

SCONTENTS

STORES L^AT_EX CONTENTS

V2.7 2026-06-01^{*}

©2019–2026 by Pablo González L[†]

CTAN: <https://www.ctan.org/pkg/scontents>

 <https://github.com/pablgonz/scontents>

Abstract

This package allows to store L^AT_EX code, including “*verbatim*”, in *sequences* using the `l3seq` module of `expl3`. The *stored content* can be used as many times as desired in the document, additionally you can write to *external files* or show it in *verbatim style*.

Contents

1	Description of the package	1	5.9	The environment <code>verbatimsc</code>	7
2	Motivation and Acknowledgments	1	6	Other commands provided	7
3	License and Requirements	2	6.1	The command <code>\countsc</code>	7
4	The <code>scontents</code> package	2	6.2	The command <code>\cleanseqsc</code>	8
4.1	Installation	2	7	The <code>scontents</code> package in action	8
4.2	Loading and options	2	8	Examples	8
4.3	The TAB character	2	8.1	From <code>answers</code> package	9
4.4	Configuration of the options	3	8.2	From <code>filecontentsdef</code> package	9
4.5	Options Overview	3	8.3	From TeX-SX	9
5	User interface	3	8.4	Customization of <code>verbatimsc</code>	11
5.1	The environment <code>scontents</code>	4	8.5	The command <code>\mergesc</code> in action	14
5.2	The command <code>\newenvsc</code>	5	8.6	The tagged PDF example	15
5.3	The command <code>\Scontents</code>	5	9	Change history	16
5.4	The command <code>\getstored</code>	6	10	Index of Documentation	18
5.5	The command <code>\foreachsc</code>	6	11	References	18
5.6	The command <code>\typestored</code>	6	12	Implementation	20
5.7	The command <code>\meaningsc</code>	6	13	Index of Implementation	49
5.8	The command <code>\mergesc</code>	7			

1 Description of the package

The `SCONTENTS` package allows to *store contents* in *sequences* or *external files*. In some ways it is similar to the `filecontentsdef` package, with the difference in which the *content* is stored. The idea behind this package is to get an approach to ConT_EXt “*buffers*” by making use *sequences*.


2 Motivation and Acknowledgments

In L^AT_EX there is no direct way to record content for later use, although you can do this using `\macro`, recording *verbatim content* is a problem, usually you can avoid this by creating external files or boxes.

The general idea of this package is to try to imitate this implementation “*buffers*” that has ConT_EXt which allows you to save content in memory, including *verbatim*, to be used later. The `filecontentsdef`[2] package solves this problem and since `expl3`[1] has an excellent way to manage data, ideas were combined giving rise to this package.

This package would not be possible without the great work of JEAN FRANÇOIS BURNOL who was kind enough to take my requirements into account and add the `filecontentsdefmacro` environment to `filecontentsdef` package. Also a special thanks to Phelype Oleinik who has collaborated and adapted a large part of the code and all L^AT_EX team for their great work and to the different members of the TeX-SX community who have provided great answers and ideas. Here a note of the main ones:

1. Stack datastructure using LaTeX
2. LaTeX equivalent of ConT_EXt buffers
3. Storing an array of strings in a command
4. Collecting contents of environment and store them for later retrieval
5. Collect contents of an environment (that contains *verbatim* content)

 Starting with version 2.3 the `SCONTENTS` package is fully compatible with *tagged* PDF and will have L^AT_EX release 2026-06-01 (T_EX Live 2026) and update `expl3` release as a minimum requirement.

^{*}This file describes a documentation for v2.7, last revised 2026-06-01.

[†]E-mail: pablgonz@educarchile.cl.

3 License and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (lpl), version 1.3 or later (<https://www.latex-project.org/lpl.txt>). The software has the status “maintained”.

The package `scontents` requires an updated version of \LaTeX to work (minimum required to \LaTeX release 2026-06-01 and `expl3` update). This package can be used with `plain`, `context`, `xelatex`, `lualatex`, `pdflatex` and the classical workflow `latex»dvips»ps2pdf`.

4 The scontents package

4.1 Installation

The package `scontents` is present in \TeX Live and $\text{MiK}\text{\TeX}$, use the package manager to install. For manual installation, download [scontents.zip](#) and unzip it, run `luatex scontents.ins` and move all files to appropriate locations, then run `mktexlsr`. To produce the documentation with source code run `luatex scontents.ins` and `lualatex scontents.dtx` three times.

<code>scontents.tex</code>	»	<code>TDS:tex/generic/scontents/</code>
<code>scontents-code.tex</code>	»	<code>TDS:tex/generic/scontents/</code>
<code>scontents.sty</code>	»	<code>TDS:tex/latex/scontents/</code>
<code>t-scontents.mkiv</code>	»	<code>TDS:tex/context/third/scontents/</code>
<code>scontents.pdf</code>	»	<code>TDS:doc/latex/scontents/</code>
<code>README.md</code>	»	<code>TDS:doc/latex/scontents/</code>
<code>scontents.dtx</code>	»	<code>TDS:source/latex/scontents/</code>
<code>scontents.ins</code>	»	<code>TDS:source/latex/scontents/</code>

4.2 Loading and options

The package is loaded in the usual way:

For \LaTeX users

```
\usepackage[⟨key = val⟩]{scontents}
```

For plain \TeX users


```
\input scontents.tex
```

For $\text{Con}\text{\TeX}$ t users


```
\usemodule{scontents}
```

- The package options are not available for plain \TeX and $\text{Con}\text{\TeX}$ t and must be passed using `\setupsc` (see §4.4). $\text{Con}\text{\TeX}$ t users should use `-luatex`, the implementation does not support LuaMetaTeX.

4.3 The TAB character

Some users use the horizontal TAB ‘’ character from keyboard to indent the source code of the document and depending on the text editor used, some will use real TAB (*hard tabs*), others with *soft tabs* (`\u` or `\uuu`) or both.

At first glance it may seem the same, but the way in which TAB character (*hard tabs*) are processed according to the context in which they are found within a file, both in *⟨reading⟩*¹ and *⟨writing⟩*² are different and may have adverse consequences.

In a standard \LaTeX document, the character TAB ‘’ are treated as explicit spaces (in most contexts) and is the behavior when *⟨stored content⟩*, but when *⟨writing files⟩* these are preserved.

With a \TeX Live distribution, the TAB character is *printable* for `latex`, `pdflatex` and `lualatex`, but if you use `xelatex` you must add the `-8bit` option on the command line, otherwise you will get \TeX character TAB ‘`^^I`’ in the *⟨output file⟩*.

As a general recommendation “Do not use TAB character unless strictly necessary”, for example within a “*verbatim environment*” that supports this character such as `Verbatim` of the packages `fancyvrb`[5], `fvextra`[7] or `lstlisting` of the package `listings`[6] or when you want to generate a `MakeFile` file.

¹Check the answer given by Ulrich Diez in [Keyboard TAB character in argument v \(xparse\)](#).

²Check the answer given by Enrico Gregorio in [How to output a tabulation into a file](#).

4.4 Configuration of the options

Most of the options can be passed directly to the package or using the command `\setupsc`. All boolean keys can be passed using the equal sign ‘=’ or just the name of the key, all unknown *<keys>* will return an error. In this section are described some of the options, a summary of all options is shown in §4.5.

`\setupsc` `\setupsc{<key = val>}`

The command `\setupsc` sets the *<keys>* in a global way, it can be used both in the preamble and in the body of the document as many times as desired. However, options set in the declaration of an environment (with `\newenvsc`) have precedence over options set with `\setupsc`.

Options in the optional arguments of environments and commands have the highest precedence, overriding both options in `\newenvsc` and in `\setupsc`.

`debug-var = {<true | false>}` default: *true*

Determines whether the *<argument>* passed to commands or the *<body>* passed to provided environments is written to the .log file. This *<key>* is only available as a package option or using `\setupsc`.

`verb-font = {}` default: `\ttfamily`

Sets the ** used to display the *<stored content>* for `\typestored`, `\mergesc` and `\meaningsc` commands. This *<key>* is only available as a package option or using `\setupsc`.

`store-all = {<seq name>}` default: *not used*

It is a *<meta-key>* that sets the `store-env` key of the `scontents` environment and the `store-cmd` key of the `\Scontents` command. This *<key>* is only available as a package option or using `\setupsc`.

`overwrite = {<true | false>}` default: *false*

Sets whether the *<files>* generated by `write-out` and `write-env` from the `scontents` environment will be rewritten. This *<key>* is available as a package option, for `\setupsc`, for `\Scontents*`, for the base environment `scontents` and all environments created using `\newenvsc`.


`print-all = {<true | false>}` default: *false*

It is a *<meta-key>* that sets the `print-env` key of the `scontents` environment and the `print-cmd` key of the `\Scontents` command. This *<key>* is only available as a package option or using `\setupsc`.

`force-eol = {<true | false>}` default: *false*

Sets if the *last end of line* for the *<stored content>* is hidden or not. This *<key>* is necessary only if the *last line* is the closing of some environment defined by `fancyvrb`[5] or `fvextra`[7] packages as `\end{Verbatim}` or another environment that does not support a comments ‘%’ after closing `\end{«env»}%`. This *<key>* is available for the base environment `scontents`, the `\Scontents*` command and all environments created using `\newenvsc`.

`width-tab = {<integer>}` default: *1*

Sets the equivalence in *<spaces>* for the character TAB ‘’ used when displaying *<stored content>* in *verbatim style*. The value must be a *<positive integer>*. This *<key>* is available for `\typestored`, `\mergesc` and `\meaningsc` commands.

4.5 Options Overview

key	package	<code>\setupsc</code>	<code>scontents</code>	<code>\Scontents</code>	<code>\Scontents*</code>	<code>\typestored</code>	<code>\mergesc</code>	<code>\meaningsc</code>
<code>debug-var</code>	✓	✓	✗	✗	✗	✗	✗	✗
<code>verb-font</code>	✓	✓	✗	✗	✗	✗	✗	✗
<code>store-all</code>	✓	✓	✗	✗	✗	✗	✗	✗
<code>overwrite</code>	✓	✓	✓	✗	✓	✓	✓	✓
<code>print-all</code>	✓	✓	✗	✗	✗	✗	✗	✗
<code>force-eol</code>	✓	✓	✓	✗	✓	✗	✗	✗
<code>width-tab</code>	✓	✓	✗	✗	✗	✓	✓	✓
<code>store-env</code>	✓	✓	✓	✗	✗	✗	✗	✗
<code>store-cmd</code>	✓	✓	✗	✓	✓	✗	✗	✗
<code>print-env</code>	✓	✓	✓	✗	✗	✗	✗	✗
<code>print-cmd</code>	✓	✓	✗	✓	✓	✓	✓	✓
<code>write-env</code>	✗	✗	✓	✗	✗	✗	✗	✗
<code>write-cmd</code>	✗	✗	✗	✗	✓	✗	✗	✗
<code>write-out</code>	✗	✗	✓	✗	✓	✓	✓	✓
<code>typestored</code>	✗	✗	✗	✗	✗	✗	✗	✓
<code>meaningsc</code>	✗	✗	✗	✗	✗	✗	✗	✓

5 User interface

The user interface consists in `scontents` environment, `\Scontents` and `\Scontents*` commands to *<stored content>*, `\getstored` command to get the *<stored content>*, `\typestored` and `\mergesc` commands to print *verbatim style* the *<stored content>* along with other utilities described in this documentation.

5.1 The environment scontents

```
scontents \begin{scontents}[⟨key=val⟩]
          ⟨body env⟩
\end{scontents}
```

The `scontents` environment processes $\{ \langle body \ env \rangle \}$ and “stores” it in a $\langle sequence \rangle$ or “writes” it to an $\langle external \ file \rangle$ if desired, including *verbatim material*. After the package has been loaded, the environment can be used both in the preamble and in the body of the document.

For the correct operation `\begin{scontents}` and `\end{scontents}` must be in *different lines*, all $\langle keys \rangle$ must be passed separated by commas and *without spaces* ‘`␣`’ of the start of the environment.

Comments ‘`%`’ or *any character* after `\begin{scontents}` or $[\langle key = val \rangle]$ on the *same line* are NOT supported, the package will return an “error” message if this happens. In a similar way comments ‘`%`’ or *any character* after `\end{scontents}` on the *same line* the package will return a “warning” message.

The environment can be “nested” if it is properly balanced and does not appear “literally” in commented lines or in some *verbatim* environment or command. As an example:

```
\begin{scontents}[store-env=outer]
This text is in the outer environment (before nested).
\begin{scontents}[store-env=inner]
This text is found in the inner environment (inside of nested).
\end{scontents}
This text is in the outer environment (after nested).
\end{scontents}
```

Of course, the $\langle stored \ content \rangle$ in the *sequence* $\{ \langle inner \rangle \}$ is *only available* after $\langle stored \ content \rangle$ in the *sequence* $\{ \langle outer \rangle \}$ one has been retrieved, either by using the key `print-env` or `\getstored` command.

- It is advisable to $\langle stored \ content \rangle$ within *sequences* with different names, so as not to get lost in the order (position) in which they are stored.

Notes for plain T_EX and ConT_EXt users

In plain T_EX there is not environments as in L_AT_EX. Instead of using the environment `scontents`, one should use a “pseudo environment” delimited by `\scontents` and `\endscontents`.

```
\scontents \scontents[⟨key=val⟩]
\endscontents ⟨body env⟩
\endscontents
```

ConT_EXt users should use `\startsccontents` and `\stopsccontents`.

```
\startsccontents \startsccontents[⟨key=val⟩]
\stopsccontents ⟨body env⟩
\stopsccontents
```

Options for environment

The environment options can be configured globally using option in package or the `\setupsc` command and locally using $[\langle key = val \rangle]$ in the environment. The key `force-eol` is available for this environment.

`store-env = {⟨seq name⟩}` default: *contents*

Sets the *name* of the *sequence* in which the $\{ \langle body \ env \rangle \}$ will be stored. If the *sequence* does not exist, it will be created globally.

`print-env = {⟨true | false⟩}` default: *false*

Sets if the $\langle stored \ content \rangle$ is displayed or not at the time of running the environment. The $\langle stored \ content \rangle$ is extracted from the *sequence* $\{ \langle seq \ name \rangle \}$ set by the key `store-env` at the time it is executed.

`write-env = {⟨file.ext⟩}` default: *not used*

Sets the *name* of the $\langle external \ file \rangle$ in which the $\{ \langle body \ env \rangle \}$ of the environment will be written. The $\langle file.ext \rangle$ will be created in the working directory and the $\{ \langle body \ env \rangle \}$ will be *stored* in the *sequence* set by the key `store-env` at the time it is executed. If $\langle file.ext \rangle$ does not exist, it will be created or overwritten if the `overwrite` key is used.

`write-out = {⟨file.ext⟩}` default: *not used*

Sets the *name* of the $\langle external \ file \rangle$ in which the $\{ \langle body \ env \rangle \}$ of the environment will be written. The $\langle file.ext \rangle$ will be created in the working directory and the $\{ \langle body \ env \rangle \}$ will NOT be *stored* in the *sequence* set by the key `store-env` at the time it is executed. If $\langle file.ext \rangle$ does not exist, it will be created or overwritten if the `overwrite` key is used.

- In the keys `write-env` and `write-out` the character TAB will be written in `⟨file.ext⟩`, relative or absolute paths are not supported. X₃TeX users using character TAB must add `-8bit` at the command line, otherwise you will get T_EX character TAB ‘^^I’ in `⟨file.ext⟩`.

5.2 The command `\newenvsc`

```
\newenvsc \newenvsc{⟨env name⟩}[⟨initial keys⟩]
```

The command `\newenvsc` allows you to create `⟨new environments⟩` based on the same characteristics of the `scontents` environment. The values entered in `[⟨initial keys⟩]` will be considered as the default values for this new environment and the valid `⟨keys⟩` are `store-env` and `print-env`. For example:

```
\newenvsc{myenvstore}[store-env=myseq,print-env=false]
```

created the environment `myenvstore` that `⟨stored content⟩` in the `sequence {⟨myseq⟩}` and will NOT display the `⟨stored content⟩` when the environment it is executed.

5.3 The command `\Scontents`

```
\Scontents \Scontents[⟨key = val⟩]{⟨argument⟩}
\Scontents* [⟨key = val⟩]{⟨argument⟩}
\Scontents* [⟨key = val⟩]⟨del⟩⟨argument⟩⟨del⟩
```

The `\Scontents` command reads the `{⟨argument⟩}` in standard mode and “stores” it in the `sequence` set by the key `store-cmd` at the time it is executed. It is not possible to pass *verbatim things*, but it is possible to use the implementation of `\Verb` delimited by *braces* ‘{’ provided by the `fvextra`[7] package for *verbatim one line* and `\lstinline` from `listings`[6] package, but it is preferable to use the *starred argument* ‘*’.

The `\Scontents*` command reads the `{⟨argument⟩}` under *verbatim catcode* regimen and “stores” it in the `sequence` set by the key `store-cmd` at the time it is executed. If its “first” delimiter is a *brace* ‘{’, it will be assumed that the `{⟨argument⟩}` is nested within *braces*. Otherwise it will be assumed that `{⟨argument⟩}` is delimited by that first delimiter `⟨del⟩` like command `\verb`. Blank lines are preserved, escaped braces ‘\{’ and ‘\}’ must also be balanced if the `{⟨argument⟩}` is used with *braces* and character TAB typed from the keyboard are converted into spaces.

The *starred argument* ‘*’ and `[⟨key = val⟩]` must NOT be separated by *spaces* ‘ ’ between them and the command. Both versions can be used anywhere in the document and cannot be used as an `{⟨argument⟩}` for other command.

Options for command

The command options can be configured globally using option in package or the `\setupsc` command and locally using `[⟨key = val⟩]`.

```
store-cmd = {⟨seq name⟩} default: contents
```

Sets the *name* of the *sequence* in which the `{⟨argument⟩}` is stored. If the *sequence* does not exist, it will be created globally.

```
print-cmd = {⟨true | false⟩} default: false
```

Sets if the `⟨stored content⟩` is displayed or not at the time of running the command. The `⟨stored content⟩` is extracted from the *sequence* `{⟨seq name⟩}` set by the key `store-cmd` at the time it is executed.

Options only for starred version

The `force-eol` and `overwrite` keys is available for this *starred version*.

```
write-cmd = {⟨file.ext⟩} default: not used
```

Sets the *name* of the `⟨external file⟩` in which the `{⟨argument⟩}` will be written. The `⟨file.ext⟩` will be created in the working directory and the `{⟨argument⟩}` will be *stored* in the *sequence* set by the key `store-cmd` at the time it is executed. If `⟨file.ext⟩` does not exist, it will be created or overwritten if the `overwrite` key is used.

```
write-out = {⟨file.ext⟩} default: not used
```

Sets the *name* of the `⟨external file⟩` in which the `{⟨argument⟩}` will be written. The `⟨file.ext⟩` will be created in the working directory and the `{⟨argument⟩}` will NOT be *stored* in any *sequence* set by the key `store-cmd` at the time it is executed. If `⟨file.ext⟩` does not exist, it will be created or overwritten if the `overwrite` key is used.

- In the keys `write-cmd` and `write-out` the character TAB will be written in `⟨file.ext⟩`, relative or absolute paths are not supported. X₃TeX users using character TAB must add `-8bit` at the command line, otherwise you will get T_EX character TAB ‘^^I’ in `⟨file.ext⟩`.

5.4 The command `\getstored`

```
\getstored <getstored>[<index>]{<seq name>}
```

The `\getstored` command retrieves the *<stored content>* according to the *integer value* set in *<index>* which corresponds to the *position* of the *<stored content>* in the *sequence* *{<seq name>}*.

The command is robust and can be used as an *{<argument>}* for another command. If the *optional argument* is not passed, the default value is the “last” *<stored content>* in the *sequence* *{<seq name>}*.

5.5 The command `\foreachsc`

```
\foreachsc <foreachsc>[<key = val>]{<seq name>}
```

The command `\foreachsc` iterates through and executes the command `\getstored` on the *<stored content>* in the *sequence* *{<seq name>}*. If the *optional argument* is not passed run `\getstored` on all *<stored content>* in the *sequence* *{<seq name>}*.

Options for command

`sep = {<code>}` default: *empty*

Establishes the *separation* between each *<stored content>* in the *sequence* *{<seq name>}*. For example, you can use `sep={\[\10pt]}` for vertical separation of *<stored contents>*.

`step = {<integer>}` default: *1*

Sets the *increment* (*<step>*) applied to the value set by key `start` for each *<stored content>* in the *sequence* *{<seq name>}*. The value must be a *<positive integer>*.

`start = {<integer>}` default: *1*

Set the *position* of the *<stored content>* within the *sequence* *{<seq name>}* from which to *start* executing. The value must be a *<positive integer>*.

`stop = {<integer>}` default: *total*

Set the *position* of the *<stored content>* within the *sequence* *{<seq name>}* at which execution ends. The value must be a *<positive integer>*.

`before = {<code>}` default: *empty*

Sets the *{<code>}* that will be executed *before* each *<stored content>* within the *sequence* *{<seq name>}*. The *{<code>}* must be passed *between braces* ‘{ }’.

`after = {<code>}` default: *empty*

Sets the *{<code>}* that will be executed *after* each *<stored content>* within the *sequence* *{<seq name>}*. The *{<code>}* must be passed *between braces* ‘{ }’.

`wrapper = {<code> {#1} more code}` default: *empty*

Wraps the *<stored content>* within the *sequence* *{<seq name>}* referenced by *{#1}*. The *{<code>}* must be passed *between braces* ‘{ }’. For example `\foreachsc[wrapper={\makebox[1em][l]{#1}}]{contents}`.

5.6 The command `\typestored`

```
\typestored <typestored>[<index, start-stop, 1-end, keys>]{<seq name>}
```

The command `\typestored` places the *<stored content>* in the *sequence* *{<seq name>}* into the internally `verbatimsc` environment (§5.9). The *integer value* set at *<index>* corresponds to the *position* of the *<stored content>* in the *sequence* *{<seq name>}* will be printed, if *<1-end>* is used “all” *<stored content>* in the *sequence* *{<seq name>}* will be printed.

The *integer values* set by *<start-stop>* define the *range* of the *<stored content>* in the *sequence* *{<seq name>}* that will be printed, the rest of the accepted *<keys>* are `print-cmd` with default value `true`, `write-out`, `width-tab` and `overwrite`.

If the *optional argument* is not passed, the *first* *<stored content>* in the *sequence* *{<seq name>}* will be printed.

- In the key `write-out` the character TAB will be written in *<file.ext>*, relative or absolute paths are not supported. X_YLaTeX users using character TAB must add `-8bit` at the command line, otherwise you will get TeX character TAB ‘^^I’ in *<file.ext>*.

5.7 The command `\meaningsc`

```
\meaningsc <meaningsc>[<index, start-stop, 1-end, keys>]{<seq name>}
```

The command `\meaningsc` executes `\meaning` on the *<stored content>* in the *sequence* *{<seq name>}*. The *<index>*, *<start-stop>*, *<1-end>* and *<keys>* they have the same behavior as in the command `\typestored`. If the *optional argument* is not passed it defaults to the first *<stored content>* in the *sequence* *{<seq name>}*.

5.8 The command `\mergesc`

\mergesc `\mergesc[⟨typestored | meaningsc, keys⟩]{⟨⟨seq A⟩[⟨index⟩], {⟨seq B⟩[⟨start - stop⟩], {⟨seq C⟩[⟨1-end⟩]}}`

The command `\mergesc` assembles the *⟨stored content⟩* in the *sequences* $\{\langle seq A \rangle[1], \langle seq B \rangle[2-5] \text{ and } \langle seq C \rangle[1-end]$ and then executes `\typestored` (§5.6) if the `typestored` key is active or `\meaningsc` (§5.7) if the `meaningsc` key is active.

The $\{\langle argument \rangle\}$ taken by this command is a *comma separated list* of the form $\{\langle seq name \rangle\}$ followed by either $[⟨index⟩]$, $[⟨start-stop⟩]$ or $[⟨1-end⟩]$. The use of the keys `typestored` or `meaningsc` are “mandatory” and disjoint from each other, the rest of the accepted *⟨keys⟩* are `print-cmd`, `write-out`, `width-tab` and `overwrite`.

The use of the `write-out` key with this command follows the same rules already described, the main advantage is that allows joining *⟨stored content⟩* *without rewriting* the file over and over again, by design \TeX does not have an *append mode* for writing files, this effectively allows you to write chunks of code and then merge them into a single file.

5.9 The environment `verbatimsc`

verbatimsc The environment used by `\typestored` and `\mergesc` to display the *⟨stored content⟩* in *verbatim style*. The environment is compatible with *tagged PDF* and can be customized in the following ways after loading the `SCONTENTS` package:

Using the packages `fvextra`[7] or `fancyvrb`[5]:

```
\ExplSyntaxOn
\cs_undefine:N \verbatimsc
\cs_undefine:N \endverbatimsc
\ExplSyntaxOff
\usepackage{fancyvrb}
\DefineVerbatimEnvironment{verbatimsc}{Verbatim}{numbers=left}
```

Using the package `minted`[8]:

```
\ExplSyntaxOn
\cs_undefine:N \verbatimsc
\cs_undefine:N \endverbatimsc
\ExplSyntaxOff
\usepackage{minted}
\newminted{tex}{linenos}
\newenvironment{verbatimsc}{\VerbatimEnvironment\begin{texcode}}{\end{texcode}}
```

Using the package `listings`[6]:

```
\ExplSyntaxOn
\cs_undefine:N \verbatimsc
\cs_undefine:N \endverbatimsc
\ExplSyntaxOff
\usepackage{listings}
\lstnewenvironment{verbatimsc}
{
  \lstset{
    basicstyle=\small\ttfamily,
    columns=fullflexible,
    language=[LaTeX]TeX,
    numbers=left,
    numberstyle=\tiny\color{gray},
    keywordstyle=\color{red}
  }
}{}
\end{verbatimsc}
```

🌱 At the moment, the `fvextra`[7] and `fancyvrb`[5] packages partially support *tagged PDF*.

6 Other commands provided

6.1 The command `\countsc`

\countsc `\countsc{⟨seq name⟩}`

The command `\countsc` counts a number of *⟨stored content⟩* in the *sequence* $\{\langle seq name \rangle\}$.

6.2 The command `\cleanseqsc`

```
\cleanseqsc \cleanseqsc{⟨seq name⟩}
```

The command `\cleanseqsc` removes all *⟨stored content⟩* in the *sequence {⟨seq name⟩}*.

7 The S CONTENTS package in action

Remember the abstract on the first page?, this is it:

Abstract

This package allows to store \LaTeX code, including “*verbatim*”, in *⟨sequences⟩* using the `l3seq` module of `expl3`. The *⟨stored content⟩* can be used as many times as desired in the document, additionally you can write to *⟨external files⟩* or show it in *⟨verbatim style⟩*.

And the description of the package?

The S CONTENTS package allows to *⟨store contents⟩* in *⟨sequences⟩* or *⟨external files⟩*. In some ways it is similar to the `filecontentsdef` package, with the difference in which the *⟨content⟩* is stored. The idea behind this package is to get an approach to ConTeXt “*buffers*” by making use *⟨sequences⟩*.

I’ve only written:

```
\begin{abstract}
This package allows to store \hologo{LaTeX} code, including \enquote{\emph{verbatim}},
in \mymeta{sequences} using the \mypkg{l3seq} module of \mypkg{expl3}. The \mymeta{stored
content} can be used as many times as desired in the document, additionally you can write
to \mymeta{external files} or show it in \mymeta{verbatim style}.
\end{abstract}
```

and

The `\mypkg{*}{scontents}` package allows to `\mymeta{store contents}` in `\mymeta{sequences}` or `\mymeta{external files}`. In some ways it is similar to the `\mypkg{filecontentsdef}` package, with the difference in which the `\mymeta{content}` is stored. The idea behind this package is to get an approach to `\hologo{ConTeXt}` `\emph{\enquote{buffers}}` by making use `\mymeta{sequences}`.

Of course, I didn’t copy and paste. The real code they were written with is:

```
1 \begin{scontents}[store-env=abstract,print-env=true]
2 \begin{abstract}
3 This package allows to store \hologo{LaTeX} code, including \enquote{\emph{verbatim}},
4 in \mymeta{sequences} using the \mypkg{l3seq} module of \mypkg{expl3}. The \mymeta{stored
5 content} can be used as many times as desired in the document, additionally you can write
6 to \mymeta{external files} or show it in \mymeta{verbatim style}.
7 \end{abstract}
8 \end{scontents}
```

and

```
1 \begin{scontents}[store-env=description, print-env=true]
2 The \mypkg{*}{scontents} package allows to \mymeta{store contents} in \mymeta{sequences}
3 or \mymeta{external files}. In some ways it is similar to the \mypkg{filecontentsdef}
4 package, with the difference in which the \mymeta{content} is stored. The idea behind
5 this package is to get an approach to \hologo{ConTeXt} \emph{\enquote{buffers}} by
6 making use \mymeta{sequences}.
7 \end{scontents}
```

I stored the content in memory and then ran `\getstored` and `\typestored`. This is one of the ways you can use S CONTENTS.

8 Examples

These are some adapted examples that have served as inspiration for the creation of this package. The examples are attached to this documentation and can be extracted from your PDF viewer or from the command line by running:


```
$ pdfdetach -saveall scontents.pdf
```

and then you can use the excellent `arara`³ tool to compile them.

³The cool \TeX automation tool: <https://www.ctan.org/pkg/arara>

8.1 From answers package

Example 1

Adaptation of example 1 of the package `answers`[\[17\]](#) .


```

1 % arara: pdflatex: { branch: developer}
2 % arara: clean: { extensions: [ aux, log] }
3 \documentclass{article}
4 \usepackage[store-cmd=solutions]{scontents}
5 \newtheorem{ex}{Exercise}
6 \setlength{\parindent}{0pt}
7 \pagestyle{empty}
8 \begin{document}
9 \section{Problems}
10 \begin{ex}
11 First exercise
12 \Scontents{First solution.}
13 \end{ex}
14
15 \begin{ex}
16 Second exercise
17 \Scontents{Second solution.}
18 \end{ex}
19
20 \section{Solutions}
21 \foreachsc[sep={\\[10pt]}]{solutions}
22 \end{document}

```

8.2 From filecontentsdef package

Example 2

Adaptation of example from package `filecontentsdef`[\[2\]](#) .

```

1 % arara: pdflatex: { branch: developer}
2 % arara: clean: { extensions: [ aux, log] }
3 \documentclass{article}
4 \usepackage[store-env=defexercise,store-cmd=defexercise]{scontents}
5 \setlength{\parindent}{0pt}
6 \pagestyle{empty}
7 \begin{document}
8 % not starred
9 \Scontents{
10 Prove that  $[x^n+y^n=z^n]$  is not solvable in positive integers if  $n$  is at
11 most  $3$ .\par
12 }
13 % starred
14 \Scontents*|Refute the existence of black holes in less than  $140$  characters.|
15 % write environment to \jobname.txt
16 \begin{scontents}[write-env=\jobname.txt]
17 \def\NSA{\NSA}%
18 Prove that factorization is easily done via probabilistic algorithms and
19 advance evidence from knowledge of the names of its employees in the
20 seventies that the \NSA\ has known that for  $40$  years.\par
21 \end{scontents}
22 % see all stored
23 \begin{itemize}
24 \foreachsc[before={\item }]{defexercise}
25 \end{itemize}
26 % \getstored are robust :)
27 \section{\getstored[2]{defexercise}}
28 \end{document}

```

8.3 From TeX-SX

Example 3

Adapted from `LaTeX equivalent of ConTeXt buffers` .

```

1 % arara: pdflatex: { branch: developer}
2 % arara: clean: { extensions: [ aux, log] }

```

```

3 \documentclass{article}
4 \usepackage[store-cmd=tikz]{scontents}
5 \usepackage{tikz}
6 \setlength{\parindent}{0pt}
7 \pagestyle{empty}
8 \Scontents{\matrix{ \node (a) {$a$} ; & \node (b) {$b$} ; \\ } ;}
9 \Scontents{\matrix[ampersand replacement=\&
10 { \node (a) {$a$} ; \& \node (b) {$b$} ; \\ } ;}
11 \Scontents{\matrix{\node (a) {$a$} ; & \node (b) {$b$} ; \\ } ;}
12 \begin{document}
13 \section{tikzpicture}
14 \begin{tikzpicture}
15 \getstored[1]{tikz}
16 \end{tikzpicture}
17
18 \begin{tikzpicture}
19 \getstored[2]{tikz}
20 \end{tikzpicture}
21
22 \begin{tikzpicture}
23 \getstored{tikz}
24 \end{tikzpicture}
25
26 \begin{scontents}[store-env=buffer]
27 Hello World!
28
29 This is a \verb*|fake poor man's buffer :)|.
30 \end{scontents}
31
32 \section{source tikz}
33 \typestored[1]{tikz}
34 \typestored[2]{tikz}
35 \typestored[3]{tikz}
36
37 \section{fake buffer}
38 \subsection{real content}
39 \getstored[1]{buffer}
40 \subsection{verbatim style}
41 \typestored[1]{buffer}
42 \subsection{meaning}
43 \meaningsc[1]{buffer}
44
45 \section{tikz again}
46 \foreachsc[before={\begin{tikzpicture}},after={\end{tikzpicture}},sep={\\[10pt]}]{tikz}
47 \end{document}

```

Example 4

Adapted from [Collecting contents of environment and store them for later retrieval](#) .

```

1 % arara: pdflatex: { branch: developer}
2 % arara: clean: { extensions: [ aux, log ] }
3 \documentclass{article}
4 \usepackage{scontents}
5 \setlength{\parindent}{0pt}
6 \pagestyle{empty}
7 \begin{document}
8 \begin{scontents}[store-env=main]
9 Something for main A.
10 \end{scontents}
11
12 \begin{scontents}[store-env=main]
13 Something for \verb|main B|.
14 \end{scontents}
15
16 \begin{scontents}[store-env=other]
17 Something for \verb|other|.
18 \end{scontents}
19
20 \textbf{Let's print them}
21
22 This is first stored in main: \getstored[1]{main}\par

```

```

23 This is second stored in main: \getstored{main}\par
24 This is stored in other: \getstored{other}
25
26 \textbf{Print all of stored in main}\par
27 \foreachsc[sep={\\[10pt]}]{main}
28 \end{document}

```

Example 5

Adapted from [Collect contents of an environment \(that contains verbatim content\)](#) .

```

1 % arara: pdflatex: { branch: developer}
2 % arara: clean: { extensions: [ aux, log] }
3 \documentclass{article}
4 \usepackage{scontents}
5 \setlength{\parindent}{0pt}
6 \pagestyle{empty}
7 \begin{document}
8 \section{Problem stated the first time}
9 \begin{scontents}[print-env=true,store-env=problem]
10 This is normal text.
11 \verb|This is from the verb command.|
12 \verb*|This is from the verb* command.|
13 This is normal text.
14 \begin{verbatim}
15 This is from the verbatim environment:
16 &{%{}}~
17 \end{verbatim}
18 \end{scontents}
19 \section{Problem restated}
20 \getstored[1]{problem}
21 \section{Problem restated once more}
22 \getstored[1]{problem}
23 \end{document}

```

Example 6

Adapted from [Environment hiding its content](#) .

```

1 % arara: pdflatex: { branch: developer}
2 % arara: clean: { extensions: [ aux, log] }
3 \documentclass[10pt]{article}
4 \usepackage{scontents}
5 \newenvsc{forshort}[store-env=forshort,print-env=false]
6 \setlength{\parindent}{0pt}
7 \pagestyle{empty}
8 \begin{document}
9
10 Something in the whole course.
11
12 \begin{forshort}
13     Just a summary...
14 \end{forshort}
15
16 \end{document}

```

8.4 Customization of verbatimsc

Example 7

Customization of `verbatimsc` using the `fvextra`[\[7\]](#) and `tcolorbox`[\[14\]](#) package .

```

1 % arara: pdflatex: { branch: developer}
2 % arara: clean: { extensions: [ aux, log] }
3 \documentclass{article}
4 \usepackage{scontents}
5 \ExplSyntaxOn
6 \cs_undefine:N \verbatimsc
7 \cs_undefine:N \endverbatimsc
8 \ExplSyntaxOff
9 \usepackage{fvextra}
10 \usepackage{xcolor}

```

```

11 \definecolor{mygray}{gray}{0.9}
12 \usepackage{tcolorbox}
13 \newenvironment{verbatimsc}%
14 {\VerbatimEnvironment
15 \begin{tcolorbox}[colback=mygray, boxsep=0pt, arc=0pt, boxrule=0pt]
16 \begin{Verbatim}[fontsize=\scriptsize, breaklines, breakafter=*, breaksymbolsep=0.5em,
17 breakaftersymbolpre={\,\tiny\ensuremath{\rfloor}}]}%
18 {\end{Verbatim}}%
19 \end{tcolorbox}}
20 \setlength{\parindent}{0pt}
21 \pagestyle{empty}
22 \begin{document}
23 \section[Test \texttt{\textbackslash begin\{scontents\}} with \texttt{fancyvrb}]
24 Test \verb+{scontents}+ \par
25
26 \begin{scontents}
27 Using \verb+scontents+ env no \verb+[key=val]+, save in seq \verb+contents+
28 with index 1.
29
30 Prove new \Verb*{ fancyvrb with braces } and environment \verb+Verbatim*+
31 \begin{verbatim}
32   verbatim environment
33 \end{verbatim}
34 \end{scontents}
35
36 \section[Test \texttt{\textbackslash Scontents} with \texttt{fancyvrb}]
37 \Scontents{ We have coded this in \LaTeX:  $E=mc^2$ .}
38
39 \section[Test \texttt{\textbackslash getstored}]
40 \getstored[1]{contents}\par
41 \getstored{contents}
42
43 \section[Test \texttt{\textbackslash meaningsc}]
44 \meaningsc[1]{contents}\par
45 \meaningsc[2]{contents}
46
47 \section[Test \texttt{\textbackslash typestored}]
48 \typestored[1]{contents}
49 \typestored[2]{contents}
50 \end{document}

```

Example 8

Customization of `verbatimsc` using the `listings`[6] package .

```

1 % arara: pdflatex: { branch: developer}
2 % arara: clean: { extensions: [ aux, log] }
3 \documentclass{article}
4 \usepackage{scontents}
5 \ExplSyntaxOn
6 \cs_undefine:N \verbatimsc
7 \cs_undefine:N \endverbatimsc
8 \ExplSyntaxOff
9 \usepackage{xcolor}
10 \usepackage{listings}
11 \lstnewenvironment{verbatimsc}
12 {
13   \lstset{
14     basicstyle=\small\ttfamily,
15     breaklines=true,
16     columns=fullflexible,
17     language=[LaTeX]TeX,
18     numbers=left,
19     numbersep=1em,
20     numberstyle=\tiny\color{gray},
21     keywordstyle=\color{red}
22   }
23 }{}
24 \setlength{\parindent}{0pt}
25 \pagestyle{empty}
26 \begin{document}
27 \section[Test \texttt{\textbackslash begin\{scontents\}} with \texttt{listings}]

```

```

28 Test \verb+{scontents}+ \par
29
30 \begin{scontents}
31 Using \verb+scontents+ env no \verb+[key=val]+, save in seq \verb+contents+ with index 1.\par
32
33 Prove \lstinline[basicstyle=\ttfamily]| \lstinline | and environment \verb+Verbatim*+
34 \begin{verbatim}
35     verbatim environment
36 \end{verbatim}
37 \end{scontents}
38
39 \section[Test \texttt{\textbackslash Scontents*} with \texttt{listings}]
40
41 \Scontents*+We have coded this in \lstinline[basicstyle=\ttfamily]| \LaTeX:  $E=mc^2$ 
42 and more.+
43
44 \section[Test \texttt{\textbackslash getstored}]
45 \getstored{contents}\par
46 \getstored[1]{contents}
47
48 \section[Test \texttt{\textbackslash typestored}]
49 \typestored[1]{contents}
50 \typestored[2]{contents}
51 \end{document}

```

Example 9

Customization of `verbatimsc` using the `minted`[8] package .

```

1 % arara: xelatex: { branch: developer, shell: true, options: [-8bit]}
2 % arara: xelatex: { branch: developer, shell: true, options: [-8bit]}
3 % arara: clean: { extensions: [ aux, log] }
4 \documentclass{article}
5 \usepackage{scontents}
6 \ExplSyntaxOn
7 \cs_undefine:N \verbatimsc
8 \cs_undefine:N \endverbatimsc
9 \ExplSyntaxOff
10 \usepackage{minted}
11 \newminted{tex}{linenos}
12 \newenvironment{verbatimsc}{\VerbatimEnvironment\begin{texcode}}{\end{texcode}}
13 \pagestyle{empty}
14 \setlength{\parindent}{0pt}
15 \begin{document}
16 \section[Test \texttt{\textbackslash begin\{scontents\}} with \texttt{minted}]
17 Test \verb+{scontents}+ \par
18
19 \begin{scontents}[overwrite,write-env=\jobname.tsc,force-eol=true]
20 Using \verb+scontents+ env no \verb+[key=val]+, save in seq \verb+contents+
21 with index 1.\par
22
23 Prove new \Verb*{ new fvxtra with braces } and environment \verb+Verbatim*+
24 \begin{Verbatim}[obeytabs, showtabs, tab=\rightarrowfill, tabcolor=red]
25 No tab
26         One real tab
27             Two real Tab plus         one tab
28 \end{Verbatim}
29 \end{scontents}
30
31 \section[See \Verb{\jobname.tsc}]
32 Read \Verb{\jobname.tsc} (shows TABs as red arrows):
33 \VerbatimInput[obeytabs, showtabs, tab=\rightarrowfill, tabcolor=red]{\jobname.tsc}
34
35 \section[Test \texttt{\textbackslash Scontents} with \texttt{minted}]
36
37 \Scontents{ We have coded \par this in \LaTeX:  $E=mc^2$ .}
38
39 \section[Test \texttt{\textbackslash getstored}]
40 \getstored[1]{contents}\par
41 \getstored{contents}
42
43 \section[Test \texttt{\textbackslash typestored}]


```

```

44 \typestored[1]{contents}
45 \typestored[2]{contents}
46 \end{document}

```

8.5 The command \mergesc in action

The command `\mergesc` in action, adapted from Denis Bitouzé request at <https://github.com/pablgonz/scontents/issues/2> .

```

1 % arara: pdflatex: { branch: developer}
2 % arara: clean: { extensions: [ aux, log] }
3 \documentclass{article}
4 \usepackage{scontents}
5 % Fix part of a MCE that should go before babel's loading
6 \begin{scontents}[store-env=mce]
7 \documentclass[french]{article}
8 \usepackage[T1]{fontenc}
9 \usepackage[utf8]{inputenc}
10 \usepackage{lmodern}
11 \usepackage[a4paper]{geometry}
12 \end{scontents}
13 % Fix part of a MCE that should go after (>=) babel's loading
14 \begin{scontents}[store-env=mce]
15 \usepackage{babel}
16 \begin{document}
17 \end{scontents}
18 % Fix part of a MCE that should go after its body
19 \begin{scontents}[store-env=mce]
20 \end{document}
21 \end{scontents}
22 \begin{document}
23 \section{First annswer}
24 % Variable part of a MCE that should added to the fixed preamble, before babel's loading
25 \begin{scontents}[store-env=mce-1]
26 \usepackage{amsmath}
27 \end{scontents}
28 % Variable part of a MCE being the code snippet
29 \begin{scontents}[store-env=mce-1]
30 \begin{align}
31   0 & \& \neq 1 \\\
32   1 & \& \neq 0
33 \end{align}
34 \end{scontents}
35 \begin{description}
36 \item[Preamble's addition]\leavevmode
37   \typestored[1]{mce-1}
38 \item[Code snippet]\leavevmode
39   \typestored[2]{mce-1}
40 \item[MCE]\leavevmode
41   \mergesc[typestored, print-cmd=true]
42     {
43       {mce}[1], {mce-1}[1], {mce}[2], {mce-1}[2], {mce}[3]
44     }
45 \end{description}
46 \section{Second annswer}
47 % Variable part of a MCE that should added to the fixed preamble, before babel's loading
48 \begin{scontents}[store-env=mce-2]
49 \usepackage{amsmath}
50 \end{scontents}
51 % Variable part of a MCE being the code snippet
52 \begin{scontents}[store-env=mce-2]
53 \begin{flalign}
54   0 & \& \neq 1 \\\
55   1 & \& \neq 0
56 \end{flalign}
57 \end{scontents}
58
59 \begin{description}
60 \item[Preamble's addition]\leavevmode
61   \typestored[1]{mce-2}
62 \item[Code snippet]\leavevmode
63   \typestored[2]{mce-2}

```



```

64 \item[MCE]\leavevmode
65   \mergesc[typetored, print-cmd=true, write-out=mce.txt, overwrite=true]
66   {
67     {mce}[1], {mce-2}[1], {mce}[2], {mce-2}[2], {mce}[3]
68   }
69 \end{description}
70 \end{document}

```


8.6 The tagged PDF example

This example is just to show the compatibility of **SCONTENTS** with *tagged* PDF using `lualatex`. The attached files here are just for testing .

```

1 % arara: lualatex: { branch: developer}
2 % arara: lualatex: { branch: developer}
3 % arara: clean: { extensions: [ aux, log] }
4 \DocumentMetadata{tagging=on, lang=en-US, pdfversion=2.0, pdfstandard=ua-2}
5 \documentclass{article}
6 \usepackage{scontents,unicode-math,hyperref}
7 \hypersetup{pdftitle = {Test scontents package},}
8 \begin{document}
9   Some
10
11 \begin{scontents}[print-env=true]
12   First code \verb|\foo|
13
14   And more code \verb|\bar|
15 \end{scontents}
16
17 Text
18
19 \begin{scontents}[print-env=true]
20   Second code \verb|\foo|
21
22   And more code \verb|\bar|
23 \end{scontents}
24
25 Text
26
27 \Scontents*{code \verb|\baz|}
28
29 % \typetored
30 \typetored[1]{contents}
31
32 % \mergesc
33 \mergesc[typetored]{ {contents}[1-end] }
34
35 % \getstored
36 \getstored[2]{contents}
37 \end{document}

```

 This example have been checked using `veraPDF` together with `ngpdf`.

9 Change history

In this section you will find some (not all) of the changes in `SCONTENTS` development, from the first public implementation using the `filecontentsdef`[2] package to the current version with only `expl3`[1].

- | | |
|--------------------------------|---|
| v2.7 (ctan), 2026-06-01 | <ul style="list-style-type: none"> – Internal environment <code>verbatimsc</code> update for <i>tagged</i> PDF. – Update requirement to \LaTeX release 2026-06-01. – Add <code>debug-var</code> key for writing variables to the <code>.log</code> file. – Update and improvements documentation and examples. |
| v2.6 (ctan), 2025-11-20 | <ul style="list-style-type: none"> – Internal environment <code>verbatimsc</code> update for <i>tagged</i> PDF. – Update requirement to \LaTeX release 2025-11-01. |
| v2.5 (ctan), 2025-11-01 | <ul style="list-style-type: none"> – Replacing <code>\scantokens</code> (primitive) with <code>\tl_retokenize:n</code>. – Cleanup warnings and details returned by <code>expltools</code>. – Update minimum required of <code>expl3</code> to 2025-07-08. |
| v2.4 (ctan), 2025-05-15 | <ul style="list-style-type: none"> – Optimization of expansion code from <code>'x'</code> to <code>'e'</code>. – Restructuring code for documentation and implementation. – Add new keys for <code>\typestored</code> and <code>\meaningsc</code>. – Check the version of <code>expl3</code> in plain \TeX and \ConTeXt. |
| v2.3 (ctan), 2025-04-23 | <ul style="list-style-type: none"> – Adapting the <code>verbatimsc</code> environment for <i>tagged</i> PDF. – Update minimum required to \LaTeX release 2024-11-01. – Safer code for replacement <code>\obeyedline</code>. |
| v2.2 (ctan), 2025-03-26 | <ul style="list-style-type: none"> – Fix internal definition for some functions. – Replace <code>\peek_charcode_ignore_spaces:NTF</code> by <code>\peek_charcode:NTF</code>. – Set correct code for <code>\obeyedline</code> implement in \LaTeX release 2024-06-01. |
| v2.1 (ctan), 2024-06-14 | <ul style="list-style-type: none"> – Fix <code>\cleanseqsc</code> command. – Add <code>\mergesc</code> command. – Fix internal definition for <code>seq</code> var. – Fix internal code for <code>\typestored</code>. – Replace <code>\cs_argument_spec:N</code> by <code>\cs_parameter_spec:N</code>. – Detect <code>l3keys2e</code> package (obsolete in june 2022 \LaTeX release). – Minor adjustments in the documentation. |
| v2.0 (ctan), 2022-04-04 | <ul style="list-style-type: none"> – Adapting the <code>verbatimsc</code> environment (compatibility <code>verbatim</code> package). – Removed compatibility layer for older \LaTeX releases. – Fix loader in plain \TeX and \ConTeXt. – Minor adjustments in the documentation. |
| v1.9 (ctan), 2020-01-21 | <ul style="list-style-type: none"> – Update and improvements in the internal code. – Updating the generic code for I/O verification. – Add <code>write-cmd</code> and <code>write-out</code> keys for <code>\Scontents*</code>. – Fix <code>sep</code> key in <code>\foreachsc</code>. |
| v1.8 (ctan), 2019-11-18 | <ul style="list-style-type: none"> – Add <code>\newenvsc</code> command. – Fix nested environment in plain \TeX and \ConTeXt. – Modified default value in <code>\getstored</code>. – Add <code>overwrite</code> key to reduce I/O operations. – Deleted an unnecessary group in the code. |
| v1.7 (ctan), 2019-10-29 | <ul style="list-style-type: none"> – The <code>verbatimsc</code> environment was rewritten. – Minor adjustments in documentation. |
| v1.6 (ctan), 2019-10-26 | <ul style="list-style-type: none"> – The internal behavior of <code>\getstored</code> has been modified. – The internal behavior of <code>\foreachsc</code> has been modified. – Corrected file extension for \ConTeXt. – Remove spurious warning. |
| v1.5 (ctan), 2019-10-24 | <ul style="list-style-type: none"> – Add support for plain \TeX and \ConTeXt. – Split internal code for optimization. – Add support for vertical spaces in <code>[⟨key = val⟩]</code>. – Add <code>\foreachsc</code> command. – Check if <code>verbatim</code> package is loaded. |
| v1.4 (ctan), 2019-10-03 | <ul style="list-style-type: none"> – Add <code>store-all</code> key. – Messages and keys were separated. – Restructuring of documentation. – Now the version of <code>expl3</code> is checked instead of <code>xparse</code>. – The internal behavior of <code>force-eol</code> has been modified. |
| v1.3 (ctan), 2019-09-24 | <ul style="list-style-type: none"> – The environment <code>scontents</code> can now nest. – Added <code>force-eol</code>, <code>verb-font</code> and <code>width-tab</code> keys. – The extra space has been removed when you run <code>\getstored</code>. – Internal code has been rewritten more efficiently. – Remove <i>starred argument</i> <code>'*</code> for <code>\typestored</code>. – Remove <code>filecontentsdef</code> dependency. |

- v1.2 (ctan), 2019-08-28**
 - Changing `\regex_replace_all:` for `\tl_replace_all:`.
 - Restructuring of documentation.
 - Added copy of `\tex_scantokens:`.
- v1.1 (ctan), 2019-08-12**
 - Extension of documentation.
 - Replace `\tex_endinput:D` by `\file_input_stop:`.
- v1.0 (ctan), 2019-07-30**
 - First public release.

10 Index of Documentation

The italic numbers denote the pages where the corresponding entry is described.

C	
Commands provide by SCONTENTS :	
\Sccontents*	3, 5
\Sccontents	3, 5
\cleanseqsc	8
\countsc	7
\endscontents	4
\foreachsc	6
\getstored	3, 4, 6
\meaningsc	3, 6
\mergesc	3, 7
\newenvsc	3, 5
\scontents	4
\setupsc	3-5
\startscontents	4
\stopscontents	4
\typestored	3, 6, 7
E	
Environments provide by SCONTENTS :	
scontents	3-5
verbatimsc	6, 11-13
Environments:	
Verbatim	2
filecontentsdefmacro	1
lstlisting	2
K	
Keys provide by SCONTENTS :	
after	6
before	6
debug-var	3
force-eol	3-5
meaningsc	3, 7
P	
Packages:	
answers	9
expl3	1, 2, 8, 16
fancyvrb	2, 3, 7
filecontentsdef	1, 8, 9, 16
fvextra	2, 3, 5, 7, 11
l3keys2e	16
l3seq	1, 8
listings	2, 5, 7, 12
minted	7, 13
scontents	1, 2, 7, 8, 15, 16
tcolorbox	11
verbatim	16
xparse	16
overwrite	3-7
print-all	3
print-cmd	3, 5-7
print-env	3-5
sep	6
start	6
step	6
stop	6
store-all	3
store-cmd	3, 5
store-env	3-5
typestored	3, 7
verb-font	3
width-tab	3, 6, 7
wrapper	6
write-cmd	3, 5
write-env	3-5
write-out	3-7

11 References

[1] The L^AT_EX Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/expl3>, 2026.

[2] BURNOL, JEAN FRANÇOIS. “The filecontentsdef package”. Available from CTAN, <https://ctan.org/pkg/filecontentsdef>, 2019.

[3] The L^AT_EX Project. “The xparse package”. Available from CTAN, <https://www.ctan.org/pkg/xparse>, 2024.

[4] NIEDERBERGER, CLEMENS. “xsim – eXercise Sheets IMproved”. Available from CTAN, <https://www.ctan.org/pkg/xsim>, 2023.

[5] VAN ZANDT, TIMOTHY. “The fancyvrb package - Fancy Verbatims in L^AT_EX”. Available from CTAN, <https://www.ctan.org/pkg/fancyvrb>, 2026.

[6] HOFFMANN, JOBST. “The listings package”. Available from CTAN, <https://www.ctan.org/pkg/listings>, 2025.

[7] POORE, GEOFFREY M. “The fvextra package - Highlighted source code in L^AT_EX”. Available from CTAN, <https://www.ctan.org/pkg/minted>, 2026.

[8] POORE, GEOFFREY M. “The minted package - Highlighted source code in L^AT_EX”. Available from CTAN, <https://www.ctan.org/pkg/minted>, 2026.

[9] The L^AT_EX Project. “The L^AT_EX₃ Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2026.

- [10] The \LaTeX Project. “The \LaTeX 2 _{ϵ} sources”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2026.
- [11] The \LaTeX Project. “ \LaTeX for authors current version”. Available from CTAN, <https://ctan.org/pkg/latex-base>, 2026.
- [12] FISCHER, ULRIKE. “tagpdf – \LaTeX kernel code for PDF tagging”. Available from CTAN, <https://www.ctan.org/pkg/tagpdf>, 2026.
- [13] The \LaTeX Project. “latex-lab – \LaTeX laboratory”. Available from CTAN, <https://www.ctan.org/pkg/latex-lab>, 2026.
- [14] F. STURM, THOMAS. “tcolorbox – Coloured boxes, for \LaTeX examples and theorems, etc”. Available from CTAN, <https://ctan.org/pkg/tcolorbox>, 2026.
- [15] The \LaTeX Project. “verbatim – Reimplementation of and extensions to \LaTeX verbatim”. Available from CTAN, <https://www.ctan.org/pkg/verbatim>, 2023.
- [16] WRIGHT, JOSEPH. “Programming key-value in expl3”. Available from TUGBOAT, <https://www.tug.org/TUGboat/tb31-1/tb97wright-l3keys.pdf>, 2010.
- [17] WRIGHT, JOSEPH. “answers – Setting questions (or exercises) and answers”. Available from CTAN, <https://ctan.org/pkg/answers>, 2014.

12 Implementation

The most recent publicly released version of `SCONTENTS` is available at CTAN: <https://www.ctan.org/pkg/scontents>. Historical and developmental versions are available at <https://github.com/pablgonz/scontents>. While general feedback via email is welcomed, specific bugs or feature requests should be reported through the issue tracker: <https://github.com/pablgonz/scontents/issues>.

- All variables and functions defined in this package are private and are NOT intended to work or be used by another package or module.

12.1 Declaration of the package

First we set up the module name for DocStrip l3doc class:

```
1 <@@=scontents>
```

Now we define some common macros to hold the package date and version:

```
2 <*loader>
3 \def\ScontentsFileDate{2026-06-01}%
4 \def\ScontentsFileVersion{2.7}%
5 \def\ScontentsFileDescription{Stores LaTeX contents in memory or files}%
```

The \LaTeX loader is quite simple, we just need to make sure of the minimum version for correct operation and then set interfaces up. The choice of \LaTeX release 2026-06-01 is the latest available in \TeX Live 2026 and is necessary to be able to implement the package's full compatibility with *tagged* PDF.

```
6 <*latex>
7 \NeedsTeXFormat{LaTeX2e}[2026-06-01]
8 \ProvidesExplPackage
9   {scontents} {\ScontentsFileDate} {\ScontentsFileVersion} {\ScontentsFileDescription}
10 </latex>
```

12.1.1 Load expl3-generic and xparse-generic in plain \TeX and \ConTeXt

```
\c__scontents_version_str
\l__scontents_char_value_int
```

The plain \TeX and \ConTeXt loaders are similar (probably because I don't know how to make a proper \ConTeXt module :-). We define a string variable `\c__scontents_version_str` with version info (just in case) and add `\ExplSyntaxOn` to be able to load the frozen `xparse[3]`.

```
11 <!!latex>
12 <context>\writestatus{loading}{User Module scontents v\ScontentsFileVersion}
13 <context>\unprotect
14 \input expl3-generic.tex
15 \ExplSyntaxOn
16 \str_const:ce { c__scontents_version_str } { \ScontentsFileDate\space
17   v\ScontentsFileVersion\space \ScontentsFileDescription }
18 \iow_log:e { Package: ~ scontents ~ \str_use:c { c__scontents_version_str } }
```

Outside of \LaTeX we can't usually use `xparse[3]` now `ltxcmd[10]` part of the \LaTeX kernel. However since the old `xparse[3]` provide `xparse-generic.tex` is loadable in any format.

```
19 \int_new:N \l__scontents_char_value_int
20 \int_set:Nn \l__scontents_char_value_int { \char_value_catcode:n { \@ } }
21 \char_set_catcode_letter:N \@
22 \file_input:n { xparse-generic.tex }
23 \char_set_catcode:nn { \@ } { \l__scontents_char_value_int }
24 </!latex>
```

(End of definition for `\c__scontents_version_str` and `\l__scontents_char_value_int`.)

12.1.2 Checking expl3 version

For plain \TeX , \LaTeX and \ConTeXt we must check the minimum requirement, in this case `\tl_retokenize:n` which was added in release 2025-07-08 of `expl3` included in \TeX Live 2025.

```
25 \cs_if_exist:NF \tl_retokenize:n
26 {
27   \msg_new:nnn { scontents } { expl-too-old }
28   {
29     Please~install~an~up~to~date~TeX~distribution~or~update~using~
30     your~TeX~package~manager~or~from~CTAN. \\\
31     See~documentation.~Loading~scontents~will~abort!
32   }
33   \msg_fatal:nn { scontents } { expl-too-old }
34   \ExplSyntaxOff
35   \file_input_stop:
```



```
36 }
```

12.1.3 Preventing double loading in plain T_EX

In plain T_EX, check that the package isn't being loaded twice (E_T_X and ConT_EXt already defend against that):

```
37 ⟨*plain⟩
38 \tl_if_exist:NT \l__scontents_cmd_name_tl
39 {
40   \msg_new:nnn { scontents } { already-loaded }
41   {
42     The~'scontents'~package~is~already~loaded.~Aborting~input~\msg_line_context:.
43   }
44   \msg_warning:nn { scontents } { already-loaded }
45   \ExplSyntaxOff
46   \file_input_stop:
47 }
48 ⟨/plain⟩
49 ⟨/loader⟩
```

12.1.4 Checking proper loader

Checking that the package was loaded with the proper loader code. This code was copied from `expl3-code.tex`.

```
50 ⟨*core⟩
51 \def\ScontentsCoreFileDate{2026-06-01}%
52 \begingroup
53   \catcode32=10
54   \endlinechar=32
55   \def\next{\endgroup}%
56   \expandafter\ifx\csname PackageError\endcsname\relax
57     \begingroup
58       \def\next{\endgroup\endgroup}%
59       \def\PackageError#1#2#3%
60       {%
61         \endgroup
62         \errhelp{#3}%
63         \errmessage{#1 Error: #2!}%
64       }%
65     \fi
66     \expandafter\ifx\csname ScontentsFileDate\endcsname\relax
67       \def\next
68       {%
69         \PackageError{scontents}{No scontents loader detected}
70         {%
71           You have attempted to use the scontents code directly rather than using
72           the correct loader. Loading of scontents will abort.
73         }%
74       \endgroup
75       \endinput
76     }%
77   \else
78     \ifx\ScontentsFileDate\ScontentsCoreFileDate
79     \else
80       \def\next
81       {%
82         \PackageError{scontents}{Mismatched scontents files detected}
83         {%
84           You have attempted to load scontents with mismatched files:
85           probably you have one or more files 'locally installed' which
86           are in conflict. Loading of scontents will abort.
87         }%
88       \endgroup
89       \endinput
90     }%
91   \fi
92 \fi
93 \next
94 ⟨/core⟩
```

12.2 Variables and functions by format

We define and set variables and one function that must be handled separately in order to work properly with plain \TeX , \LaTeX and Con \TeX t.

`__scontents_format_case:nnn` Sometimes we need to detect the format from within a macro:

```

95 <*loader>
96 \cs_new:Npn \__scontents_format_case:nnn #1 #2 #3
97 <latex> {#1} % LaTeX
98 <plain> {#2} % Plain/Generic
99 <context> {#3} % ConTeXt

```

(End of definition for `__scontents_format_case:nnn`.)

`\c__scontents_newline_tl` A constant token list `\c__scontents_newline_tl` to store ‘`^^J`’ according to the formats

```

100 <!context>
101 \tl_const:Nc \c__scontents_newline_tl { \iow_char:N ^^J }
102 </!context>

```

(End of definition for `\c__scontents_newline_tl`.)

In Con \TeX t we must take a precaution when running under LMTX. This is an adaptation of the file `t-lua-widow-control.mkx` part of Max Chernoff’s `lua-widow-control` package, `\contextlmtxmode` is described at <https://source.contextgarden.net/tex/context/base/mkx/context.mkx>.

```

103 <context>
104 \bool_if:NTF \contextlmtxmode
105 {
106   \msg_new:nnn { scontents } { luametatex }
107   {
108     The~'scontents'~package~doesn't~work~under~LMTX.
109   }
110   \msg_error:nn { scontents } { luametatex }
111   % \tl_const:Nc \c__scontents_newline_tl { \sys_obeyed_line }
112 }
113 {
114   \tl_const:Nc \c__scontents_newline_tl { \iow_char:N ^^J }
115 }
116 </context>

```

For some reason unknown to me, it doesn’t work under LMTX, Max gave me this in the chat....ask what it does `\sys_obeyed_line` and where it’s documented. <https://chat.stackexchange.com/transcript/message/6>

`\g__scontents_end_verbatimsc_tl` The global token list `\g__scontents_end_verbatimsc_tl` match when ending `verbatimsc` (§12.13).

```

\c__scontents_end_env_tl
\l__scontents_env_name_tl
117 \tl_new:N \g__scontents_end_verbatimsc_tl
118 \tl_gset_rescan:Nnn \g__scontents_end_verbatimsc_tl
119 {
120   \char_set_catcode_other:N \
121 <*latex>
122   \char_set_catcode_other:N \{
123   \char_set_catcode_other:N \}
124 </latex>
125 }
126 <latex> { \end{verbatimsc} }
127 <plain> { \endverbatimsc }
128 <context> { \stopverbatimsc }

```

The constant token list `\c__scontents_end_env_tl` match when ending environments defined by `\newenvsc`, the token list `\l__scontents_env_name_tl` storing the *name* of environments defined by `\newenvsc` (§12.11).

```

129 \tl_new:N \l__scontents_env_name_tl
130 \tl_const:Nc \c__scontents_end_env_tl
131 {
132   \c_backslash_str
133 <latex|plain> end
134 <context> stop
135 <latex> \c_left_brace_str
136 \exp_not:N \l__scontents_env_name_tl
137 <latex> \c_right_brace_str
138 }

```

(End of definition for `\g__scontents_end_verbatimsc_tl`, `\c__scontents_end_env_tl`, and `\l__scontents_env_name_tl`.)

12.3 Loading the package core

Now we load the core `SCONTENTS` code:

```

139 \file_input:n { scontents-code.tex }
140 \loader

```

12.4 Keys for the package

store-env
store-cmd
verb-font
print-env
print-cmd
force-eol
overwrite
width-tab
print-all
store-all
debug-var

We create some common `\keys` that will be used by the options passed to the package as well as by the environments and commands defined.

```

141 \*core
142 \keys_define:nn { scontents }
143 {
144   store-env .tl_set:N      = \l__scontents_name_seq_env_tl,
145   store-env .initial:n     = contents,
146   store-env .value_required:n = true,
147   store-cmd .tl_set:N      = \l__scontents_name_seq_cmd_tl,
148   store-cmd .initial:n     = contents,
149   store-cmd .value_required:n = true,
150   verb-font .tl_set:N      = \l__scontents_verb_font_tl,
151   verb-font .value_required:n = true,
152   print-env .bool_set:N    = \l__scontents_print_env_bool,
153   print-env .initial:n     = false,
154   print-env .default:n     = true,
155   print-cmd .bool_set:N    = \l__scontents_print_cmd_bool,
156   print-cmd .initial:n     = false,
157   print-cmd .default:n     = true,
158   force-eol .bool_set:N    = \l__scontents_forced_eol_bool,
159   force-eol .initial:n     = false,
160   force-eol .default:n     = true,
161   overwrite .bool_set:N    = \l__scontents_overwrite_bool,
162   overwrite .initial:n     = false,
163   overwrite .default:n     = true,
164   width-tab .int_set:N     = \l__scontents_tab_width_int,
165   width-tab .initial:n     = 1,
166   width-tab .value_required:n = true,
167   print-all .meta:n       = { print-env = #1, print-cmd = #1 },
168   print-all .default:n    = true,
169   store-all .meta:n       = { store-env = #1, store-cmd = #1 },
170   store-all .value_required:n = true,
171   debug-var .bool_set:N    = \l__scontents_debug_var_bool,
172   debug-var .initial:n     = true,
173   debug-var .default:n     = true,
174 }
175 \core

```

Set default value for `verb-font` key.

```

176 \loader\keys_define:nn { scontents }
177 \latex { verb-font .initial:n = \ttfamily }
178 \plain|context { verb-font .initial:n = \tt }

```

In `TEX` mode process the `\keys` as options passed on to the package and will return an error when they are.

```

179 \*latex
180 \ProcessKeyOptions [ scontents ]
181 \latex

```

(End of definition for `store-env` and others.)

12.5 Internal variables

\l__scontents_save_every_body_lines_tl
\l__scontents_processed_body_lines_tl
\l__scontents_environment_keys_tl
\l__scontents_nesting_env_int
\l__scontents_nesting_aux_int

The token list `\l__scontents_save_every_body_lines_tl` holds the `{\body env}` of an environment, `scontents` by default, as it's being read, the token list `\l__scontents_processed_body_lines_tl` saves all sanitized lines saved in `\l__scontents_save_every_body_lines_tl`.

The token list `\l__scontents_environment_keys_tl` saves the `\keys` passed to the *optional argument* after they are sanitized and the integer variables `\l__scontents_nesting_env_int` together with `\l__scontents_nesting_aux_int` are used to analyze the nesting of the environment.

- All of these variables are used in the implementation of `\newenvsc` (§12.11) and the environments base functions (§12.10).

```

182 < *core >
183 \tl_new:N \l__scontents_save_every_body_lines_tl
184 \tl_new:N \l__scontents_processed_body_lines_tl
185 \tl_new:N \l__scontents_environment_keys_tl
186 \int_new:N \l__scontents_nesting_env_int
187 \int_new:N \l__scontents_nesting_aux_int

```

(End of definition for `\l__scontents_save_every_body_lines_tl` and others.)

`\l__scontents_cmd_name_tl` The token list `\l__scontents_cmd_name_tl` saves the *name* of the commands `\Scontents`, `\foreachsc`, `\typestored`, `\meaningsc` and `\mergesc`.

```

188 \tl_new:N \l__scontents_cmd_name_tl

```

(End of definition for `\l__scontents_cmd_name_tl`.)

`\l__scontents_Scontents_arg_tl` The token lists `\l__scontents_Scontents_arg_tl`, `\l__scontents_foreachsc_arg_tl`, `\l__scontents_typedstored_arg_tl` and `\l__scontents_meaningsc_arg_tl` save the $\{\langle argument \rangle\}$ passed to the `\Scontents` (§12.14), `\foreachsc` (§12.16), `\typestored` (§12.17) and `\meaningsc` (§12.18) commands.

```

189 \tl_new:N \l__scontents_Scontents_arg_tl
190 \tl_new:N \l__scontents_foreachsc_arg_tl
191 \tl_new:N \l__scontents_typedstored_arg_tl
192 \tl_new:N \l__scontents_meaningsc_arg_tl

```

(End of definition for `\l__scontents_Scontents_arg_tl` and others.)

`\l__scontents_mergesc_arg_tl` The token list `\l__scontents_mergesc_arg_tl` save the $\{\langle argument \rangle\}$ and the token list `\l__scontents_mergesc_keys_tl` save the $\langle keys \rangle$ passed to the `\mergesc` (§12.19) command.

`\l__scontents_mergesc_keys_tl` The string variable `\l__scontents_current_seq_name_str` stores the name of the *current sequence* passed as an $\{\langle argument \rangle\}$ to the `\typestored` and `\meaningsc` commands and is used by the function `__scontents_parse_type_meaning_key:n`.

```

193 \tl_new:N \l__scontents_mergesc_arg_tl
194 \tl_new:N \l__scontents_mergesc_keys_tl
195 \str_new:N \l__scontents_current_seq_name_str

```

(End of definition for `\l__scontents_mergesc_arg_tl`, `\l__scontents_mergesc_keys_tl`, and `\l__scontents_current_seq_name_str`.)

`\l__scontents_file_name_tl` The token list `\l__scontents_file_name_tl` is used for store the name of the $\langle output file \rangle$, when there's one. Its value is set by the keys `write-env`, `write-out` and `write-cmd` (§12.7).

`\l__scontents_file_write_iow` The variable `\l__scontents_file_write_iow` is an *output stream* for write the $\{\langle body env \rangle\}$ of an environment or $\{\langle argument \rangle\}$ for command to a $\langle output file \rangle$ when the keys `write-env`, `write-out` or `write-cmd` are active.

The boolean variables `\l__scontents_writing_bool` and `\l__scontents_storing_bool` (true by default) set by the `write-out`, `write-env` and `write-cmd` keys determine whether the content is stored and written or just written to a $\langle output file \rangle$.

The boolean variable `\l__scontents_writable_bool` keeps track of whether we should write to a file, it is in write-only or in mode overwrite when the key `overwrite` is used.

🔗 This variable is used by the function `__scontents_file_if_writable:nTF` (see 12.10.2).

```

196 \tl_new:N \l__scontents_file_name_tl
197 \iow_new:N \l__scontents_file_write_iow
198 \bool_new:N \l__scontents_writing_bool
199 \bool_new:N \l__scontents_storing_bool
200 \bool_set_true:N \l__scontents_storing_bool
201 \bool_new:N \l__scontents_writable_bool

```

(End of definition for `\l__scontents_file_name_tl` and others.)

`\l__scontents_foreach_print_seq` Internal variables used by $\langle keys \rangle$ (§12.8.2) and implementation of `\foreachsc` command (§12.16).

```

202 \seq_new:N \l__scontents_foreach_print_seq
203 \tl_new:N \g__scontents_foreach_exec_tl
204 \tl_new:N \l__scontents_foreach_before_tl
205 \bool_new:N \l__scontents_foreach_before_bool
206 \tl_new:N \l__scontents_foreach_after_tl
207 \bool_new:N \l__scontents_foreach_after_bool
208 \int_new:N \l__scontents_foreach_stop_int
209 \bool_new:N \l__scontents_foreach_stop_bool
210 \bool_new:N \l__scontents_foreach_wrapper_bool

```

(End of definition for `\l__scontents_foreach_print_seq` and others.)

`\l__scontents_seq_item_seq` The sequence variable `\l__scontents_seq_item_seq` save the *indexes* in the *sequence* of the items requested to `\tystored`, `\mergesc` or `\meaningsc` and the sequence `\g__scontents_name_sc!internal_seq` assemble this.

```
211 \seq_new:N \l__scontents_seq_item_seq
212 \seq_new:c { g__scontents_name_sc!internal_seq }
```

(End of definition for `\l__scontents_seq_item_seq` and `\g__scontents_name_sc!internal_seq`.)

`\g__scontents_last_stored_tl` The token list `\g__scontents_last_stored_tl` used by the function `__scontents_lastfrom_seq:n` (§12.9) to execute the last *stored content* outside the group.

```
213 \tl_new:N \g__scontents_last_stored_tl
```

(End of definition for `\g__scontents_last_stored_tl`.)

`\c__scontents_hidden_space_str` The variable `\c__scontents_hidden_space_str` is a constant *string* to used to hide the *forced space* added by T_EX when recording content in a macro. This *string* contains the *reserved phrase* ‘`%^^Ascheol%`’ which is added to the end of the `{⟨argument⟩}` stored in *sequence* when the key `force-eol` is false.

```
214 \str_const:Nc \c__scontents_hidden_space_str
215 { \c_percent_str \c_circumflex_str \c_circumflex_str A scheol \c_percent_str }
```

(End of definition for `\c__scontents_hidden_space_str`.)

`\l__scontents_save_sf_int` Internal variables used by functions `__scontents_bsphack:` and `__scontents_esphack:` (§12.6.2).

```
\l__scontents_save_skip
216 \int_new:N \l__scontents_save_sf_int
217 \skip_new:N \l__scontents_save_skip
```

(End of definition for `\l__scontents_save_sf_int` and `\l__scontents_save_skip`.)

`\q__scontents_stop` Some quarks and scan’s used along the code as macro delimiters.

```
\q__scontents_mark
\__scontents_stop
\__scontents_mark
218 \quark_new:N \q__scontents_stop
219 \quark_new:N \q__scontents_mark
220 \scan_new:N \__scontents_stop
221 \scan_new:N \__scontents_mark
222 ⟨/core⟩
```

(End of definition for `\q__scontents_stop` and others.)

`\l__scontents_plain_bool` The boolean variable `\l__scontents_plain_bool` used in the plain T_EX implementation of the `verbatimsc` environment (§12.13).

```
223 ⟨*plain⟩
224 \bool_new:N \l__scontents_plain_bool
225 ⟨/plain⟩
```

(End of definition for `\l__scontents_plain_bool`.)

12.6 Utility functions

`\tl_if_empty: fTF` Some nonstandard kernel variant.

```
\tl_replace_all:NeV
\tl_retokenize:e
226 ⟨*core⟩
227 \prg_generate_conditional_variant:Nnn \tl_if_empty:n { f } { p, TF }
228 \cs_generate_variant:Nn \tl_replace_all:Nnn { NeV }
229 \cs_generate_variant:Nn \tl_retokenize:n { e }
```

(End of definition for `\tl_if_empty: fTF`, `\tl_replace_all:NeV`, and `\tl_retokenize:e`.)

`__scontents_use_delimit_by_s_stop:nw` Some functions used in the implementation of `\mergesc` (§12.19) and `scontents` (§12.12).

```
\__scontents_use_i_delimit_by_s_stop:nw
\__scontents_use_none_delimit_by_s_stop:w
\__scontents_use_none_delimit_by_q_stop:w
230 \cs_new:Npn \__scontents_use_delimit_by_s_stop:nw #1 \__scontents_stop {#1}
231 \cs_new:Npn \__scontents_use_i_delimit_by_s_stop:nw #1 #2 \__scontents_stop {#1}
232 \cs_new:Npn \__scontents_use_none_delimit_by_s_stop:w #1 \__scontents_stop { }
233 \cs_new:Npn \__scontents_use_none_delimit_by_q_stop:w #1 \q__scontents_stop { }
```

(End of definition for `__scontents_use_delimit_by_s_stop:nw` and others.)

`__scontents_tl_if_head_is_q_mark:nTF` The conditional function `__scontents_tl_if_head_is_q_mark:n` tests if the head of the token list is `\q__scontents_mark`.

```

234 \prg_new_protected_conditional:Npnn \__scontents_tl_if_head_is_q_mark:n #1
235 { T, F, TF }
236 {
237   \exp_after:wN \if_meaning:w
238     \exp_after:wN
239     \q__scontents_mark \__scontents_use_i_delimit_by_s_stop:nw #1 ? \s__scontents_stop
240     \prg_return_true:
241   \else:
242     \prg_return_false:
243   \fi:
244 }
```

(End of definition for `__scontents_tl_if_head_is_q_mark:nTF`.)

`__scontents_file_if_writable:n` The conditional function `__scontents_file_if_writable:n` used by the `write-env`, `write-cmd`, `write-out` and `overwrite` keys.

```

245 \prg_new_protected_conditional:Npnn \__scontents_file_if_writable:n #1 { T, F, TF }
246 {
247   \bool_if:NTF \l__scontents_writing_bool
248   {
249     \file_if_exist:nTF {#1}
250     {
251       \bool_if:NTF \l__scontents_overwrite_bool
252       {
253         \msg_warning:nne { scontents } { overwrite-file } {#1}
254         \prg_return_true:
255       }
256       {
257         \msg_warning:nne { scontents } { not-writing } {#1}
258         \prg_return_false:
259       }
260     }
261     {
262       \msg_warning:nne { scontents } { writing-file } {#1}
263       \prg_return_true:
264     }
265   }
266   { \prg_return_false: }
267 }
```

(End of definition for `__scontents_file_if_writable:n` and others.)

`__scontents_file_write_cmd:nn` The function `__scontents_file_write_cmd:nn` used by the `write-env`, `write-cmd`, `write-out` and `overwrite` keys for commands.

```

268 \cs_new_protected:Npn \__scontents_file_write_cmd:nn #1#2
269 {
270   \__scontents_file_if_writable:nT {#1}
271   {
272     \iow_open:Nn \l__scontents_file_write_iow {#1}
273     \iow_now:Nn \l__scontents_file_write_iow {#2}
274     \iow_close:N \l__scontents_file_write_iow
275   }
276 }
277 \cs_generate_variant:Nn \__scontents_file_write_cmd:nn { VV }
```

(End of definition for `__scontents_file_write_cmd:nn`.)

12.6.1 Functions for TAB and verbatim

`__scontents_tab:` Control sequences to replace tab `^^I` and form feed `^^L` characters.

```

278 \cs_new:Nc \__scontents_tab: { \c_space_tl }
279 \cs_new:Nn \__scontents_par: { ^^J ^^J }
```

(End of definition for `__scontents_tab:` and `__scontents_par:.`)

`__scontents_tabs_to_spaces:` In a *verbatim* context the TAB character is made active and set equal to `__scontents_tabs_to_spaces:`, to produce as many spaces as the `width-tab` key was set to.


```

280 \cs_new:Nn \__scontents_tabs_to_spaces:
281   { \prg_replicate:nn { \l__scontents_tab_width_int } { ~ } }

```

(End of definition for `__scontents_tabs_to_spaces:.`)

`__scontents_do_noligs:N` The function `__scontents_do_noligs:N` is an alternative definition for \TeX 2_{ϵ} 's `\do@noligs` which makes sure to not consume following space tokens. The \TeX 2_{ϵ} version ends with `\char`#1`, which leaves \TeX still looking for an *optional space*.

```

282 \cs_new_protected:Npn \__scontents_do_noligs:N #1
283   {
284     \char_set_catcode_active:N #1
285     \cs_set:cpe { __scontents_active_char_ \token_to_str:N #1 : }
286     {
287       \mode_leave_vertical:
288       \tex_kern:D \c_zero_dim
289       \tex_char:D ``\exp_not:N #1
290     }
291     \char_set_active_eq:Nc #1 { __scontents_active_char_ \token_to_str:N #1 : }
292   }

```

(End of definition for `__scontents_do_noligs:N`.)

`__scontents_set_active_eq:NN` `__scontents_make_control_chars_active:` `__scontents_plain_disable_outer_par:` Shortcut definitions for common catcode changes. The `'^^L'` needs a special treatment in non- \TeX mode because in plain \TeX it is an `\outer` token.

```

293 \cs_new_protected:Npn \__scontents_set_active_eq:NN #1
294   {
295     \char_set_catcode_active:N #1
296     \char_set_active_eq:NN #1
297   }
298 </core>
299 <*loader>
300 \group_begin:
301 <plain> \char_set_catcode_active:n { ``\* }
302 \cs_new_protected:Nn \__scontents_plain_disable_outer_par:
303 <*plain>
304   {
305     \group_begin:
306     \char_set_lccode:nn { ``\* } { ``^^L }
307     \tex_lowercase:D { \group_end:
308       \tex_let:D * \scan_stop:
309     }
310   }
311 </plain>
312 <latex|context> { }
313 \group_end:
314 </loader>
315 <*core>
316 \group_begin:
317 \char_set_catcode_active:N \*
318 \cs_new_protected:Nn \__scontents_make_control_chars_active:
319   {
320     \__scontents_plain_disable_outer_par:
321     \__scontents_set_active_eq:NN ^^I \__scontents_tab:
322     \__scontents_set_active_eq:NN ^^L \__scontents_par:
323     \__scontents_set_active_eq:NN ^^M \__scontents_ret:w
324   }
325 \group_end:
326 </core>

```

(End of definition for `__scontents_set_active_eq:NN`, `__scontents_make_control_chars_active:`, and `__scontents_plain_disable_outer_par:.`)

12.6.2 Functions `\@bsphack` and `\@esphack`

`__scontents_bsphack:` `__scontents_esphack:` We emulate `\@bsphack` and `\@esphack` for plain \TeX . This is necessary to prevent *unwanted spaces* when the `print-cmd` key is false.

```

327 <*core>
328 \cs_new_protected:Nn \__scontents_bsphack:
329   {

```

```

330 \scan_stop:
331 \mode_if_horizontal:T
332 {
333     \skip_set_eq:NN \l__scontents_save_skip \tex_lastskip:D
334     \int_set_eq:NN \l__scontents_save_sf_int \tex_spacefactor:D
335 }
336 }
337 \cs_new_protected:Nn \l__scontents_esphack:
338 {
339     \scan_stop:
340     \mode_if_horizontal:T
341     {
342         \int_set_eq:NN \tex_spacefactor:D \l__scontents_save_sf_int
343         \dim_compare:nNnT { \l__scontents_save_skip } > { \c_zero_skip }
344         {
345             \skip_if_eq:nnT { \tex_lastskip:D } { \c_zero_skip }
346             {
347                 \nobreak
348                 \skip_horizontal:n { \c_zero_skip }
349             }
350             \tex_ignorespaces:D
351         }
352     }
353 }
354 </core>
355 <*latex>
356 \cs_gset_eq:NN \l__scontents_bsphack: \@bsphack
357 \cs_gset_eq:NN \l__scontents_esphack: \@esphack
358 </latex>

```

(End of definition for `\l__scontents_bsphack:` and `\l__scontents_esphack:`.)

12.7 Keys for environment

We define a set of *keys* for environment `scontents`.

```

write-env
write-out
print-env
store-env
force-eol
overwrite
unknown
359 <core>
360 \keys_define:nn { scontents / scontents }
361 {
362     write-env .code:n          = {
363         \bool_set_true:N \l__scontents_storing_bool
364         \bool_set_true:N \l__scontents_writing_bool
365         \tl_set:Nn \l__scontents_file_name_tl {#1}
366     },
367     write-out .code:n          = {
368         \bool_set_false:N \l__scontents_storing_bool
369         \bool_set_true:N \l__scontents_writing_bool
370         \tl_set:Nn \l__scontents_file_name_tl {#1}
371     },
372     write-env .value_required:n = true,
373     write-out .value_required:n = true,
374     print-env .meta:nn          = { scontents } { print-env = #1 },
375     print-env .default:n         = true,
376     store-env .meta:nn          = { scontents } { store-env = #1 },
377     force-eol .meta:nn          = { scontents } { force-eol = #1 },
378     force-eol .default:n         = true,
379     overwrite .meta:nn          = { scontents } { overwrite = #1 },
380     overwrite .default:n         = true,
381     unknown .code:n             = { \l__scontents_unknown_keys_env:n {#1} },
382 }

```

(End of definition for `write-env` and others.)

12.7.1 Handling unknown keys for environment `scontents`

The *keys* are save in the string variable `\l_keys_key_str` and the value (if any) is passed as an argument to each *function*.

```

\l__scontents_unknown_keys_env:n
\l__scontents_unknown_keys_env:n

```

We check the *keys* passed to the environment `scontents` and process it with `\l__scontents_parse_environment_keys:n` if the *key* is *unknown* we return an error message.

```

383 \cs_new_protected:Npn \l__scontents_unknown_keys_env:n #1

```

```

384 { \exp_args:NV \__scontents_unknown_keys_env:nn \l_keys_key_str {#1} }
385 \cs_new_protected:Npn \__scontents_unknown_keys_env:nn #1#2
386 {
387   \tl_if_blank:nTF {#2}
388     { \msg_error:nnn { scontents } { env-key-unknown } {#1} }
389     { \msg_error:nnnn { scontents } { env-key-value-unknown } {#1} {#2} }
390 }

```

(End of definition for `__scontents_unknown_keys_env:n` and `__scontents_unknown_keys_env:nn`.)

12.8 Keys for commands

We add the `<keys>` divided into subgroups to handle *errors* and *unknown* `<keys>` separately.

12.8.1 Keys for command `\Scontents`

We define a set of `<keys>` for commands `\Scontents` and `\Scontents*`.

```

write-cmd  We define a set of <keys> for commands \Scontents and \Scontents*.
write-out
print-cmd
store-cmd
force-eol
overwrite
unknown
391 \keys_define:nn { scontents / Scontents }
392 {
393   write-cmd .code:n          = {
394     \bool_set_true:N \__scontents_storing_bool
395     \bool_set_true:N \__scontents_writing_bool
396     \tl_set:Nn \l__scontents_file_name_tl {#1}
397   },
398   write-out .code:n         = {
399     \bool_set_false:N \__scontents_storing_bool
400     \bool_set_true:N \__scontents_writing_bool
401     \tl_set:Nn \l__scontents_file_name_tl {#1}
402   },
403   write-cmd .value_required:n = true,
404   write-out .value_required:n = true,
405   print-cmd .meta:nn         = { scontents } { print-cmd = #1 },
406   print-cmd .default:n       = true,
407   store-cmd .meta:nn         = { scontents } { store-cmd = #1 },
408   force-eol .meta:nn         = { scontents } { force-eol = #1 },
409   force-eol .default:n       = true,
410   overwrite .meta:nn         = { scontents } { overwrite = #1 },
411   overwrite .default:n       = true,
412   unknown .code:n            = { \__scontents_unknown_keys_cmd:n {#1} },
413 }

```

(End of definition for `write-cmd` and others.)

12.8.2 Keys for command `\foreachsc`

We define a set of `<keys>` for command `\foreachsc`.

```

before  We define a set of <keys> for command \foreachsc.
after
start
stop
step
wrapper
sep
unknown
414 \keys_define:nn { scontents / foreachsc }
415 {
416   before .code:n          = {
417     \bool_set_true:N \__scontents_foreach_before_bool
418     \tl_set:Nn \l__scontents_foreach_before_tl {#1}
419   },
420   before .value_required:n = true,
421   after .code:n           = {
422     \bool_set_true:N \__scontents_foreach_after_bool
423     \tl_set:Nn \l__scontents_foreach_after_tl {#1}
424   },
425   after .value_required:n = true,
426   start .int_set:N        = \l__scontents_foreach_start_int,
427   start .value_required:n = true,
428   start .initial:n        = 1,
429   stop .code:n            = {
430     \bool_set_true:N \__scontents_foreach_stop_bool
431     \int_set:Nn \l__scontents_foreach_stop_int {#1}
432   },
433   stop .value_required:n = true,
434   step .int_set:N        = \l__scontents_foreach_step_int,
435   step .value_required:n = true,
436   step .initial:n        = 1,
437   wrapper .code:n        = {
438     \bool_set_true:N \__scontents_foreach_wrapper_bool

```

```

439         \cs_set_protected:Npn
440         \__scontents_foreach_wrapper:n #1 {#1}
441     },
442     wrapper .value_required:n = true,
443     sep .tl_set:N = \l__scontents_foreach_sep_tl,
444     sep .initial:n = { },
445     sep .value_required:n = true,
446     unknown .code:n = { \__scontents_unknown_keys_cmd:n {#1} },
447 }

```

(End of definition for *before* and *others*.)

12.8.3 Handling unknown keys for \Scontents and \foreachsc

```

\__scontents_unknown_keys_cmd:n
\__scontents_unknown_keys_cmd:nn

```

We check the *⟨keys⟩* passed to commands \Scontents, \Scontents* or \foreachsc and process it with __scontents_unknown_keys_cmd:n if the *⟨key⟩* is *unknown* we return an error message.

```

448 \cs_new_protected:Npn \__scontents_unknown_keys_cmd:n #1
449 { \exp_args:NV \__scontents_unknown_keys_cmd:nn \l_keys_key_str {#1} }
450 \cs_new_protected:Npn \__scontents_unknown_keys_cmd:nn #1#2
451 {
452     \tl_if_blank:nTF {#2}
453     { \msg_error:nnn { scontents } { cmd-key-unknown } {#1} }
454     { \msg_error:nnnn { scontents } { cmd-key-value-unknown } {#1} {#2} }
455 }

```

(End of definition for __scontents_unknown_keys_cmd:n and __scontents_unknown_keys_cmd:nn.)

12.8.4 Keys for commands \typestored and \meaningsc

```

width-tab
write-out
overwrite
print-cmd
unknown

```

We define a *⟨keys⟩* for command \typestored and \meaningsc. Both commands accept the same type of *optional arguments*, just define a common *⟨keys⟩*. Here we will implement the *write-out*, *overwrite* and *print-cmd* keys which are necessary in the implementation of the \mergesc command (§12.19).

```

456 \keys_define:nn { scontents / typemeaning }
457 {
458     width-tab .meta:nn = { scontents } { width-tab = #1 },
459     write-out .code:n = {
460         \bool_set_false:N \l__scontents_storing_bool
461         \bool_set_true:N \l__scontents_writing_bool
462         \tl_set:Nn \l__scontents_file_name_tl {#1}
463     },
464     write-out .value_required:n = true,
465     overwrite .meta:nn = { scontents } { overwrite = #1 },
466     overwrite .default:n = true,
467     print-cmd .bool_set:N = \l__scontents_print_verb_style_bool,
468     print-cmd .initial:n = true,
469     print-cmd .default:n = true,
470     unknown .code:n = { \__scontents_parse_type_meaning_key:n {#1} },
471 }

```

(End of definition for *width-tab* and *others*.)

12.8.5 Keys for command \mergesc

```

typestored
meaningsc
\__scontents_mergesc_cmd:nn

```

We define two *⟨keys⟩* typestored and meaningsc as *mandatory*, returning an “error” through the function __scontents_mergesc_cmd:nn.

```

472 \keys_define:nn { scontents / mergesc }
473 {
474     typestored .code:n =
475     {
476         \cs_set_eq:NN \__scontents_mergesc_cmd:nn \__scontents_typestored:nn
477     },
478     typestored .value_forbidden:n = true,
479     meaningsc .code:n =
480     {
481         \cs_set_eq:NN \__scontents_mergesc_cmd:nn \__scontents_meaningsc:nn
482     },
483     meaningsc .value_forbidden:n = true,
484 }
485 \cs_new_protected:Npn \__scontents_mergesc_cmd:nn #1 #2
486 {
487     \msg_error:nn { scontents } { mergesc-missing-key }

```

488 }

(End of definition for `\typestored`, `\meaningsc`, and `__scontents_mergesc_cmd:nn`.)

12.8.6 Parsing keys for `\typestored`, `\meaningsc` and `\mergesc`

```
\__scontents_parse_type_meaning_key:n
\__scontents_parse_type_meaning_key:nn
\__scontents_parse_type_meaning_range:w
  \__scontents_range_parser:nnnn
  \__scontents_range_parser:nnen
  \__scontents_range_parser_aux:nnn
```

The `\typestored`, `\meaningsc` and `\mergesc` commands (which internally uses the previous two) accept an *optional argument* containing the *index* position, *1-end* or the range of *start-stop* positions of the *stored content* in the *sequence* along with other *keys*.

To avoid the awkward `\typestored[...] [⟨keys⟩] {...}` syntax, we'll make the commands have a single *optional argument* which is processed by `l3keys[9]`, and the *unknown* *keys* are brought here to `__scontents_parse_type_meaning_key:n` to process.

First we check if the *key* is an integer using `\int_to_roman:n`. If it is, we check that the value passed to the *key* is blank (otherwise something odd as `1=1` might have been used). If everything is correct, then set the value of the integer which holds the *index*, otherwise raise an error about an *unknown* option.

```
489 \cs_new_protected:Npn \__scontents_parse_type_meaning_key:n #1
490 { \exp_args:NV \__scontents_parse_type_meaning_key:nn \l_keys_key_str {#1} }
491 \cs_new_protected:Npn \__scontents_parse_type_meaning_key:nn #1#2
492 {
493   \tl_if_blank:nTF {#2}
494     { \__scontents_parse_type_meaning_range:w #1 - \q__scontents_mark - \s__scontents_mark }
495     { \msg_error:nnee { scontents } { cmd-key-value-unknown } {#1} {#2} }
496   }
497 \cs_new_protected:Npn \__scontents_parse_type_meaning_range:w #1 - #2 - #3 \s__scontents_mark
498 {
499   \__scontents_range_parser:nnen {#1} {#2}
500   { \seq_count:c { g__scontents_name_ \l__scontents_current_seq_name_str_seq } }
501   { \msg_error:nnn { scontents } { cmd-key-unknown } {#1} }
502 }
503 \cs_new_protected:Npn \__scontents_range_parser:nnnn #1 #2 #3 #4
504 {
505   \exp_args:Nee \__scontents_range_parser_aux:nnn
506   { \str_if_eq:nnTF {#1} { end } {#3} { \exp_not:n {#1} } }
507   { \str_if_eq:nnTF {#2} { end } {#3} { \exp_not:n {#2} } }
508   { #4 }
509 }
510 \cs_generate_variant:Nn \__scontents_range_parser:nnnn { nnen }
511 \cs_new_protected:Npn \__scontents_range_parser_aux:nnn #1 #2 #3
512 {
513   \__scontents_tl_if_head_is_q_mark:nTF {#2}
514   {
515     \tl_if_empty:ftTF { \int_to_roman:n { -0 #1 } }
516     { \seq_put_right:Ne \l__scontents_seq_item_seq { \int_eval:n {#1} } }
517     { #3 {#1} }
518   }
519   {
520     \bool_lazy_and:nnTF
521     { \tl_if_empty_p:f { \int_to_roman:n { -0 #1 } } }
522     { \tl_if_empty_p:f { \int_to_roman:n { -0 #2 } } }
523     {
524       \int_compare:nNnTF {#2} > {#1}
525       { \int_step_inline:nnnn {#1} { 1 } {#2} }
526       { \int_step_inline:nnnn {#1} { -1 } {#2} }
527       { \seq_put_right:Nn \l__scontents_seq_item_seq {#1} }
528     }
529     { #3 { #1-#2 } }
530   }
531 }
532 </core>
```

(End of definition for `__scontents_parse_type_meaning_key:n` and others.)

12.9 Functions for sequences

The *storage system* of the package `SCONTENTS` is done using `seq` variables. Here we set up the macros that will manage the variables.

```
\__scontents_append_contents:nn
\__scontents_append_contents:Ve
```

The function `__scontents_append_contents:nn` creates a *sequence* `{⟨seq name⟩}` pass to `#1` if one didn't exist and appends the `{⟨body env⟩}` for environments or `{⟨argument⟩}` for commands to the right of the *sequence* `{⟨seq name⟩}` pass to `#2`.

```

533  ⟨*core⟩
534  \cs_new_protected:Npn \__scontents_append_contents:nn #1#2
535  {
536    \tl_if_blank:nT {#1}
537      { \msg_error:nn { scontents } { empty-store-cmd } }
538    \seq_if_exist:cF { g__scontents_name_#1_seq }
539      { \seq_new:c { g__scontents_name_#1_seq } }
540    \seq_gput_right:cn { g__scontents_name_#1_seq } {#2}
541  }
542  \cs_generate_variant:Nn \__scontents_append_contents:nn { Ve }

```

(End of definition for __scontents_append_contents:nn.)

__scontents_store_to_seq:NN The function __scontents_store_to_seq:NN writes the recorded contents in #1 to the log if debug-var is true, and stores it in #2.

```

543  \cs_new_protected:Npn \__scontents_store_to_seq:NN #1#2
544  {
545    \bool_if:NT \__scontents_debug_var_bool
546      {
547        \tl_log:N #1
548      }
549    \__scontents_append_contents:Ve #2 { \exp_not:V #1 }
550  }

```

(End of definition for __scontents_store_to_seq:NN.)

__scontents_finish_storing:NNN The __scontents_finish_storing:NNN function will first check if we are in standard *storing mode*, that is, the `write-out` key is NOT active, then the state of the variable __scontents_forced_eol_bool set by the `force-eol` key is checked and if this is “false” (default value) we will add \c__scontents_hidden_space_str to the end of the token list passed in {#1} which contains {⟨body env⟩} for the generic environment `scontents` and the {⟨argument⟩} for the `\Scontents` command.

Then the function __scontents_store_to_seq:NN is applied to “store” in the *sequence* passed in {#2} and finally the state of the boolean variable passed in {#3} established by the `print-env` and `print-cmd` keys is checked and if it is “true”, {⟨body env⟩} or {⟨argument⟩} will be printed from the *sequence* in which it was stored by means of the __scontents_lastfrom_seq:V function.

```

551  \cs_new_protected:Npn \__scontents_finish_storing:NNN #1 #2 #3
552  {
553    \bool_if:NT \__scontents_storing_bool
554      {
555        \bool_if:NF \__scontents_forced_eol_bool
556        { \tl_put_right:Ne #1 { \c__scontents_hidden_space_str } }
557        \__scontents_store_to_seq:NN #1 #2
558        \bool_if:NT #3 { \__scontents_lastfrom_seq:V #2 }
559      }
560  }

```

(End of definition for __scontents_finish_storing:NNN.)

__scontents_getfrom_seq:Nn The function __scontents_getfrom_seq:Nn retrieves the ⟨stored content⟩ pass to {#1} from the *sequence* {⟨seq name⟩} pass to {#2}.

__scontents_getfrom_seq:nNn
 __scontents_getfrom_seq_aux:nnn
 __scontents_getfrom_seq:nn
 __scontents_getfrom_seq:nnn

```

561  \cs_new:Npn \__scontents_getfrom_seq:Nn #1#2
562  {
563    \seq_if_exist:cTF { g__scontents_name_#2_seq }
564      {
565        \exp_args:Nf \__scontents_getfrom_seq:nNn
566        { \seq_count:c { g__scontents_name_#2_seq } } #1 {#2}
567      }
568      { \msg_expandable_error:nnn { scontents } { undefined-storage } {#2} }
569  }
570  \cs_new:Npn \__scontents_getfrom_seq:nNn #1 #2 #3
571  {
572    \seq_map_tokens:Nn #2 { \__scontents_getfrom_seq_aux:nnn {#1} {#3} }
573  }
574  \cs_new:Npn \__scontents_getfrom_seq_aux:nnn #1 #2 #3
575  {
576    \exp_args:Nnf \use:n { \__scontents_getfrom_seq:nnn {#1} } { \int_eval:n {#3} } {#2}
577  }
578  \cs_new:Npn \__scontents_getfrom_seq:nn #1#2

```



```

579 {
580   \seq_if_exist:cTF { g__scontents_name_#2_seq }
581   {
582     \exp_args:Nf \__scontents_getfrom_seq:nnn
583     { \seq_count:c { g__scontents_name_#2_seq } }
584     {#1} {#2}
585   }
586   { \msg_expandable_error:nnn { scontents } { undefined-storage } {#2} }
587 }
588 \cs_new:Npn \__scontents_getfrom_seq:nnn #1#2#3
589 {
590   \bool_lazy_or:nnTF
591   { \int_compare_p:nNn {#2} = { 0 } }
592   { \int_compare_p:nNn { \int_abs:n {#2} } > {#1} }
593   { \msg_expandable_error:nnnnn { scontents } { index-out-of-range } {#2} {#3} {#1} }
594   { \seq_item:cn { g__scontents_name_#3_seq } {#2} }
595 }

```

(End of definition for `__scontents_getfrom_seq:Nn` and others.)

`__scontents_lastfrom_seq:n` The function `__scontents_lastfrom_seq:n` save the last *stored content* from the *sequence* pass to `#1` in `\g__scontents_last_tl` then rescan with the function `\tl_retokenize:V` when the keys `print-env` or `print-cmd` are active.

```

596 \cs_new_protected:Npn \__scontents_lastfrom_seq:n #1
597 {
598   \tl_gset:Nx \g__scontents_last_stored_tl
599   {
600     \seq_item:cn { g__scontents_name_#1_seq } { -1 }
601   }
602   \group_insert_after:N \tl_retokenize:V
603   \group_insert_after:N \g__scontents_last_stored_tl
604   \group_insert_after:N \tl_gclear:N
605   \group_insert_after:N \g__scontents_last_stored_tl
606 }
607 \cs_generate_variant:Nn \__scontents_lastfrom_seq:n { V }
608 </core>

```

(End of definition for `__scontents_lastfrom_seq:n`.)

12.10 Base functions for environments

In version 1.8 (2019-11-18) the command `\newenvsc` (§12.11) was implemented, allowing you to create environments with the same behavior as the base environment `scontents`. Since that version, the base environment `scontents` (§12.12) is defined using `\newenvsc`.

The references to `\begin{scontents}` or `\end{scontents}` described in this section are for illustrative purposes only, but apply to any environment defined using `\newenvsc`.

12.10.1 Functions for keyval of environment

`__scontents_grab_opt_arg:n` The function `__scontents_grab_opt_arg:w` is called from the `scontents` environment with the tokens following the `\begin{scontents}` when the next character is a '['. This function is defined using `ltxcmd[10]` to exploit its delimited argument processor.

```

609 <core>
610 \NewDocumentCommand \__scontents_grab_opt_arg:w { r[] }
611 {
612   \__scontents_grab_opt_arg:n {#1}
613 }

```

The function is called from a context where `^^M` is active, so `__scontents_normalise_line_ends:N` is used to replace active `^^M` characters by spaces.

```

614 \cs_new_protected:Npn \__scontents_grab_opt_arg:n #1
615 {
616   \tl_if_novalue:nF {#1}
617   {
618     \tl_set:Nn \__scontents_environment_keys_tl {#1}
619     \__scontents_normalise_line_ends:N \__scontents_environment_keys_tl
620     \keys_set:nV { scontents / scontents } \__scontents_environment_keys_tl
621   }
622   \__scontents_start_after_option:w
623 }

```

624 $\langle /core \rangle$

(End of definition for `_scontents_grab_opt_arg:n` and `_scontents_grab_opt_arg:w`.)

12.10.2 Functions for save the body of environment

```
\_scontents_start_environment:w
\_scontents_start_after_option:w
\_scontents_check_line_process:en
  \_scontents_stop_environment:
```

Here we make ‘`^^I`’, ‘`^^L`’ and ‘`^^M`’ active characters so that the end of line can be “seen” to be used as a delimiter, and $\text{T}_{\text{E}}\text{X}$ doesn’t try to eliminate space-like characters.

First we check if the immediate next token after `\begin{scontents}` is a ‘`[`’. If it is, then `_scontents_grab_opt_arg:w` is called to do the heavy lifting. `_scontents_grab_opt_arg:w` processes the optional argument and calls `_scontents_start_after_option:w`.

The function `_scontents_start_after_option:w` also checks for trailing tokens after the optional argument and issues an error if any. In all cases, the function `_scontents_check_line_process:en` checks that everything past `\begin{scontents}` is empty and then process the environment.

The function `_scontents_check_line_process:en` calls the function `_scontents_file_tl_write_start:V` which will then read the contents of the environment and optionally store them in a token list or write to an *external file*.

When that’s done, the function `_scontents_file_write_stop:N` does the cleanup. This part of the code is inspired and adapted from the code of the package `xsim`[4] by Clemens Niederberger.

```
625  $\langle *core \rangle$ 
626 \group_begin:
627   \char_set_catcode_active:N \^^I
628   \char_set_catcode_active:N \^^L
629   \char_set_catcode_active:N \^^M
630   \cs_new_protected:Npn \_scontents_normalise_line_ends:N #1
631     { \tl_replace_all:Nnn #1 { ^^M } { ~ } }
632   \cs_new_protected:Npn \_scontents_start_environment:w #1 ^^M
633     {
634       \tl_if_head_is_N_type:nTF {#1}
635       {
636         \str_if_eq:eeTF { \tl_head:n {#1} } { [ ]
637           { \_scontents_grab_opt_arg:w #1 ^^M }
638           { \_scontents_check_line_process:en { } {#1} }
639         }
640         { \_scontents_check_line_process:en { } {#1} }
641       }
642   \cs_new_protected:Npn \_scontents_start_after_option:w #1 ^^M
643     { \_scontents_check_line_process:en { [...] } {#1} }
644   \cs_new_protected:Npn \_scontents_check_line_process:en #1 #2
645     {
646       \tl_if_blank:nF {#2}
647       {
648         \msg_error:nnee { scontents } { junk-after-begin }
649         { after~\c_backslash_str begin { \_scontents_env_name_tl } #1 } {#2}
650       }
651       \_scontents_make_control_chars_active:
652       \_scontents_file_tl_write_start:V \_scontents_file_name_tl
653     }
654   \cs_new_protected:Nn \_scontents_stop_environment:
655     {
656       \_scontents_file_write_stop:N \_scontents_processed_body_lines_tl
657       \bool_lazy_and:nnT
658         { \l_scontents_storing_bool }
659         { \tl_if_empty_p:N \_scontents_processed_body_lines_tl }
660         {
661           \msg_warning:nne { scontents } { empty-environment } { \_scontents_env_name_tl }
662         }
663     }
```

(End of definition for `_scontents_start_environment:w` and others.)

```
\_scontents_file_tl_write_start:n
\_scontents_file_tl_write_start:V
\_scontents_verb_processor_iterate:w
\_scontents_verb_processor_iterate:nnn
\_scontents_setup_verb_processor:
  \_scontents_file_write_stop:N
\_scontents_remove_leading_nl:n
\_scontents_remove_leading_nl:w
```

This is the main macro/function to collect the $\langle body\ env \rangle$ of a verbatim environment. The function `_scontents_file_tl_write_start:n` starts a *group*, opens the *output file*, if necessary, sets *verbatim catcodes*, and then issues ‘`^^M`’ (set equal to `_scontents_ret:w`) to read $\langle body\ env \rangle$ of the environment line by line until reaching its end. The output token list will be appended with an active ‘`^^J`’ character and the line just read, and this line is written to the *output file*, if any. At the end of the environment the *output file* is closed (if it was open), and the output token list is smuggled out of the verbatim group. A

leading ‘^^J’ is removed from the token list using `__scontents_remove_leading_nl:n`, which expects an active ‘^^J’ token at the head of the token list; a low level TeX “error” is raised otherwise.

```

664 \cs_new_protected:Npn \__scontents_file_tl_write_start:n #1
665 {
666   \group_begin:
667   \__scontents_file_if_writable:nTF {#1}
668   {
669     \bool_set_true:N \l__scontents_writable_bool
670     \iow_open:Nn \l__scontents_file_write_iow {#1}
671   }
672   { \bool_set_false:N \l__scontents_writable_bool }
673   \tl_clear:N \l__scontents_save_every_body_lines_tl
674   \seq_map_function:NN \l_char_special_seq \char_set_catcode_other:N
675   \int_step_function:nnnN { 128 } { 1 } { 255 } \char_set_catcode_letter:n
676   \cs_set_protected:Npe \__scontents_ret:w ##1 ^^M
677   {
678     \exp_not:N \__scontents_verb_processor_iterate:w
679     ##1 \c__scontents_end_env_tl
680     \c__scontents_end_env_tl
681     \exp_not:N \q__scontents_stop
682   }
683   \__scontents_make_control_chars_active:
684   \__scontents_ret:w
685 }
686 \cs_new:Nn \__scontents_setup_verb_processor:
687 {
688   \use:e
689   {
690     \cs_set:Npn \exp_not:N \__scontents_verb_processor_iterate:w
691     ##1 \c__scontents_end_env_tl
692     ##2 \c__scontents_end_env_tl
693     ##3 \exp_not:N \q__scontents_stop
694     } { \__scontents_verb_processor_iterate:nnn {##1} {##2} {##3} }
695   }
696 \cs_new:Npn \__scontents_verb_processor_iterate:nnn #1 #2 #3
697 {
698   \tl_if_blank:nTF {#3}
699   {
700     \__scontents_analyse_nesting:n {#1}
701     \__scontents_verb_processor_output:n {#1}
702   }
703   {
704     \__scontents_if_nested:TF
705     {
706       \__scontents_nesting_decr:
707       \__scontents_verb_processor_output:e
708       { \exp_not:n {#1} \c__scontents_end_env_tl \exp_not:n {#2} }
709     }
710     {
711       \tl_if_blank:nF {#1}
712       { \__scontents_verb_processor_output:n {#1} }
713       \cs_set_protected:Npe \__scontents_ret:w
714       {
715         \__scontents_env_end_function:
716         \bool_lazy_or:nnF
717         { \tl_if_blank_p:n {#2} }
718         { \str_if_eq_p:ee {#2} { \c_percent_str } }
719         {
720           \str_if_eq:VnF \c__scontents_hidden_space_str {#2}
721           {
722             \msg_warning:nnnn { scontents } { rescanning-text }
723             {#2} { \tl_use:N \l__scontents_env_name_tl }
724           }
725           \tl_retokenize:n {#2}
726         }
727       }
728       \char_set_active_eq:NN ^^M \__scontents_ret:w
729     }
730   }
731   ^^M
732 }

```

```

733 \cs_new:Nn \__scontents_env_end_function:
734 {
735   \__scontents_format_case:nnn
736   { \exp_not:N \end { \if_false: } \fi: }
737   { \exp_after:wN \exp_not:N \cs:w end }
738   { \exp_after:wN \exp_not:N \cs:w stop }
739   \tl_use:N \l__scontents_env_name_tl
740   \__scontents_format_case:nnn
741   { \if_false: { \fi: } }
742   { \cs_end: }
743   { \cs_end: }
744 }
745 \cs_new_protected:Npn \__scontents_file_write_stop:N #1
746 {
747   \bool_if:NT \l__scontents_writable_bool
748   { \iow_close:N \l__scontents_file_write_iow }
749   \use:e
750   {
751     \group_end:
752     \bool_if:NT \l__scontents_storing_bool
753     {
754       \tl_set:Nn \exp_not:N #1
755       {
756         \exp_args:NV
757         \__scontents_remove_leading_nl:n \l__scontents_save_every_body_lines_tl
758       }
759     }
760   }
761 }
762 \cs_new:Npn \__scontents_remove_leading_nl:n #1
763 {
764   \tl_if_head_is_N_type:nTF {#1}
765   {
766     \exp_args:Nf
767     \__scontents_remove_leading_nl:nn
768     { \tl_head:n {#1} } {#1}
769   }
770   { \exp_not:n {#1} }
771 }
772 \cs_new:Npn \__scontents_remove_leading_nl:nn #1 #2
773 {
774   \token_if_eq_meaning:NNTF ^^J #1
775   { \exp_not:o { \__scontents_remove_leading_nl:w #2 } }
776   { \exp_not:n {#2} }
777 }
778 \cs_new:Npn \__scontents_remove_leading_nl:w ^^J { }

```

(End of definition for `__scontents_file_tl_write_start:n` and others.)

`__scontents_verb_processor_output:n`
`__scontents_verb_processor_output:e`

The function `__scontents_verb_processor_output:n` does the output of each line read, to a token list and to a file, depending on the booleans `\l__scontents_writing_bool` and `\l__scontents_storing_bool`.

```

779 \cs_new_protected:Npn \__scontents_verb_processor_output:n #1
780 {
781   \bool_if:NT \l__scontents_writable_bool
782   { \iow_now:Nn \l__scontents_file_write_iow {#1} }
783   \bool_if:NT \l__scontents_storing_bool
784   { \tl_put_right:Nn \l__scontents_save_every_body_lines_tl { ^^J #1 } }
785 }
786 \group_end:
787 \cs_generate_variant:Nn \__scontents_verb_processor_output:n { e }
788 \cs_generate_variant:Nn \__scontents_file_tl_write_start:n { V }

```

(End of definition for `__scontents_verb_processor_output:n`.)

`__scontents_analyse_nesting:n`
`__scontents_analyse_nesting:w`
`__scontents_nesting_decr:`
`__scontents_use_none_delimit_by_q_stop:w`
`__scontents_if_nested:TF`

The function `__scontents_analyse_nesting:n` scans nested `\begin{contents}` and steps a `\l__scontents_nesting_env_int` counter. The `__scontents_if_nested:` conditional tests if we're in a nested environment, and `__scontents_nesting_decr:` reduces the nesting level, if an `\end{contents}` is found.

```

789 \cs_new_protected:Npn \__scontents_analyse_nesting:n #1
790 {
791   \int_zero:N \l__scontents_nesting_aux_int
792   \__scontents_analyse_nesting_format:n {#1}
793   \int_compare:nNnT { \l__scontents_nesting_aux_int } > { 1 }
794   { \msg_warning:nn { scontents } { multiple-begin } }
795 }
796 \cs_new_protected:Nn \__scontents_nesting_incr:
797 {
798   \int_incr:N \l__scontents_nesting_env_int
799   \int_incr:N \l__scontents_nesting_aux_int
800 }
801 \cs_new_protected:Nn \__scontents_nesting_decr:
802 {
803   \int_decr:N \l__scontents_nesting_env_int
804 }
805 \prg_new_protected_conditional:Npnn \__scontents_if_nested: { TF }
806 {
807   \int_compare:nNnTF { \l__scontents_nesting_env_int } > { \c_zero_int }
808   { \prg_return_true: }
809   { \prg_return_false: }
810 }

```

Multiple `\end{scontents}` in the same line are NOT supported...

In \LaTeX , environments start with `\begin{«env»}`, so checking if a string contains `\begin{scontents}` is straightforward. Since no `}` can appear inside `«env»`, then just a macro delimited by `'` is enough.

```

811 \use:e
812 {
813   \cs_new_protected:Npn \exp_not:N \__scontents_analyse_nesting_latex:w #1
814   \c_backslash_str begin \c_left_brace_str #2 \c_right_brace_str
815 } {
816   \__scontents_tl_if_head_is_q_mark:nTF {#2}
817   { \__scontents_use_none_delimit_by_q_stop:w }
818   {
819     \str_if_eq:VnT \l__scontents_env_name_tl {#2}
820     { \__scontents_nesting_incr: }
821     \__scontents_analyse_nesting_latex:w
822   }
823 }
824 \cs_new_protected:Npe \__scontents_analyse_nesting_latex:n #1
825 {
826   \__scontents_analyse_nesting_latex:w #1
827   \c_backslash_str begin
828   \c_left_brace_str \exp_not:N \q__scontents_mark \c_right_brace_str
829   \exp_not:N \q__scontents_stop
830 }

```

In other formats, however, we don't have an “end anchor” to delimit the environment name, so a delimited macro won't help. We have to search for the entire environment command (usually `\scontents` and `\startscontents`).

```

831 \cs_new_protected:Npn \__scontents_analyse_nesting_generic_process:nn #1 #2
832 {
833   \tl_if_head_is_N_type:nTF {#2}
834   {
835     \__scontents_tl_if_head_is_q_mark:nF {#2}
836     {
837       \__scontents_nesting_incr:
838       \__scontents_analyse_nesting_generic:w #2 \q__scontents_stop
839     }
840   }
841   { \__scontents_analyse_nesting_generic:w #2 \q__scontents_stop }
842 }
843 \cs_new_protected:Npn \__scontents_analyse_nesting_generic:nn #1 #2
844 {
845   \__scontents_define_generic_nesting_function:n {#1}
846   \use:e
847   {
848     \exp_not:N \__scontents_analyse_nesting_generic:w #2
849     \c_backslash_str #1 \tl_use:N \l__scontents_env_name_tl
850     \exp_not:N \q__scontents_mark \exp_not:N \q__scontents_stop
851   }

```

```

852 }
853 \cs_new_protected:Npn \__scontents_define_generic_nesting_function:n #1
854 {
855   \use:e
856   {
857     \cs_set_protected:Npn \exp_not:N \__scontents_analyse_nesting_generic:w ##1
858     \c_backslash_str #1 \tl_use:N \l__scontents_env_name_tl
859     ##2 \exp_not:N \q__scontents_stop
860   } { \__scontents_analyse_nesting_generic_process:nn {##1} {##2} }
861 }
862 </core>

```

Now we just need to call the `__scontents_analyse_nesting_format:n` function to analyze the nesting.

```

863 <*loader>
864 <latex>\cs_new_eq:NN \__scontents_analyse_nesting_format:n
865 <latex> \__scontents_analyse_nesting_latex:n
866 <!latex>\cs_new_protected:Npn \__scontents_analyse_nesting_format:n
867 <plain> { \__scontents_analyse_nesting_generic:nn { } }
868 <context> { \__scontents_analyse_nesting_generic:nn { start } }
869 </loader>

```

(End of definition for `__scontents_analyse_nesting:n` and others.)

12.11 The command `\newenvsc`

In version 1.8 (2019-11-18) the command `\newenvsc` was implemented, allowing you to create environments with the same behavior as the base environment `scontents`. To achieve this, we will create new environments so that they wrap around the base functions (§12.10).

`__scontents_generic_begin:` The function `__scontents_generic_begin:` leaves the ‘`^M`’ character active and calls the generic environment start function `__scontents_start_environment:w`.

```

870 <*core>
871 \cs_new_protected:Npn \__scontents_generic_begin:
872 {
873   \char_set_catcode_active:N ^M
874   \__scontents_start_environment:w
875 }

```

The function `__scontents_generic_end:` calls the generic environment stop function `__scontents_stop_environment:` and finally calls the function `__scontents_finish_storing:NNN` which stores `{<body env>}` in the *sequence* `{<seq name>}` and prints it from the *sequence* if the `print-env` key is active.

```

876 \cs_new_protected:Npn \__scontents_generic_end:
877 {
878   \__scontents_stop_environment:
879   \__scontents_finish_storing:NNN
880   \l__scontents_processed_body_lines_tl
881   \l__scontents_name_seq_env_tl
882   \l__scontents_print_env_bool
883 }

```

(End of definition for `__scontents_generic_begin:` and `__scontents_generic_end:`.)

`__scontents_setting_env:nn` The function `__scontents_setting_env:nn` receives the environment *name* passed in `{#1}` and save it in the variable `\l__scontents_env_name_tl` along with the initial `<keys>` passed in `{#2}`.

`__scontents_define_env:nnn`

Two functions will be created `__scontents_#1_begin:` and `__scontents_#1_end:` which will internally call the `__scontents_generic_begin:` and `__scontents_generic_end:` functions and expand the arguments of the function `__scontents_define_env:nnn` function.

```

884 \cs_new_protected:Npn \__scontents_setting_env:nn #1 #2
885 {
886   \cs_new_protected:cpn { __scontents_#1_begin: }
887   {
888     \tl_set:Nn \l__scontents_env_name_tl {#1}
889     \keys_set:nn { scontents } {#2}
890     \__scontents_setup_verb_processor:
891     \__scontents_generic_begin:
892   }
893   \cs_new_protected:cpn { __scontents_#1_end: }
894   { \__scontents_generic_end: }
895   \exp_args:Nooo \__scontents_define_env:nnn % http://nooooooooooooooooo.com :) jeje

```

```

896     { \tl_to_str:n {#1} }
897     { \cs:w __scontents_#1_begin: \cs_end: }
898     { \cs:w __scontents_#1_end: \cs_end: }
899   }
900 </core>

```

The function `__scontents_define_env:nnn` will create the environments for \LaTeX , plain \TeX and \ConTeXt .

```

901 <*loader>
902 \cs_new_protected:Npn \__scontents_define_env:nnn #1 #2 #3
903 {
904   <latex|plain>   \NewDocumentEnvironment {#1} { }
905   <context>       \cs_new_protected:cpn { start #1 }
906   {
907     <!!latex>     \group_begin:
908                 #2
909   }
910   <context>       \cs_new_protected:cpn { stop #1 }
911   {
912                 #3
913     <!!latex>     \group_end:
914   }
915 }

```

(End of definition for `__scontents_setting_env:nn` and `__scontents_define_env:nnn`.)

`\newenvsc` Now we just need to create the user command `\newenvsc` for \LaTeX , plain \TeX and \ConTeXt .

```

916 \NewDocumentCommand \newenvsc { m O{} }
917 {
918   <latex|plain>   \cs_if_exist:cTF { #1 }
919   <context>       \cs_if_exist:cTF { start #1 }
920   { \msg_error:nnn { scontents } { env-already-defined } {#1} }
921   { \__scontents_setting_env:nn {#1} {#2} }
922 }
923 </loader>

```

(End of definition for `\newenvsc`. This function is documented on page 5.)

12.12 The environment scontents

`scontents` Finally defining the `scontents` environment should be easy :)

```

\contents
\contents
\endscontents
\startcontents
\stopscontents

```

(End of definition for `scontents` and others. These functions are documented on page 4.)

12.13 The environment verbatimsc

The `verbatimsc` environment is, in a way, a customized version of the standard `verbatim` environment provided by \LaTeX . For correct operation in plain \TeX , \LaTeX and \ConTeXt , we must add a couple of additional functions.

`\dospecials` The `verbatim` environment in \LaTeX requires `\dospecials`. In case it doesn't exist (at the time `SCONTENTS` is loaded) we define `\dospecials` to use the `\l_char_special_seq`.

```

927 <!!latex>
928 \cs_if_exist:NF \dospecials
929 {
930   \cs_new:Npn \dospecials
931   { \seq_map_function:NN \l_char_special_seq \do }
932 }
933 </!!latex>

```

(End of definition for `\dospecials`.)

`__scontents_xverb:w` The environment `verbatimsc` needs to literally find the end of this `\end{verbatimsc}` in the case of \LaTeX . Here we set this for plain \TeX , \LaTeX , and \ConTeXt .

```

\end{verbatimsc}
\endverbatimsc
\stopverbatimsc

```

```

934 <*loader>

```

```

935 ⟨*!context⟩
936 \use:e
937 {
938   \cs_new_protected:Npn \exp_not:N \__scontents_xverb:w
939     #1 \g__scontents_end_verbatimsc_tl
940   ⟨latex⟩      { #1 \exp_not:N \end{verbatimsc} }
941   ⟨plain⟩      { #1 \exp_not:N \endverbatimsc }
942   ⟨context⟩    { #1 \exp_not:N \stopverbatimsc }
943 }
944 ⟨/!context⟩
945 ⟨/loader⟩

```

(End of definition for `__scontents_xverb:w` and others.)

12.13.1 plain T_EX version off `verbatimsc`

In plain T_EX we emulate L^AT_EX’s `verbatim` environment.

```

\verbatimsc
\endverbatimsc
\__scontents_verbatimsc_aux:
  \__scontents_vobeyspaces:
    \__scontents_xverb:
  \__scontents_nolig_list:
    \__scontents_xobeysp:
946 ⟨*plain⟩
947 \cs_new_protected:Npn \verbatimsc
948 {
949   \group_begin:
950     \__scontents_verbatimsc_aux: \frenchspacing \__scontents_vobeyspaces:
951     \__scontents_xverb:
952   }
953 \cs_new_protected:Npn \endverbatimsc
954   { \group_end: }
955 \cs_new_protected:Nn \__scontents_verbatimsc_aux:
956   {
957     \skip_vertical:N \parskip
958     \dim_zero:N \parindent
959     \skip_set:Nn \parfillskip { 0pt plus 1fil }
960     \skip_set:Nn \parskip { 0pt plus 0pt minus 0pt }
961     \tex_par:D
962     \bool_set_false:N \l__scontents_plain_bool
963     \cs_set:Npn \par
964       {
965         \bool_if:NTF \l__scontents_plain_bool
966           {
967             \mode_leave_vertical:
968             \null
969             \tex_par:D
970             \penalty \interlinepenalty
971           }
972           {
973             \bool_set_true:N \l__scontents_plain_bool
974             \mode_if_horizontal:T
975               { \tex_par:D \penalty \interlinepenalty }
976           }
977         }
978     \cs_set_eq:NN \do \char_set_catcode_other:N
979     \dospecials \obeylines
980     \tl_use:N \l__scontents_verb_font_tl
981     \cs_set_eq:NN \do \__scontents_do_noligs:N
982     \__scontents_nolig_list:
983     \tex_everypar:D \exp_after:wN
984       { \tex_the:D \tex_everypar:D \tex_unpenalty:D }
985   }
986 \cs_new_protected:Nn \__scontents_nolig_list:
987   { \do\` \do\< \do\> \do\, \do\' \do\~ }
988 \cs_new_protected:Nn \__scontents_vobeyspaces:
989   { \__scontents_set_active_eq:NN \ \__scontents_xobeysp: }
990 \cs_new_protected:Nn \__scontents_xobeysp:
991   { \mode_leave_vertical: \nobreak \ }
992 ⟨/plain⟩

```

(End of definition for `\verbatimsc` and others.)

12.13.2 ConT_EXt version off `verbatimsc`

In ConT_EXt we use our own tool `\definetyping`.

```

\startverbatimsc
\stopverbatimsc
993 ⟨*loader⟩

```



```

994 <context>\definetyping[verbatimsc]
995 </loader>

```

(End of definition for `\startverbatimsc` and `\stopverbatimsc`.)

12.13.3 L^AT_EX version of `verbatimsc`

`__scontents_verbatimsc_instance:` To be compatible with *tagged* PDF we must define the environment `verbatimsc` in terms of the `xtemplate`[10] module integrated into the L^AT_EX kernel and the new `blocks-code` from `latex-lab`[13]. This code is adapted directly from Mrs. Ulrike Fischer’s answer to *New verbatim environment with block code (tagged-pdf)*.

```

996 <*loader>
997 <*latex>
998 \cs_new_protected:Nn \__scontents_verbatimsc_instance:
999 {
1000   \DeclareInstance{blockenv}{verbatimsc}{std}
1001   {
1002     name          = verbatimsc,
1003     tag-name      = \UseStructureName{block/verbatim},
1004     tagging-recipe = standard,
1005     tagging-suppress-paras = true,
1006     block-instance = verbatim,
1007     para-instance = justify,
1008     final-code    = \legacyverbatimsetup{invisible},
1009   }
1010 }
1011 \NewDocumentEnvironment { verbatimsc } { }
1012 {
1013   \IfDocumentMetadataTF
1014   {
1015     \__scontents_verbatimsc_instance:
1016     \UseInstance{blockenv}{verbatimsc}{}
1017     \@setupverbinvisiblespace\frenchspacing\@vobeyspaces
1018     \__scontents_xverb:
1019   }
1020   {
1021     \cs_set_eq:cN { @xverbatim } \__scontents_xverb:
1022     \verbatim
1023   }
1024 }

```

The `\endverbatim` in the second argument of the `verbatimsc` environment is only needed for compatibility with the `verbatim`[15] package.

```

1025 {
1026   \IfDocumentMetadataTF
1027   {
1028     \BlockEnvEnd
1029   }
1030   { \endverbatim }
1031 }
1032 </latex>
1033 </loader>

```

(End of definition for `__scontents_verbatimsc_instance:` and `verbatimsc`. This function is documented on page 7.)

12.14 The command `\Scontents`

`\Scontents` User command `\Scontents` to *<stored content>* in a *sequence*, adapted from code by Ulrich Diez in *Stringify input - \string on token list* and code by user @siracusa in *Convert a macro from Latexze to expl3*.

```

1034 <*core>
1035 \NewDocumentCommand \Scontents { !s !O{} }
1036 {
1037   \tl_set:Nn \l__scontents_cmd_name_tl { Scontents }
1038   \__scontents_Scontents_code:nn {#1} {#2}
1039 }

```

The internal function `__scontents_Scontents_code:nn` first executes `__scontents_bsphack:`, opens a group and checks the *<keys>* passed in `{#2}`, leaves the TAB character active and then checks the *starred argument* ‘*’ passed in `{#1}`. If the latter is present it will call the function `__scontents_Scontents_verb_arg:w` otherwise it will call the function `__scontents_Scontents_norm_arg:n`.

```

1040 \cs_new_protected:Npn \__scontents_Scontents_code:nn #1 #2
1041 {
1042   \__scontents_bsphack:
1043   \group_begin:
1044     \tl_if_novalue:nF {#2}
1045     { \keys_set:nn { scontents / Scontents } {#2} }
1046     \char_set_catcode_active:n { 9 }
1047     \bool_if:NTF #1
1048     { \__scontents_Scontents_verb_arg:w }
1049     { \__scontents_Scontents_norm_arg:n }
1050 }

```

The function `__scontents_Scontents_verb_arg:w` saves the $\langle argument \rangle$ passed in $\{#1\}$ under the *verbatim catcode* using *v*-type argument from `ltxcmd` in `\l__scontents_Scontents_arg_tl` and replaces all ‘`^^M`’ and `\obeyedline` added in L^AT_EX release 2024-06-01 by ‘`^^J`’ and then calls the function `__scontents_Scontents_finish:`. Here we will apply `\RenewDocumentCommand` since `\obeyedline` can be modified by the user and if so the code would return a low-level error.

```

1051 \NewDocumentCommand \__scontents_Scontents_verb_arg:w { +v }
1052 {
1053   \tl_set:Nn \l__scontents_Scontents_arg_tl {#1}
1054   \tl_if_empty:NT \l__enumext_ref_key_arg_tl
1055   {
1056     \msg_warning:nne { scontents } { empty-argument-command } { \l__scontents_Scontents_arg_tl }
1057   }
1058   \cs_if_exist:NT \obeyedline
1059   {
1060     \RenewDocumentCommand \obeyedline { } { \iow_char:N ^^M }
1061     \tl_replace_all:Nee \l__scontents_Scontents_arg_tl { \obeyedline } { \iow_char:N ^^M }
1062   }
1063   \tl_replace_all:NeV \l__scontents_Scontents_arg_tl { \iow_char:N ^^M } \c__scontents_newline_tl
1064   \__scontents_Scontents_finish:
1065 }

```

The function `__scontents_Scontents_norm_arg:n` saves the $\langle argument \rangle$ passed in $\{#1\}$ in the *standard catcode* regimen and then calls the function `__scontents_Scontents_finish:`.

```

1066 \cs_new_protected:Npn \__scontents_Scontents_norm_arg:n #1
1067 {
1068   \tl_set:Nn \l__scontents_Scontents_arg_tl {#1}
1069   \__scontents_Scontents_finish:
1070 }

```

The function `__scontents_Scontents_finish:` will first call the function `__scontents_file_write_cmd:VV` used by the `write-out` and `write-cmd` keys, then call the function `__scontents_finish_storing:NNN` to store the $\langle argument \rangle$ passed to the `\Scontents` command saved in the `\l__scontents_Scontents_arg_tl` variable in the *sequence* `\l__scontents_name_seq_cmd_tl` and print it from this *sequence* according to the state of the `\l__scontents_print_cmd_bool` variable set by the `print-cmd` key. Finally it will close the *group* opened at the beginning of the command definition and run `__scontents_esphack:` if the `print-cmd` key is not active.

```

1071 \cs_new_protected:Nn \__scontents_Scontents_finish:
1072 {
1073   \__scontents_file_write_cmd:VV \l__scontents_file_name_tl \l__scontents_Scontents_arg_tl
1074   \__scontents_finish_storing:NNN
1075   \l__scontents_Scontents_arg_tl \l__scontents_name_seq_cmd_tl \l__scontents_print_cmd_bool
1076   \use:e
1077   {
1078     \group_end:
1079     \bool_if:NF \l__scontents_print_cmd_bool { \__scontents_esphack: }
1080   }
1081 }

```

(End of definition for `\Scontents` and others. This function is documented on page 5.)

12.15 The command `\getstored`

`\getstored` User command `\getstored` to extract $\langle stored content \rangle$ in *sequence* (robust).

```

\__scontents_getstored:nn
1082 \NewDocumentCommand \getstored { 0{-1} m }
1083 {
1084   \__scontents_getstored:nn {#1} {#2}
1085 }

```

The internal function `__scontents_getstored:nn` will set the end of line then apply the function `\tl_retokenize:e` on the *stored content* at the *index* given by `{#1}` in the sequence `{#2}` via the function `__scontents_getfrom_seq:nn`.

```

1086 \cs_new_protected:Npn \__scontents_getstored:nn #1 #2
1087 {
1088     \tl_retokenize:e
1089     {
1090         \__scontents_getfrom_seq:nn {#1} {#2}
1091     }
1092 }

```

(End of definition for `\getstored` and `__scontents_getstored:nn`. This function is documented on page 6.)

12.16 The command `\foreachsc`

`\foreachsc`

User command `\foreachsc` to loop over *stored content* in *sequence*.

`__scontents_foreachsc:nn`
`__scontents_foreach_add_body:n`

```

1093 \NewDocumentCommand \foreachsc { o m }
1094 {
1095     \tl_set:Nn \__scontents_cmd_name_tl { foreachsc }
1096     \__scontents_foreachsc:nn {#1} {#2}
1097 }
1098 \cs_new_protected:Npn \__scontents_foreachsc:nn #1 #2
1099 {
1100     \group_begin:
1101     \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / foreachsc } {#1} }
1102     \tl_set:Nn \l__scontents_foreachsc_arg_tl {#2}
1103     \seq_clear:N \l__scontents_foreach_print_seq
1104     \bool_if:NF \l__scontents_foreach_stop_bool
1105     {
1106         \int_set:Nn \l__scontents_foreach_stop_int
1107         { \seq_count:c { g__scontents_name_#2_seq } }
1108     }
1109     \int_step_function:nnnN
1110     { \l__scontents_foreach_start_int }
1111     { \l__scontents_foreach_step_int }
1112     { \l__scontents_foreach_stop_int }
1113     \__scontents_foreach_add_body:n
1114     \tl_gset:Ne \g__scontents_foreach_exec_tl
1115     {
1116         \exp_args:NNV \seq_use:Nn
1117         \l__scontents_foreach_print_seq \l__scontents_foreach_sep_tl
1118     }
1119     \group_end:
1120     \exp_after:wN \tl_gclear:N
1121     \exp_after:wN \g__scontents_foreach_exec_tl
1122     \g__scontents_foreach_exec_tl
1123 }
1124 \cs_new_protected:Npn \__scontents_foreach_add_body:n #1
1125 {
1126     \seq_put_right:Ne \l__scontents_foreach_print_seq
1127     {
1128         \bool_if:NT \l__scontents_foreach_before_bool
1129         { \exp_not:V \l__scontents_foreach_before_tl }
1130         \bool_if:NTF \l__scontents_foreach_wrapper_bool
1131         { \__scontents_foreach_wrapper:n }
1132         { \use:n }
1133         { \getstored [#1] { \tl_use:N \l__scontents_foreachsc_arg_tl } }
1134         \bool_if:NT \l__scontents_foreach_after_bool
1135         { \exp_not:V \l__scontents_foreach_after_tl }
1136     }
1137 }

```

(End of definition for `\foreachsc`, `__scontents_foreachsc:nn`, and `__scontents_foreach_add_body:n`. This function is documented on page 6.)

12.17 The command `\typestored`

`\typestored`

The `\typestored` commands fetches a buffer from memory, prints it to the log file, and then calls `__scontents_typestored:N`.

`__scontents_typestored:nn`
`__scontents_typestored:N`
`__scontents_xverb:w`

```

1138 \NewDocumentCommand \typestored { o m }

```

```

1139 {
1140   \tl_set:Nn \l__scontents_cmd_name_tl { typestored }
1141   \__scontents_typestored:nn {#1} {#2}
1142 }
1143 \cs_new_protected:Npn \__scontents_typestored:nn #1 #2
1144 {
1145   \__scontents_bsphack:
1146   \group_begin:
1147     \seq_clear:N \l__scontents_seq_item_seq
1148     \str_set:Nx \l__scontents_current_seq_name_str {#2}
1149     \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / typemeaning } {#1} }
1150     \seq_if_empty:NT \l__scontents_seq_item_seq
1151       { \seq_set_from_clist:Nn \l__scontents_seq_item_seq { 1 } }
1152     \tl_set:Nx \l__scontents_typestored_arg_tl
1153       { \__scontents_getfrom_seq:Nn \l__scontents_seq_item_seq {#2} }
1154     \__scontents_remove_trailing_eol:N \l__scontents_typestored_arg_tl
1155     \tl_replace_all:NxV \l__scontents_typestored_arg_tl { \c__scontents_hidden_space_str } \c__scontents_hidden_space_str
1156     \bool_if:NT \l__scontents_debug_var_bool
1157       {
1158         \tl_log:N \l__scontents_typestored_arg_tl
1159       }
1160     \tl_if_empty:NF \l__scontents_typestored_arg_tl
1161       {
1162         \bool_if:NT \l__scontents_print_verb_style_bool
1163           {
1164             \__scontents_typestored:N \l__scontents_typestored_arg_tl
1165           }
1166       }
1167     \__scontents_file_write_cmd:VV \l__scontents_file_name_tl \l__scontents_typestored_arg_tl
1168     \use:e
1169     {
1170   \group_end:
1171   \bool_if:NF \l__scontents_print_verb_style_bool { \__scontents_esphack: }
1172   }
1173 }

```

The `__scontents_typestored:N` macro is defined with active carriage return (ASCII 13) characters to *mimic* an actual verbatim environment “on the loose”. The contents of the environment are placed in a `verbatimsc` environment and rescanned using `\tl_retokenize:e`.

```

1174 \group_begin:
1175   \char_set_catcode_active:N ^^M
1176   \cs_new_protected:Npn \__scontents_typestored:N #1
1177   {
1178     \tl_if_blank:VT #1
1179       { \msg_error:nnn { scontents } { empty-variable } {#1} }
1180     \cs_set_eq:NN \__scontents_verb_print_EOL: ^^M
1181     \cs_set_eq:NN ^^M \scan_stop:
1182     \cs_set_eq:cN { do@noligs } \__scontents_do_noligs:N
1183     \int_set:Nn \tex_newlinechar:D { ``^^J }
1184     \tl_retokenize:e
1185     {
1186       \__scontents_format_case:nnn
1187         { \exp_not:N \begin{verbatimsc} } % LaTeX
1188         { \verbatimsc } % Plain/Generic
1189         { \startverbatimsc } % ConTeXt
1190       ^^M
1191       \exp_not:V #1 ^^M
1192       \g__scontents_end_verbatimsc_tl
1193     }
1194     \cs_set_eq:NN ^^M \__scontents_verb_print_EOL:
1195   }
1196 \group_end:
1197 \cs_new_protected:Nn \__scontents_xverb:
1198 {
1199   \char_set_catcode_active:n { 9 }
1200   \char_set_active_eq:nN { 9 } \__scontents_tabs_to_spaces:
1201   \__scontents_xverb:w
1202 }

```

(End of definition for `\typestored` and others. This function is documented on page 6.)

12.18 The command `\meaningsc`

`\meaningsc` User command `\meaningsc` to see content stored in seq.

```

__scontents_meaningsc:nn
__scontents_meaningsc:n

1203 \NewDocumentCommand \meaningsc { o m }
1204 {
1205     \tl_set:Nn \l__scontents_cmd_name_tl { meaningsc }
1206     \__scontents_meaningsc:nn {#1} {#2}
1207 }
1208 \cs_new_protected:Npn \__scontents_meaningsc:nn #1 #2
1209 {
1210     \__scontents_bsphack:
1211     \group_begin:
1212         \seq_clear:N \l__scontents_seq_item_seq
1213         \str_set:Ne \l__scontents_current_seq_name_str {#2}
1214         \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / typemeaning } {#1} }
1215         \seq_if_empty:NT \l__scontents_seq_item_seq
1216             { \seq_set_from_clist:Nn \l__scontents_seq_item_seq { 1 } }
1217         \__scontents_meaningsc:n {#2}
1218         \use:e
1219         {
1220             \group_end:
1221             \bool_if:NF \l__scontents_print_verb_style_bool { \__scontents_esphack: }
1222             }
1223     }
1224     \group_begin:
1225     \char_set_catcode_active:N ^^I
1226     \cs_new_protected:Npn \__scontents_meaningsc:n #1
1227     {
1228         \tl_set:Ne \l__scontents_meaningsc_arg_tl
1229             { \__scontents_getfrom_seq:Nn \l__scontents_seq_item_seq {#1} }
1230         \tl_replace_all:Nen \l__scontents_meaningsc_arg_tl { \iow_char:N ^^J } { ~ }
1231         \tl_replace_all:Nen \l__scontents_meaningsc_arg_tl { \c__scontents_hidden_space_str } { ~ }
1232         \bool_if:NT \l__scontents_debug_var_bool
1233             {
1234                 \tl_log:N \l__scontents_meaningsc_arg_tl
1235             }
1236         \tl_use:N \l__scontents_verb_font_tl
1237         \tl_replace_all:Nne \l__scontents_meaningsc_arg_tl { ^^I } { \__scontents_tabs_to_spaces: }
1238         \tl_if_empty:NF \l__scontents_meaningsc_arg_tl
1239             {
1240                 \bool_if:NT \l__scontents_print_verb_style_bool
1241                 {
1242                     \cs_replacement_spec:N \l__scontents_meaningsc_arg_tl
1243                 }
1244             }
1245         \__scontents_file_write_cmd:VV \l__scontents_file_name_tl \l__scontents_meaningsc_arg_tl
1246     }
1247 \group_end:

```

(End of definition for `\meaningsc`, `__scontents_meaningsc:nn`, and `__scontents_meaningsc:n`. This function is documented on page 6.)

12.19 The command `\mergesc`

The `\mergesc` command parses a comma separated list given as $\langle argument \rangle$, and just assembles it as a temporary *internal sequence*, then passes it to the `\typestored` or `\meaningsc` command.

`\mergesc`

The `\mergesc` command parses a list given as argument, and just assembles it as a temporary internal sequence, then passes it to the requested command.

```

__scontents_mergesc_code:nn
__scontents_mergesc_parse_list:n
__scontents_remove_trailing_eol:N
__scontents_remove_trailing_eol:w
__scontents_parse_mergesc:nw
__scontents_parse_mergesc_aux:nw
__scontents_parse_mergesc_range:nw

1248 \NewDocumentCommand \mergesc { o m }
1249 {
1250     \tl_set:Nn \l__scontents_cmd_name_tl { mergesc }
1251     \__scontents_mergesc_code:nn {#1} {#2}
1252 }
1253 \cs_new_protected:Npn \__scontents_mergesc_code:nn #1 #2
1254 {
1255     \group_begin:
1256         \tl_clear:N \l__scontents_mergesc_keys_tl
1257         \tl_if_novalue:nF {#1}
1258             {
1259                 \keys_set_known:nnN { scontents / mergesc } {#1} \l__scontents_mergesc_keys_tl

```

```

1260     }
1261     \seq_gclear:c { g__scontents_name_sc!internal_seq }
1262     \__scontents_mergesc_parse_list:n {#2}
1263     \exp_args:Ne \__scontents_mergesc_cmd:nn
1264     { 1-end, \exp_not:V \l__scontents_mergesc_keys_tl } { sc!internal }
1265     \group_end:
1266 }

1267 \cs_new_protected:Npn \__scontents_mergesc_parse_list:n #1
1268 {
1269     \clist_map_inline:nn {#1} { \__scontents_parse_mergesc:nw ##1 \s__scontents_stop }
1270     \seq_gpop_right:cN { g__scontents_name_sc!internal_seq } \l__scontents_mergesc_arg_tl
1271     \__scontents_remove_trailing_eol:N \l__scontents_mergesc_arg_tl
1272     \seq_gput_right:cV { g__scontents_name_sc!internal_seq } \l__scontents_mergesc_arg_tl
1273 }
1274 \cs_new_protected:Npe \__scontents_remove_trailing_eol:N #1
1275 {
1276     \exp_not:N \exp_after:wN \exp_not:N \__scontents_remove_trailing_eol:w
1277     #1 \s__scontents_stop \c__scontents_hidden_space_str \s__scontents_stop \s__scontents_mark #1
1278 }
1279 \use:e
1280 {
1281     \cs_new_protected:Npn \exp_not:N \__scontents_remove_trailing_eol:w #1
1282     \c__scontents_hidden_space_str \s__scontents_stop #2 \s__scontents_mark #3
1283 } {
1284     \tl_set:Ne #3
1285     {
1286         \tl_if_empty:nTF {#2}
1287         { \exp_not:o { \__scontents_use_delimit_by_s_stop:nw #1 } }
1288         { \exp_not:n {#1} }
1289     }
1290 }
1291 \cs_new_protected:Npn \__scontents_parse_mergesc:nw #1
1292 {
1293     \peek_remove_spaces:n
1294     {
1295         \peek_charcode:NTF [ % ]
1296         { \__scontents_parse_mergesc_aux:nw {#1} }
1297         { \__scontents_parse_mergesc_aux:nw {#1} [ 1-\seq_count:c { g__scontents_name_#1_seq } ] }
1298     }
1299 }
1300 \cs_new_protected:Npn \__scontents_parse_mergesc_aux:nw #1 [#2]
1301 {
1302     \seq_clear:N \l__scontents_seq_item_seq
1303     \clist_map_inline:nn {#2}
1304     { \__scontents_parse_mergesc_range:nw {#1} ##1 - \q__scontents_mark - \s__scontents_mark }
1305     \seq_map_inline:Nn \l__scontents_seq_item_seq
1306     {
1307         \seq_gput_right:ce { g__scontents_name_sc!internal_seq }
1308         { \seq_item:cn { g__scontents_name_#1_seq } {##1} }
1309     }
1310     \__scontents_use_none_delimit_by_s_stop:w
1311 }
1312 \cs_new_protected:Npn \__scontents_parse_mergesc_range:nw #1 #2 - #3 - #4 \s__scontents_mark
1313 {
1314     \cs_set_protected:Npn \__scontents_tmp:w ##1
1315     {
1316         \msg_error:nneee { scontents } { index-out-of-range }
1317         {##1} {#1} { \seq_count:c { g__scontents_name_#1_seq } }
1318     }
1319     \__scontents_range_parser:nnen {#2} {#3}
1320     { \seq_count:c { g__scontents_name_#1_seq } }
1321     { \__scontents_tmp:w }
1322 }

```

(End of definition for `\mergesc` and others. This function is documented on page 7.)

12.20 The command `\setupsc`

`\setupsc` User command `\setupsc` to setup module for `\keys_set:nn { scontents }`.

```

1323 \NewDocumentCommand \setupsc { +m }

```

```

1324 {
1325     \keys_set:nn { scontents } {#1}
1326 }

```

(End of definition for \setupsc. This function is documented on page 3.)

12.21 The command \countsc

`\countsc` User command `\countsc` to count number of *stored contents* in the sequence.

```

1327 \NewExpandableDocumentCommand \countsc { m }
1328 {
1329     \seq_count:c { g__scontents_name_#1_seq }
1330 }

```

(End of definition for \countsc. This function is documented on page 7.)

12.22 The command \cleanseqsc

`\cleanseqsc` A user command `\cleanseqsc` to clear (remove) all *stored contents* in the sequence.

```

1331 \NewDocumentCommand \cleanseqsc { m }
1332 {
1333     \seq_gclear_new:c { g__scontents_name_#1_seq }
1334 }

```

(End of definition for \cleanseqsc. This function is documented on page 8.)

12.23 Warning and error messages

Warning and error messages used throughout the package.

```

1335 \msg_new:nnn { scontents } { junk-after-begin }
1336 {
1337     Junk~characters~#1~\msg_line_context: :
1338     \\ \\
1339     #2
1340 }
1341 \msg_new:nnnn { scontents } { env-already-defined }
1342 { Environment~'#1'~already-defined! }
1343 {
1344     You~have~used~\newenvsc
1345     with~an~environment~that~already~has~a~definition. \\ \\
1346     The~existing~definition~of~'#1'~will~not~be~altered.
1347 }
1348 \msg_new:nnn { scontents } { empty-argument-command }
1349 { Empty~argument~for~'Scontents'~\msg_line_context:. }
1350 \msg_new:nnn { scontents } { empty-environment }
1351 { environment~'#1'~empty~\msg_line_context:. }
1352 \msg_new:nnn { scontents } { empty-variable }
1353 { Variable~'#1'~empty~\msg_line_context:. }
1354 \msg_new:nnn { scontents } { overwrite-file }
1355 { Overwriting~file~'#1'. }
1356 \msg_new:nnn { scontents } { writing-file }
1357 { Writing~file~'#1'. }
1358 \msg_new:nnn { scontents } { not-writing }
1359 { File~'#1'~already~exists.~Not~writing. }
1360 \msg_new:nnn { scontents } { rescanning-text }
1361 { Rescanning~text~'#1'~after~\c_backslash_str end{#2}~\msg_line_context:. }
1362 \msg_new:nnn { scontents } { multiple-begin }
1363 { Multiple~\c_backslash_str begin{ \l__scontents_env_name_tl }~\msg_line_context:. }
1364 \msg_new:nnn { scontents } { undefined-storage }
1365 { Storage~named~'#1'~is~not~defined. }
1366 \msg_new:nnnn { scontents } { mergesc-missing-key }
1367 {
1368     Need~mandatory~key~'typestored'~or~'meaningsc'~for~\
1369     command~\c_backslash_str mergesc~\msg_line_context:.
1370 }
1371 {
1372     The~command~\c_backslash_str mergesc~need~a~mandatory~key~typestored~or~meaningsc.\\
1373     Check~that~you~have~spelled~the~key~name~correctly.
1374 }
1375 \msg_new:nnn { scontents } { index-out-of-range }

```

```

1376 {
1377   \int_compare:nNnTF {#1} = { 0 }
1378   { Index~of~sequence~cannot~be~zero. }
1379   {
1380     Index~'#1'~out~of~range~for~'#2',~
1381     \int_compare:nNnTF {#1} > { 0 }
1382     { Max = } { Min = - } #3.
1383   }
1384 }
1385 \msg_new:nnnn { scontents } { env-key-unknown }
1386 {
1387   The~key~'#1'~is~unknown~by~environment~
1388   '\l__scontents_env_name_tl'~and~is~being~ignored.
1389 }
1390 {
1391   The~environment~'\l__scontents_env_name_tl'~does~not~have~a~key~called~'#1'.\\
1392   Check~that~you~have~spelled~the~key~name~correctly.
1393 }
1394 \msg_new:nnnn { scontents } { env-key-value-unknown }
1395 {
1396   The~key~'#1=#2'~is~unknown~by~environment~
1397   '\l__scontents_env_name_tl'~and~is~being~ignored.
1398 }
1399 {
1400   The~environment~'\l__scontents_env_name_tl'~does~not~have~a~key~called~'#1'.\\
1401   Check~that~you~have~spelled~the~key~name~correctly.
1402 }
1403 \msg_new:nnnn { scontents } { cmd-key-unknown }
1404 {
1405   The~key~'#1'~is~unknown~by~command~\c_backslash_str \l__scontents_cmd_name_tl \c_space_tl
1406   and~is~being~ignored~ \msg_line_context:.
1407 }
1408 {
1409   The~command~\c_backslash_str \l__scontents_cmd_name_tl \c_space_tl
1410   does~not~have~a~key~called~'#1'.\\
1411   Check~that~you~have~spelled~the~key~name~correctly.
1412 }
1413 \msg_new:nnnn { scontents } { cmd-key-value-unknown }
1414 {
1415   The~key~'#1=#2'~is~unknown~by~command~\c_backslash_str \l__scontents_cmd_name_tl \c_space_tl
1416   and~is~being~ignored~ \msg_line_context:.
1417 }
1418 {
1419   The~command~\c_backslash_str \l__scontents_cmd_name_tl \c_space_tl
1420   does~not~have~a~key~called~'#1'.\\
1421   Check~that~you~have~spelled~the~key~name~correctly.
1422 }
1423 </core>

```

12.24 Finish package

Finish package implementation.

```

1424 <plain|context>\ExplSyntaxOff
1425 <plain|context>\endinput
1426 <latex|core>\file_input_stop:

```


13 Index of Implementation

The italic numbers denote the pages where the corresponding entry is described, the numbers underlined and all others indicate the line on which they are implemented in the package code.

Symbols	
\'	987
*	301, 306, 317
\,	987
\-	987
\<	987
\>	987
\\	30, 120, 1338, 1345, 1368, 1372, 1391, 1400, 1410, 1420
\`	987
A	
after	414
B	
before	414
\beginngroup	52, 57
\BlockEnvEnd	1028
bool commands:	
\bool_if:NTF	104, 247, 251, 545, 553, 555, 558, 747, 752, 781, 783, 965, 1047, 1079, 1104, 1128, 1130, 1134, 1156, 1162, 1171, 1221, 1232, 1240
\bool_lazy_and:nnTF	520, 657
\bool_lazy_or:nnTF	590, 716
\bool_new:N	198, 199, 201, 205, 207, 209, 210, 224
\bool_set_false:N	368, 399, 460, 672, 962
\bool_set_true:N	200, 363, 364, 369, 394, 395, 400, 417, 422, 430, 438, 461, 669, 973
C	
\catcode	53
char commands:	
\char_set_active_eq:NN	291, 296, 728
\char_set_active_eq:nN	1200
\char_set_catcode:nn	23
\char_set_catcode_active:N	284, 295, 317, 627, 628, 629, 873, 1175, 1225
\char_set_catcode_active:n	301, 1046, 1199
\char_set_catcode_letter:N	21
\char_set_catcode_letter:n	675
\char_set_catcode_other:N	120, 122, 123, 674, 978
\char_set_lccode:nn	306
\l_char_special_seq	39, 674, 931
\char_value_catcode:n	20
\cleanseqsc	8, 47, 1331
clist commands:	
\clist_map_inline:nn	1269, 1303
\contextlmtxmode	104
\countsc	7, 47, 1327
cs commands:	
\cs:w	737, 738, 897, 898
\cs_end:	742, 743, 897, 898
\cs_generate_variant:Nn	228, 229, 277, 510, 542, 607, 787, 788
\cs_gset_eq:NN	356, 357
\cs_if_exist:NTF	25, 918, 919, 928, 1058
\cs_new:Nn	278, 279, 280, 686, 733
\cs_new:Npn	96, 230, 231, 232, 233, 561, 570, 574, 578, 588, 696, 762, 772, 778, 930
\cs_new_eq:NN	864
\cs_new_protected:Nn	302, 318, 328, 337, 654, 796, 801, 955, 986, 988, 990, 998, 1071, 1197
\cs_new_protected:Npe	824, 1274
\cs_new_protected:Npn	268, 282, 293, 383, 385, 448, 450, 485, 489, 491, 497, 503, 511, 534, 543, 551, 596, 614, 630, 632, 642, 644, 664, 745, 779, 789, 813, 831, 843, 853, 866, 871, 876, 884, 886, 893, 902, 905, 910, 938, 947, 953, 1040, 1066, 1086, 1098, 1124, 1143, 1176, 1208, 1226, 1253, 1267, 1281, 1291, 1300, 1312
\cs_replacement_spec:N	1242
\cs_set:Npe	285
\cs_set:Npn	690, 963
\cs_set_eq:NN	476, 481, 978, 981, 1021, 1180, 1181, 1182, 1194
\cs_set_protected:Npe	676, 713
\cs_set_protected:Npn	439, 857, 1314
\csname	56, 66
D	
debug-var	141
\DeclareInstance	1000
\def	3, 4, 5, 51, 55, 58, 59, 67, 80
\definetyping	994
dim commands:	
\dim_compare:nNnTF	343
\dim_zero:N	958
\c_zero_dim	288
\do	931, 978, 981, 987
\dospecials	927, 979
E	
\else	77, 79
else commands:	
\else:	241
\end	126, 736, 940
\endcsname	56, 66
\endgroup	55, 58, 61, 74, 88
\endinput	75, 89, 1425
\endlinechar	54
\endscontents	4, 924
\endverbatim	1030
\endverbatimsc	127, 934, 946
\end{verbatimsc}	934
enumext internal commands:	
\l_enumext_ref_key_arg_tl	1054
Environments:	
verbatim	39, 40
\errhelp	62
\errmessage	63
exp commands:	
\exp_after:wN	237, 238, 737, 738, 983, 1120, 1121, 1276
\exp_args:Ne	1263
\exp_args:Nee	505
\exp_args:Nf	565, 582, 766
\exp_args:Nnf	576
\exp_args:NNV	1116
\exp_args:Nooo	895
\exp_args:NV	384, 449, 490, 756

\exp_not:N	136, 289, 678, 681, 690, 693, 736, 737, 738, 754, 813, 828, 829, 848, 850, 857, 859, 938, 940, 941, 942, 1187, 1276, 1281
\exp_not:n	506, 507, 549, 708, 770, 775, 776, 1129, 1135, 1191, 1264, 1287, 1288
\expandafter	56, 66
\ExplSyntaxOff	34, 45, 1424
\ExplSyntaxOn	15

F

\fi	65, 91, 92
fi commands:	
\fi:	243, 736, 741
file commands:	
\file_if_exist:nTF	249
\file_input:n	22, 139
\file_input_stop:	35, 46, 1426
force-eol	141, 359, 391
\foreachsc	6, 29, 30, 43, 1093
\frenchspacing	950, 1017

G

\getstored	6, 42, 1082, 1133
group commands:	
\group_begin:	300, 305, 316, 626, 666, 907, 949, 1043, 1100, 1146, 1174, 1211, 1224, 1255
\group_end:	307, 313, 325, 751, 786, 913, 954, 1078, 1119, 1170, 1196, 1220, 1247, 1265
\group_insert_after:N	602, 603, 604, 605

I

if commands:	
\if_false:	736, 741
\if_meaning:w	237
\IfDocumentMetadataTF	1013, 1026
\ifx	56, 66, 78
\input	14
int commands:	
\int_abs:n	592
\int_compare:nNnTF	524, 793, 807, 1377, 1381
\int_compare_p:nNn	591, 592
\int_decr:N	803
\int_eval:n	516, 576
\int_incr:N	798, 799
\int_new:N	19, 186, 187, 208, 216
\int_set:Nn	20, 431, 1106, 1183
\int_set_eq:NN	334, 342
\int_step_function:nnnN	675, 1109
\int_step_inline:nnnn	525, 526
\int_to_roman:n	515, 521, 522
\int_zero:N	791
\c_zero_int	807
\interlinepenalty	970, 975
iow commands:	
\iow_char:N	101, 114, 1060, 1061, 1063, 1230
\iow_close:N	274, 748
\iow_log:n	18
\iow_new:N	197
\iow_now:Nn	273, 782
\iow_open:Nn	272, 670

K

Keys provide by SCONTENTS :	
debug-var	32
meaningsc	30
typetored	30

keys commands:

\keys_define:nn	142, 176, 360, 391, 414, 456, 472
\l_keys_key_str	384, 449, 490
\keys_set:nn	620, 889, 1045, 1101, 1149, 1214, 1325
\keys_set_known:nnN	1259

L

\legacyverbatimsetup	1008
----------------------	------

M

\meaningsc	6, 30, 31, 45, 1203
meaningsc	472
\mergesc	7, 30, 31, 45, 1248
mode commands:	
\mode_if_horizontal:TF	331, 340, 974
\mode_leave_vertical:	287, 967, 991
msg commands:	
\msg_error:nn	110, 487, 537
\msg_error:nnn	388, 453, 501, 920, 1179
\msg_error:nnnn	389, 454, 495, 648
\msg_error:nnnnn	1316
\msg_expandable_error:nnn	568, 586
\msg_expandable_error:nnnnn	593
\msg_fatal:nn	33
\msg_line_context:	42, 1337, 1349, 1351, 1353, 1361, 1363, 1369, 1406, 1416
\msg_new:nnn	27, 40, 106, 1335, 1348, 1350, 1352, 1354, 1356, 1358, 1360, 1362, 1364, 1375
\msg_new:nnnn	1341, 1366, 1385, 1394, 1403, 1413
\msg_warning:nn	44, 794
\msg_warning:nnn	253, 257, 262, 661, 1056
\msg_warning:nnnn	722

N

\NeedsTeXFormat	7
\NewDocumentCommand	610, 916, 1035, 1051, 1082, 1093, 1138, 1203, 1248, 1323, 1331
\NewDocumentEnvironment	904, 1011
\newenvsc	5, 38, 916, 925, 1344
\NewExpandableDocumentCommand	1327
\next	55, 58, 67, 80, 93
\nobreak	347, 991
\null	968

O

\obeyedline	1058, 1060, 1061
\obeylines	979
overwrite	141, 359, 391, 456

P

\PackageError	59, 69, 82
Packages:	
expl3	20
l3keys	31
latex-lab	41
ltxcmd	20, 33, 42
lua-widow-control	22
scontents	20, 23, 31, 39
verbatim	41
xparse	20
xsim	34
xtemplate	41
\par	963
\parfillskip	959
\parindent	958
\parskip	957, 960

peek commands:

\peek_charcode:NTF 1295
 \peek_remove_spaces:n 1293
 \penalty 970, 975
 prg commands:
 \prg_generate_conditional_variant:Nnn .. 227
 \prg_new_protected_conditional:Npnn . 234, 245, 805
 \prg_replicate:nn 281
 \prg_return_false: 242, 258, 266, 809
 \prg_return_true: 240, 254, 263, 808
 print-all 141
 print-cmd 141, 391, 456
 print-env 141, 359
 \ProcessKeyOptions 180
 \ProvidesExplPackage 8

Q

quark commands:

\quark_new:N 218, 219
 quark internal commands:
 \q__scontents_mark 26, 218, 239, 494, 828, 850, 1304
 \q__scontents_stop 218, 233, 681, 693, 829, 838, 841, 850, 859

R

\relax 56, 66
 \RenewDocumentCommand 1060

S

scan commands:

\scan_new:N 220, 221
 \scan_stop: 308, 330, 339, 1181

scan internal commands:

\s__scontents_mark 218, 494, 497, 1277, 1282, 1304, 1312
 \s__scontents_stop .. 218, 230, 231, 232, 239, 1269, 1277, 1282

\Scontents 5, 29, 30, 41, 1034

\scontents 4, 924

scontents 4, 924

scontents internal commands:

__scontents_analyse_nesting:n 36, 700, 789, 789
 __scontents_analyse_nesting:w 789
 __scontents_analyse_nesting_format:n 38, 792, 864, 866
 __scontents_analyse_nesting_generic:nn . 843, 867, 868
 __scontents_analyse_nesting_generic:w .. 838, 841, 848, 857
 __scontents_analyse_nesting_generic_--
 process:nn 831, 860
 __scontents_analyse_nesting_latex:n 824, 865
 __scontents_analyse_nesting_latex:w 813, 821, 826
 __scontents_append_contents:nn 31, 533, 534, 542, 549
 __scontents_bsphack: .. 25, 41, 327, 328, 356, 1042, 1145, 1210
 \l__scontents_char_value_int 11
 __scontents_check_line_process:nn 34, 625, 638, 640, 643, 644
 \l__scontents_cmd_name_tl . 24, 38, 188, 1037, 1095, 1140, 1205, 1250, 1405, 1409, 1415, 1419
 \l__scontents_current_seq_name_str 24, 193, 500, 1148, 1213

\l__scontents_debug_var_bool 171, 545, 1156, 1232
 __scontents_define_env:nnn . 38, 39, 884, 895, 902
 __scontents_define_generic_nesting_--
 function:n 845, 853
 __scontents_do_noligs:N . 27, 282, 282, 981, 1182
 \c__scontents_end_env_tl 22, 117, 679, 680, 691, 692, 708
 \g__scontents_end_verbatimsc_tl .. 22, 117, 939, 1192
 __scontents_env_end_function: 715, 733
 \l__scontents_env_name_tl 22, 38, 117, 649, 661, 723, 739, 819, 849, 858, 888, 1363, 1388, 1391, 1397, 1400
 \l__scontents_environment_keys_tl . 23, 182, 618, 619, 620
 __scontents_esphack: .. 25, 42, 327, 337, 357, 1079, 1171, 1221
 __scontents_file_if_writable:n .. 26, 245, 245
 __scontents_file_if_writable:nTF . 24, 245, 270, 667
 \l__scontents_file_name_tl . 24, 196, 365, 370, 396, 401, 462, 652, 1073, 1167, 1245
 __scontents_file_tl_write_start:n 34, 652, 664, 664, 788
 __scontents_file_write_cmd:nn . 26, 42, 268, 268, 277, 1073, 1167, 1245
 \l__scontents_file_write_iow .. 24, 196, 272, 273, 274, 670, 748, 782
 __scontents_file_write_stop:N 34, 656, 664, 745
 __scontents_finish_storing:NNN 32, 38, 42, 551, 551, 879, 1074
 \l__scontents_forced_eol_bool 32, 158, 555
 __scontents_foreach_add_body:n 1093, 1113, 1124
 \l__scontents_foreach_after_bool 202, 422, 1134
 \l__scontents_foreach_after_tl . 202, 423, 1135
 \l__scontents_foreach_before_bool 202, 417, 1128
 \l__scontents_foreach_before_tl 202, 418, 1129
 \g__scontents_foreach_exec_tl . 202, 1114, 1121, 1122
 \l__scontents_foreach_print_seq 202, 1103, 1117, 1126
 \l__scontents_foreach_sep_tl 443, 1117
 \l__scontents_foreach_start_int ... 426, 1110
 \l__scontents_foreach_step_int 434, 1111
 \l__scontents_foreach_stop_bool 209, 430, 1104
 \l__scontents_foreach_stop_int . 202, 431, 1106, 1112
 __scontents_foreach_wrapper:n 440, 1131
 \l__scontents_foreach_wrapper_bool .. 202, 438, 1130
 __scontents_foreachsc:nn 1093, 1096, 1098
 \l__scontents_foreachsc_arg_tl 24, 189, 1102, 1133
 __scontents_format_case:nnn 95, 96, 735, 740, 1186
 __scontents_generic_begin: ... 38, 870, 871, 891
 __scontents_generic_end: 38, 870, 876, 894
 __scontents_getfrom_seq:Nn .. 32, 561, 561, 1153, 1229
 __scontents_getfrom_seq:nn .. 43, 561, 578, 1090
 __scontents_getfrom_seq:nNn 561, 565, 570
 __scontents_getfrom_seq:nnn . 561, 576, 582, 588
 __scontents_getfrom_seq_aux:nnn . 561, 572, 574
 __scontents_getstored:nn ... 43, 1082, 1084, 1086
 __scontents_grab_opt_arg:n 609, 612, 614
 __scontents_grab_opt_arg:w . 33, 34, 609, 610, 637
 \c__scontents_hidden_space_str . 25, 32, 214, 556,

720, 1155, 1231, 1277, 1282
 _scontents_if_nested: 36, 805
 _scontents_if_nested:TF 704, 789
 \g_scontents_last_stored_tl 25, 213, 598, 603, 605
 \g_scontents_last_tl 33
 _scontents_lastfrom_seq:n . 25, 32, 33, 558, 596, 596, 607
 _scontents_make_control_chars_active: . 293, 318, 651, 683
 _scontents_meaningsc:n 1203, 1217, 1226
 _scontents_meaningsc:nn . 481, 1203, 1206, 1208
 \l_scontents_meaningsc_arg_tl . . 24, 189, 1228, 1230, 1231, 1234, 1237, 1238, 1242, 1245
 \l_scontents_mergesc_arg_tl . 24, 189, 193, 1270, 1271, 1272
 _scontents_mergesc_cmd:nn 30, 472, 476, 481, 485, 1263
 _scontents_mergesc_code:nn . . 1248, 1251, 1253
 \l_scontents_mergesc_keys_tl 24, 193, 1256, 1259, 1264
 _scontents_mergesc_parse_list:n . . 1248, 1262, 1267
 \g_scontents_name_sc!internal_seq . . . 25, 211
 \l_scontents_name_seq_cmd_tl . . . 42, 147, 1075
 \l_scontents_name_seq_env_tl 144, 881
 \l_scontents_nesting_aux_int . 23, 182, 791, 793, 799
 _scontents_nesting_decr: 36, 706, 789, 801
 \l_scontents_nesting_env_int . . 23, 36, 182, 798, 803, 807
 _scontents_nesting_incr: 796, 820, 837
 \c_scontents_newline_tl . 22, 100, 111, 114, 1063, 1155
 _scontents_nolig_list: 946, 982, 986
 _scontents_normalise_line_ends:N 33, 619, 630
 \l_scontents_overwrite_bool 161, 251
 _scontents_par: 278, 279, 322
 _scontents_parse_environment_keys:n . . . 28
 _scontents_parse_mergesc:nw . 1248, 1269, 1291
 _scontents_parse_mergesc_aux:nw . . 1248, 1296, 1297, 1300
 _scontents_parse_mergesc_range:nw 1248, 1304, 1312
 _scontents_parse_type_meaning_key:n 24, 470, 489, 489
 _scontents_parse_type_meaning_key:nn . . 489, 490, 491
 _scontents_parse_type_meaning_range:w . 489, 494, 497
 _scontents_parse_typemeaning_key:n 31
 \l_scontents_plain_bool . . 25, 223, 962, 965, 973
 _scontents_plain_disable_outer_par: 293, 302, 320
 \l_scontents_print_cmd_bool 42, 155, 1075, 1079
 \l_scontents_print_env_bool 152, 882
 \l_scontents_print_verb_style_bool 467, 1162, 1171, 1221, 1240
 \l_scontents_processed_body_lines_tl 23, 182, 656, 659, 880
 _scontents_range_parser:nnnn 489, 499, 503, 510, 1319
 _scontents_range_parser_aux:nnn 489, 505, 511
 _scontents_remove_leading_nl:n . . 35, 664, 757, 762
 _scontents_remove_leading_nl:nn . . . 767, 772
 _scontents_remove_leading_nl:w . 664, 775, 778
 _scontents_remove_trailing_eol:N . 1154, 1248, 1271, 1274
 _scontents_remove_trailing_eol:w . 1248, 1276, 1281
 _scontents_ret:w 34, 323, 676, 684, 713, 728
 \l_scontents_save_every_body_lines_tl 23, 182, 673, 757, 784
 \l_scontents_save_sf_int 216, 334, 342
 \l_scontents_save_skip 216, 333, 343
 \l_scontents_Scontents_arg_tl 24, 42, 189, 1053, 1056, 1061, 1063, 1068, 1073, 1075
 _scontents_Scontents_code:nn . 41, 1034, 1038, 1040
 _scontents_Scontents_finish: . 42, 1064, 1069, 1071
 _scontents_Scontents_norm_arg:n . 41, 42, 1034, 1049, 1066
 _scontents_Scontents_verb_arg:w . 41, 42, 1034, 1048, 1051
 \l_scontents_seq_item_seq 25, 211, 516, 527, 1147, 1150, 1151, 1153, 1212, 1215, 1216, 1229, 1302, 1305
 _scontents_set_active_eq:NN 293, 293, 321, 322, 323, 989
 _scontents_setting_env:nn . . . 38, 884, 884, 921
 _scontents_setup_verb_processor: 664, 686, 890
 _scontents_start_after_option:w . 34, 622, 625, 642
 _scontents_start_environment:w . . 38, 625, 632, 874
 _scontents_stop_environment: 38, 625, 654, 878
 _scontents_store_to_seq:NN . . 32, 543, 543, 557
 \l_scontents_storing_bool . 24, 36, 199, 200, 363, 368, 394, 399, 460, 553, 658, 752, 783
 _scontents_storing_bool 196
 _scontents_tab: 278, 278, 321
 \l_scontents_tab_width_int 164, 281
 _scontents_tabs_to_spaces: . 26, 280, 280, 1200, 1237
 _scontents_tl_if_head_is_q_mark:n . . 26, 234
 _scontents_tl_if_head_is_q_mark:nTF 234, 513, 816, 835
 _scontents_tmp:w 1314, 1321
 _scontents_tystored:N 43, 44, 1138, 1164, 1176
 _scontents_tystored:nn . 476, 1138, 1141, 1143
 \l_scontents_tystored_arg_tl . 24, 189, 1152, 1154, 1155, 1158, 1160, 1164, 1167
 _scontents_unknown_keys_cmd:n 30, 412, 446, 448, 448
 _scontents_unknown_keys_cmd:nn . 448, 449, 450
 _scontents_unknown_keys_env:n . 381, 383, 383
 _scontents_unknown_keys_env:nn . 383, 384, 385
 _scontents_use_delimit_by_s_stop:nw 230, 230, 1287
 _scontents_use_i_delimit_by_s_stop:nw . 230, 231, 239
 _scontents_use_none_delimit_by_q_stop:w 230, 233, 789, 817
 _scontents_use_none_delimit_by_s_stop:w 230, 232, 1310
 \l_scontents_verb_font_tl 150, 980, 1236
 _scontents_verb_print_EOL: 1180, 1194
 _scontents_verb_processor_iterate:nnn . 664,

694, 696	
_scontents_verb_processor_iterate:w	664, 678, 690
_scontents_verb_processor_output:n	36, 701, 707, 712, 779, 779, 787
_scontents_verbatimsc_aux:	946, 950, 955
_scontents_verbatimsc_instance:	996, 998, 1015
\c_scontents_version_str	20, 11
_scontents_vobeyspaces:	946, 950, 988
\l_scontents_writable_bool	24, 196, 669, 672, 747, 781
\l_scontents_writing_bool	24, 36, 196, 247, 364, 369, 395, 400, 461
_scontents_xobeysp:	946, 989, 990
_scontents_xverb:	946, 951, 1018, 1021, 1197
_scontents_xverb:w	934, 938, 1138, 1201
\ScontentsCoreFileDate	51, 78
\ScontentsFileDate	3, 9, 16, 78
\ScontentsFileDescription	5, 9, 17
\ScontentsFileVersion	4, 9, 12, 17
sep	414
seq commands:	
\seq_clear:N	1103, 1147, 1212, 1302
\seq_count:N	500, 566, 583, 1107, 1297, 1317, 1320, 1329
\seq_gclear:N	1261
\seq_gclear_new:N	1333
\seq_gpop_right:NN	1270
\seq_gput_right:Nn	540, 1272, 1307
\seq_if_empty:NTF	1150, 1215
\seq_if_exist:NTF	538, 563, 580
\seq_item:Nn	594, 600, 1308
\seq_map_function:NN	674, 931
\seq_map_inline:Nn	1305
\seq_map_tokens:Nn	572
\seq_new:N	202, 211, 212, 539
\seq_put_right:Nn	516, 527, 1126
\seq_set_from_clist:Nn	1151, 1216
\seq_use:Nn	1116
\setupsc	3, 46, 1323
skip commands:	
\skip_horizontal:n	348
\skip_if_eq:nnTF	345
\skip_new:N	217
\skip_set:Nn	959, 960
\skip_set_eq:NN	333
\skip_vertical:N	957
\c_zero_skip	343, 345, 348
\space	16, 17
start	414
\startscontents	4, 924
\startverbatimsc	993, 1189
step	414
stop	414
\stopscontents	4, 924
\stopverbatimsc	128, 934, 993
store-all	141
store-cmd	141, 391
store-env	141, 359
str commands:	
\c_backslash_str	132, 649, 814, 827, 849, 858, 1361, 1363, 1369, 1372, 1405, 1409, 1415, 1419
\c_circumflex_str	215
\c_left_brace_str	135, 814, 828
\c_percent_str	215, 718
\c_right_brace_str	137, 814, 828
\str_const:Nn	16, 214
\str_if_eq:nnTF	506, 507, 636, 720, 819
\str_if_eq_p:nn	718
\str_new:N	195
\str_set:Nn	1148, 1213
\str_use:N	18
syst commands:	
\syst_obeyed_line	111
T	
TeX and L ^A T _E X 2 _ε commands:	
\@	20, 21, 23
\@bsphack	27, 356
\@esphack	27, 357
\@setupverbinvisiblespace	1017
\@vobeyspaces	1017
tex commands:	
\tex_char:D	289
\tex_everypar:D	983, 984
\tex_ignorespaces:D	350
\tex_kern:D	288
\tex_lastskip:D	333, 345
\tex_let:D	308
\tex_lowercase:D	307
\tex_newlinechar:D	1183
\tex_par:D	961, 969, 975
\tex_spacefactor:D	334, 342
\tex_the:D	984
\tex_unpenalty:D	984
tl commands:	
\c_space_tl	278, 1405, 1409, 1415, 1419
\tl_clear:N	673, 1256
\tl_const:Nn	101, 111, 114, 130
\tl_gclear:N	604, 1120
\tl_gset:Nn	598, 1114
\tl_gset_rescan:Nnn	118
\tl_head:n	636, 768
\tl_if_blank:nTF	387, 452, 493, 536, 646, 698, 711, 1178
\tl_if_blank_p:n	717
\tl_if_empty:n	227
\tl_if_empty:NTF	1054, 1160, 1238
\tl_if_empty:nTF	226, 515, 1286
\tl_if_empty_p:N	659
\tl_if_empty_p:n	521, 522
\tl_if_exist:NTF	38
\tl_if_head_is_N_type:nTF	634, 764, 833
\tl_if_novalue:nTF	616, 1044, 1101, 1149, 1214, 1257
\tl_log:N	547, 1158, 1234
\tl_new:N	117, 129, 183, 184, 185, 188, 189, 190, 191, 192, 193, 194, 196, 203, 204, 206, 213
\tl_put_right:Nn	556, 784
\tl_replace_all:Nnn	226, 228, 631, 1061, 1063, 1155, 1230, 1231, 1237
\tl_retokenize:n	25, 226, 229, 602, 725, 1088, 1184
\tl_set:Nn	365, 370, 396, 401, 418, 423, 462, 618, 754, 888, 1037, 1053, 1068, 1095, 1102, 1140, 1152, 1205, 1228, 1250, 1284
\tl_to_str:n	896
\tl_use:N	723, 739, 849, 858, 980, 1133, 1236
token commands:	
\token_if_eq_meaning:NNTF	774
\token_to_str:N	285, 291
\tt	178
\ttfamily	177

\typestored	6, 30, 31, 43, <u>1138</u>	\verbatim	1022
typestored	<u>472</u>	\verbatimsc	<u>946</u> , 1188
U		verbatimsc	7, <u>996</u>
unknown	<u>359</u> , <u>391</u> , <u>414</u> , <u>456</u>	W	
\unprotect	13	width-tab	<u>141</u> , <u>456</u>
use commands:		wrapper	<u>414</u>
\use:n 576, 688, 749, 811, 846, 855, 936, 1076, 1132, 1168,		write-cmd	<u>391</u>
1218, 1279		write-env	<u>359</u>
\UseInstance	1016	write-out	<u>359</u> , <u>391</u> , <u>456</u>
\UseStructureName	1003	\writestatus	12
V			
verb-font	<u>141</u>		