A couple of things involving environments

Will Robertson

2014/05/04  vo.3

Abstract

This package provides two things, one for document authors and one for macro authors. For the document authors, a new method, \NewEnviron, for defining environments that might be more convenient on occasion. And for the package writers, amsmath’s \collect\BODY command, and a long version of the same, \Collect\BODY.

1 Introduction

This packages provides new commands for defining environments:

\NewEnviron{test}{%\fbox{\parbox{1.5cm}{\BODY}}\color{red}\fbox{\parbox{1.5cm}{\BODY}}} \begin{test}pargrafpargraf\end{test}

\RenewEnviron has the same syntax to redefine a pre-existing environment.

2 For the document author

\LaTeX’s standard method of defining environments looks like this (ignoring arguments for now):

\newenvironment{⟨name⟩}{⟨pre code⟩}{⟨post code⟩}.

The advantage to using environments is that their contents are not treated as a macro argument, so there are fewer restrictions on what can exist inside, and the processing can be more efficient for long pieces of document text.

The disadvantage of environments is that sometimes you really do want to collect up their body and apply some sort of command to the whole thing. This package provides a way to define such environments:

\NewEnviron{⟨name⟩}{⟨macro code⟩}{⟨final code⟩}.

You saw an example in the introduction; the body of the environment is contained within the macro \BODY, and \end{⟨name⟩} is the code executed at \end{⟨name⟩} (more on this later).
2.1 Environment arguments

If you want to use arguments to the environment, these are specified as usual:
\NewEnviron\{\langle name\rangle\}[\langle N. args\rangle][\langle opt. arg.\rangle][\langle macro code\rangle][\langle final code\rangle]

where \langle macro code\rangle has arguments #1, #2, ..., as per traditional \LaTeX \ environ-

ent mandatory and optional arguments.

Here’s an example with two arguments; one optional argument (#1, which
is \today if omitted) and one mandatory argument (#2):

\begin{test}{Title}par\par graf\end{test}
\begin{test}[Yesterday]{Title}par\par graf\end{test}

\begin{test}{Title}par\par graf\end{test}

This is the code executed at $\end{\langle name\rangle}$ of the environment. For the
purposes of this package it is only designed (but is very useful indeed) for cleanup code such as space gobbling in the input text.

This macro sets a default value for the \langle final code\rangle (unless manually spec-

ified) in each subsequent environment created with \NewEnviron. The default is to

define each new environment postfixed by \ignorespacesafterend, like this:
\environfinalcode{\ignorespacesafterend}

Here’s a silly example:

\begin{test}{finish}par\par graf\end{test}

Careful, \environfinalcode cannot contain square brackets without first

protecting them with braces (e.g., \environfinalcode{[end]} will not work but
\environfinalcode{{[end]}} will). This is because the optional argument to
\NewEnviron itself uses square brackets as argument delimiters.
2.3 The \BODY command

Using \BODY as the body of the environment might clash with a command defined by another package. To overcome such conflicts, rename this command with
\environbodyname\langle command\rangle
at which point \NewEnviron will use \langle command\rangle instead of \BODY. Here’s an example:

\NewEnviron{FOO}{\fbox{\BODY}}
\environbodyname\envbody
\NewEnviron{foo}{\fbox{\envbody}}
\begin{FOO}FOO\end{FOO}
\begin{foo}foo\end{foo}

For the macro author

The amsmath package contains a macro that facilitates the functionality in the previous section, which package writers may wish to use directly. The canonical command is \collect@body, which I’ve also defined in \long form to be useable for multi-paragraph environments (\Collect@Body). Here’s how it’s used:

[ hello there ]
\long\def\wrap#1{[#1]}
\newenvironment{test}{\Collect@Body\wrap}{hello}
\begin{test}hello there\end{test}

And here’s a crude example with environment arguments:

[—arg— hello there ]
\long\def\wrap#1{[\arg#1]}
def\arg#1{---#1---\par}
\newenvironment{test}{\Collect@Body\wrap}{hello}
\begin{test}{arg}hello there\end{test}
4 Test

Here’s an example or two to ensure everything that you’d think should work, in fact, does:

\begin{test}
  outer
  *aa*
  inner
  (bb)
  "inner
  (bb)"
  \\
  outer
  *aa*
  inner
  (bb)
  "inner
  (bb)"
\end{test}
File I

environ implementation

This is the package.

\ProvidesPackage{environ}[2014/05/04 v0.3 A new way to define environments]
\RequirePackage{trimspaces}

Change History

v0.2
\NewEnviron: Added.

v0.3
\environbodyname: Works properly and now documented.
\RenewEnviron: Fixed for non-environ commands.

5 Begin

\environbodyname {#1}: control sequence
Changes the control sequence used to represent the environment body in its
definition. Not to be used as a user command; but maybe one day it will be.
Don’t change this after defining any \NewEnviron environments!
\def\environbodyname#1{\def\env@BODY{#1}}
\environbodyname\BODY

\environfinalcode {#1}: code
This is the {⟨code⟩} that’s executed by default at \end{⟨env. name⟩}:
\def\environfinalcode#1{\%
\def\env@finalcode{#1}}
\environfinalcode{\ignorespacesafterend}

\longdef@c LATEX3-inspired shorthands.
\def\longdef@c#1{\%
\expandafter\long\expandafter\def\csname#1\endcsname}

6 \collect@body-related code

\collect@body Now, amsmath defines \collect@body for us. But that package may not be
loaded, and we don’t want to have to load the whole thing just for this one
macro.
\unless\ifdefined\collect@body
\newtoks@envbody
\def\collect@body#1{\%
\@envbody{\expandafter#1}\expandafter{\the@envbody}}%
And now we define our own 'long' version.

\longdef\Collect@Body#1{\@envbody{% 
\expandafter\LetCsName\currenvir\endcsname\collect@body 
\edef\process@envbody{% 
\expandafter\noexpand\csname\currenvir\endcsname} 
\process@envbody 
}
\longdef\Push@Begins#1\begin#2{% 
\ifx\end#2\else 
\b\expandafter\push@begins \fi}%
\longdef\Addto@Envbody#1{% 
\global\@envbody\expandafter{\the\@envbody#1}%
\longdef\Collect@@Body#1\end#2{% 
\edef\begin@stack{\Push@Begins#1\begin\end\expandafter\@gobble\begin@stack} 
\ifx\@empty\begin@stack 
\endgroup 
\@checkend{#2}%
\addto@envbody{#1} 
\else 
\addto@envbody{#1\end{#2}} 
\fi 
\process@envbody 
}
7 User-level syntax

\RenewEnviron \NewEnviron

This is the new one.

\def\NewEnviron{%
  \let\env@newenvironment\newenvironment
  \env@NewEnviron}

\def\RenewEnviron{%
  \let\env@newenvironment\renewenvironment
  \env@NewEnviron}

Input argument parsing     The first optional argument:
\def\env@NewEnviron#1{%
  @ifnextchar[
    {\env@new@i{#1}}
    {\env@new@iii{#1}{}}}

And the second:
\def\env@new@i#1[#2]{%
  @ifnextchar[
    {\env@new@ii{#1}[#2]}
    {\env@new@iii{#1}{[#2]}}}

And the second: (cont.)
\def\env@new@ii#1[#2][#3]{%
  \env@new@iii{#1}{[#2][#3]}}

The final optional argument:
\long\def\env@new@iii#1#2#3{%
  @temptokena={\env@new{#1}{#2}{#3}}%
  @ifnextchar[%
    \the@temptokena
  ]%}
  \expandafter\the\expandafter
  @emptokena\expandafter[\env@finalcode]%
  }}

Environment creation code
\env@new

{#1}: name of the environment
{#2}: possible optional args (either ‘(empty)’ or ‘[N]’ or ‘[N] [default]’)
Save the definition of \texttt{\env@BODY} so we know what to look for.

Define the new environment to Collect its body and execute \texttt{\env@#1@parse} on it.

\texttt{\env@#1@parse} executes the body twice: the first time to save the body while ignoring the arguments; and the second time to process the environment definition itself while ignoring the environment body:

These must be defined on a per-environment basis in order to get the argument gobbling right: (because there are a variable number of arguments)

If \texttt{\env@BODY} were variable, this macro would have to be saved for every environment definition individually; at the moment we just use a global definition. Use \texttt{\trim@spaces} to remove surrounding space:

This is the same as a \texttt{\@gobble@nil} but long and less likely to exist in the environment body: