The alttex package

Arno L. Trautmann*

Version 0.a.1 December 27, 2008

This is the package `alttex` which will try to give an experimental new way to write XeLaTeX code. So far it is mostly done with very dirty code and actually it's a collection of things that come into my mind during boring lectures. Maybe someone will have fun with the following code fragments.

Contents

1 introduction **2**

2 Textmode **4**
2.1 no escape 4
2.2 tables 4

3 Math stuff **5**
3.1 braces 5

4 itemize and similar things **6**
4.1 itemize with a single character 6

*arno.trautmann@gmx.de

1 introduction

The problem I have with LaTeX[1] is the antique way of typing. Because most people still use a hopelessly outdated keyboard layout („qwerty“ or slightly adapted versions of that), LaTeX doesn't make use of some cool features. I'm not talking about writing chinese or arabic text! Maybe this example will make the idea clear:

In standard LaTeX, one has to write

```
This is the normal text, then comes the itemization:
\begin{itemize}
  \item text for first item
  \item \begin{itemize}
        \item this is an item inside an item…
       \item[$\Rightarrow$] Here an item with a formula: $\int_a^b x^2 dx$
        \end{itemize}
  \item and the outer itemize goes on…
\end{itemize}
```

Using this package and having a superior keyboard layout[2], you can simply write:[3]

```
This is the normal text, then comes the itemization:

• text for first item
•

  ▸ this is an item inside an item
  ▸[⇒] Here an item with a formula: $∫_a^b x² dx$

• and the outer itemize goes on…

And your normal text goes on…
```

Well, actually I'm lying now because this is not fully implemented so far. But it's the aim of this package to provide this – besides many, many other funny and cool things. The aim is to offer a more „wysiwyg“ way, without loosing anything of logical markup. One still can re\define the • if he doesn't like the way his items look. I have just started to write the package, there will be much more stuff here in the future.

Ok, enough blahblah, now comes the code. We begin with the uninteresting preamble stuff:

```
1 \ProvidesPackage{alttex}
```

[1] I'll write LaTeX instead of XeLaTeX—saves me two keystrokes. Most of the code below *only* works with XeLaTeX. If you need support for [utf8]inputenc or LuaLaTeX, please contact the author.

[2] E. g. the ergonomic layout NEO.

[3] The lmodern font I'm using here does not have the symbol for the inner item , so we change to DejaVu Sans Mono here.

```

\RequirePackage{amsmath}
```

`\usepackage` Now, this is the first highlight. It is an extremely simple and stupid approach to load missing packages on-the-fly, just like MikTEX does. We re`\def`ine the `\usepackage` and hope, it works. Only working with texlive! If you're using MikTEX, put a

```
\let\usepacke\oldpackage
```

into your preamble, *directly* after loading `alttex`. If this does not work, delete the following lines from your `alttex.sty`.

```
\let\oldpackage\usepackage
\def\usepackage#1{
  \AtBeginDocument{hallo}
  \IfFileExists{#1.sty}{
    \oldpackage{#1}
  }{
    \immediate\write18{tlmgr install #1}
  }
}
```

So far, this code seems to be a bit buggy, but it should work anyhow.

Now load some nice packages and testing wether you're running XeLaTeX or not.

```
\RequirePackage{exscale}
\RequirePackage{ifxetex}
\ifxetex
\typeout{Loading XeTeX, everything's fine.}
\else
  \typeout{^^J%
  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J%
  ! This package can only be compiled with XeLaTeX.^^J%
  ! pdfLaTeX cannot handle unicode the way it is used here.^^J%
  ! If you want to have support for [utf8]inputenc, please contact the author.^^J%
  ! If you want to use LuaLaTeX, give it a try:^^J%
  ! comment out the lines 32,33,35-43.^^J%
  ! Please e-mail me the result of your experiences!^^J%
  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J%
  }
  \errmessage{No XeLaTeX, no alttex. See the log for more information.}
  \endinput
\fi

```

We need `exscale` to write really big formulae, and `ifxetex` to check wether one uses the correct engine.

2 Textmode

2.1 no escape

`\noescape` You want to write plain text. Maybe you're annoyed by always escaping characters like _ # & { } $ ~ and so on. `\noescape` allows you to never escape anything—except the \, which still might be used for `\textit{}` or so. Or maybe not… because the { } are not escaped. Have to think about this one. Maybe the \ will be redefined to define { } by itself.

```
\def\noescape{
  \catcode`\_= 11%
  \catcode`\^= 11%
  \catcode`\#= 11%
  \catcode`\&= 11%
  %\catcode`\{= 11%
  %\catcode`\}= 11%
  \catcode`\$= 11%
  \catcode`\~= 11%
 \makeatletter% I just noticed this is not necessary… but I'll leave it for some strange \thin@gs or
  \catcode`\%= 11
}
```

l

`\oldescape` Of course this has to be reset when doing anything like formula, tabular etc. Maybe I will be able to change the behaviour automatically. Thas idea has been inspired by a discussion on the ConTEXt mailinglist.

```
\def\oldescape{
  \catcode`\%= 14%
  \catcode`\_= 8%
  \catcode`\^= 7%
  \catcode`\#= 6%
  \catcode`\&= 4%
  %\catcode`\{= 1%
  %\catcode`\}= 2%
  \catcode`\$= 3%
  \catcode`\~= 13%
  \makeatother%
}
```

2.2 tables

The way one has to type extensive tables is quite complex – and the resulting code is often not really readable. I don't have an idea how to change this, but I'm thinking about it. Just a reminder to myself… mail me any suggestions for this!

3 Math stuff

3.1 braces

\newbraces
\oldbraces

Now this is something most LaTeX-beginners don't recognize and wonder why the formula looks so ugly: The braces () do not fit to the hight of the formula. This can be achieved by putting `\left` and `\right` in front of the braces. But actually, this is annoying! In almost any case you want this behaviour, so this should be the standard. So we redefine the way braces are handled. With `\newbraces` the ( ) always fit. If you prefer the normal LaTeX way, use `\oldbraces` to reset everything. This new behaviour should be extended to other characters like `| [ { <` and so on. Maybe in version 0.0.1…

I would have never been able to implement this without the help of the mailinglist members of `tex-d-l@listserv.dfn.de`!

Maybe one could "temporarily hardcode" the hight of [and then use this…

The redefinition of `\mathstrut` is necessary when using amsmath (you will use amsmath when typesetting formulae, won't you?), because the hight of formulae is determinated by the hight of a brace. But using ( ) as `\active` characters, we need another brace here. So we take `[`. This will probably also change. But the code is working fine for ( ).

```
\makeatletter
\def\resetMathstrut@{%
    \setbox\z@\hbox{%
      \mathchardef\@tempa\mathcode`\[\relax
      \def\@tempb##1"##2##3{\the\textfont"##3\char"}%
      \expandafter\@tempb\meaning\@tempa \relax
    }%
  \ht\Mathstrutbox@\ht\z@ \dp\Mathstrutbox@\dp\z@
}
\makeatother

\edef\oldbraces{
  \mathcode`(\the\mathcode`(
  \mathcode`)\the\mathcode`)
}
\begingroup
  \catcode`(\active \xdef({\left\string(}
  \catcode`)\active \xdef){\right\string)}
\endgroup

\def\newbraces{
  \mathcode`("8000
  \mathcode`)"8000
}
```

hugedisplaymath

Sometimes, especially in presentations, you might need an really big formula. Imagine two hours of struggle with transformations—and finally there is the beautiful formula. Now you can say

```
\begin{hugedisplaymath} E = mc^2 \end{hugedisplaymath}
```

There should be several steps of size, maybe.

```
\def\hugedisplaymath{
  \makeatletter
    \makeatother
  \Huge
    \begin{equation*}
}
\def\endhugedisplaymath{
  \end{equation*}
}
```

4 itemize and similar things

4.1 itemize with a single character

• instead of \item

Here we use an active character (mostly a unicode character bullet •) for the whole construct. And another one for nested itemizations (like a triangular bullet ‣).

This does—guess it—not work correctly so far. I'm trying to find a tricky way so that the ending character is not necessary any more. So far one has to end an itemize with something like an - (em-dash). There will also be a possibility to change the characters responsible for the whole action.[4]

\newitemi
\newitemii

The following ugly peace of code is writen by me, defining the conditional insertion of the \begin{itemize}. This will be assigned to an active character using \makeitemi and \makeitemii, respectively.

```
\def\notinside{}
\def\inside{}
\let\insideitemizei\outside
\let\insideitemizeii\outside

\def\newitemi{
  \ifx\insideitemizei\inside
    \expandafter\item%
  \else
    \begin{itemize}
      \global\let\insideitemizei\inside
      \expandafter\item%
  \fi
}

\def\newitemii{
  \ifx\insideitemizeii\inside
    \expandafter\item%
  \else
    \begin{itemize}
```

[4] The triangular bullet sign does not appear here – the font is lacking it…

```
      \global\let\insideitemizeii\inside
      \expandafter\item%
  \fi
}
```

Ok, the following code is stolen from the `shortvrb` package, and I don't understand anything of it. But I keep on trying… nevertheless, it's working fine, as far as I can see.

`\makeitemi` `\makeitemii`

With this macro, you can define the character you want to use for first-level itemize. (Guess the sense of `\makeitemii`…) Default ist • for first-level and for second-level. Maybe this will be extended till fourth level. More doesn't seem to make any sense.

```
%
\makeatletter
\def\makeitemi#1{%
  \expandafter\ifx\csname cc\string#1\endcsname\relax
    \add@special{#1}%
    \expandafter
    \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
    \begingroup
      \catcode`\~\active  \lccode`\~`#1%
      \lowercase{%
      \global\expandafter\let
         \csname ac\string#1\endcsname~%
      \expandafter\gdef\expandafter~\expandafter{\newitemi}}%
    \endgroup
    \global\catcode`#1\active
  \else
  \fi
}

\def\makeitemii#1{%
  \expandafter\ifx\csname cc\string#1\endcsname\relax
    \add@special{#1}%
    \expandafter
    \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
    \begingroup
      \catcode`\~\active  \lccode`\~`#1%
      \lowercase{%
      \global\expandafter\let
         \csname ac\string#1\endcsname~%
      \expandafter\gdef\expandafter~\expandafter{\newitemii}}%
    \endgroup
    \global\catcode`#1\active
  \else
  \fi
}
```

Now there are the two helperfunctions – no guess what they are really doing.

```
\def\add@special#1{%
  \rem@special{#1}%
  \expandafter\gdef\expandafter\dospecials\expandafter
{\dospecials \do #1}%
  \expandafter\gdef\expandafter\@sanitize\expandafter
{\@sanitize \@makeother #1}}
\def\rem@special#1{%
  \def\do##1{%
    \ifnum`#1=`##1 \else \noexpand\do\noexpand##1\fi}%
  \xdef\dospecials{\dospecials}%
  \begingroup
    \def\@makeother##1{%
      \ifnum`#1=`##1 \else \noexpand\@makeother\noexpand##1\fi}%
    \xdef\@sanitize{\@sanitize}%
  \endgroup}
\makeatother
```

1 2 3 And we do just the same stuff with `\enumerate`. But here we take the 1 as first level item, the 2 as second level etc. This may be confusing some way, but just try it.

For the implementation: copy-pasted the code above, nothing interesting so far.

```
\let\insideenumi\outside
\let\insideenumii\outside

\def\newenumi{
  \ifx\insideenumi\inside
    \expandafter\item%
  \else
    \begin{enumerate}
      \global\let\insideenumi\inside
      \expandafter\item%
  \fi
}

\def\newenumii{
  \ifx\insideenumii\inside
    \expandafter\item%
  \else
    \begin{enumerate}
      \global\let\insideenumii\inside
      \expandafter\item%
  \fi
}

```

We use the same methods as above, still not understanding, what they are doing. Just changing two lines of code and hoping, everything will be fine.

```
\makeatletter
\def\makeenumi#1{%
```

```
  \expandafter\ifx\csname cc\string#1\endcsname\relax
    \add@special{#1}%
    \expandafter
    \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
    \begingroup
      \catcode`\~\active  \lccode`\~`#1%
      \lowercase{%
      \global\expandafter\let
         \csname ac\string#1\endcsname~%
      \expandafter\gdef\expandafter~\expandafter{\newenumi}}%
    \endgroup
    \global\catcode`#1\active
  \else
  \fi
}

\def\makeenumii#1{%
  \expandafter\ifx\csname cc\string#1\endcsname\relax
    \add@special{#1}%
    \expandafter
    \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
    \begingroup
      \catcode`\~\active  \lccode`\~`#1%
      \lowercase{%
      \global\expandafter\let
         \csname ac\string#1\endcsname~%
      \expandafter\gdef\expandafter~\expandafter{\newenumii}}%
    \endgroup
    \global\catcode`#1\active
  \else
  \fi
}
\makeatother

```

Finally, we set the default characters for the items and enumerations:

```
\makeitemi•
\makeitemii
\makeenumi¹
\makeenumii²
```

Tabelle, blindtext

And that's it. Happy altTEXing!