A couple of things involving environments

Will Robertson

2007/09/23 v0.1

Abstract

This package provides two things, one for document authors and one for macro authors. For the document authors, a new method of defining environments that might be more convenient on occassion. And for the package writers, amsmath's `\collect@body` command, and a long version of the same, `\Collect@Body`.

1 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 is not treated as a macro argument, so there are less 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 its body and apply some sort of command to the whole thing. This package provides a way to define such environments,

`\NewEnvironment{`⟨*name*⟩`}{`⟨*macro code*⟩`}` ,

where `{`⟨*macro code*⟩`}` has argument `#1` as everything inside the environment. `\RenewEnvironment` can be used to redefine a preëxisting environment.

As an example, consider putting a box around an environment. This requires more than a one-line solution without this package[1], but now we can write

```
\NewEnvironment{test}{\fbox{\parbox{3cm}{#1}}}
\begin{test}
  par\par
  graf
\end{test}
```

par
graf

[1] And there're packages to do things much more nicely than I'm showing here anyway.

Now, this kind of environment definition makes collecting arguments a little cumbersome, but it's certainly possible. Arguments are defined with a separate macro that 'gobbles up' the arguments inside the environment before the body is passed to {⟨*macro code*⟩}.

\EnvironArgs{⟨*name*⟩}[⟨*N. args*⟩][⟨*opt. arg.*⟩]{⟨*arg. macro code*⟩}

This follows the same syntax of defining a macro with several arguments and a possible optional argument at the beginning. Here's an example:

```
\NewEnvironment{test}{(#1)\tmp}
\EnvironArgs{test}[2][before]{\def\tmp{\par---#1/#2---}}
\begin{test}{after}
  par

  graf
\end{test}
```

(par graf) —before/after—

I've tried to ensure that whitespace is ignored at the appropriate places; without this additional code, there would be a space before 'par' and after 'graf' in the examples above.

Note that arguments to the environment have to be explicitly passed into the macro code defining what the environment actually does. This is a bit of a shame, and a more convenient syntax could be

```
\NewEnv{test}[2][before]{\EnvBody\par---#1/#2---}
```

This is a probable future addition I'll make to the package.

Oh, one last thing. These environments are defined to be ended by `\ignorespacesafterend`, which means that if they're used in a paragraph then the `\end{...}` command will gobble space after it. Let me know if this is a problem. It's kind of the way I like to do things but it might not fit everyone.

2 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 an 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}
```

File I
environ implementation

This is the package.

```
\ProvidesPackage{environ}[2007/09/23 v0.1 A new environment syntax]
```

`\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\@emptytoks
  \newtoks\@envbody
  \def\collect@body#1{%
    \@envbody{\expandafter#1\expandafter{\the\@envbody}}%
    \edef\process@envbody{\the\@envbody\noexpand\end{\@currenvir}}%
    \@envbody\@emptytoks \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\@emptytoks \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}
```

`\NewEnvironment` `#1` : Environment name
`#2` : Macro definition applied to env. body

Here's our new environment definition macro. First of all wrap it up appropriately for new- or renew-

```
\newcommand\NewEnvironment{%
  \let\env@newenvironment\newenvironment
  \let\env@newcommand\newcommand
  \Make@Environment}
\newcommand\RenewEnvironment{%
  \let\env@newenvironment\renewenvironment
  \let\env@newcommand\renewcommand
  \Make@Environment}
```

And here we go:

```
\newcommand\Make@Environment[2]{%
```

Initial 'argument' parser, does nothing but remove leading space:

```
  \expandafter\let\csname env@args@#1\endcsname\ignorespaces
```

We use `\Collect@Body` to grab the argument (always `\long`)

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

Now precede the env. body by the argument parsing command, which may or may not be defined in `\EnvironArgs` (and `\unskip` removes trailing space)

```
\expandafter\env@newcommand\csname env@@#1\endcsname[1]{%
  \csname env@@@#1\endcsname{%
    \csname env@args@\@currenvir\endcsname##1\unskip}}%
```

And then pass it all off to the environment macro (`#2`),

```
\expandafter\env@newcommand\csname env@@@#1\endcsname[1]{#2}}
```

`\EnvironArgs` `#1` : Environment name
`[`⟨*Number of arguments*⟩`]` `[`⟨*Optional argument*⟩`]` `#2` : Argument macro code
Tedious argument parsing:

```
\newcommand\EnvironArgs[1]{%
  \@ifnextchar[
    {\Env@Args{#1}}
    {\Env@Args{#1}[0]}}
```

Tedious argument parsing:

```
\long\def\Env@Args#1[#2]{%
  \@ifnextchar[
    {\Env@@@Args{#1}[#2]}
    {\Env@@Args{#1}[#2]}}
```

This is when there is no optional argument. In this case and the next, we simply define a command that is inserted when the argument body is processed (see `\NewEnvironment`). `\ignorespaces` removes leading space after the arguments.

```
\long\def\Env@@Args#1[#2]#3{%
  \expandafter\renewcommand\csname env@args@#1\endcsname[#2]{%
    #3\ignorespaces}}
```

Same as above when there is an optional argument:

```
\long\def\Env@@@Args#1[#2][#3]#4{%
  \expandafter\renewcommand\csname env@args@#1\endcsname[#2][#3]{%
    #4\ignorespaces}}
```