

# The `alrtex` package

Arno L. Trautmann\*

Version 0.a.2 January 2, 2009

This is the package `alrtex` which will try to give an experimental new way to write  $\text{X}\text{\LaTeX}$ <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

|          |   |          |
|----------|---|----------|
| <b>1</b> | <b>introduction</b>                         | <b>2</b> |
| <b>2</b> | <b>Textmode</b>                             | <b>4</b> |
| 2.1      | no escape . . . . .                         | 4        |
| 2.2      | tabular . . . . .                           | 4        |
| <b>3</b> | <b>Math stuff</b>                           | <b>5</b> |
| 3.1      | braces . . . . .                            | 5        |
| 3.2      | huge display math . . . . .                 | 6        |
| 3.3      | unicode math . . . . .                      | 6        |
| <b>4</b> | <b>Lists and such things</b>                | <b>7</b> |
| 4.1      | itemize with a single character . . . . .   | 7        |
| 4.2      | enumerate with a single character . . . . . | 9        |

---

\*arno.trautmann@gmx.de

<sup>1</sup>If you don't know about  $\text{X}\text{\LaTeX}$ , see the appendix.4.2

# 1 introduction

The problem I have with  $\text{\LaTeX}$ <sup>2</sup> 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 Because most people still use a hopelessly outdated keyboard layout („qwerty“ or slightly adapted versions of that),  $\text{\LaTeX}$  doesn't make use of some cool features.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.



## 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.

## 3 Math stuff

### 3.1 braces

`\newbraces` Now this is something most  $\text{\LaTeX}$ -beginners don't recognize and wonder why the  
`\oldbraces` 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  $\text{\LaTeX}$  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](mailto: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 `()`.

```
58 \makeatletter
59 \def\resetMathstrut@{%
60   \setbox\z@\hbox{%
61     \mathchardef\@tempa\mathcode`\[\relax
62     \def\@tempb##1"##2##3{\the\textfont"##3\char"%
63     \expandafter\@tempb\meaning\@tempa \relax
64   }%
65   \ht\Mathstrutbox=\ht\z@ \dp\Mathstrutbox=\dp\z@
66 }
67 \makeatother
68
69
70 \edef\oldbraces{
71   \mathcode`\(\the\mathcode` (
72   \mathcode`) \the\mathcode`)
73 }
74 \begingroup
75   \catcode`\active \xdef({\left\string(}
76   \catcode`\active \xdef){\right\string)}
77 \endgroup
78 \def\newbraces{
```

The newbraces does *not* work at the moment!

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

```

79
80 \mathcode`("8000
81 \mathcode`) "8000
82 }

```

### 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.

```

83 \def\hugedisplaymath{
84   \makeatletter
85   \makeatother
86   \Huge
87   \begin{equation*}
88 }
89 \def\endhugedisplaymath{
90   \end{equation*}
91 }

```

### 3.3 unicode math

Typing math in  $\TeX$  is no great fun – you have to write things like `\int` instead of  $\int$  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.

```

92 \makeatletter
93 \def\altmath#1#2{%
94   \expandafter\ifx\csname cc\string#1\endcsname\relax
95     \add@special{#1}%
96     \expandafter
97     \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
98     \begingroup
99       \catcode`\~\active \lccode`\~`#1%
100     \lowercase{%
101       \global\expandafter\let
102       \csname ac\string#1\endcsname~%
103       \expandafter\gdef\expandafter~\expandafter{#2}}%
104     \endgroup
105     \global\catcode`#1\active
106   \else
107     \fi

```

```
108 }
```

```
109 \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!

```
110 \def\makealtmath{
```

```
111 \altmath{\alpha}\alpha
```

```
112 \altmath{\beta}\beta
```

```
113 \altmath{\gamma}\gamma
```

```
114 \altmath{\delta}\delta
```

```
115
```

```
116 \altmath{=>}\rightarrow
```

```
117 \altmath{<=}\leftarrow
```

```
118 \altmath{<=>}\leftrightarrows
```

```
119
```

```
120 \altmath{\int}\int
```

```
121 \altmath{\forall}\forall
```

```
122 }
```

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

## 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.<sup>5</sup>

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.

```
123 \def\outside{o}
```

```
124 \def\inside{i}
```

```
125 \let\insideitemizei\outside
```

```
126 \let\insideitemizeii\outside
```

The end of itemizei and itemizeii:

```
127 \def\bullet{\end{itemize}}
```

```
128 \def\blacktriangleright{\end{itemize}}
```

```
129
```

```
130 \def\newitemi{
```

```
131 \ifx\insideitemizei\inside
```

<sup>5</sup>The triangular bullet sign does not appear here – the font is lacking it...

insideitemize wird nicht  
zurückgesetzt!!

```

132   \expandafter\item%
133   \else
134     \begin{itemize}
135       \let\insideitemizei\inside
136       \expandafter\item%
137     \fi
138 }
139
140 \def\newitemii{
141   \ifx\insideitemizeii\inside
142     \expandafter\item%
143   \else
144     \begin{itemize}
145       \let\insideitemizeii\inside
146       \expandafter\item%
147     \fi
148 }

```

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  
`\makeitemii` 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.

```

149 %
150 \makeatletter
151 \def\makeitemi#1{%
152   \expandafter\ifx\csname cc\string#1\endcsname\relax
153     \add@special{#1}%
154     \expandafter
155     \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
156     \begingroup
157       \catcode`\~\active \lccode`\~`#1%
158       \lowercase{%
159         \global\expandafter\let
160         \csname ac\string#1\endcsname~%
161         \expandafter\gdef\expandafter~\expandafter{\newitemi}}%
162     \endgroup
163     \global\catcode`#1\active
164   \else
165   \fi
166 }
167
168 \def\makeitemii#1{%
169   \expandafter\ifx\csname cc\string#1\endcsname\relax
170     \add@special{#1}%
171     \expandafter
172     \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%

```



```

173 \begingroup
174 \catcode`\~\active \lccode`\~`#1%
175 \lowercase{%
176 \global\expandafter\let
177 \csname ac\string#1\endcsname~%
178 \expandafter\gdef\expandafter~\expandafter{\newitemii}}%
179 \endgroup
180 \global\catcode`#1\active
181 \else
182 \fi
183 }

```

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

```

184 \def\add@special#1{%
185 \rem@special{#1}%
186 \expandafter\gdef\expandafter\dospecials\expandafter
187 {\dospecials \do #1}%
188 \expandafter\gdef\expandafter\@sanitize\expandafter
189 {\@sanitize \@makeother #1}}
190 \def\rem@special#1{%
191 \def\do##1{%
192 \ifnum`#1=##1 \else \noexpand\do\noexpand##1\fi}%
193 \xdef\dospecials{\dospecials}%
194 \begingroup
195 \def\@makeother##1{%
196 \ifnum`#1=##1 \else \noexpand\@makeother\noexpand##1\fi}%
197 \xdef\@sanitize{\@sanitize}%
198 \endgroup}
199 \makeatother

```

## 4.2 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>6</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.

```

200 \def\^1{\end{enumerate}}
201 \def\^2{\end{enumerate}}
202
203
204 \let\insideenumi\outside
205 \let\insideenumii\outside
206
207 \def\newenumi{
208 \ifx\insideenumi\inside

```

---

<sup>6</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`.

```

209 \expandafter\item%
210 \else
211 \begin{enumerate}
212 \let\insideenumi\inside
213 \expandafter\item%
214 \fi
215 }
216
217 \def\newenumii{
218 \ifx\insideenumii\inside
219 \expandafter\item%
220 \else
221 \begin{enumerate}
222 \let\insideenumii\inside
223 \expandafter\item%
224 \fi
225 }
226

```

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.

```

227 \makeatletter
228 \def\makeenumi#1{%
229 \expandafter\ifx\csname cc\string#1\endcsname\relax
230 \add@special{#1}%
231 \expandafter
232 \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
233 \begingroup
234 \catcode`\~\active \lccode`\~`#1%
235 \lowercase{%
236 \global\expandafter\let
237 \csname ac\string#1\endcsname~%
238 \expandafter\gdef\expandafter~\expandafter{\newenumii}}%
239 \endgroup
240 \global\catcode`#1\active
241 \else
242 \fi
243 }
244
245 \def\makeenumii#1{%
246 \expandafter\ifx\csname cc\string#1\endcsname\relax
247 \add@special{#1}%
248 \expandafter
249 \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
250 \begingroup
251 \catcode`\~\active \lccode`\~`#1%
252 \lowercase{%
253 \global\expandafter\let
254 \csname ac\string#1\endcsname~%
255 \expandafter\gdef\expandafter~\expandafter{\newenumii}}%

```

```

256     \endgroup
257     \global\catcode`#1\active
258   \else
259     \fi
260 }
261 \makeatother
262

```

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

```

263 \makeitemi•
264 \makeitemii▶
265 \makeenumi1
266 \makeenumii2

```

And that's it.

Happy altT<sub>E</sub>Xing!

## 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, *do not* use `\usepackage{inputenc}` and `\usepackage{fontenc}`, but *do use*

`\usepackage{xltxra}`.

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 TeX would not understand – X<sub>Y</sub>TeX 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` etc. Of course, you don't compile with the command `latex file.tex`, but `xelatex file.tex`. You get a pdf as output. Nevertheless, X<sub>Y</sub>TeX is not pdfTeX, so you cannot use microtypographic extensions... :(

If you have any trouble using X<sub>Y</sub>LaTeX, just 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`?
-