rescansync — Re-scan tokens with synctex information *

user202729

Released 2022/07/09

Abstract

Allow users to execute saved code to typeset text while preserving SyncTEX information.

1 Simple interface

`rescansyncSaveenvPacked` Saves the environment body into the specified macro after the environment name, with additional information required to execute it later.

The macro is stored in some internal format. If the user want to modify the content, use one of the advanced interface described below.

Requires the saveenv and currfile package to be loaded. If you want to save the environment body using some other package, use the advanced interface.

`\rescansyncPacked` `\rescansyncPacked` ⟨*macro*⟩

Execute the stored content in ⟨*macro*⟩.

The content must be stored with `rescansyncSaveenvPacked` environment or similar.

To give an usage example: the following code

```
% the following line will save the content "123 456"
% and some auxiliary information into the macro \mycontent.
\begin{rescansyncSaveenvPacked}{\mycontent}
123

456
\end{rescansyncSaveenvPacked}

% the following line will typeset 789 as usual.
789

% the following line will typeset "123 456".
\rescansyncPacked{\mycontent}
```

will typeset “789 123 456” as 3 separate paragraphs, with SyncTEX information preserved.

If the engine is not LuaTEX, there will still be limited SyncTEX information that points to a temporary file, but the line number is preserved.

`rescansyncSaveenvghostPacked` The usage of this environment is similar to the `rescansyncSaveenvPacked` above,

*This file describes version v0.0.0, last revised 2022/07/09.

except that the content is still typeset with SyncTeX information preserved while it's stored.

In the example above, if this environment is used instead

```
\begin{rescansyncSaveenvghostPacked}{\mycontent}
123

456
\end{rescansyncSaveenvghostPacked}

789

\rescansyncPacked{\mycontent}
```

the content typeset will be "123 456 789 123 456".

As with `saveenvghost` environment (read saveenv package documentation for more details), the SyncTeX information of the first section is guaranteed to be preserved, but there might be some performance hit.

2 Advanced interface

`\rescansync:nn` `\rescansync:nn {⟨content⟩} {⟨line offset⟩}`

Execute (rescan) the {⟨*content*⟩}. Requires currfile.

Details: {⟨*content*⟩} will be detokenized, and characters with char code 10 will be interpreted as a line break (the behavior is inherited from `\iow_now:Nn` function. As a consequence, it's not allowed to write literal character with char code 10 to the file; however this is not very useful regardless because on some operating systems this is equivalent to a real newline)

{⟨*line offset*⟩} is some non-negative number. If it's 0 then the first line of {⟨*content*⟩} corresponds to the first line in the target file.

Remark: If newverbs, xparse or filecontentsdef is used to collect multiline verbatim environment, they have the line separation characters separated by character with char code 13 by default, you need to manually replace them.

Even though the SyncTeX information points to the correct file, if there's some error the temporary file name (which has the form RS⟨*number*⟩-⟨*file name*⟩ will be shown.

Engines other than LuaTeX has the limitations described above.

`\rescansync:nnn` `\rescansync:nnn {⟨content⟩} {⟨line offset⟩} {⟨file name⟩}`

Similar to above, but use {⟨*file name*⟩} as the temporary file name. (only important in error messages, the SyncTeX information points to the file that the content is scanned.)

`\rescansync:nnnn` `\rescansync:nnnn {⟨content⟩} {⟨line offset⟩} {⟨file name⟩} {⟨SyncTeX tag⟩}`

Similar to above, but you're allowed to specify the SyncTeX tag.

More details: each file ever read has an associated SyncTeX tag value, which is a number. The SyncTeX information will point to that file.

On engines other than LuaTeX, this feature is not supported and the function returns empty.

\rescansync_gettag: ⋆ `\rescansync_gettag:`

Function that fully expands to some token list that represents the SyncTEX tag that corresponds to some file.

Note that if a file is `\input` more than once (or if `\file_get:nnN` expl3 function is used on that file, which internally uses the TEX `\input` primitive), the tag value will be different and forward search may fail to work.

As a full example, assume there's a file named `a.tex`:

```
1 \tl_set:Nx \mytag {\rescansync_gettag:}
```

After it's executed, a file `b.tex` executes:

```
1 \rescansync:nnnV {abcdef} {3} {rescanned-a} \mytag
```

then the text `abcdef` will be typesetted, the SyncTEX information points to line $1+3=4$ in the file `a.tex` (as the code `abcdef` is on the first line of the `{`⟨*content*⟩`}`, and the line offset is 3).

If there's any error while typesetting `abcdef`, the error file name will be reported as for example, `RS1-rescanned-a.tex` (the number `1` might vary) instead of `a.tex`.

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.