Latex, packages and Beamer*

Andrey Lukyanenko, CSE, Aalto University

Spring, 2015

In this course we are not going to discuss basics in Latex, so it is assumed that basic information is known.[1]

1. **Editor**.
   The first thing needed for Latex is a good editor. Of course anyone could use any text-editor (such as **nano**, **vi**, **emacs** and **gedit**, etc), and whenever is needed to call additional tools (such as **ispell** for spell checking, etc), but full-functional latex-editor can still benifit for its convinience, functionality, build-in mass used latex elements and explicit editing options.

   Previously, for Linux primary was one such a tool, called **Kile**. With popularity of **Gnome** desktop environment it required to install additional 1GB packages related to **KDE** desktop. Now, recently released version 3 of **Texmaker** made a good light-weight substitute for **Kile** in Gnome.

   Both of the tools supports spell-checking (ispell), line numbering, good navigation, built-in fast compilation tools (we however still prefer own Makefiles), rich symbol sets and hot-keys. There is also windows only option **TeXnicCenter** or **gedit** plugins for latex.

   Reader may install any of the aforementioned editors or choose own, but we will concentrate here on **Texmaker**.

   The first thing to do is understand short-cuts as it saves a lot of time. (Go to **menu->options->Configure Texmaker** and **Editor** tab contains the comprehensive list of short-cuts available and set). By default the following often used are set.

Key	Command
Ctrl+T	After selecting text can comment it (even multiple lines at once)
Ctrl+I	\textit
Ctrl+B	\textbf
Ctrl+Shift+S	\textsl
Ctrl+Shift+C	\textsc
Ctrl+Shift+T	\texttt
Ctrl+Shift+E	\emph
Ctrl+Shift+A	\textsf
Ctrl+Shift+M	Inline math mode $...$
Alt+Shift+M	Display math mode \[...\]

   The author of this tutorial for convenience define own short-cuts using double set of commands in order to remove overlapping with environment in the form **Ctrl+Space,Ctrl+V** for **verbatim** environment or **Ctrl+Space,Ctrl+T** for tabular.

2. **Basic structure.** Every document has the form which is enough minimum to compile:

```
\documentclass[options]{class}

\begin{document}

\end{document}
```

   options can be

   - **10pt, 11pt, 12pt**
   - **a4paper, letterpaper, a5paper, b5paper, executivepaper, legalpaper**
   - **fleqn** – Formulas a left aligned.

*Good Latex documentation online at **http://en.wikibooks.org/wiki/LaTeX/**.

[1]Additional external material includes: Latex tutorial **http://www.tug.org/twg/mactex/tutorials/ltxprimer-1.0.pdf**, Documentation **http://www.latex-project.org/guides/**, especially "The (Not So) Short Introduction to LaTeX2e" **http://ctan.tug.org/tex-archive/info/lshort/english/lshort.pdf**. Symbols **ftp://ftp.funet.fi/pub/TeX/CTAN/info/symbols/comprehensive/symbols-a4.pdf**, Mathematical documentation **ftp://ftp.ams.org/ams/doc/amsmath/amsldoc.pdf**.

- **leqno** – Nombers of formulas on the left.
- **titlepage, notitlepage**
- **onecolumn, twocolumn**
- **twoside, oneside**
- **landscape**
- **openright, openany**
- **draft**

class can be one of set: **article, IEEEtran, proc, minimal, report, book, slides, memoir, letter, beamer.** The last one is an modern way of doing slides.

3. **List of obvious commands.**

   The following list command is expected to be known to reader: **begin, end, title, author, date, maketitle, usepackage, part, chapter, section, section*, subsection, subsubsection, paragraph, subparagraph, abstract, tableofcont appendix, itemize, enumerate, tabular**.

   For font modifiers: **textnormal, emph, textrm, textsf, texttt, textup, textit, textsl, textsc, uppercase, textbf, textmd**.

   For text sizes **tiny, scriptsize, footnotesize, small, normalsize, large, Large, LARGE, huge, Huge**.

4. **Description**.

   Description is similar to itemize and enumerate list, except that the points are defined by a user.

```
\begin{description}
\item [One] The first statement
\item [Two] The second
\begin{enumerate}
\item one
\item two
\end{enumerate}
\item [Others] \hfill \\ Other...
\end{description}
```

One The first statement

Two The second

(a) one

(b) two

Others
Other

5. **Quotes**

   When using quotes a common mistake is to use double quote sign "(\textquotedbl), which is incorrect for scientific articles. Use `` for opening double quote and '' for closing, e.g. ``hello'' corresponds to “hello”. The same is for single quotes `hello' will be compiled as ‘hello’.

6. **Tables**

7. **Newcommand**.
   Often it is required to use an environments more complex or simple but combined a lot. For this purpose it is benefitial to use own defined command. It can easilty be done using **newcommand** operation. In this and other crash tutorials we were using following:

```
\newcommand{\cmmd}[1]{\texttt{\bfseries{#1}}}
\newcommand{\cmit}[1]{\textit{\bfseries{#1}}}
\newcommand{\prgm}[1]{\texttt{{#1}}}
```

   Where **\cmmd** was used for strong bold text, compare **\cmmd{"/usr/bin"}** command produces **"/usr/bin"** and **"\textbf{"/usr/bin"}"** produces **/usr/bin**. In a latter case the slash sign is not what it supposed to be, it is text separator.

   Another useful commands are

```
\newcommand{\el}{\mathcal{L}}
\newcommand{\todo}[1]{{\bf\color{blue} TODO: #1}}   # \usepackage[x11names, rgb]{xcolor} is needed
\newcommand{\fixme}[1]{{\bf\color{red} FIXME: #1}}
```

   Now **\el** is for a mathematical symbol in math brackets $\mathcal{L}$, and showing what should be todo: **TODO: change this text**, or even **FIXME: fix it**.

8. **Verbatim, comment and url**.

   Sometimes it is useful to add a text which is not formatted by latex. For this purpose use verbatim environment.

   - To process text as it is use environment `verbatim`. It will simply output anything from inside to output file.

```
\begin{verbatim}
This is \textbf{bold} text
and math \(\epsilon\).
\end{verbatim}
```

```
This is \textbf{bold} text
and math \(\epsilon\).
```

   - In order to be able to format text inside of `verbatim` use `alltt` (needed also `\usepackage{alltt}`.

```
\begin{alltt}
This is \textbf{bold} text
and math \(\epsilon\).
\end{alltt}
```

```
This is bold text
and math ϵ.
```

   - There is a possibility to use a shortened version of `verbatim` environment:

     `\verb+\textbf+` Notice that the separator sign can be anything, in the example above it is `+` sign.

   - The verbatim environment also has comment which can be used to comment out set of lines:

```
\begin{comment}
This is \textbf{bold} text
and math \(\epsilon\).
\end{comment}
```

```
*empty here*
```

     Anyone may can create own comment command simply defining new command as `\newcommand{\comment}[1]{}`

   - **Urls.**
     Define in the beginning a package `url` in order to wrap around urls. E.g. `\usepackage{url}` and in the body of document `\url{http://en.wikibooks.org/wiki/LaTeX/}` to produce `http://en.wikibooks.org/wiki/LaTeX/`.
     However, it will simplify formatting of urls only, in order to add "clickable" url in the end document include also package hyperref: `\usepackage{hyperref}`. As an alternative to url can be **`href`** which hides actual rule. For example above: `\href{http://en.wikibooks.org/wiki/LaTeX/}{Latex wiki}` produces Latex wiki.

9. **Math environment.**

   One aspect here which sometimes omitted is bracket matching in Math environment. Any bracket pairs `( ) { } [ ] | |` should correspond to each other explicitly.

   For example, `$(\int_a^b (f(x)\cdot g(x)) dx)$` produces such math equation ($(\int_a^b (f(x)\cdot g(x))dx)$, where all brackets are of the same size.
   Compare it to `$\left(\int_a^b \left(f(x)\cdot g(x)\right) dx\right)$` which produces $\left(\int_a^b \left(f(x)\cdot g(x)\right) dx\right)$.

   Cases environment allows to create constructions of the forms:

```
\[
 u(x) =
  \begin{cases}
   \exp{x} & \text{if } x \geq 0 \\
   1       & \text{if } x < 0
  \end{cases}
\]
```

$$u(x) = \begin{cases} \exp x & \text{if } x \geq 0 \\ 1 & \text{if } x < 0 \end{cases}$$

   Never use **`eqnarray`** instead use **`align`**, see for example

```
equation:
\begin{equation*}
  z_0 = d = 0
\end{equation*}
\begin{equation*}
  z_{n+1} = z_n^2+c
\end{equation*}
align:
\begin{align*}
  z_0 &= d = 0 \\
  z_{n+1} &= z_n^2+c
\end{align*}
eqnarray:
\begin{eqnarray*}
  z_0 &=& d = 0 \\
  z_{n+1} &=& z_n^2+c
\end{eqnarray*}
```

equation:

$$z_0 = d = 0$$
$$z_{n+1} = z_n^2 + c$$

align:

$$z_0 = d = 0$$
$$z_{n+1} = z_n^2 + c$$

eqnarray:

$$\begin{aligned} z_0 &= d = 0 \\ z_{n+1} &= z_n^2 + c \end{aligned}$$

10. **Algorithms**

```
\begin{algorithmic}
\REQUIRE $n \geq 0 \vee x \neq 0$
\ENSURE $y = x^n$
\STATE $y \Leftarrow 1$
\IF{$n < 0$}
\STATE $X \Leftarrow 1 / x$
\STATE $N \Leftarrow -n$
\ELSE
\STATE $X \Leftarrow x$
\STATE $N \Leftarrow n$
\ENDIF
\WHILE{$N \neq 0$}
\IF{$N$ is even}
\STATE $X \Leftarrow X \times X$
\STATE $N \Leftarrow N / 2$
\ELSE[$N$ is odd]
\STATE $y \Leftarrow y \times X$
\STATE $N \Leftarrow N - 1$
\ENDIF
\ENDWHILE
\end{algorithmic}
```

Require: $n \geq 0 \vee x \neq 0$
Ensure: $y = x^n$
  $y \Leftarrow 1$
  if $n < 0$ **then**
    $X \Leftarrow 1/x$
    $N \Leftarrow -n$
  else
    $X \Leftarrow x$
    $N \Leftarrow n$
  end if
  while $N \neq 0$ **do**
    if N is even **then**
      $X \Leftarrow X \times X$
      $N \Leftarrow N/2$
    else $\{N$ is odd$\}$
      $y \Leftarrow y \times X$
      $N \Leftarrow N - 1$
    end if
  end while

11. **Page margins.**

Page placement is defined by the following values:

```
\oddsidemargin = 31pt
```

```
\topmargin = 20pt
\headheight = 12pt
\headsep = 25pt
\textheight = 592pt
\textwidth = 390pt
\marginparsep = 10pt
\marginparwidth = 35pt
\footskip = 30pt
\marginparpush = 7pt
\hoffset = 0pt
\voffset = 0pt
\paperwidth = 597pt
\paperheight = 845pt
```

Each command may be edited as `\setlength{\textwidth}{300pt}` or using `geometry` package.

`\usepackage[top=tlength, bottom=blength, left=llength, right=rlength]{geometry}`

For example:
`\usepackage[margin=1in, paperwidth=5.5in, paperheight=8.5in]{geometry}`

Pages are filled automatically by latex compiler, however user may control line breaks, vertical spaces, horizontal spaces and page breaks.

`\\`	Breaks line at the point.
`\\[10pt]`	Breaks line at the point and add vertical skip 10pt.
`\newpage`	Breaks page at the point, and add the next text on the next page.
`\vspace{30pt}`	If possible adds vertical space of 30pt length.
`\vspace*{30pt}`	Strictly adds vertical space of 30pt length .
`\hspace{1in}`	If possible adds horizontal space of 1 inch length.
`\hspace{1in}`	Strictly adds horizontal space of 1 inch length.
`\clearpage`	Stops current page, flushing everything onto it.
`\quad`	Horizontal tab (11pt).
`\qquad`	Horizontal double tab.
`\!`	Negative horizontal space.
`~`	Make sure that space between words is not separted by line break. E.g. `author~\cite{book1}`

12. **Graphics**.

Latex compiler has two graphic versions. The first one is for `PS` file processing with command **`latex`** (and `DVI`); it accepts one format (EPS files). The second one is `PDF` file which is produced using **`pdflatex`** and it accepts different set of formats (`JPG, PNG, PDF`).

There are a various set of converters from one format to another: **`epstopdf, ps2pdf, fig2dev, fig2ps, fig2ps, ....`**

The main command which includes graphics is `\includegraphics[set of attributes]{graphical_file}`. There is no need to put file extension in `graphical_file`, during compilation of `DVI/PS` files with extension eps/ps will be searched and during compilation using `pdflatex` files with extensions pdf/jpg/png. The use of extension is needed only when you want to remove ambiguity (i.e., there is file with both pdf and png extensions).

The set of options is as following:

width	Defines width of the image.
height	Defines height of the image.
keepaspectratio	Should the image keep the ratio during scaling (including when width and height does not correspond to the ratio)?
scale	Defines the retio of scaling.
angle	Defines the angle to rotate image.
trim	Defines the sizes of borders in order to trim image accordingly.
clip	Should be true for trim option to use.
page	Define page explicitly from multipage pdf files.

Whenever graphic files are not in current directory, the directory to take as an input may be defined using `\graphicspath` parameter, e.g., `\graphicspath{{./images/}}`.

To accompany images with caption, as well as produce additional manipulations figure environment may be used:

```
\begin{figure}[ht]
\centering
\includegraphics[width=0.8\textwidth]{graph}
\caption{Caption for the graph: here we see an image}
\label{fig:graph}
\end{figure}
```

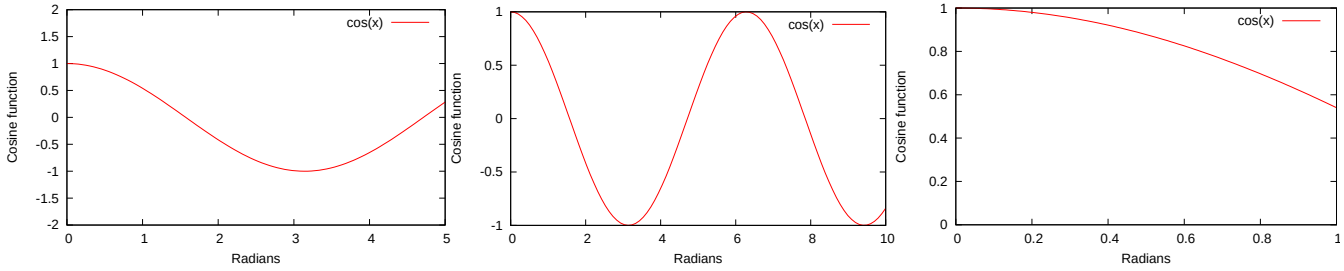
Label should go after caption and later in the text be referenced by `\ref{fig:graph}` command.

As a substitute for figure image **wrapfig** may be used (first `\usepackage{wrapfig}`, then `\begin{wrapfigure}[lineheight]{ali` whenever the text should go around an image (or for side capture use package `sidecap` and environment `SCfigure`).

13. **subfig** environment.

    In research articles a lot is needed to put a set of figures on the same line, to help with it **subfig** environment may help a lot.

```
\begin{figure}
  \centering
  \subfloat[Plot A]{\label{fig:pa}\includegraphics[width=0.3\textwidth]{graph1}}
  \subfloat[Plot B]{\label{fig:pb}\includegraphics[width=0.3\textwidth]{graph2}}
  \subfloat[Plot C]{\label{fig:pc}\includegraphics[width=0.3\textwidth]{graph3}}
  \caption{Pictures of animals}
  \label{fig:animals}
\end{figure}
```



(a) Plot A (b) Plot B (c) Plot C

Figure 1: Pictures of animals

Notice that for convinience it good to put in `\label` specifier for the type of label, e.g., for figures it is **fig:**, for tables it is **tbl:**, for sections it is **sec:** and for equations it is **eq:**.

14. **picture** environment.

    Picture environment provides one of simplest ways to draw. It uses command **put, line, vector, circle, oval,...**. Everything

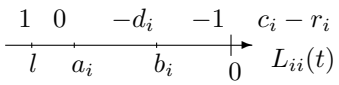
```
\begin{picture}(100,15)
  \put(0,8){\vector(1,0){93}}
  \put(26,7){\line(0,1){2}}
  \put(57,7){\line(0,1){2}}
\put(56,-2){\textbf{$b_i$}}
\put(25,-2){\textbf{$a_i$}}

  \put(85,4){\line(0,1){8}}
  \put(84,-5){$0$}

\put(100,0){$L_{ii}(t)$}
\put(95,15){$c_i-r_i$}

\put(10,7){\line(0,1){2}}
\put(9,-2){$1$}

\put(5,15){\textbf{$1$}}
\put(18,15){$0$}
\put(40,15){$-d_i$}
\put(70,15){$-1$}
\end{picture}
```

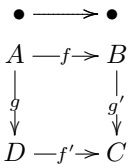


gnuplot may produce picture environment directly.

As an alternative for **picture** environment may be **xy** package (use `\usepackage[all]{xy}` to add it). It may produce such things as

```
\begin{displaymath}
    \xymatrix{ \bullet \ar[r] \ar@{.>}[r] & \bullet }
\end{displaymath}

\begin{displaymath}
    \xymatrix{
        A \ar[r]|f \ar[d]|g & B \ar[d]|{g'} \\
        D \ar[r]|{f'}       & C }
\end{displaymath}
```

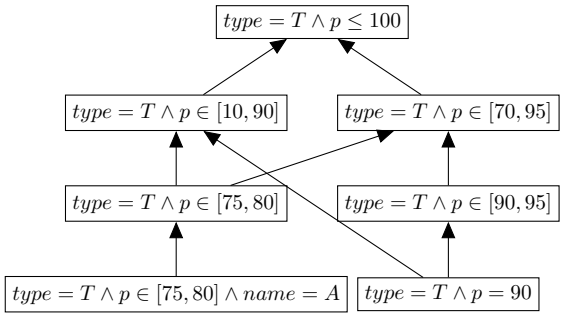


15. **TikZ/PGF**.

To use it **tikz** package is required (**\usepackage{tikz}**). However, it is not enough, it requires to define set of sub-libraries needed. Those includes but not limited to **tree, automata, shapes, arrows, background, ...**.[2] For the following example we use **\usetikzlibrary{arrows,shapes,decorations.pathmorphing,backgrounds,fit}** just after usepackage declaration of tikz.

```
\begin{tikzpicture}[scale=1.2]
  \node (root) at (0,0) [rectangle,draw,scale=0.8] {$type = T \land p \leq 100$};
  \node (l1n1) at (-1.5,-1) [rectangle,draw,scale=0.8] {$type = T \land p \in [10,90]$};
  \node (l1n2) at (1.5,-1) [rectangle,draw,scale=0.8] {$type = T \land p \in [70,95]$};
  \node (l2n1) at (-1.5,-2) [rectangle,draw,scale=0.8] {$type = T \land p \in [75,80]$};
  \node (l2n2) at (1.5,-2) [rectangle,draw,scale=0.8] {$type = T \land p \in [90,95]$};
  \node (l3n1) at (-1.5,-3) [rectangle,draw,scale=0.8] {$type = T \land p \in [75,80] \land name = A$};
  \node (l3n2) at (1.5,-3) [rectangle,draw,scale=0.8] {$type = T \land p = 90$};

  \begin{scope}[>=triangle 45]
    \draw [->] (l1n1) -- (root);
    \draw [->] (l1n2) -- (root);
    \draw [->] (l2n1) -- (l1n1);
    \draw [->] (l2n1) -- (l1n2);
    \draw [->] (l2n2) -- (l1n2);
    \draw [->] (l3n1) -- (l2n1);
    \draw [->] (l3n2) -- (l2n2);
    \draw [->] (l3n2) -- (l1n1);
  \end{scope}
\end{tikzpicture}
```



TikZ/PGF is one additional useful package may be worth to use.

[2] A comprehensive list of available options can be found in **mirror.ctan.org/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf**.

16. **Beamer**

To produce a presentation it is enough to add `\documentclass{beamer}` in the top of the document. After that a presentation will be produced as output (primary using pdflatex).

The structure of the document is

```
\begin{document}
  \begin{frame}
    \frametitle{This is the first slide}
    %Content goes here
  \end{frame}
  \begin{frame}
    \frametitle{This is the second slide}
    \framesubtitle{A bit more information about this}
    %More content goes here
  \end{frame}

\end{document}
```

There is also **`title, subtitle, author, date, subject`** may be specified. Inside of each frame standard latex commands may be used, with little exception.

To produce pauses during output use `\pause` command. Then everytime it will be stopped and additional mouse click is required to get to continue.
Beamer allows also do define an order of the text

```
\begin{itemize}
  \item This one is always shown
  \item<1-> The first time
  \item<2-> The second time
  \item<1-> Also the first time
  \only<1-> This one is shown at the first time, but it will hide soon.
\end{itemize}
```

To produce multiple columns use `columns` environment, e.g.,

```
\begin{frame}
    \begin{columns}[c] % the "c" option specifies center vertical alignment
    \column{.5\textwidth} % column designated by a command
     Contents of the first column
    \column{.5\textwidth}
     Contents split \\ into two lines
    \end{columns}
\end{frame}
```

and blocks are available through

```
\begin{frame}
   \begin{block}{This is a Block}
      This is important information
   \end{block}

   \begin{alertblock}{This is an Alert block}
   This is an important alert
   \end{alertblock}

   \begin{exampleblock}{This is an Example block}
   This is an example
   \end{exampleblock}

\end{frame}
```

The main point here is that Aalto provides own templates (which are still partly in preparations but available throughout a request). Available in Aalto Inside `https://wiki.aalto.fi/display/aaltolatex/Aalto+University+Beamer+themes`.

The raw a little bit, and should be ready before the end of the year. I use the older version of the aalto beamers class (which available for you at `http://users.tkk.fi/u/laser/crash/aalto.tgz`). As soon as newer aalto slides will be available I will update this part.

The beamer presentation example (Lecture 1 of this course) is available by address `http://users.tkk.fi/u/laser/crash/crash_lecture_1.tex`.