A couple of things involving environments

Will Robertson

2014/05/04 v0.3

Abstract

This package provides two things, one for document authors and one for macro authors. For the document authors, a new method, `\NewEnviron`, fo 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:

par graf	par graf

```
\NewEnviron{test}{%
  \fbox{\parbox{1.5cm}{\BODY}}\color{red}
  \fbox{\parbox{1.5cm}{\BODY}}}
\begin{test}
  par\par graf
\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 `[`⟨*final code*⟩`]` 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{`⟨*name*⟩`}[`⟨*N. args*⟩`][`⟨*opt. arg.*⟩`]{`⟨*macro code*⟩`}[`⟨*final code*⟩`]`

where `{`⟨*macro code*⟩`}` has arguments `#1`, `#2`, ..., as per traditional LaTeX environment 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`):

Title
par
graf
(May 4, 2014)

Title
par
graf
(Yesterday)

```
\NewEnviron{test}[2][\today]{%
  \fbox{\parbox{3cm}{%
    \textbf{#2}\\
    \BODY\\
    (#1)}}}

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

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

2.2 [⟨*final code*⟩]

This is the code executed at `\end{`⟨*name*⟩`}` 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.

`\environfinalcode` This macro sets a default value for the `[`⟨*final code*⟩`]` (unless manually specified) 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:

par
graf
(finish)

```
\environfinalcode{(finish)}
\NewEnviron{test}{\fbox{\parbox{3cm}{\BODY}}}
\begin{test}
  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

\environbodyname

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\⟨*command*⟩

at which point \NewEnviron will use \⟨*command*⟩ instead of \BODY. Here's an example:

FOO foo

```
\NewEnviron{FOO}{\fbox{\BODY}}
\environbodyname\envbody
\NewEnviron{foo}{\fbox{\envbody}}

\begin{FOO}FOO\end{FOO}
\begin{foo}foo\end{foo}
```

3 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}{}
\begin{test}
  hello

  there
\end{test}
```

And here's a crude example with environment arguments:

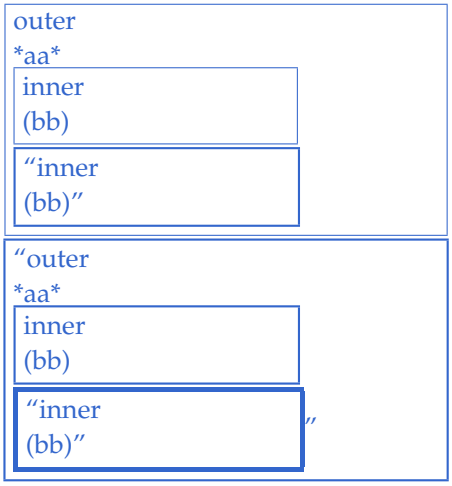
[—arg—
hello
there]

```
\long\def\wrap#1{[\arg#1]}
\def\arg#1{---#1---\par}
\newenvironment{test}{\Collect@Body\wrap}{}
\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:




```
\NewEnviron{test}{%
  \fbox{\parbox{\linewidth-
    0.1cm*\currentgrouplevel}{\BODY}}
  \setlength\fboxrule{2\fboxrule}
  \fbox{\parbox{\linewidth-
    0.1cm*\currentgrouplevel}{``\BODY''}}}

\begin{test}
  outer\par
  \def\tmp#1{*#1*}%
  \tmp{aa}\par
  \begin{test}
    inner\par
    \def\tmp#1{(#1)}\tmp{bb}
  \end{test}
\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

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}}%
```

```
    \edef\process@envbody{\the\@envbody\noexpand\end{\@currenvir}}%
    \@envbody={}%
    \def\begin@stack{b}%
    \begingroup
    \expandafter\let\csname\@currenvir\endcsname\collect@@body
    \edef\process@envbody{%
      \expandafter\noexpand\csname\@currenvir\endcsname}%
    \process@envbody
  }
  \def\push@begins#1\begin#2{%
    \ifx\end#2\else
      b\expandafter\push@begins
    \fi}
  \def\addto@envbody#1{%
    \global\@envbody\expandafter{\the\@envbody#1}}
  \def\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}
\fi
```

`\Collect@Body` And now we define our own 'long' version.

```
\long\def\Collect@Body#1{%
  \@envbody{\expandafter#1\expandafter{\the\@envbody}}%
  \edef\process@envbody{\the\@envbody\noexpand\end{\@currenvir}}%
  \@envbody={}%
  \def\begin@stack{b}%
  \begingroup
  \expandafter\let\csname\@currenvir\endcsname\Collect@@Body
  \edef\process@envbody{%
    \expandafter\noexpand\csname\@currenvir\endcsname}%
  \process@envbody
}
\long\def\Push@Begins#1\begin#2{%
  \ifx\end#2\else
    b\expandafter\Push@Begins
  \fi}
\long\def\Addto@Envbody#1{%
  \global\@envbody\expandafter{\the\@envbody#1}}
\long\def\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 This is the new one.

\NewEnviron

```
\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
      \@temptokena\expandafter[\env@finalcode]%
  }}
```

Environment creation code

\env@new `{#1}`: name of the environment
`{#2}`: possible optional args (either '⟨*empty*⟩' or '`[N]`' or '`[N][default]`')

`{#3}`: environment code
`[#4]`: final code

```
\long\def\env@new#1#2#3[#4]{%
```

Save the definition of `\env@BODY` so we know what to look for.

```
  \longdef@c{env@#1@BODY\expandafter}\expandafter{\env@BODY}%
```

Define the new environment to Collect its body and execute `env@#1@parse` on it.

```
  \env@newenvironment{#1}{%
    \expandafter\Collect@Body\csname env@#1@parse\endcsname
  }{#4}%
```

`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:

```
  \longdef@c{env@#1@parse}##1{%
    \csname env@#1@save@env\endcsname##1\env@nil
    \csname env@#1@process\endcsname##1\env@nil}%
```

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)

```
  \expandafter\let\csname env@#1@save@env\endcsname\relax
  \expandafter\let\csname env@#1@process\endcsname\relax
  \expandafter\newcommand
    \csname env@#1@save@env\endcsname#2{%
      \expandafter\expandafter\expandafter
        \env@save\csname env@#1@BODY\endcsname}%
  \expandafter\newcommand
    \csname env@#1@process\endcsname#2{#3\env@ignore}%
}
```

`\env@save` If `\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 `\trim@spaces` to remove surrounding space:

```
\long\def\env@save#1#2\env@nil{%
  \edef#1{%
    \unexpanded\expandafter
      \expandafter\expandafter{\trim@spaces{#2}}}}
```

This is the same as a `\@gobblenil` but long and less likely to exist in the environment body:

```
\long\def\env@ignore#1\env@nil{}
```