

# The **alttex** package

Arno L. Trautmann\*

Version 0.a.3 April 26, 2009

This is the package **alttex** which will try to give an experimental new way to write  $\text{\LaTeX}$ <sup>1</sup> 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

<sup>1</sup>If you don't know about  $\text{\LaTeX}$ , see the appendix.??

# 1 introduction

The problem I have with  $\text{\LaTeX}^2$  is the antique way of typing. Because most people still use a hopelessly outdated keyboard layout («qwerty» or slightly adapted versions of that),  $\text{\LaTeX}$  doesn't make use of some cool features.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  $\text{\LaTeX}$ , 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<sup>3</sup>, you can simply write:<sup>4</sup>

This is the normal text, then comes the itemization:

- text for first item
- - this is an item inside an item
  - [ $\Rightarrow$ ] Here an item with a formula:  $\int_a^b x^2 dx$
- 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:

---

<sup>2</sup>I'll write  $\text{\LaTeX}$  instead of  $\text{\XeLaTeX}$ —saves me two keystrokes. Most of the code below *only* works with  $\text{\XeLaTeX}$ . If you need support for `[utf8]inputenc` or  $\text{\LuaTeX}$ , please contact the author.

<sup>3</sup>E.g. the ergonomic layout Neo: <http://neo-layout.org/>

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

`\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\define the `\usepackage` and hope, it works. Only working with texlive! If you're using MikTeX, put a

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

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

```

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
```

3

## 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.

```
32 \def\noescape{
33   \catcode`\_ = 11%
34   \catcode`\^ = 11%
35   \catcode`\# = 11%
36   \catcode`\& = 11%
37   %\catcode`\{ = 11%
38   %\catcode`\} = 11%
39   \catcode`\$ = 11%
40   \catcode`\~ = 11%
41   \makeatletter%
42   \catcode`\% = 11
43 }
```

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.

```
44 \def\oldescape{
45   \catcode`\% = 14%
46   \catcode`\_ = 8%
47   \catcode`\^ = 7%
48   \catcode`\# = 6%
49   \catcode`\& = 4%
50   %\catcode`\{ = 1%
51   %\catcode`\} = 2%
52   \catcode`\$ = 3%
53   \catcode`\~ = 13%
54   \makeatother%
55 }
```

### 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!

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

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

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

```
56 \def\-{\\hhline}
57 \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.

## 2.3 excel tabulars

`\exceltabular` Often one uses a program to calculate tabulars of numbers. To insert it into L<sup>A</sup>T<sub>E</sub>X, one has to do some work. Here we try to copy-paste the tabular from excel, Calc or any other program to a file `mytabular.txt` (or any other ending). Then you say `\exceltabular{mytabular}` (you do not need the ending, therefore it doesn't matter) and you get the tabular in a standard format. I will extend this to enable caption, variable number of columns, kind of rule used etc. This is just a very first test.

This is the definition of the command:

```
58 \def\exceltabular#1{
59   \catcode\^^I=4\relax
60   \eolintabular%
61   \begin{tabular}{|c|c|c|}\hhline%
62   \input{#1}%
63   \end{tabular}%
64   \catcode\^^M=5\relax
65 }
```

And a little helper function to make the `<enter>` `\active`. Again, thanks to the people on the mailinglists.

```
66 \def\mybreak{\\hhline}
67 \begingroup
68   \lccode\~=\^^M%
69   \lowercase{%
70     \endgroup
71     \def\eolintabular{%
72       \catcode\^^M=\active
73       \let~\mybreak
74     }%
75 }
```

## 2.4 tabbing

This will be analog to the `\exceltabular`. You write your tabbing using tabs and `<enter>`. That's it :)

`\alttabbing`

```
76 %
77 %   \end{macrocode}
78 %
```

```

79 % \end{macro}
80
81 % \section{Math stuff}
82 % \subsection{braces}
83 % \begin{macro}{\newbraces}
84 % \begin{macro}{\oldbraces}
85 % 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 height 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 \verb~| [ { <~ and so on. Maybe in some later version.
86 %
87 % I would have never been able to implement this without the help of the mailinglist members of |tex-d-l@listserv.dfn.de| !\todo{The newbraces does \emph{not} work at the moment!}
88 %
89 % The redefinition of |\mathstrut| is necessary when using amsmath (you will use amsmath when typesetting formulae, won't you?), because the height of formulae is determined by the height 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 ( ).\todo{Maybe one could "temporarily hardcode" the height of [ and then use this...}
90 % \begin{macrocode}
91 \makeatletter
92 \def\resetMathstrut@{%
93   \setbox\z@\hbox{%
94     \mathchardef\@tempa\mathcode`\[ \relax
95     \def\@tempb##1"##2##3{\the\textfont"##3\char"}%
96     \expandafter\@tempb\meaning\@tempa \relax
97   }%
98   \ht\Mathstrutbox@\ht\z@ \dp\Mathstrutbox@\dp\z@
99 }
100 \makeatother
101
102 {\catcode`\(\active \xdef{\left\string{}}
103 {\catcode`\)\active \xdef{\right\string{}}}
104
105 \def\newbraces{
106   \mathcode`("8000
107   \mathcode`) "8000
108 }
109
110 \edef\oldbraces{
111   \mathcode`\(\the\mathcode` (
112   \mathcode`\)\the\mathcode` )
113 }

```

## 2.5 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.

```
114 \def\hugedisplaymath{
115   \makeatletter
116   \makeatother
117   \Huge
118   \begin{equation*}
119 }
120 \def\endhugedisplaymath{
121   \end{equation*}
122 }
```

## 2.6 unicode math

Typing math in T<sub>E</sub>X 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 T<sub>E</sub>X-command.

```
123 \makeatletter
124 \def\altmath#1#2{%
125   \expandafter\ifx\csname cc\string#1\endcsname\relax
126     \add@special{#1}%
127     \expandafter
128     \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
129     \begingroup
130       \catcode`\~\active \lccode`\~`#1%
131       \lowercase{%
132         \global\expandafter\let
133         \csname ac\string#1\endcsname~%
134         \expandafter\gdef\expandafter~\expandafter{#2}}%
135     \endgroup
136     \global\catcode`#1\active
137   \else
138     \fi
139 }
140 \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!

```

141 \def\makealtmath{
142 \altmath{\alpha}\alpha
143 \altmath{\beta}\beta
144 \altmath{\gamma}\gamma
145 \altmath{\delta}\delta
146
147 \altmath{\Rightarrow}\Rightarrow
148 \altmath{\Leftarrow}\Leftarrow
149 \altmath{\Leftrightarrow}\Leftrightarrow
150
151 \altmath{\int}\int
152 \altmath{\forall}\forall
153 }

```

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

## 2.7 Lazy underscript and superscript

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 in the sub/superscript. So let's try to implement the possibility to type `$F_{\mu\nu} F^{\mu\nu}$`.

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

```

154 \let\oldunderscore_\relax
155 \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.

```

156 \catcode`\_ =13
157 \def_{%
158   \ifmode
159     \catcode`\ =2\relax%
160     \catcode`\^M=2\relax%
161     \expandafter\oldunderscore\bgroup%
162   \else%
163     \textunderscore%
164   \fi%
165 }
166
167 \iffalse
168 This does not work so far...
169 \catcode`\^ =13
170 \def^{%
171   \ifmode
172     \catcode`\ =2\relax%
173     \catcode`\^M=2\relax%

```

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.



```

174 \expandafter\oldhat\bgroup%
175 \else%
176 \oldhat%
177 \fi%
178 }
179 \fi

```

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

```
180 \let\advancedunderscore_
```

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

```

181 \def\oldUnder{
182 \global\catcode`\_ =8\relax
183 }
184 \def\newUnder{
185 \global\let_\advancedunderscore
186 }

```

## 3 Lists and such things

### 3.1 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.

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

```

187 \def\outside{o}
188 \def\inside{i}
189 \let\insideitemizei\outside
190 \let\insideitemizeii\outside

```

The end of itemizei and itemizeii:

```

191 \def\altenditemize{
192 \if\altlastitem l%
193 \let\altlastitem0%
194 \else%
195 \end{itemize}%
196 \let\insideitemizei\outside%
197 \fi%
198 }
199
200 \begingroup

```

```

201 \lccode`~=`^^M%
202 \lowercase{%
203 \endgroup
204 \def\makeenteractive{%
205 \catcode`^^M=\active
206 \let~\altenditemize
207 }%
208 }
209

```

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

```

232 %
233 \makeatletter
234 \def\makeitemi#1{%
235   \expandafter\ifx\csname cc\string#1\endcsname\relax
236     \add@special{#1}%
237   \expandafter
238   \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
239   \begingroup
240     \catcode`\~\active \lccode`\~`#1%
241     \lowercase{#1%

```

```

242     \global\expandafter\let
243     \csname ac\string#1\endcsname~%
244     \expandafter\gdef\expandafter~\expandafter{\newitemi}}%
245   \endgroup
246   \global\catcode`#1\active
247 \else
248 \fi
249 }
250
251 \def\makeitemi#1{%
252   \expandafter\ifx\csname cc\string#1\endcsname\relax
253     \add@special{#1}%
254     \expandafter
255     \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
256     \begingroup
257       \catcode`\~\active \lccode`\~`#1%
258       \lowercase{%
259         \global\expandafter\let
260         \csname ac\string#1\endcsname~%
261         \expandafter\gdef\expandafter~\expandafter{\newitemi}}%
262       \endgroup
263       \global\catcode`#1\active
264   \else
265   \fi
266 }

```

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

```

267 \def\add@special#1{%
268   \rem@special{#1}%
269   \expandafter\gdef\expandafter\dospecials\expandafter
270   {\dospecials \do #1}%
271   \expandafter\gdef\expandafter\@sanitize\expandafter
272   {\@sanitize \@makeoother #1}}
273 \def\rem@special#1{%
274   \def\do##1{%
275     \ifnum`#1=##1 \else \noexpand\do\noexpand##1\fi}%
276   \xdef\dospecials{\dospecials}%
277   \begingroup
278     \def\@makeoother##1{%
279       \ifnum`#1=##1 \else \noexpand\@makeoother\noexpand##1\fi}%
280     \xdef\@sanitize{\@sanitize}%
281   \endgroup}
282 \makeatother

```

### 3.2 enumerate with a single character

<sup>1</sup>, <sup>2</sup> And we do just the same stuff with `\enumerate`. But here we take the character <sup>1</sup> as first level item, the <sup>2</sup><sup>5</sup> 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.

```

283 \def\^1{\end{enumerate}}
284 \def\^2{\end{enumerate}}
285
286 \let\insideenumi\outside
287 \let\insideenumii\outside
288
289 \def\newenumi{
290   \ifx\insideenumi\inside
291     \expandafter\item%
292   \else
293     \begin{enumerate}
294       \let\insideenumi\inside
295       \expandafter\item%
296   \fi
297 }
298
299 \def\newenumii{
300   \ifx\insideenumii\inside
301     \expandafter\item%
302   \else
303     \begin{enumerate}
304       \let\insideenumii\inside
305       \expandafter\item%
306   \fi
307 }
308

```

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.

```

309 \makeatletter
310 \def\makeenumi#1{%
311   \expandafter\ifx\csname cc\string#1\endcsname\relax
312     \add@special{#1}%
313   \expandafter
314   \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
315   \begingroup
316     \catcode`\~\active \lccode`\~`#1%
317     \lowercase{%
318       \global\expandafter\let

```

---

<sup>5</sup>Maybe this is a very stupid idea, because now the <sup>2</sup> 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`.

```

319         \csname ac\string#1\endcsname~%
320         \expandafter\gdef\expandafter~\expandafter{\newenumi}}%
321     \endgroup
322     \global\catcode`#1\active
323 \else
324 \fi
325 }
326
327 \def\makeenumii#1{%
328     \expandafter\ifx\csname cc\string#1\endcsname\relax
329         \add@special{#1}%
330         \expandafter
331         \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
332         \begingroup
333             \catcode`\~\active \lccode`\~`#1%
334             \lowercase{%
335                 \global\expandafter\let
336                 \csname ac\string#1\endcsname~%
337                 \expandafter\gdef\expandafter~\expandafter{\newenumii}}%
338         \endgroup
339         \global\catcode`#1\active
340     \else
341     \fi
342 }
343 \makeatother
344

```

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

```

345 \makeitemi•
346 \makeitemii▶
347 \makeenumi1
348 \makeenumii2

```

And that's it.

Happy  $\text{\LaTeX}$ ing!

## A very short introduction to X<sub>Y</sub>LaTeX

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

```
\usepackage{xlttra}
```

instead of

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

This loads some files that provide all the cool stuff X<sub>Y</sub>LaTeX offers. You don't have to take care of letters T<sub>E</sub>X would not understand – X<sub>Y</sub>LaTeX 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 `latex file.tex` and you get a pdf as output. Mostly, your editor will not have a shortcut to start X<sub>Y</sub>LaTeX. In that case, you have to compile via the command line. If you know your editor well enough, you may be able to create a shortcut that will run `xelatex file.tex` for you. Notice that you will need an editor that is utf8-capable! One last warning: While X<sub>Y</sub>LaTeX is not an pdf<sub>T</sub>E<sub>X</sub> successor, you cannot use microtypographic extensions. Maybe in the future there will be an implementation that uses advanced OpenType-features, but at the moment there is no microtypography possible!

If you have any trouble using X<sub>Y</sub>LaTeX, just e-mail me!

## todo

Here a section with some ideas that could be implemented.

- Use <sup>2</sup> as square in mathmode and possibly <sup>1</sup> as `\footnote`?
- Do something to enable easy tabular
- If there is only one char after an `_`, there should no space be needed.
- Maybe there could be a ConTeXt-version of this file.