The \texttt{mhsetup} package

Morten Høgholm (maintained by Lars Madsen)

2017/03/31

Abstract

The \texttt{mhsetup} package provides tools for a \LaTeX{} programming environment similar to the one described in \texttt{expl3} on CTAN although not as extensive. It is a required part of both the \texttt{mathtools} and \texttt{empheq} packages.

The description below was made before the extensive changes made to the expl3 code available from the \LaTeX{} Project website.

1 The new internal syntax

The \LaTeX{}3 package \texttt{ldcsetup} defines the command \texttt{\InternalSyntaxOn} which makes \_ and : letters and then automatically restores the category codes at the end of the package. This usually works fine but when you try to load \texttt{amstext} you will experience that \TeX{} goes into an infinite loop. Packages containing code like \texttt{\@for\@tempa:=\@tempb\do{...}} will not work correctly either, thus we provide an alternative version here with the pair of commands \texttt{\MHInternalSyntaxOn} and \texttt{\MHInternalSyntaxOff}. They are to be used only as a pair, because \texttt{\MHInternalSyntaxOn} defines \texttt{\MHInternalSyntaxOff} so that it restores the category codes correctly.

2 Handling optional arguments

The \LaTeX{}3 package \texttt{ldcsetup} defines the command \texttt{\InternalSyntaxOn} which makes \_ and : letters and then automatically restores the category codes at the end of the package. This usually works fine but when you try to load \texttt{amstext} you will experience that \TeX{} goes into an infinite loop. Packages containing code like \texttt{\@for\@tempa:=\@tempb\do{...}} will not work correctly either, thus we provide an alternative version here with the pair of commands \texttt{\MHInternalSyntaxOn} and \texttt{\MHInternalSyntaxOff}. They are to be used only as a pair, because \texttt{\MHInternalSyntaxOn} defines \texttt{\MHInternalSyntaxOff} so that it restores the category codes correctly.

\MHInternalSyntaxOn
\MHInternalSyntaxOff

The standard behavior of scanning for optional arguments in \LaTeX{} allows any number of spaces preceding the optional argument and that is not always good in math. For that reason \texttt{amsmath} makes sure that commands like \texttt{\textbackslash{}\disallows} spaces before the optional argument but at the same time it fails to provide “safe” environments. What would you expect from the following input?

\[
\begin{gathered}
\texttt{\textbackslash{}[}
\begin{gathered}
[v] = 100 \text{ \textbackslash{} }
[t] = 200
\end{gathered}
\end{gathered}
\texttt{\textbackslash{}]}\]

\ MHInternalSyntaxOn
\MHInternalSyntaxOff

\*This package has version number v1.3, last revised on 2017/03/31.
\LaTeX{} will see the \texttt{[v]} as an optional argument of \texttt{gathered} and use it. In this case the test inside \texttt{gathered} checks if it’s a \texttt{t} or \texttt{b} and if it’s neither it’ll choose \texttt{vcenter} internally. So you get no warning, only missing output. Another example, this time from the \texttt{empheq} package used with its \texttt{overload} option: If preceding spaces are allowed, the input

\begin{verbatim}
\begin{gather}
 [a] = [b]
\end{gather}
\end{verbatim}

results in the rather strange error message

! Package keyval Error: a undefined.

When using \texttt{\newcommand} etc. for defining commands and environments with optional arguments, the peek ahead is done by \texttt{\kernel@ifnextchar} (since \LaTeX{} release 2003/12/01, else \texttt{@ifnextchar}) and it is hardwired at definition time by \texttt{\@xargdef}. With the commands \texttt{\MHPrecedingSpacesOff} and \texttt{\MHPrecedingSpacesOn} \texttt{mhsetup} provides an interface to define commands and environments where the optional argument cannot have preceding spaces. You simply wrap them around the definitions:

\begin{verbatim}
\MHPrecedingSpacesOff
\newenvironment*{test}[1][default]{Start, arg: (#1)}{Ending.}
\MHPrecedingSpacesOn
\begin{test}
 [text]
\end{test}
\begin{test}
 [text]
\end{test}
\end{verbatim}

Start, arg: (default) \texttt{[text]} Ending. Start, arg: \texttt{(text)} Ending.

It is of somewhat limited use in commands (control words in \LaTeX{} terminology), because \LaTeX{} discards the spaces. The exception is \texttt{control symbols} where \LaTeX{} obeys following spaces but there are rather few of them available. All is not lost however. In the \texttt{aligned} environment from \texttt{amsmath} (shown below) a command is used as argument grabber.

\begin{verbatim}
\newenvironment{aligned}{%  \let\@testopt\alignsafe@testopt
\aligned@a% }
{%  \crcr\egroup
  \restorecolumn@}
\end{verbatim}

1
By applying our trick on the grabber function, we get a space obeying version:

\MHPrecedingSpacesOff
\newcommand*{\aligned@a}{\start@aligned{#1}\m@ne}
\MHPrecedingSpacesOn

This way a nested aligned environment is still safe from empty first cells.

3 First bits of a new programming environment

Almost copy of \InternalSyntaxOn.

\def\MHInternalSyntaxOn{
  \edef\MHInternalSyntaxOff{%
    \catcode\noexpand\~\the\catcode\~\relax
    \catcode\noexpand\ =\the\catcode\ \relax
    \catcode\noexpand\^^I\the\catcode\^^I\relax
    \catcode\noexpand\@\the\catcode\@\relax
    \catcode\noexpand\:=\the\catcode\:\relax
    \catcode\noexpand\_\the\catcode\_\relax
    \endlinechar\the\endlinechar\relax
  }%
  \catcode\~=10\relax
  \catcode\ =9\relax
  \catcode\^^I=9\relax
  \makeatletter
  \catcode\_=11\relax
  \catcode\\%=11\relax
  \endlinechar=\%
  \relax
}
\MHInternalSyntaxOn
\MHAtEndOfPackage{\MHInternalSyntaxOff}

3.2 Programming tools

The whole idea is to provide programming tools that are convenient but not yet widely available. I hope this’ll be obsolete soon!

Firstly we setup a few helper functions.
\MH_let:NwN  An alias for \let.
\let\MH_let:NwN \let

\MH_let:cN  This one takes a \csname-endcsname name and \lets it to a single macro. We'll use this to set up our conditionals.
\def\MH_let:cN #1#2{
  \expandafter\MH_let:NwN\csname#1\endcsname#2}

\MH_let:cc  This one has takes a \csname-endcsname name and \lets it to a another \csname-endcsname name. To be used in constructions with weird characters like * or alike in them and can take a \global prefix if wanted (we want that later on).
\def\MH_let:cc #1#2{
  \expandafter\MH_let:NwN\csname#1\expandafter\endcsname\csname#2\endcsname}

\MH_new_boolean:n \MH_set_boolean_F:n \MH_set_boolean_T:n \MH_if_boolean:nTF \MH_if_boolean:nT \MH_if_boolean:nF

Sets up conditionals. For instance
\MH_new_boolean:n {⟨name⟩}
defines the boolean ⟨name⟩ but also the conditional \MH_if_boolean_⟨name⟩: to be used in the ordinary
\MH_if_boolean_⟨name⟩:
  ⟨true code⟩
\MH_else:
  ⟨false code⟩
\MH_fi:

There is also a more “\LaTeX-like” interface available by using the commands
\MH_if_boolean:nT{⟨name⟩}⟨{arg}⟩
which will execute the argument if the current value of the boolean is ‘true’ while
\MH_if_boolean:nF{⟨name⟩}⟨{arg}⟩
is the equivalent with ‘false’. Finally we have
\MH_if_boolean:nTF{⟨name⟩}⟨{true code}⟩⟨{false code}⟩.

This is the interface I have used in this package.
Initially \MH_if_boolean_⟨name⟩: is ‘false’. This can be changed by saying
\TeX: \MH_boolean_⟨name⟩_true: or
\LaTeX: \MH_set_boolean_T:n{⟨name⟩}
and changed back again by
\TeX: \MH_boolean_⟨name⟩_false: or
\LaTeX: \MH_set_boolean_F:n{⟨name⟩}
And yes, we’re also using alternative names for \texttt{\textbackslash else} and \texttt{\textbackslash fi} now. That way a simple search and replace will be all that is needed for this package to be a certified \LaTeX3 package (well, maybe a little more is needed, but not much).

\begin{verbatim}
def\MH_new_boolean:n #1{
  \expandafter\@ifdefinable\csname MH_if_boolean_#1:\endcsname{
    \\@namedef{MH_boolean_#1_true:}{\iftrue}
    \\@namedef{MH_boolean_#1_false:}{\iffalse}
    \\@nameuse{MH_boolean_#1_false:}%
  }
}
def\MH_set_boolean_F:n #1{ \@nameuse{MH_boolean_#1_false:} }
def\MH_set_boolean_T:n #1{ \@nameuse{MH_boolean_#1_true:} }
def\MH_if_boolean:nTF #1{
  \\@nameuse{MH_if_boolean_#1:}
  \expandafter\@firstoftwo\MH_else:
  \expandafter\@secondoftwo\MH_fi:
}
def\MH_if_boolean:nT #1{
  \\@nameuse{MH_if_boolean_#1:}
  \expandafter\@firstofone\MH_else:
  \expandafter\@gobble\MH_fi:
}
def\MH_if_boolean:nF #1{
  \\@nameuse{MH_if_boolean_#1:}
  \expandafter\@gobble\MH_else:
  \expandafter\@firstofone\MH_fi:
}
\AtBeginDocument{
  \\@ifundefined{MH_if:w}{\MH_let:NwN \MH_if:w =\if}{}
  \\@ifundefined{MH_if_meaning:NN}{\MH_let:NwN \MH_if_meaning:NN =\ifx}{}
  \\@ifundefined{MH_else:}{\MH_let:NwN \MH_else:=\else}{}
  \\@ifundefined{MH_fi:}{\MH_let:NwN \MH_fi:=\fi}{}
  \\@ifundefined{MH_if_num:w}{\MH_let:NwN \MH_if_num:w =\ifnum}{}
  \\@ifundefined{MH_if_dim:w}{\MH_let:NwN \MH_if_dim:w =\ifdim}{}
  \\@ifundefined{MH_if_case:w}{\MH_let:NwN \MH_if_case:w =\ifcase}{}
  \\@ifundefined{MH_or:}{\MH_let:NwN \MH_or:=\or}{}
  \\\MH_cs_to_str:N Strip off the backslash of a macro name.
\end{verbatim}
We might as well make use of some of the extended features from \texttt{\textasciitilde\textsc{pX}}. We use \texttt{\textbackslash dimexpr} for some simple calculations as it saves a lot of the scanning that goes on inside \texttt{calc}. The \texttt{\textbackslash protected} primitive comes in handy when we want to declare a robust command, that cannot be ‘robustified’ with \texttt{\textbackslash DeclareRobustCommand}. If we don’t have \texttt{\textasciitilde\textsc{pX}} we’ll just let our private macros be aliases for the less effective alternatives.

A way to make aliases with \texttt{keyval}. This will come in handy later.

I need to be able to pick up individual arguments in a list of four (similar to \texttt{\textbackslash@secondoftwo}).

Scanning for the next character but disallow spaces.
The code for the space sensitive peek ahead.

\kernel@ifnextchar
\MH_kernel_xargdef:nwwn
\MH_nospace_xargdef:nwwn
\MHPrecedingSpacesOff
\MHPrecedingSpacesOn
\MH_group_align_safe_begin:
\MH_group_align_safe_end: