LaTeX course at ICT School — Tutorial 1

Alexandre Labrosse
KTH *Royal Institute of Technology*

2012, Jan. 24th

To start the tutorial you must have downloaded and installed a full LaTeX environment, including a distribution and a text editor. Difficulty increases along questions. From question 8, you can just jump to problems and come back later to go further and deeper.

1 First document

Question 1

We've seen the `minimal` class during lecture. Why is it a good candidate to test that your installation is working? Write a "Hello world!" document (or whatever you want) using that document class and compile it with pdflatex.

Solution: The `minimal` class is present in every distribution of LaTeX and does not need any other package to run. This way you can check that your distribution is properly installed by solely compiling a minimal document. When you encounter a problem with LaTeX, you are strongly advised to build a minimal example of your problem before posting a question onto a message board or a forum. Why not starting by a `minimal` document class?

Question 2

In the directory where you saved your source file you should see now multiple other files. What each is made for?

Solution:

file.aux transports the information from one run of the compiler to the other. Such information are mainly references.

file.log is intended for human use. One can read what happens during the compilation process.

Question 3

Add a sentence in your document with accented characters. Include the package `inputenc` with the appropriate option and check that you get what expected when compiling.

Solution: I hope you have found your proper encoding. If not, don't stay stuck and ask someone, either *viva voce*, either through KTH Social.

Question 4

Keeping the minimal class, add a title to your document. Does it work? Why? Correct the source file so that it compiles properly.

Solution: As the name suggest, the minimal class is minimal. There is no title mechanism included, neither sectioning mechanisms. The solution here is to use another document class that provides it (`article`, `report`, `book`, etc.). You may also write a set of commands that provide same functionality using standard TeX.

Question 5

Add `sections`, `subsections`, `paragraphs`, and so on to see all available sectioning commands.

Solution: Was it great?

Question 6

Add a table of content, compile only one time and see the result. Is it what you might have expected? Why? (Hint: take a look at the auxiliary files in the directory.)

Solution: After the first compilation, you observe that the table of contents is empty, except for the title. But there is now a file called `file.toc` in the working directory that contains `\contentsline` commands. In fact, the compiler cannot output the table of contents at the beginning of the document before having read the whole source file to look for sectioning commands. A second compilation now do the trick.

Question 7

Find the documentation of the `lipsum` package. Read it and before using the package, understand what you think it is meant for. How many command does it define? Then produce a four or five pages document with sectioning commands.

Solution: The `lipsum` package defines four commands. In fact, if you look at the source code of the package, you can see that more than 160 commands are defined. Most of them can't be used in a document directly because of the protecting `@`. It is meant for producing "dummy" text to fill a document. This is useful when testing or demonstrating. The starred version of the `\lipsum` command allow to host this dummy text inside the body of a paragraph. The optional argument allows to select one of the 150 paragraphs of the text.

Question 8

Find some informations about the `\pagestyle`, `\thispagestyle` and `\pagenumbering` command. Use it to number pages like this: i, ii, iii, iv, v… Then suppress only first page numbering, then for all pages. (Hint: keep in mind that `\pagestyle` is effective when starting a new page and then on and `\thispagestyle` only on the current page.)

Solution: For the roman numbering (i, ii, iii, iv, v, etc.), just place the command `\pagenumbering{roman}` either in the preamble, either in the body. This could be used to number differently the "front matter" of your document from the "main matter" (see the documentation of the `book` package).

Using `\thispagestyle{empty}` allows to suppress page numbering on the current page, that is to say the page currently being built by the compiler. To suppress page numbering on the first page, use it just after `\begin{document}`

To suppress all numbering, use `\pagestyle{empty}`. Beware `\begin{document}` that starts a new page without considering current page style, so you need both `\pagestyle` and `\thispagestyle`.

Question 9

For the sake of completeness, take a look at `fancyhdr` package and try to reproduce the headers and footers of this tutorial—including the difference between first page and next ones. The copyright may be very difficult to place, so forget about it or take a look at the `minipage` environment.

Solution:

```
\newcommand\copyfoot{%
  \textcopyright~Alexandre~{\scshape Labrosse}~2012 This
  document may be modified or distributed under the terms of the GNU Free
  Documentation License.}

\usepackage{fancyhdr}
\fancypagestyle{empty}{%
  \fancyhf{}
  \renewcommand\headrulewidth{0pt}
  \renewcommand\footrulewidth{0pt}
  \fancyfoot[L]{%
    \begin{minipage}{10cm}
      \flushleft\footnotesize\copyfoot
    \end{minipage}}
  \fancyfoot[R]{\thepage}}
\fancypagestyle{plain}{%
  \fancyhf{}
  \renewcommand\headrulewidth{0pt}
  \renewcommand\footrulewidth{0pt}
  \fancyhead[L]{\scshape \LaTeX{} course at ICT School}
  \fancyhead[R]{\itshape Tutorial~1}
  \fancyfoot[L]{%
    \begin{minipage}{10cm}
      \flushleft\footnotesize\copyfoot
    \end{minipage}}
  \fancyfoot[R]{\thepage}}

  ...

\begin{document}
  \pagestyle{plain}
  \maketitle\thispagestyle{empty}

  ...
```

Question 10

Now you can just play with the following:

1. change the document class (`article`, `report`, `book`), add eventually some `\part` commands to see the effect,
2. change paragraph behaviour (indentation, vertical skip between lines or between paragraphs)
3. change options of the document class according to the slides (font size, draft, page size, etc.)
4. when using the `twoside` option, try `\clearpage` and `\cleardoublepage` commands.

2 Problems

Question 1

What is the output of the following line? (Try to guess before testing.) Propose three solutions to get the intended behaviour. Compare each.

```
I love the \LaTeX typesetting system.
```

Solution: The output is: I love the LaTeXtypesetting system.

In fact, the `\LaTeX` command gathers all coming white spaces. We have three solutions to suppress this behaviour:

1. grouping around the `\LaTeX` command to limit its scope:
   ```
   I love the {\LaTeX} typesetting system.
   ```
   I love the LaTeX typesetting system.
2. giving the command an empty argument to stop eating white spaces:
   ```
   I love the \LaTeX{} typesetting system.
   ```
   I love the LaTeX typesetting system.
3. adding a forced space after the command:
   ```
   I love the \LaTeX\ typesetting system.
   ```
   I love the LaTeX typesetting system.

All solutions are equally good, except for use in the definition of a new command where you should avoid the last one, which may force an unwanted space.

Question 2

Try to reproduce the title of this tutorial, like this: (Size does not matter, but centering does.)

LaTeX course at ICT School — Tutorial 1

Solution:

```
\begin{center}
  {\LaTeX{} course at ICT School --- Tutorial~1}
\end{center}
```

Question 3

In many country, quoting or using a word from a foreign language is made in italics. For that purpose, we want to define a command called `\foreign`. Why? Propose such a command definition. Test it and criticize your solution.

Solution: Using a pure LaTeX syntax, one might think out this solution:

```
\newcommand\foreign[1]{\textit{#1}}
```

A problem may occur if the command is used in an already italicized environment. If the behaviour of the `\emph` command suits you (for now), you can define:

```
\newcommand\foreign[1]{\emph{#1}}
```

or a better solution that makes the `\foreign` command independent of later redefinition of `\emph`:

```
\let\foreign\emph
\renewcommand\emph[1]{\textbf{#1}}
```

A last solution is to define your own command without using high level commands. The example provides small caps when the surrounding text is italicized, italics otherwise:

```
\usepackage{ifthen}

\makeatletter
  \newcommand\ifit[2]{\ifthenelse{\equal{\f@shape}{it}}{#1}{#2}}
\makeatother

\newcommand\foreign[1]{\ifit{\textsc{#1}}{\textit{#1}}}
```