The alttex package

Arno L. Trautmann[*]

Version 0.a.2 January 9, 2009

This is the package `alttex` which will try to give an experimental new way to write XƎLaTeX[1] 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

[*]arno.trautmann@gmx.de
[1]If you don't know about XƎLaTeX, see the appendix.4.2

1 introduction

The problem I have with LaTeX[2] 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 Because most people still use a hopelessly outdated keyboard layout (»qwerty« or slightly adapted versions of that), LaTeX doesn't make use of some cool features.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[3], you can simply write:[4]

```
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`\def`ine 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:

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

[3]E. g. the ergonomic layout Neo: `http://neo-layout.org/`

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

```
 1 \ProvidesPackage{alttex}
 2
 3 \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`.

```
 4 \let\oldpackage\usepackage
 5 \def\usepackage#1{
 6   \IfFileExists{#1.sty}{
 7     \oldpackage{#1}
 8   }{
 9     \immediate\write18{tlmgr install #1}
10   }
11 }
```

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 XƎLATEX or not.

```
12 \RequirePackage{exscale}
13 \RequirePackage{ifxetex}
14 \RequirePackage{hhline}
15 \ifxetex
16 \typeout{Loading XeTeX, everything's fine.}
17 \else
18   \typeout{^^J%
19   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J%
20   ! This package can only be compiled with XeLaTeX.^^J%
21   ! pdfLaTeX cannot handle unicode the way it is used here.^^J%
22   ! If you want to have support for [utf8]inputenc, please contact the au-
   thor.^^J%
23   ! If you want to use LuaLaTeX, give it a try:^^J%
24   ! comment out the lines 32,33,35–43.^^J%
25   ! Please e-mail me the result of your experiences!^^J%
26   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J%
27   }
28   \errmessage{No XeLaTeX, no alttex. See the log for more information.}
29   \endinput
30 \fi
31
```

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

The `\makeatletter` is not necessary. But it fitted into this line, so I will leave it here.

\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. This 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 tabular

The way one has to type extensive tabulars is quite complex – and the resulting code is often not easy to read. I don't have good ideas how to change this, but I'm thinking about it. Mail me any suggestions for this!

I will try to implement cool stuff from the hhline-package.

This will be the first attempt to make tabulars easier: Mostly you want an `\hline` after an `\\`. So let's try something like:

\§ for \\\hhline Type `\–` (an en-dash) at the end of a line, and you get an `\hhline`. Type `\=` to get a double line

```
\def\–{\hhline}
\def\={\hhline}
```

This is shurely not a good symbol for this purpose, but I don't have a better idea so far. At least it's a ”bar“, so one can guess what it should do.

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 some later version.

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

The newbraces does *not* work at the moment!

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 ( ).

Maybe one could ”temporarily hardcode“ the hight of [and then use this...

```
\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

{\catcode`(\active \xdef({\left\string(}}
{\catcode`)\active \xdef){\right\string)}}

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

\edef\oldbraces{
  \mathcode`(\the\mathcode`(
  \mathcode`)\the\mathcode`)
}
```

3.2 huge display math

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*}
}
```

3.3 unicode math

Typing math in TEX is no great fun – you have to write things like `\int` instead of ∫ and so on. Have a look at the following formula:
`\int_\infty^\infty \sum_a`

The code again is stolen and I don't understand, why it does what it does, but it does it: The first argument is the character you want to use for "unicode math", the second one is the TEX-command.

```
\makeatletter
\def\altmath#1#2{%
  \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{#2}}%
    \endgroup
    \global\catcode`#1\active
  \else
  \fi
}
\makeatother
```

We will make a switch to turn this stuff on or off, so it does not interfere with the unicode-math package. This list will increase by time. If you are missing a symbol, just send me the `\altmath{X}{\Xcode}`-line. I would be very thankful if anybody could send me a whole list of symbols!

```
\def\makealtmath{
\altmath{α}\alpha
\altmath{β}\beta
\altmath{γ}\gamma
\altmath{δ}\delta

\altmath{⇒}\Rightarrow
\altmath{⇐}\Leftarrow
\altmath{⇔}\Leftrightarrow

\altmath{∫}\int
\altmath{∀}\forall
}
```

There will be an `\makenormalmath`-switch as well.

3.4 Lazy underscript and superscript

An underscore at the end of an inline-formula has to be ended with } or egroup. That is not nice…

The redefinition of hat does not work because TeX uses it for definition of catcodes. There has to be a really tricky way to get around that.

Sometimes one has to make extensive use of subscripts and superscripts, e. g. when typing long formulae including tensors. Then it is a bit annoying to always write the `{}`, especially when there are only two letters.

First, store the actual meaning of _ and ^ in `\oldunderscore` and `\oldhat`.

```
\let\oldunderscore_\relax
\let\oldhat^\relax
```

Now set _ as `\active` char and define it the way we want it to behave. For this, we need the space char and end-of-line char to be an egroup char. So the underscript group is ended by space or eol and we don't need to close it explicitly.

```
\catcode`\_=13
\def_{%
  \ifmmode
    \catcode`\ =2\relax%
    \catcode`\^^M=2\relax%
    \expandafter\oldunderscore\bgroup%
  \else%
    \textunderscore%
  \fi%
}

\iffalse
This does not work so far…
\catcode`\^=13
\def^{%
  \ifmmode
    \catcode`\ =2\relax%
    \catcode`\^^M=2\relax%
    \expandafter\oldhat\bgroup%
  \else%
    \oldhat%
  \fi%
}
\fi
```

To give the possibility to swith between normal and `alttex` behaviour, store the new underscore.

The newUnder does not work so far.

```
\let\advancedunderscore_
```

And the switches. By default, `_` is active. Type `\oldUnder` to get the normal `_`.

```
\def\oldUnder{
  \global\catcode`\_=8\relax
}
\def\newUnder{
  \global\let_\advancedunderscore
}
```

4 Lists and such 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.[5]

insideitemize wird nicht zurückgesetzt!!

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\outside{o}
\def\inside{i}
\let\insideitemizei\outside
\let\insideitemizeii\outside
```

The end of itemizei and itemizeii:

```
\def\•{\end{itemize}}
\def\‣{\end{itemize}}

\def\newitemi{%
  \ifx\insideitemizei\inside%
    \setcounter{lastitem}{0}%
    \expandafter\item%
  \else%
```

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

```
    \begin{itemize}%
    \let\insideitemizei\inside%
    \catcode`\ſ=5%
    \catcode`\€=14%
    \catcode`\^^M=\active\def^^M{\end{itemize}}\item €ſ
  \fi
}

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

4.2 enumerate with a single character

¹, ² And we do just the same stuff with \enumerate. But here we take the character ¹ as first level item, the ²[6] 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.

```
\def\¹{\end{enumerate}}
\def\²{\end{enumerate}}

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

\def\newenumi{
  \ifx\insideenumi\inside
    \expandafter\item%
```

[6]Maybe this is a very stupid idea, because now the ² cannot be used as a square in mathmode. Of course there could be a test ifmmode, but I rather would like to find a better character for enumerate.

```
  \else
    \begin{enumerate}
      \let\insideenumi\inside
      \expandafter\item%
  \fi
}

\def\newenumii{
  \ifx\insideenumii\inside
    \expandafter\item%
  \else
    \begin{enumerate}
      \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
```

And that's it.

Happy altTeXing!

A very short introduction to XƎLaTeX

Everything you have to know about XƎLaTeX to use this package: Write your LaTeX file just as you are used to. But save it as utf8-encoded, and say

```
\usepackage{xltxra}
```

instead of

```
\usepackage[latin1]{inputenc} and \usepackage[T1]{fontenc}
```

This loads some files that provide all the cool stuff XƎLaTeX offers. You don't have to take care of letters TeX would not understand – XƎTeX understands every character you type. But sometimes the font may not have the symbol for this – then you can use `\fontspec{fontname}`, where `fontname` is the name of a font on your system, e. g. `Arno Pro`, `Linux Libertine`, `LT Zapfino One` etc.

Then, you compile your document with the command `xelatex file.tex`, instead of `xelatex file.tex` and you get a pdf as output. Nevertheless, XƎTeX is not an pdfTeX successor, so you cannot use microtypographic extensions.

If you have any trouble using XƎLaTeX, just e-mail me!

todo

Here a section with some ideas that could be implemented.

- Use `²` as square in mathmode and possibly `¹` as `\footnote`?
- Do something to enable easy tabular
- If there is only one char after an `_`, there should no space be needed.