The \texttt{mhsetup} package* 

Morten Høgholm (maintained by Lars Madsen and The LaTeX3 team) 

2021/03/18 

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 \LaTeX3 package \texttt{ldcsetup} defines the command \texttt{\textbackslash 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 \LaTeX3 package \texttt{ldcsetup} defines the command \texttt{\textbackslash 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. 

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 \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{verbatim}
begin{gathered}
[v] = 100 \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ [t] = 200
\end{gathered}
\end{verbatim}

*This package has version number v1.4, last revised on 2021/03/18.
LaTeX will see the \[v\] 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 \textit{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}


It is of somewhat limited use in commands (control words in \TeX{} terminology), because \TeX{} discards the spaces. The exception is \textit{control symbols} where \TeX{} 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
\endgroup
\restorecolumn@}
\end{verbatim}

2
By applying our trick on the grabber function, we get a space obeying version:

```
\MHPrecedingSpacesOff
\renewcommand*{\aligned@section}{\start@aligned{#1}\m@ne}
\MHPrecedingSpacesOn
```

This way a nested \texttt{aligned} environment is still safe from empty first cells.

### 3 First bits of a new programming environment

1 ⟨∗package⟩
2 \ProvidesPackage{mhsetup}%
3 [2021/03/18 v1.4 programming setup (MH)]

#### 3.1 The new internal syntax

```
\MHInternalSyntaxOn
\MHInternalSyntaxOff
Almost copy of \texttt{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
\AtEndOfPackage{\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.
An alias for \let.

This one takes a \csname-endcsname name and \lets it to a single macro. We’ll use this to setup our conditionals.

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).

Sets up conditionals. For instance

\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

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 “\TeX-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 \else and \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).  

```
\def\MH_new_boolean:n #1{  
  \expandafter\@ifdefinable\csname MH_if_boolean_#1: \endcsname{  
    \@namedef{MH_boolean_#1_true:}{\MH_let:cN{MH_if_boolean_#1:}\iftrue}  
    \@namedef{MH_boolean_#1_false:}{\MH_let:cN{MH_if_boolean_#1:}\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:  
}
```

\MH_if:w Copies of \TeX\ primitives.

\MH_if_meaning:NN \MH_else: \MH_if_num:w \MH_if_dim:w \MH_if_case:w \MH_or:w

\MH_cs_to_str:N Strip off the backslash of a macro name.

5
We might as well make use of some of the extended features from $\varepsilon$-TEX. We use \texttt{dimexpr} for some simple calculations as it saves a lot of the scanning that goes on inside \texttt{calc}. The \texttt{\MH_protected} primitive comes in handy when we want to declare a robust command, that cannot be ‘robustified’ with \texttt{\DeclareRobustCommand}. If we don’t have $\varepsilon$-TEX we’ll just let our private macros be aliases for the less effective alternatives.

\begin{verbatim}
\def\MH_setlength:dn {\setlength}
\def\MH_addtolength:dn {\addtolength}
\end{verbatim}

A way to make aliases with keyval. This will come in handy later.

\begin{verbatim}
\def\MH_keyval_alias_with_addon:nnnn #1#2#3#4{
  \@namedef{KV@#1@#2}{\@nameuse{KV@#1@#3}#4}
  \@namedef{KV@#1@#2@default}{\@nameuse{KV@#1@#3@default}#4}}
\end{verbatim}

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

\begin{verbatim}
\def\MH_use_choice_i:nnnn #1#2#3#4{#1}
\def\MH_use_choice_ii:nnnn #1#2#3#4{#2}
\def\MH_use_choice_iii:nnnn #1#2#3#4{#3}
\def\MH_use_choice_iv:nnnn #1#2#3#4{#4}
\end{verbatim}

Scanning for the next character but disallow spaces.

\begin{verbatim}
\def\MH_nospace_ifnextchar:Nnn \MH_nospace_nextchar:\n\MH_nospace_testopt:nn {\long\def\MH_nospace_ifnextchar:Nnn {\MH_nospace_nextchar:}}
\end{verbatim}
\MH_nospace_ifnextchar:Nnn[  
  {#1}  
  {#1[#2]}  
]  
def\MH_nospace_protected_testopt:n #1{  
  \MH_if_meaning:NN \protect\@typeset@protect  
  \expandafter\MH_nospace_testopt:nn  
  \MH_else:  
  \@x@protect#1  
  \MH_fi:  
}

\kernel@ifnextchar The code for the space sensitive peek ahead.
\MH_kernel_xargdef:nwwn  
\MH_nospace_xargdef:nwwn  
\MHPrecedingSpacesOff  
\MHPrecedingSpacesOn  
\def \MH_group_align_safe_begin: {\iffalse{\fi\ifnum0='}{\fi}}  
\def \MH_group_align_safe_end: {\ifnum0='\fi}{\fi}}  
\MH_group_align_safe_begin:  
\MH_group_align_safe_end:  
\providecommand*\MHPrecedingSpacesOff{  
  \MH_preceding_spaces_off:  
  \MH_preceding_spaces_on:  
}

\def \MH_group_align_safe_begin: {\iffalse{\fi\ifnum0='}{\fi}}  
\def \MH_group_align_safe_end: {\ifnum0='\fi}{\fi}}  
\endinput