saveenv — Save environment content verbatim *

user202729

Released 2022/12/23

Abstract

Allow users to collect environment content verbatim.

1 Motivation

This package provides tools to create your own verbatim environments, and works in all values of `\endlinechar`.

2 Comparison to existing similar packages

- scontents package is similar; however it does not allow programmer to programmatically retrieve the macro content, only do a limited number of actions with them (execute, typeset as verbatim code, etc.).
- fancyvrb, xsimverb and verbatimcopy provides some internal macro for defining similar environments, however they're internal and/or too specialized (writes to file etc.).
- filecontentsdef and newverbs provides very similar facilities (`filecontentsdefmacro` environment and `\Collectverbenv` command respectively); however it requires `\endlinechar=13` and does not support `\ExplSyntaxOn` environment.
- verbatim provides facilities to define custom verbatim environment that processes the content line-by-line, however the interface is more complicated.

  For comparison, the following code

```
\ExplSyntaxOn
\makeatletter
\cs_generate_variant:Nn \tl_build_gput_right:Nn {No}
\NewDocumentEnvironment{}{}{
    \@bsphack

    \endlinechar=`\^^M ~  \let\do\@makeother \dospecials
    \@makeother \^^I
    %\cctab_select:N \c_other_cctab
    % the above cannot be used, as \verbatim@ relies on
```

*This file describes version v0.0.1, last revised 2022/12/23.

```
    % the environment name having letter catcode

    \catcode `\^^M \active

    \tl_build_gbegin:N \__all_data
    \def\verbatim@processline{
        \tl_build_gput_right:No \__all_data{\the\verbatim@line ^^J}
    }
    \verbatim@
}{
    \tl_build_gend:N \__all_data
    \str_gset:NV \__all_data \__all_data

    % can do something else with \__all_data here

    \@esphack
}
\makeatother
\ExplSyntaxOff
```

defines an environment that saves the data similar to `saveenv` environment described below (with the overhead of `\tl_build_*` functions), but inside `\ExplSyntaxOn` environment it generates a spurious space at the beginning of the string.

3 Main environment

`saveenv` Environment that saves its body.

For example, the following code

```
\begin{saveenv}{\data}
123
456
\end{saveenv}
```

will save the string `123`⟨*newline*⟩`456`⟨*newline*⟩ globally into `\data`.

Remark: it is consistent to keep the trailing newline, as

```
\begin{saveenv}{\data}
\end{saveenv}
```

will make `\data` empty, and

```
\begin{saveenv}{\data}

\end{saveenv}
```

will make `\data` consist of a single ⟨*newline*⟩.

The braces around `{\data}` is optional; however, in the unlikely case if `\endlinechar` has the "letter" catcode, it might be absorbed and gives unexpected result.

A newline is represented as a token with charcode 10 (`^^J`) and catcode other.

Note that this is unusual, for comparison filecontentsdef stores it as an active `^^M`.

In the current implementation, it can be used in functions such as `\iow_now:Nn` and newline characters will appear as newline. (the number 10 is hard coded in the implementation of `\iow_now:Nn`, nevertheless I'm not sure if this behavior is guaranteed. Manually replacing them with `\iow_newline:` tokens and `x`-expand the result would be guaranteed to work)

The assignment is global, and done before the macro `\endsaveenv` is executed.

The data is represented as an expl3-string, that is, a sequence of tokens with catcode 12 (for non-space characters) or 10 (for space character).

In other words, the token list is equal to its own `\detokenize`.

`saveenvghost` Similar to above; however the content inside is still typesetted/executed, and the SyncTEX information is preserved.

Note that this environment is implemented by reading the whole file from the beginning, therefore there might be some performance hit for large files and multiple usages of the environment. Use sparingly.

`saveenvkeeplast` Similar to above; however the final newline and the space characters before `\end{`⟨*environment name*⟩`}` are preserved.

For example, the example above with `saveenv` replaced with `saveenvkeeplast` will save the string `123`⟨*newline*⟩`456`⟨*newline*⟩ into `\data` instead.

4 Reinsert environments

Sometimes it's desirable to execute something (e.g. do some local assignments) after the group ends.

There are a few options:

- use `\aftergroup`,
- close the group, execute the code and open a new one (remember to preserve the value of `\@currenvir`),
- use one of the environments below.

`saveenvreinsert` Environment that takes two arguments, the ⟨*str var*⟩ to be set and the code to be put after the group end.

Usage example: If the following code is executed

```
\begin{saveenvreinsert}{\data}{\myfunction {args etc.}}
123
456
\end{saveenvreinsert}
some other content
```

the effect would be identical to as if `\data` is set to `123`⟨*newline*⟩`456`⟨*newline*⟩, then the following code is executed

```
\myfunction {args etc.}some other content
```

If the second argument is empty, this environment is identical to the `saveenv` environment.

Note that in the example above `\myfunction` is executed after the group ends, while any code in the second block of the `\NewDocumentEnvironment` definition would be executed before the group ends (such as in the example below), thus any local assignment will not persist.

`saveenvkeeplastreinsert` Same as above, but the trailing whitespaces (after the last newline) are preserved.

5 Extending the environments

All of the environments are extensible.

Follow the instruction in https://tex.stackexchange.com/q/14683/250119 to define an environment based on an existing one.

For example, the following definition

```
\ExplSyntaxOn
\NewDocumentEnvironment{custom}{}{
    \saveenv \data
}{
    \endsaveenv
    \tl_show:N \data
}
```

will define an environment `custom` that prints out the content inside the environment using `\tl_show:N`.

Note that in this case `\tl_show:N \data` is executed before the group ends.

6 Limitation

- This package does not process the content line-by-line. Therefore:
  - The memory consumption might be larger than approaches that process the content line-by-line, in case you only need to do something with the line.
    Nevertheless, in modern computers, the overhead is negligible.
  - In case it's desired to typeset the content afterwards, it's difficult to preserve the SyncTeX data.
    (although with LuaLaTeX it's possible, see rescansync package)
- Currently nested environments with the same name are not supported (unlike scontents package).
- Note that there must be nothing after the `\begin{`⟨*environment name*⟩`}` or the `\end{`⟨*environment name*⟩`}` line.
- Note that `\end{`⟨*environment name*⟩`}` must not be in the middle of any line.

- The `\endlinechar` must not be tokenized in advance (or if it is, its catcode must be 12/other). This might happen when for example your environment looks ahead for optional argument. See https://tex.stackexchange.com/q/649331/250119 for an example and one way how to fix it.

  Alternatively, you can

  – manually remove the end line token,

  – set `\endlinechar=-1\relax` inside the environment before calling `\saveenv`.

  The value of `\endlinechar` will automatically be reset after the group of the environment is closed.

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.