The alttex package

Arno L. Trautmann*

Version 0.a, 2008 December 17

This is the package `alttex` which will try to give an experimental new way to write LaTeXcode. So far it is mostly done with very dirty and actually it's a collection of things I think about in boring lectures. Maybe someone will have fun with the following code.

Contents

1 introduction

The problem I have with LaTeX is the antique way of typing. Because most people still use a hopeless outdated keyboard layout („qertzy“ or slightly adapted versions of that), LaTeXdoesn't make use of some cool features.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 another item
        \end{itemize}
  \item and the outer itemize goes on…
```

*arno.trautmann@gmx.de

```
\end{itemize}
```

Using this package, you can simply write[1]

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

• text for first itemi
•

    this is an item inside an item
    another item

• and the outer itemize goes on…

And your normal text goes on…
```

Well, actually I'm lying now. But it's the aim of this package to provide this – besides many, many other funny and cool things. I have just started it, there will be much more stuff here.

```
\ProvidesPackage{alttex}

\RequirePackage{exscale} % For huge math

```

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%
  \catcode`\%= 11
}
```

[1] The lmodern font I'm using here does not have the symbol for the inner item…

\oldescape Of course this has to be reset when doing anything like formula, tabular etc.

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

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!

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

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 not—guess it— 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 `-`. There will also be a possibility to change the characters responsible for the whole action.

• instead of `\item`

```
\newcounter{itemi}
\setcounter{itemi}{0}

\catcode`\•=\active
\catcode`\ =\active

\def•{
  \ifvmode \ifnum \theitemi = 0 %außerhalb einer itemize
    \begin{itemize}\setcounter{itemi}{1}
      \item
```

```
    \else
      \item %zum Fortsetzen einer Liste
    \fi
  \else
    \item % normales item innerhalb einer Liste
  \fi
}

\def {
  \ifvmode
    \begin{itemize}
      \item
  \else
    \item
  \fi
}

\def\-{\end{itemize}}


\newcounter{insideitemize}
\setcounter{insideitemize}{0}
\newcounter{insideitem}
\setcounter{insideitem}{0}

\catcode`\•=\active

\iffalse
\def•{
 \ifnum \theinsideitemize = 0 % Außerhalb einer itemize-Umgebung  initialisieren
    \begin{itemize}
      \iffalse
      \catcode`\^^M=\active
      \def^^M{\myeol} \catcode`\^^M=5%
      \fi
      \setcounter{insideitemize}{1} % Nun innerhalb einer itemize
      \setcounter{insideitem}{1}% und innerhalb eines Items
      \expandafter\item
  \else
    \makeatletter
      \ifthenelse{\boolean{@inlabel}}{%
    \makeatother
      tach
}{%
    \makeatother
      \setcounter{insideitem}{1}% innerhalb eines items
      \expandafter\item
}
  \fi
}
```

```
\fi

\def\myeol{%
  \ifnum \theinsideitem = 0%
    \end{itemize}%
    \catcode`\^^M=5%
    \setcounter{insideitem}{0}%
  \else%
    \setcounter{insideitem}{0}%
\fi%
}
```