.

The Little Things That Make Your Work Look Professional

- Use accent grave for left quote marks, the apostrophe for right quote marks.
  - Thus, ``` ``quote'' ``` will give “quote”.
  - Do *not* use the ASCII double-quote character; a sure mark of the rank amateur!

The Little Things That Make Your Work Look Professional

- Use accent grave for left quote marks, the apostrophe for right quote marks.
  - Thus, ``` ``quote'' ``` will give “quote”.
  - Do *not* use the ASCII double-quote character; a sure mark of the rank amateur!
- Form dashes correctly:

The Little Things That Make Your Work Look Professional

- Use accent grave for left quote marks, the apostrophe for right quote marks.
  - Thus, `` ``quote'' `` will give “quote”.
  - Do *not* use the ASCII double-quote character; a sure mark of the rank amateur!
- Form dashes correctly:
  - Use - as a hyphen, as in “multi-use”.

The Little Things That Make Your Work Look Professional

- Use accent grave for left quote marks, the apostrophe for right quote marks.
  - Thus, ``` ``quote'' ``` will give “quote”.
  - Do *not* use the ASCII double-quote character; a sure mark of the rank amateur!
- Form dashes correctly:
  - Use - as a hyphen, as in “multi-use”.
  - Use two hyphens to form en-dash –, as in date or number ranges. (“Read pp. 44–55 of the text.”)

The Little Things That Make Your Work Look Professional

- Use accent grave for left quote marks, the apostrophe for right quote marks.
  - Thus, ``` ``quote'' ``` will give “quote”.
  - Do *not* use the ASCII double-quote character; a sure mark of the rank amateur!
- Form dashes correctly:
  - Use - as a hyphen, as in “multi-use”.
  - Use two hyphens to form en-dash –, as in date or number ranges. (“Read pp. 44–55 of the text.”)
  - Use three hyphens to form em-dash, which is a punctuation dash used for an emphatic break. (“No candidate—not even Clinton—has said enough about climate.”)

Math Mode

- LaTeX has powerful capabilities to format mathematics.

Math Mode

- LaTeX has powerful capabilities to format mathematics.
- You have to go into "math mode". This is indicated by dollar signs $ or by entering certain math environments.

Math Mode

- LaTeX has powerful capabilities to format mathematics.
- You have to go into "math mode". This is indicated by dollar signs $ or by entering certain math environments.
- You can have in-line math symbols: e.g., `$e^{i\pi}=-1$` gives $e^{i\pi} = -1$.

Math Mode

- LaTeX has powerful capabilities to format mathematics.
- You have to go into “math mode”. This is indicated by dollar signs \$ or by entering certain math environments.
- You can have in-line math symbols: e.g., `$e^{i\pi}=-1$` gives $e^{i\pi} = -1$.
- Greek letters can be done in math mode: `$\alpha$` gives α.

Math Mode

- LaTeX has powerful capabilities to format mathematics.
- You have to go into “math mode”. This is indicated by dollar signs $ or by entering certain math environments.
- You can have in-line math symbols: e.g., `$e^{i\pi}=-1$` gives $e^{i\pi} = -1$.
- Greek letters can be done in math mode: `$\alpha$` gives α.
- Or you can display your math in a huge variety of ways (especially using `amsmath`).

Display Math

```
\begin{equation}
   \int_{-\infty}^{\infty} \,e^{-x^2} dx = \sqrt{\pi}
\end{equation}
```

gives

$$\int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi}. \tag{2}$$

Combining Text and Math

- The `amsmath` package has a very useful command `\text` which allows one to put text in a formula:

Combining Text and Math

- The `amsmath` package has a very useful command `\text` which allows one to put text in a formula:
- `$\text{$x^3+3x=7$  when $x=1$}$` gives

$$x^3 + 3x = 7 \text{ when } x = 1.$$

Combining Text and Math

- The `amsmath` package has a very useful command `\text` which allows one to put text in a formula:
- `$\text{$x^3+3x=7$  when $x=1$}$` gives

$$x^3 + 3x = 7 \text{ when } x = 1.$$

- Note: I put the whole line in `\text` because that gets the spacing right!

Combining Text and Math

- The `amsmath` package has a very useful command `\text` which allows one to put text in a formula:
- `$\text{$x^3+3x=7$  when $x=1$}$` gives

$$x^3+3x=7 \text{ when } x=1.$$

- Note: I put the whole line in `\text` because that gets the spacing right!
- Word to the wise: use math mode for symbols in text that are mathematical. E.g., say `let $p$ be a prime such that`, *not* `let p be a prime such that...`

Logic Symbols

- LaTeX has all the standard logic symbols: $\exists$, $\forall$, $\vee$, $\supset$, $\neg$, etc.

Logic Symbols

- LaTeX has all the standard logic symbols: $\exists$, $\forall$, $\vee$, $\supset$, $\neg$, etc.
- There is a package that does Fitch notation, and several that do truth trees (though this remains a work in progress).

Macros

- You can define your own symbols using `\newcommand`.

Macros

- You can define your own symbols using `\newcommand`.
- To get the Russell up-side-down iota, you can use `\newcommand{\Russell}{\rotatebox[origin=c]{180}{$\iota$}}` (This requires the `graphicx` package.)

Macros

- You can define your own symbols using `\newcommand`.
- To get the Russell up-side-down iota, you can use `\newcommand{\Russell}{\rotatebox[origin=c]{180}{$\iota$}}` (This requires the `graphicx` package.)
- Example of usage: `$\Russell xFx$` gives ℩xFx, which is read, "The x such that Fx".

Macros with Parameters

- `\newcommand{\ket}[1]{|#1\rangle}`
  defines a Dirac "ket" that can have any symbol put in place of the `#1`.

Macros with Parameters

- `\newcommand{\ket}[1]{|#1\rangle}`
  defines a Dirac “ket” that can have any symbol put in place of the `#1`.
- So `$\ket{\psi}$` gives $|\psi\rangle$.

Citation

- Two options: use the bibliography function built into LaTeX, or use BibTeX.

Citation

- Two options: use the bibliography function built into LaTeX, or use BibTeX.
- Example of inbuilt bibliography:

```
{\frenchspacing\raggedright
\begin{thebibliography}{99}
\bibitem{AA80}
  Aharonov, Y., and D. Albert (1980):  ‘States and
 Observables in Relativistic Quantum  Field Theories’,
\textsl{Physical Review D} 21, 3316--3324.
\end{thebibliography}
}
```

Citation

- Two options: use the bibliography function built into LaTeX, or use BibTeX.
- Example of inbuilt bibliography:

```
{\frenchspacing\raggedright
\begin{thebibliography}{99}
\bibitem{AA80}
  Aharonov, Y., and D. Albert (1980):  ‘States and
 Observables in Relativistic Quantum  Field Theories’,
\textsl{Physical Review D} 21, 3316--3324.
\end{thebibliography}
}
```

- Use `\cite[p. 3316--7]{AA80}` to cite this reference.

BibTeX

- Create a `yourbib.bib` file, which is a database of all your references.

BibTeX

- Create a `yourbib.bib` file, which is a database of all your references.
- Where you want the bibliography to appear, include this code:

```
\bibliographystyle{plain}
\bibliography{yourbib}
```

BibTeX

- Create a `yourbib.bib` file, which is a database of all your references.
- Where you want the bibliography to appear, include this code:

```
\bibliographystyle{plain}
\bibliography{yourbib}
```

- Advantages:

BibTeX

- Create a `yourbib.bib` file, which is a database of all your references.
- Where you want the bibliography to appear, include this code:

```
\bibliographystyle{plain}
\bibliography{yourbib}
```

- Advantages:
  - There are many style files (and you can create your own).

BibTeX

- Create a `yourbib.bib` file, which is a database of all your references.
- Where you want the bibliography to appear, include this code:
  ```
  \bibliographystyle{plain}
  \bibliography{yourbib}
  ```
- Advantages:
  - There are many style files (and you can create your own).
  - Once you have entered a reference in your `.bib` database, you never have to enter it again.

BibTeX

- Create a `yourbib.bib` file, which is a database of all your references.
- Where you want the bibliography to appear, include this code:

```
\bibliographystyle{plain}
\bibliography{yourbib}
```

- Advantages:
    - There are many style files (and you can create your own).
    - Once you have entered a reference in your `.bib` database, you never have to enter it again.
    - It integrates with other citation management systems such as Zotero.

BibTeX

- Create a `yourbib.bib` file, which is a database of all your references.
- Where you want the bibliography to appear, include this code:
  ```
  \bibliographystyle{plain}
  \bibliography{yourbib}
  ```
- Advantages:
  - There are many style files (and you can create your own).
  - Once you have entered a reference in your `.bib` database, you never have to enter it again.
  - It integrates with other citation management systems such as Zotero.
- Disadvantage: BibTeX can be temperamental and there is a learning curve to climb.

Survival Tips for the LaTeX Rookie

- Compile your document frequently; that way you catch errors as soon as they occur.

Survival Tips for the LaTeX Rookie

- Compile your document frequently; that way you catch errors as soon as they occur.
    - Most common error is mis-matched curly brackets!

Survival Tips for the LaTeX Rookie

- Compile your document frequently; that way you catch errors as soon as they occur.
  - Most common error is mis-matched curly brackets!
- Use the defaults; don't try to construct your own document styles until you really know what you are doing.

Survival Tips for the LaTeX Rookie

- Compile your document frequently; that way you catch errors as soon as they occur.
  - Most common error is mis-matched curly brackets!
- Use the defaults; don't try to construct your own document styles until you really know what you are doing.
- As with all programming, it is good practice to *comment* any complicated code so that you won't forget how it was supposed to work!

Survival Tips for the LaTeX Rookie

- Compile your document frequently; that way you catch errors as soon as they occur.
    - Most common error is mis-matched curly brackets!
- Use the defaults; don't try to construct your own document styles until you really know what you are doing.
- As with all programming, it is good practice to *comment* any complicated code so that you won't forget how it was supposed to work!
    - Comment character: %.

Survival Tips for the LaTeX Rookie

- Compile your document frequently; that way you catch errors as soon as they occur.
  - Most common error is mis-matched curly brackets!
- Use the defaults; don't try to construct your own document styles until you really know what you are doing.
- As with all programming, it is good practice to *comment* any complicated code so that you won't forget how it was supposed to work!
  - Comment character: %.
- Warning! LaTeX is addictive!

LaTeX Resources

- See http://scholar.ulethbridge.ca/kentpeacock/resources.