

# SCONTENTS

STORES L<sup>A</sup>T<sub>E</sub>X CONTENTS

V2.4 2025-05-15\*

©2019–2025 by Pablo González L<sup>†</sup>

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

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

## Abstract

This package allows to store L<sup>A</sup>T<sub>E</sub>X code, including “*verbatim*”, in  $\langle sequences \rangle$  using the `l3seq` module of `expl3`. The  $\langle stored content \rangle$  can be used as many times as desired in the document, additionally you can write to  $\langle external files \rangle$  or show it in  $\langle verbatim style \rangle$ .

## Contents

|     |  |   |     |   |    |
|-----|--|---|-----|---|----|
| 1   | <b>Description of the package</b>      | 1 | 5.9 | The environment <code>verbatimsc</code>     | 7  |
| 2   | <b>Motivation and Acknowledgments</b>  | 1 | 6   | <b>Other commands provided</b>              | 7  |
| 3   | <b>License and Requirements</b>        | 2 | 6.1 | The command <code>\countsc</code>           | 7  |
| 4   | <b>The scontents package</b>           | 2 | 6.2 | The command <code>\cleanseqsc</code>        | 8  |
| 4.1 | Installation                           | 2 | 7   | <b>The scontents package in action</b>      | 8  |
| 4.2 | Loading and options                    | 2 | 8   | <b>Examples</b>                             | 8  |
| 4.3 | The TAB character                      | 2 | 8.1 | From answers 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   | <b>User interface</b>                  | 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   | <b>Change history</b>                       | 16 |
| 5.4 | The command <code>\getstored</code>    | 6 | 10  | <b>Index of Documentation</b>               | 17 |
| 5.5 | The command <code>\foreachsc</code>    | 6 | 11  | <b>References</b>                           | 17 |
| 5.6 | The command <code>\typestored</code>   | 6 | 12  | <b>Implementation</b>                       | 19 |
| 5.7 | The command <code>\meaningsc</code>    | 6 | 13  | <b>Index of Implementation</b>              | 48 |
| 5.8 | The command <code>\mergesc</code>      | 7 |     |   |    |

## 1 Description of the package

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

## 2 Motivation and Acknowledgments

In L<sup>A</sup>T<sub>E</sub>X there is no direct way to record content for later use, although you can do this using `\macro`, recording  $\langle verbatim content \rangle$  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 ConTeXt 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<sup>A</sup>T<sub>E</sub>X 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 ConTeXt 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<sup>A</sup>T<sub>E</sub>X release 2024-11-01 (TeX Live 2024) as a minimum requirement.

\*This file describes a documentation for v2.4, last revised 2025-05-15.

<sup>†</sup>E-mail: [pablgonz@educarchile.cl](mailto: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 2024-11-01). 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 MiK $\TeX$ , 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>      | » | TDS:tex/generic/scontents/       |
| <code>scontents-code.tex</code> | » | TDS:tex/generic/scontents/       |
| <code>scontents.sty</code>      | » | TDS:tex/latex/scontents/         |
| <code>t-scontents.mkiv</code>   | » | TDS:tex/context/third/scontents/ |
| <code>scontents.pdf</code>      | » | TDS:doc/latex/scontents/         |
| <code>README.md</code>          | » | TDS:doc/latex/scontents/         |
| <code>scontents.dtx</code>      | » | TDS:source/latex/scontents/      |
| <code>scontents.ins</code>      | » | TDS:source/latex/scontents/      |

### 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 Con $\TeX$ t users

```
\usemodule{scontents}
```

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

### 4.3 The TAB character

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

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

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

<sup>2</sup>Check 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  $\langle keys \rangle$  will return an error. In this section are described some of the options, a summary of all options is shown in §4.5.

---

`\setupsc`  $\langle setupsc \{ \langle key = val \rangle \} \rangle$

The command `\setupsc` sets the  $\langle keys \rangle$  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`.

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

Sets the  $\langle font family \rangle$  used to display the  $\langle stored content \rangle$  for `\tpestored`, `\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  $\langle meta-key \rangle$  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  $\langle files \rangle$  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*` and for the environment `scontents`.

`print-all = { \true | \false } default: false`

It is a  $\langle meta-key \rangle$  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  $\langle stored content \rangle$  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{\langle env \rangle}`%. This key is available for the `scontents` environment and the `\Scontents*` command.

`width-tab = { \integer } default: 1`

Sets the equivalence in  $\langle spaces \rangle$  for the character TAB used when displaying  $\langle stored content \rangle$  in *verbatim style*. The value must be a  $\langle positive integer \rangle$ . This key is available for `\tpestored`, `\mergesc` and `\meaningsc` commands.

## 4.5 Options Overview

Summary of available options:

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

## 5 User interface

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

## 5.1 The environment scontents

---

```
scontents \begin{scontents}[\langle key=val \rangle]
          \langle body env \rangle
\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 separation* ‘`␣`’ 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<sub>E</sub>X and ConT<sub>E</sub>Xt users

In plain T<sub>E</sub>X there is not environments as in L<sub>A</sub>T<sub>E</sub>X. Instead of using the environment `scontents`, one should use a “pseudo environment” delimited by `\scontents` and `\endscontents`.

---

```
\scontents \scontents[\langle key=val \rangle]
\endscontents \langle body env \rangle
\endscontents
```

---

ConT<sub>E</sub>Xt users should use `\startscntents` and `\stopscntents`.

---

```
\startscntents \startscntents[\langle key=val \rangle]
\stopscntents \langle body env \rangle
\stopscntents
```

---

### 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 = { $\langle seq\ name \rangle$ }` 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 = { $\langle true\ | \ false \rangle$ }` 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 = { $\langle file.ext \rangle$ }` 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 = { $\langle file.ext \rangle$ }` 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. X<sub>3</sub>TeX users using character TAB must add `-8bit` at the command line, otherwise you will get TeX 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 into *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. X<sub>3</sub>TeX users using character TAB must add `-8bit` at the command line, otherwise you will get TeX character TAB ‘^^I’ in `(file.ext)`.

## 5.4 The command `\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 <key = val>{<seq name>}
```

---

The command `\foreachsc` goes 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 `\tpestored`

---

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

---

The command `\tpestored` 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<sub>3</sub>LaTeX users using character `TAB` must add `-8bit` at the command line, otherwise you will get TeX character `‘^^I’` in *<file.ext>*.

## 5.7 The command `\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 `\tpestored`. 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 `{⟨seq A⟩[1]}`, `{⟨seq B⟩[2-5]}` and `{⟨seq C⟩[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 `{⟨argument⟩}` taken by this command is a *comma separated list* of the form `{⟨seq name⟩}` 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 it allows to join *⟨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}
  }
}
{}}
```

🔗 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` count a number of *⟨stored content⟩* in the sequence `{⟨seq name⟩}`.

## 6.2 The command `\cleanseqsc`

---

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

---

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

## 7 The `SCONTENTS` package in action

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

### Abstract

This package allows to store  $\text{\LaTeX}$  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 `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 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 `SCONTENTS`.

## 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`<sup>3</sup> tool to compile them.

<sup>3</sup>The cool  $\text{\TeX}$  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
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
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
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
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
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
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
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{\lfloor}}]}%
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
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 \linline[basicstyle=\ttfamily] | linline | 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 \linline[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: {shell: true, options: [-8bit]}
2 % arara: xelatex: {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 fextra 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
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 answer}
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 answer}
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[typstored, 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
2 % arara: clean: { extensions: [ aux, log ] }
3 \DocumentMetadata{tagging=on, lang=en-US, pdfversion=2.0, pdfstandard=ua-2, testphase=latest}
4 \documentclass{article}
5 \usepackage{scontents,unicode-math,hyperref}
6 \hypersetup{pdftitle = {Test scontents package},}
7 \begin{document}
8 Some
9
10 \begin{scontents}[print-env=true]
11   First code \verb|\foo|
12
13   And more code \verb|\bar|
14 \end{scontents}
15
16 Text
17
18 \begin{scontents}[print-env=true]
19   Second code \verb|\foo|
20
21   And more code \verb|\bar|
22 \end{scontents}
23
24 Text
25
26 \Scontents*{code \verb|\baz|}
27
28 % \typstored
29 \typstored[1]{contents}
30
31 % \mergesc
32 \mergesc[typstored]{ {contents}[1-end] }
33
34 % \getstored
35 \getstored[2]{contents}
36 \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].

- v2.4 (ctan), 2025-05-15**
  - Optimization of expansion code from ‘x’ to ‘e’.
  - Restructuring code for documentation and implementation.
  - Add new keys for `\typestored` and `\meaningsc`.
  - Check the version of `expl3` in plain  $\TeX$  and  $\ConTeXt$ .
- v2.3 (ctan), 2025-04-23**
  - Adapting the `verbatimsc` environment for *tagged* PDF.
  - Update minimum required to  $\LaTeX$  release 2024-11-01.
  - Safer code for replacement `\obeyedline`.
- v2.2 (ctan), 2025-03-26**
  - Fix internal definition for some functions.
  - Replace `\peek_charcode_ignore_spaces:NTF` by `\peek_charcode:NTF`.
  - Set correct code for `\obeyedline` implement in  $\LaTeX$  release 2024-06-01.
- v2.1 (ctan), 2024-06-14**
  - Fix `\cleanseqsc` command.
  - Add `\mergesc` command.
  - Fix internal definition for seq var.
  - Fix internal code for `\typestored`.
  - Replace `\cs_argument_spec:N` by `\cs_parameter_spec:N`.
  - Detect `l3keysze` package (obsolete in june 2022  $\LaTeX$  release).
  - Minor adjustments in the documentation.
- v2.0 (ctan), 2022-04-04**
  - Adapting the `verbatimsc` environment (compatibility `verbatim` 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**
  - Update and improvements in the internal code.
  - Updating the generic code for I/O verification.
  - Add `write-cmd` and `write-out` keys for `\Scontents*`.
  - Fix `sep` key in `\foreachsc`.
- v1.8 (ctan), 2019-11-18**
  - Add `\newenvsc` command.
  - Fix nested environment in plain  $\TeX$  and  $\ConTeXt$ .
  - Modified default value in `\getstored`.
  - Add `overwrite` key to reduce I/O operations.
  - Deleted an unnecessary group in the code.
- v1.7 (ctan), 2019-10-29**
  - The `verbatimsc` environment was rewritten.
  - Minor adjustments in documentation.
- v1.6 (ctan), 2019-10-26**
  - The internal behavior of `\getstored` has been modified.
  - The internal behavior of `\foreachsc` has been modified.
  - Corrected file extension for  $\ConTeXt$ .
  - Remove spurious warning.
- v1.5 (ctan), 2019-10-24**
  - Add support for plain  $\TeX$  and  $\ConTeXt$ .
  - Split internal code for optimization.
  - Add support for vertical spaces in `[(key = val)]`.
  - Add `\foreachsc` command.
  - Check if `verbatim` package is loaded.
- v1.4 (ctan), 2019-10-03**
  - Add `store-all` key.
  - Messages and keys were separated.
  - Restructuring of documentation.
  - Now the version of `expl3` is checked instead of `xparse`.
  - The internal behavior of `force-eol` has been modified.
- v1.3 (ctan), 2019-09-24**
  - The environment `scontents` can now nest.
  - Added `force-eol`, `verb-font` and `width-tab` keys.
  - The extra space has been removed when you run `\getstored`.
  - Internal code has been rewritten more efficiently.
  - Remove *starred argument* ‘\*’ for `\typestored`.
  - Remove `filecontentsdef` dependency.
  - Changing `\regex_replace_all:` for `\tl_replace_all:`.
- v1.2 (ctan), 2019-08-28**
  - 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.

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

## 11 References

- [1] The  $\LaTeX$  Project. “The `expl3` package”. Available from CTAN, <https://www.ctan.org/pkg/expl3>, 2025.
- [2] BURNOL, JEAN FRANÇOIS. “The `filecontentsdef` package”. Available from CTAN, <https://ctan.org/pkg/filecontentsdef>, 2019.
- [3] The  $\LaTeX$  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  $\LaTeX$ ”. Available from CTAN, <https://www.ctan.org/pkg/fancyvrb>, 2024.
- [6] HOFFMANN, JOBST. “The `listings` package”. Available from CTAN, <https://www.ctan.org/pkg/listings>, 2024.
- [7] POORE, GEOFFREY M. “The `fvextra` package - Highlighted source code in  $\LaTeX$ ”. Available from CTAN, <https://www.ctan.org/pkg/minted>, 2025.
- [8] POORE, GEOFFREY M. “The `minted` package - Highlighted source code in  $\LaTeX$ ”. Available from CTAN, <https://www.ctan.org/pkg/minted>, 2025.
- [9] The  $\LaTeX$  Project. “The  $\LaTeX_3$  Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2025.

- [10] The  $\text{\LaTeX}$  Project. “The  $\text{\LaTeX}$  2 $\epsilon$  sources”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2025.
- [11] The  $\text{\LaTeX}$  Project. “ $\text{\LaTeX}$  for authors current version”. Available from CTAN, <https://ctan.org/pkg/latex-base>, 2025.
- [12] FISCHER, ULRIKE. “tagpdf –  $\text{\LaTeX}$  kernel code for PDF tagging”. Available from CTAN, <https://www.ctan.org/pkg/tagpdf>, 2025.
- [13] The  $\text{\LaTeX}$  Project. “latex-lab –  $\text{\LaTeX}$  laboratory”. Available from CTAN, <https://www.ctan.org/pkg/latex-lab>, 2025.
- [14] F. STURM, THOMAS. “tcolorbox – Coloured boxes, for  $\text{\LaTeX}$  examples and theorems, etc”. Available from CTAN, <https://ctan.org/pkg/tcolorbox>, 2025.
- [15] The  $\text{\LaTeX}$  Project. “verbatim – Reimplementation of and extensions to  $\text{\LaTeX}$  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>\def\ScontentsFileDate{2025-05-15}%
3 <core>\def\ScontentsCoreFileDate{2025-05-15}%
4 <*loader>
5 \def\ScontentsFileVersion{2.4}%
6 \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 2024-11-01 is the latest available in  $\TeX$  Live 2024 (frozen) and is necessary to be able to implement the package's full compatibility with *tagged* PDF.

```
7 <*latex>
8 \NeedsTeXFormat{LaTeX2e}[2024-11-01]
9 \ProvidesExplPackage
10 {scontents} {\ScontentsFileDate} {\ScontentsFileVersion} {\ScontentsFileDescription}
11 </latex>
```

The plain  $\TeX$  and Con $\TeX$ t loaders are similar (probably because I don't know how to make a proper Con $\TeX$ t module :-). We define a  $\LaTeX$ -style `\ver@scontents.sty` macro with version info (just in case) and add `\ExplSyntaxOn` to be able to load the frozen `xparse[3]` later (§12.1.1).

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

For plain  $\TeX$  and Con $\TeX$ t we must check the minimum requirement, in this case `\int_step_tokens:nn` which was added in release 2025-01-14 of `expl3` included in  $\TeX$  Live 2024 (frozen).

```
20 \cs_if_exist:NF \int_step_tokens:nn
21 {
22   \msg_new:nnn { scontents } { expl-too-old }
23   {
24     Please-install-an-up-to-date-TeX-distribution-or-update-using~
25     your-TeX-package-manager-or-from-CTAN. \
26     See-documentation.~Loading-scontents-will-abort!
27   }
28   \msg_fatal:nn { scontents } { expl-too-old }
29   \ExplSyntaxOff
30   \file_input_stop:
31 }
32 </!latex>
```

In plain  $\TeX$ , check that the package isn't being loaded twice ( $\LaTeX$  and Con $\TeX$ t already defend against that):

```
33 <*plain>
34 \cs_if_exist:NT \__scontents_rescan_tokens:n
35 {
36   \msg_new:nnn { scontents } { already-loaded }
37   {
38     The-'scontents'-package-is-already-loaded.~Aborting~input~\msg_line_context:.
39   }
40   \msg_warning:nn { scontents } { already-loaded }
41   \ExplSyntaxOff
```

```

42   \file_input_stop:
43   }
44 \</plain>

```

In ConT<sub>E</sub>Xt we must take a precaution when running under LMTX since `\tex_scantokens:D` is a copy of the primitive  $\varepsilon$ -T<sub>E</sub>X `\scantokens` and `\tl_gset_rescan:Nnn` is a wrapper around it and are not available under LMTX.

- This is an adaptation of the file `t-lua-widow-control.mkxL` part of Max Chernoff's `lua-widow-control` package, `\contextlmtxmode` is described at <https://source.contextgarden.net/tex/context/base/mkxL/context.mkxL>.

```

45 \<context>
46 \bool_if:NT \contextlmtxmode
47 {
48   \msg_new:nnn { scantokens } { luametateX }
49   {
50     The~'scantokens'~package~doesn't~work~under~LMTX.
51   }
52   \msg_error:nn { scantokens } { luametateX }
53 }
54 \</context>
55 \</loader>

```

### 12.1.1 Load `xparse-generic` in plain T<sub>E</sub>X and ConT<sub>E</sub>Xt

`\l__scantokens_char_value_int` When loading the package outside of L<sub>A</sub>T<sub>E</sub>X we can't usually use `xparse[3]` now `ltxcmd[10]` part of the L<sub>A</sub>T<sub>E</sub>X kernel. However since the old `xparse[3]` provide `xparse-generic.tex` is loadable in any format.

```

56 \<loader&!latex>
57 \int_new:N \l__scantokens_char_value_int
58 \int_set:Nn \l__scantokens_char_value_int { \char_value_catcode:n { \@ } }
59 \char_set_catcode_letter:N \@
60 \file_input:n { xparse-generic.tex }
61 \char_set_catcode:nn { \@ } { \l__scantokens_char_value_int }
62 \</loader&!latex>

```

- The file `TDS:/tex/latex/l3packages/xparse/xparse-generic.tex` is always available and frozen since 2021.

(End of definition for `\l__scantokens_char_value_int`.)

### 12.1.2 Definition of variables by format

We define and set variables that must be handled separately in order to work properly with plain T<sub>E</sub>X, ConT<sub>E</sub>Xt and L<sub>A</sub>T<sub>E</sub>X.

```

\g__scantokens_end_verbatimsc_tl
\c__scantokens_end_env_tl
\l__scantokens_env_name_tl

```

The global token list `\g__scantokens_end_verbatimsc_tl` match when ending `verbatimsc` (§12.11).

```

63 \<loader>
64 \tl_new:N \g__scantokens_end_verbatimsc_tl
65 \tl_gset_rescan:Nnn \g__scantokens_end_verbatimsc_tl
66 {
67   \char_set_catcode_other:N \
68 \<*latex>
69   \char_set_catcode_other:N \{
70   \char_set_catcode_other:N \}
71 \</latex>
72 }
73 \<latex> { \end{verbatimsc} }
74 \<plain> { \endverbatimsc }
75 \<context> { \stopverbatimsc }

```

The constant token list `\c__scantokens_end_env_tl` match when ending environments defined by `\newenvsc`, the token list `\l__scantokens_env_name_tl` storing the *name* of environments defined by `\newenvsc` (§12.9).

```

76 \tl_new:N \l__scantokens_env_name_tl
77 \tl_const:Ne \c__scantokens_end_env_tl
78 {
79   \c_backslash_str
80 \<latex|plain> end
81 \<context> stop
82 \<latex> \c_left_brace_str
83   \exp_not:N \l__scantokens_env_name_tl
84 \<latex> \c_right_brace_str

```



```

85 }
86 </loader>

```

(End of definition for `\g__scntents_end_verbatimsc_tl`, `\c__scntents_end_env_tl`, and `\l__scntents_env_name_tl`.)

### 12.1.3 Loading the package core

Now we load the core `SCONTENTS` code:

```

87 (*loader)
88 \file_input:n { scntents-code.tex }

```

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

```

89 \cs_new:Npn \__scntents_format_case:nnn #1 #2 #3
90 <latex> {#1} % LaTeX
91 <plain> {#2} % Plain/Generic
92 <context> {#3} % ConTeXt
93 </loader>

```

(End of definition for `\__scntents_format_case:nnn`.)

### 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`.

```

94 (*core)
95 \begingroup
96 \catcode32=10
97 \endlinechar=32
98 \def\next{\endgroup}%
99 \expandafter\ifx\csname PackageError\endcsname\relax
100 \begingroup
101 \def\next{\endgroup\endgroup}%
102 \def\PackageError#1#2#3%
103 {%
104 \endgroup
105 \errhelp{#3}%
106 \errmessage{#1 Error: #2!}%
107 }%
108 \fi
109 \expandafter\ifx\csname ScontentsFileDate\endcsname\relax
110 \def\next
111 {%
112 \PackageError{scontents}{No scntents loader detected}
113 {%
114 You have attempted to use the scntents code directly rather than using
115 the correct loader. Loading of scntents will abort.
116 }%
117 \endgroup
118 \endinput
119 }%
120 \else
121 \ifx\ScontentsFileDate\ScontentsCoreFileDate
122 \else
123 \def\next
124 {%
125 \PackageError{scontents}{Mismatched scntents files detected}
126 {%
127 You have attempted to load scntents with mismatched files:
128 probably you have one or more files 'locally installed' which
129 are in conflict. Loading of scntents will abort.
130 }%
131 \endgroup
132 \endinput
133 }%
134 \fi
135 \fi
136 \next

```

## 12.2 Keys for the package

store-env 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.

```

store-cmd
verb-font
print-env
print-cmd
force-eol
overwrite
width-tab
print-all
store-all
137 \keys_define:nn { scontents }
138   {
139     store-env .tl_set:N          = \l__scontents_name_seq_env_tl,
140     store-env .initial:n        = contents,
141     store-env .value_required:n = true,
142     store-cmd .tl_set:N          = \l__scontents_name_seq_cmd_tl,
143     store-cmd .initial:n        = contents,
144     store-cmd .value_required:n = true,
145     verb-font .tl_set:N          = \l__scontents_verb_font_tl,
146     verb-font .value_required:n = true,
147     print-env .bool_set:N        = \l__scontents_print_env_bool,
148     print-env .initial:n         = false,
149     print-env .default:n         = true,
150     print-cmd .bool_set:N        = \l__scontents_print_cmd_bool,
151     print-cmd .initial:n         = false,
152     print-cmd .default:n         = true,
153     force-eol .bool_set:N        = \l__scontents_forced_eol_bool,
154     force-eol .initial:n         = false,
155     force-eol .default:n         = true,
156     overwrite .bool_set:N        = \l__scontents_overwrite_bool,
157     overwrite .initial:n         = false,
158     overwrite .default:n         = true,
159     width-tab .int_set:N         = \l__scontents_tab_width_int,
160     width-tab .initial:n         = 1,
161     width-tab .value_required:n = true,
162     print-all .meta:n            = { print-env = #1, print-cmd = #1 },
163     print-all .default:n         = true,
164     store-all .meta:n           = { store-env = #1, store-cmd = #1 },
165     store-all .value_required:n = true
166   }
167 </core>

```

Set default value for `verb-font` key.

```

168 <loader>\keys_define:nn { scontents }
169 <latex> { verb-font .initial:n = \ttfamily }
170 <plain|context> { verb-font .initial:n = \tt }

```

In  $\text{\TeX}$  mode process the (*keys*) as options passed on to the package and will return an error when they are.

```

171 <*latex>
172 \ProcessKeyOptions [ scontents ]
173 </latex>

```

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

## 12.3 Internal variables

`\l__scontents_save_every_body_lines_tl` The token list `\l__scontents_save_every_body_lines_tl` holds the  $\{<body env>\}$  of an environment, `\l__scontents_processed_body_lines_tl` `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`.

`\l__scontents_environment_keys_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` (§12.9) and the environments base functions (§12.8).

```

174 <*core>
175 \tl_new:N \l__scontents_save_every_body_lines_tl
176 \tl_new:N \l__scontents_processed_body_lines_tl
177 \tl_new:N \l__scontents_environment_keys_tl
178 \int_new:N \l__scontents_nesting_env_int
179 \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`.

```
180 \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 `{⟨argument⟩}` passed to the `\Scontents` (§12.12), `\foreachsc` (§12.14), `\typestored` (§12.15) and `\meaningsc` (§12.16) commands.

```
181 \tl_new:N \l__scontents_Scontents_arg_tl
182 \tl_new:N \l__scontents_foreachsc_arg_tl
183 \tl_new:N \l__scontents_typedstored_arg_tl
184 \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 `{⟨argument⟩}` and the token list `\l__scontents_mergesc_keys_tl` save the `⟨keys⟩` passed to the `\mergesc` (§12.17) 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 `{⟨argument⟩}` to the `\typestored` and `\meaningsc` commands and is used by the function `\l__scontents_parse_type_meaning_key:n`.

```
185 \tl_new:N \l__scontents_mergesc_arg_tl
186 \tl_new:N \l__scontents_mergesc_keys_tl
187 \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 `⟨output file⟩`, when there's one. Its value is set by the keys `write-env`, `write-out` and `write-cmd` (§12.5).

`\l__scontents_file_write_iow` The variable `\l__scontents_file_write_iow` is an *output stream* for write the `{⟨body env⟩}` of an environment or `{⟨argument⟩}` for command to a `⟨output file⟩` 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 `⟨output file⟩`.

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 `\l__scontents_file_if_writable:nTF` (see 12.8.2).

```
188 \tl_new:N \l__scontents_file_name_tl
189 \iow_new:N \l__scontents_file_write_iow
190 \bool_new:N \l__scontents_writing_bool
191 \bool_new:N \l__scontents_storing_bool
192 \bool_set_true:N \l__scontents_storing_bool
193 \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 `⟨keys⟩` (§12.6.2) and implementation of `\foreachsc` command (§12.14).

```
194 \seq_new:N \l__scontents_foreach_print_seq
195 \tl_new:N \g__scontents_foreach_exec_tl
196 \tl_new:N \l__scontents_foreach_before_tl
197 \bool_new:N \l__scontents_foreach_before_bool
198 \tl_new:N \l__scontents_foreach_after_tl
199 \bool_new:N \l__scontents_foreach_after_bool
200 \int_new:N \l__scontents_foreach_stop_int
201 \bool_new:N \l__scontents_foreach_stop_bool
202 \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 `\typestored`, `\mergesc` or `\meaningsc` and the sequence `\g__scontents_name_sc!internal_seq` assemble this.

```
203 \seq_new:N \l__scontents_seq_item_seq
204 \seq_new:c { \g__scontents_name_sc!internal_seq }
```

©2019–2025 by Pablo González L

(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` (§12.7) to execute the last *(stored content)* outside the group.

```
205 \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 TeX 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.

```
206 \str_const:Nc \c__scontents_hidden_space_str
207 { \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:` (§12.4.2).

```
\l__scontents_save_skip
208 \int_new:N \l__scontents_save_sf_int
209 \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
\l__scontents_stop
\l__scontents_mark
\l__scontents_stop
\l__scontents_mark
210 \quark_new:N \q__scontents_stop
211 \quark_new:N \q__scontents_mark
212 \scan_new:N \l__scontents_stop
213 \scan_new:N \l__scontents_mark
214 ⟨/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 TeX implementation of the `verbatimsc` environment (§12.11).

```
215 ⟨*plain⟩
216 \bool_new:N \l__scontents_plain_bool
217 ⟨/plain⟩
```

(End of definition for `\l__scontents_plain_bool`.)

## 12.4 Utility functions

`\__scontents_rescan_tokens:n` The function `\tl_rescan:nn` provided by `expl3` doesn’t fit the needs of this package because it does not allow *catcode changes* inside the argument, so `verbatim` used inside one of `SCONTENTS`’s command-s/environments will not work. Here we create a private copy of `\tex_scantokens:D` which will serve our purposes. See the answer by Ulrich Diez in [How do use <setup> in \tl\\_set\\_rescan:Nnn to replace \scantokens?](#)

```
218 ⟨*core⟩
219 \cs_new_protected:Npn \__scontents_rescan_tokens:n #1 { \tex_scantokens:D {#1} }
220 \cs_generate_variant:Nn \__scontents_rescan_tokens:n { V, e }
```

(End of definition for `\__scontents_rescan_tokens:n`.)

`\tl_if_empty:FTF` Some nonstandard kernel variant.

```
221 \prg_generate_conditional_variant:Nnn \tl_if_empty:n { f } { p, TF }
```

(End of definition for `\tl_if_empty:FTF`.)

`\__scontents_use_delimit_by_s_stop:nw` Some functions used in the implementation of `\mergesc` (§12.17) and `scontents` (§12.10).

```
\__scontents_use_i_delimit_by_s_stop:nw
\__scontents_use_none_delimit_by_s_stop:w
\__scontents_use_none_delimit_by_q_stop:w
222 \cs_new:Npn \__scontents_use_delimit_by_s_stop:nw #1 \s__scontents_stop {#1}
223 \cs_new:Npn \__scontents_use_i_delimit_by_s_stop:nw #1 #2 \s__scontents_stop {#1}
224 \cs_new:Npn \__scontents_use_none_delimit_by_s_stop:w #1 \s__scontents_stop { }
225 \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`.

```

226 \prg_new_protected_conditional:Npnn \__scontents_tl_if_head_is_q_mark:n #1
227 { T, F, TF }
228 {
229   \exp_after:wN \if_meaning:w
230     \exp_after:wN
231     \q__scontents_mark \__scontents_use_i_delimit_by_s_stop:nw #1 ? \__scontents_stop
232     \prg_return_true:
233   \else:
234     \prg_return_false:
235   \fi:
236 }

```

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

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

```

237 \prg_new_protected_conditional:Npnn \__scontents_file_if_writable:n #1 { T, F, TF }
238 {
239   \bool_if:NTF \l__scontents_writing_bool
240   {
241     \file_if_exist:nTF {#1}
242     {
243       \bool_if:NTF \l__scontents_overwrite_bool
244       {
245         \msg_warning:nne { scontents } { overwrite-file } {#1}
246         \prg_return_true:
247       }
248       {
249         \msg_warning:nne { scontents } { not-writing } {#1}
250         \prg_return_false:
251       }
252     }
253     {
254       \msg_warning:nne { scontents } { writing-file } {#1}
255       \prg_return_true:
256     }
257   }
258   { \prg_return_false: }
259 }

```

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

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

```

260 \cs_new_protected:Npn \__scontents_file_write_cmd:nn #1#2
261 {
262   \__scontents_file_if_writable:nT {#1}
263   {
264     \iow_open:Nn \l__scontents_file_write_iow {#1}
265     \iow_now:Nn \l__scontents_file_write_iow {#2}
266     \iow_close:N \l__scontents_file_write_iow
267   }
268 }
269 \cs_generate_variant:Nn \__scontents_file_write_cmd:nn { VV }

```

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

#### 12.4.1 Functions for TAB and verbatimsc

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

```

270 \cs_new:Nc \__scontents_tab: { \c_space_tl }
271 \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.

```

272 \cs_new:Nn \__scontents_tabs_to_spaces:
273   { \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  $\text{\TeX} 2_{\epsilon}$ 's `\do@noligs` which makes sure to not consume following space tokens. The  $\text{\TeX} 2_{\epsilon}$  version ends with `\char`#1`, which leaves  $\text{\TeX}$  still looking for an *optional space*.

```

274 \cs_new_protected:Npn \__scontents_do_noligs:N #1
275   {
276     \char_set_catcode_active:N #1
277     \cs_set:cpe { __scontents_active_char_ \token_to_str:N #1 : }
278     {
279       \mode_leave_vertical:
280       \tex_kern:D \c_zero_dim
281       \tex_char:D \exp_not:N #1
282     }
283     \char_set_active_eq:Nc #1 { __scontents_active_char_ \token_to_str:N #1 : }
284   }

```

(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- $\text{\TeX}$  mode because in plain  $\text{\TeX}$  it is an `\outer` token.

```

285 \cs_new_protected:Npn \__scontents_set_active_eq:NN #1
286   {
287     \char_set_catcode_active:N #1
288     \char_set_active_eq:NN #1
289   }
290 </core>
291 <*loader>
292 \group_begin:
293 <plain> \char_set_catcode_active:n { \* }
294 \cs_new_protected:Nn \__scontents_plain_disable_outer_par:
295 <*plain>
296   {
297     \group_begin:
298     \char_set_lccode:nn { \* } { ^^L }
299     \tex_lowercase:D { \group_end:
300     \tex_let:D * \scan_stop:
301     }
302   }
303 </plain>
304 <latex|context> { }
305 \group_end:
306 </loader>
307 <*core>
308 \group_begin:
309 \char_set_catcode_active:N \*
310 \cs_new_protected:Nn \__scontents_make_control_chars_active:
311   {
312     \__scontents_plain_disable_outer_par:
313     \__scontents_set_active_eq:NN ^^I \__scontents_tab:
314     \__scontents_set_active_eq:NN ^^L \__scontents_par:
315     \__scontents_set_active_eq:NN ^^M \__scontents_ret:w
316   }
317 \group_end:
318 </core>

```

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

#### 12.4.2 Functions `\@bsphack` and `\@esphack`

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

```

319 <*core>
320 \cs_new_protected:Nn \__scontents_bsphack:
321   {

```



```

322 \scan_stop:
323 \mode_if_horizontal:T
324 {
325     \skip_set_eq:NN \l__scontents_save_skip \tex_lastskip:D
326     \int_set_eq:NN \l__scontents_save_sf_int \tex_spacefactor:D
327 }
328 }
329 \cs_new_protected:Nn \l__scontents_esphack:
330 {
331     \scan_stop:
332     \mode_if_horizontal:T
333     {
334         \int_set_eq:NN \tex_spacefactor:D \l__scontents_save_sf_int
335         \dim_compare:nNnT { \l__scontents_save_skip } > { \c_zero_skip }
336         {
337             \skip_if_eq:nnT { \tex_lastskip:D } { \c_zero_skip }
338             {
339                 \nobreak
340                 \skip_horizontal:n { \c_zero_skip }
341             }
342             \tex_ignorespaces:D
343         }
344     }
345 }
346 </core>
347 <*latex>
348 \cs_gset_eq:NN \l__scontents_bsphack: \@bsphack
349 \cs_gset_eq:NN \l__scontents_esphack: \@esphack
350 </latex>

```

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

## 12.5 Keys for environment

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

```

write-out
write-env
print-env
store-env
force-eol
overwrite
unknown
351 <core>
352 \keys_define:nn { scontents / scontents }
353 {
354     write-env .code:n          = {
355         \bool_set_true:N \l__scontents_storing_bool
356         \bool_set_true:N \l__scontents_writing_bool
357         \tl_set:Nn \l__scontents_file_name_tl {#1}
358     },
359     write-out .code:n         = {
360         \bool_set_false:N \l__scontents_storing_bool
361         \bool_set_true:N \l__scontents_writing_bool
362         \tl_set:Nn \l__scontents_file_name_tl {#1}
363     },
364     write-env .value_required:n = true,
365     write-out .value_required:n = true,
366     print-env .meta:nn        = { scontents } { print-env = #1 },
367     print-env .default:n      = true,
368     store-env .meta:nn        = { scontents } { store-env = #1 },
369     force-eol .meta:nn        = { scontents } { force-eol = #1 },
370     force-eol .default:n      = true,
371     overwrite .meta:nn        = { scontents } { overwrite = #1 },
372     overwrite .default:n      = true,
373     unknown .code:n           = { \l__scontents_unknown_keys_env:n {#1} },
374 }

```

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

### 12.5.1 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` 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.

```

375 \cs_new_protected:Npn \l__scontents_unknown_keys_env:n #1

```

```

376 { \exp_args:NV \__scontents_unknown_keys_env:nn \l_keys_key_str {#1} }
377 \cs_new_protected:Npn \__scontents_unknown_keys_env:nn #1#2
378 {
379   \tl_if_blank:nTF {#2}
380     { \msg_error:nnn { scontents } { env-key-unknown } {#1} }
381     { \msg_error:nnnn { scontents } { env-key-value-unknown } {#1} {#2} }
382 }

```

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

## 12.6 Keys for commands

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

### 12.6.1 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
383 \keys_define:nn { scontents / Scontents }
384 {
385   write-cmd .code:n          = {
386     \bool_set_true:N \l__scontents_storing_bool
387     \bool_set_true:N \l__scontents_writing_bool
388     \tl_set:Nn \l__scontents_file_name_tl {#1}
389   },
390   write-out .code:n         = {
391     \bool_set_false:N \l__scontents_storing_bool
392     \bool_set_true:N \l__scontents_writing_bool
393     \tl_set:Nn \l__scontents_file_name_tl {#1}
394   },
395   write-cmd .value_required:n = true,
396   write-out .value_required:n = true,
397   print-cmd .meta:nn         = { scontents } { print-cmd = #1 },
398   print-cmd .default:n       = true,
399   store-cmd .meta:nn         = { scontents } { store-cmd = #1 },
400   force-eol .meta:nn         = { scontents } { force-eol = #1 },
401   force-eol .default:n       = true,
402   overwrite .meta:nn         = { scontents } { overwrite = #1 },
403   overwrite .default:n       = true,
404   unknown .code:n           = { \__scontents_unknown_keys_cmd:n {#1} },
405 }

```

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

### 12.6.2 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
406 \keys_define:nn { scontents / foreachsc }
407 {
408   before .code:n          = {
409     \bool_set_true:N \l__scontents_foreach_before_bool
410     \tl_set:Nn \l__scontents_foreach_before_tl {#1}
411   },
412   before .value_required:n = true,
413   after .code:n           = {
414     \bool_set_true:N \l__scontents_foreach_after_bool
415     \tl_set:Nn \l__scontents_foreach_after_tl {#1}
416   },
417   after .value_required:n = true,
418   start .int_set:N        = \l__scontents_foreach_start_int,
419   start .value_required:n = true,
420   start .initial:n        = 1,
421   stop .code:n            = {
422     \bool_set_true:N \l__scontents_foreach_stop_bool
423     \int_set:Nn \l__scontents_foreach_stop_int {#1}
424   },
425   stop .value_required:n = true,
426   step .int_set:N         = \l__scontents_foreach_step_int,
427   step .value_required:n = true,
428   step .initial:n         = 1,
429   wrapper .code:n         = {
430     \bool_set_true:N \l__scontents_foreach_wrapper_bool

```

```

431         \cs_set_protected:Npn
432         \__scontents_foreach_wrapper:n # #1 {#1}
433         },
434     wrapper .value_required:n = true,
435     sep .tl_set:N = \__scontents_foreach_sep_tl,
436     sep .initial:n = { },
437     sep .value_required:n = true,
438     unknown .code:n = { \__scontents_unknown_keys_cmd:n {#1} },
439 }

```

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

### 12.6.3 Handling unknown keys for \Scontents and \foreachsc

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.

```

\__scontents_unknown_keys_cmd:n
\__scontents_unknown_keys_cmd:nn

```

```

440 \cs_new_protected:Npn \__scontents_unknown_keys_cmd:n #1
441 { \exp_args:NV \__scontents_unknown_keys_cmd:nn \l_keys_key_str {#1} }
442 \cs_new_protected:Npn \__scontents_unknown_keys_cmd:nn #1#2
443 {
444     \tl_if_blank:nTF {#2}
445     { \msg_error:nnn { scontents } { cmd-key-unknown } {#1} }
446     { \msg_error:nnnn { scontents } { cmd-key-value-unknown } {#1} {#2} }
447 }

```

(End of definition for \\_\_scontents\_unknown\_keys\_cmd:n and \\_\_scontents\_unknown\_keys\_cmd:nn.)

### 12.6.4 Keys for commands \typestored and \meaningsc

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 (§12.17).

```

width-tab
write-out
overwrite
print-cmd
unknown

```

```

448 \keys_define:nn { scontents / typemeaning }
449 {
450     width-tab .meta:nn = { scontents } { width-tab = #1 },
451     write-out .code:n = {
452         \bool_set_false:N \__scontents_storing_bool
453         \bool_set_true:N \__scontents_writing_bool
454         \tl_set:Nn \__scontents_file_name_tl {#1}
455     },
456     write-out .value_required:n = true,
457     overwrite .meta:nn = { scontents } { overwrite = #1 },
458     overwrite .default:n = true,
459     print-cmd .bool_set:N = \__scontents_print_verb_style_bool,
460     print-cmd .initial:n = true,
461     print-cmd .default:n = true,
462     unknown .code:n = { \__scontents_parse_type_meaning_key:n {#1} },
463 }

```

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

### 12.6.5 Keys for command \mergesc

We define two *<keys>* *typestored* and *meaningsc* as *mandatory*, returning an “error” through the function \\_\_scontents\_mergesc\_cmd:nn.

```

typestored
meaningsc
\__scontents_mergesc_cmd:nn

```

```

464 \keys_define:nn { scontents / mergesc }
465 {
466     typestored .code:n =
467     {
468         \cs_set_eq:NN \__scontents_mergesc_cmd:nn \__scontents_typestored:nn
469     },
470     typestored .value_forbidden:n = true,
471     meanings .code:n =
472     {
473         \cs_set_eq:NN \__scontents_mergesc_cmd:nn \__scontents_meaningsc:nn
474     },
475     meanings .value_forbidden:n = true,
476 }
477 \cs_new_protected:Npn \__scontents_mergesc_cmd:nn #1 #2
478 {
479     \msg_error:nn { scontents } { mergesc-missing-key }

```

```
480 }
```

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

### 12.6.6 Parsing keys for `\typstored`, `\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 `\typstored`, `\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 `\typstored[...][<keys>]{...}` syntax, we'll make the commands have a single *optional argument* which is processed by `!3keys[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.

```
481 \cs_new_protected:Npn __scontents_parse_type_meaning_key:n #1
482   { \exp_args:NV __scontents_parse_type_meaning_key:nn \l_keys_key_str {#1} }
483 \cs_new_protected:Npn __scontents_parse_type_meaning_key:nn #1#2
484   {
485     \tl_if_blank:nTF {#2}
486       { __scontents_parse_type_meaning_range:w #1 - \q__scontents_mark - \s__scontents_mark }
487       { \msg_error:nnee { scontents } { cmd-key-value-unknown } {#1} {#2} }
488   }
489 \cs_new_protected:Npn __scontents_parse_type_meaning_range:w #1 - #2 - #3 \s__scontents_mark
490   {
491     __scontents_range_parser:nnen {#1} {#2}
492     { \seq_count:c { g__scontents_name_ \l__scontents_current_seq_name_str _seq } }
493     { \msg_error:nnn { scontents } { cmd-key-unknown } {#1} }
494   }
495 \cs_new_protected:Npn __scontents_range_parser:nnnn #1 #2 #3 #4
496   {
497     \exp_args:Nee __scontents_range_parser_aux:nnn
498     { \str_if_eq:nnTF {#1} { end } {#3} { \exp_not:n {#1} } }
499     { \str_if_eq:nnTF {#2} { end } {#3} { \exp_not:n {#2} } }
500     { #4 }
501   }
502 \cs_generate_variant:Nn __scontents_range_parser:nnnn { nnen }
503 \cs_new_protected:Npn __scontents_range_parser_aux:nnn #1 #2 #3
504   {
505     __scontents_tl_if_head_is_q_mark:nTF {#2}
506     {
507       \tl_if_empty:fTF { \int_to_roman:n { -0 #1 } }
508       { \seq_put_right:Ne \l__scontents_seq_item_seq { \int_eval:n {#1} } }
509       { #3 {#1} }
510     }
511     {
512       \bool_lazy_and:nnTF
513       { \tl_if_empty_p:f { \int_to_roman:n { -0 #1 } } }
514       { \tl_if_empty_p:f { \int_to_roman:n { -0 #2 } } }
515       {
516         \int_compare:nNnTF {#2} > {#1}
517         { \int_step_inline:nnnn {#1} { 1 } {#2} }
518         { \int_step_inline:nnnn {#1} { -1 } {#2} }
519         { \seq_put_right:Nn \l__scontents_seq_item_seq {#1} }
520       }
521       { #3 { #1-#2 } }
522     }
523   }
524 </core>
```

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

## 12.7 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`.

```

525 (*core)
526 \cs_new_protected:Npn \__scontents_append_contents:nn #1#2
527 {
528   \tl_if_blank:nT {#1}
529     { \msg_error:nn { scontents } { empty-store-cmd } }
530   \seq_if_exist:cF { g__scontents_name_#1_seq }
531     { \seq_new:c { g__scontents_name_#1_seq } }
532   \seq_gput_right:cn { g__scontents_name_#1_seq } {#2}
533 }
534 \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 and stores it in `#2`.

```

535 \cs_new_protected:Npn \__scontents_store_to_seq:NN #1#2
536 {
537   \tl_log:N #1
538   \__scontents_append_contents:Ve #2 { \exp_not:V #1 }
539 }

```

(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 `\l__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.

```

540 \cs_new_protected:Npn \__scontents_finish_storing:NNN #1 #2 #3
541 {
542   \bool_if:NT \l__scontents_storing_bool
543     {
544       \bool_if:NF \l__scontents_forced_eol_bool
545         { \tl_put_right:Ne #1 { \c__scontents_hidden_space_str } }
546       \__scontents_store_to_seq:NN #1 #2
547       \bool_if:NT #3 { \__scontents_lastfrom_seq:V #2 }
548     }
549 }

```

(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
550 \cs_new:Npn \__scontents_getfrom_seq:Nn #1#2
551 {
552   \seq_if_exist:cTF { g__scontents_name_#2_seq }
553     {
554       \exp_args:Nf \__scontents_getfrom_seq:nNn
555         { \seq_count:c { g__scontents_name_#2_seq } } #1 {#2}
556     }
557   { \msg_expandable_error:nnn { scontents } { undefined-storage } {#2} }
558 }
559 \cs_new:Npn \__scontents_getfrom_seq:nNn #1 #2 #3
560 {
561   \seq_map_tokens:Nn #2 { \__scontents_getfrom_seq_aux:nnn {#1} {#3} }
562 }
563 \cs_new:Npn \__scontents_getfrom_seq_aux:nnn #1 #2 #3
564 {
565   \exp_args:Nnf \use:n { \__scontents_getfrom_seq:nnn {#1} } { \int_eval:n {#3} } {#2}
566 }
567 \cs_new:Npn \__scontents_getfrom_seq:nn #1#2
568 {
569   \seq_if_exist:cTF { g__scontents_name_#2_seq }
570     {

```

```

571     \exp_args:Nf \__scontents_getfrom_seq:nnn
572     { \seq_count:c { g__scontents_name_#2_seq } }
573     {#1} {#2}
574   }
575   { \msg_expandable_error:nnn { scontents } { undefined-storage } {#2} }
576 }
577 \cs_new:Npn \__scontents_getfrom_seq:nnn #1#2#3
578 {
579   \bool_lazy_or:nnTF
580   { \int_compare_p:nNn {#2} = { 0 } }
581   { \int_compare_p:nNn { \int_abs:n {#2} } > {#1} }
582   { \msg_expandable_error:nnnn { scontents } { index-out-of-range } {#2} {#3} {#1} }
583   { \seq_item:cn { g__scontents_name_#3_seq } {#2} }
584 }

```

(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 `\__scontents_rescan_tokens:V` when the keys `print-env` or `print-cmd` are active.

`\__scontents_lastfrom_seq:V`

```

585 \cs_new_protected:Npn \__scontents_lastfrom_seq:n #1
586 {
587   \tl_gset:Nx \g__scontents_last_stored_tl
588   {
589     \seq_item:cn { g__scontents_name_#1_seq } { -1 }
590   }
591   \group_insert_after:N \__scontents_rescan_tokens:V
592   \group_insert_after:N \g__scontents_last_stored_tl
593   \group_insert_after:N \tl_gclear:N
594   \group_insert_after:N \g__scontents_last_stored_tl
595 }
596 \cs_generate_variant:Nn \__scontents_lastfrom_seq:n { V }
597 </core>

```

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

## 12.8 Base functions for environments

In version 1.8 (2019-11-18) the command `\newenvsc` (§12.9) was implemented, allowing you to create environments with the same behavior as the base environment `scontents`. Since that version, the base environment `scontents` (§12.10) 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`.

### 12.8.1 Functions for keyval of environment

`\__scontents_grab_opt_arg:n`

`\__scontents_grab_opt_arg:w`

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 `\lcmd[10]` to exploit its delimited argument processor.

```

598 <*core>
599 \NewDocumentCommand \__scontents_grab_opt_arg:w { r[] }
600 {
601   \__scontents_grab_opt_arg:n {#1}
602 }

```

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.

```

603 \cs_new_protected:Npn \__scontents_grab_opt_arg:n #1
604 {
605   \tl_if_novalue:nF {#1}
606   {
607     \tl_set:Nn \l__scontents_environment_keys_tl {#1}
608     \__scontents_normalise_line_ends:N \l__scontents_environment_keys_tl
609     \keys_set:nV { scontents / scontents } \l__scontents_environment_keys_tl
610   }
611   \__scontents_start_after_option:w
612 }
613 </core>

```

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



### 12.8.2 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  $\TeX$  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.

```

614 <*core>
615 \group_begin:
616   \char_set_catcode_active:N ^^I
617   \char_set_catcode_active:N ^^L
618   \char_set_catcode_active:N ^^M
619   \cs_new_protected:Npn __scontents_normalise_line_ends:N #1
620     { \tl_replace_all:Nnn #1 { ^^M } { ~ } }
621   \cs_new_protected:Npn __scontents_start_environment:w #1 ^^M
622     {
623       \tl_if_head_is_N_type:nTF {#1}
624       {
625         \str_if_eq:eeTF { \tl_head:n {#1} } { [ ]
626           { __scontents_grab_opt_arg:w #1 ^^M }
627           { __scontents_check_line_process:en { } {#1} }
628         }
629         { __scontents_check_line_process:en { } {#1} }
630       }
631   \cs_new_protected:Npn __scontents_start_after_option:w #1 ^^M
632     { __scontents_check_line_process:en { [...] } {#1} }
633   \cs_new_protected:Npn __scontents_check_line_process:en #1 #2
634     {
635       \tl_if_blank:nF {#2}
636       {
637         \msg_error:nnee { scontents } { junk-after-begin }
638         { after~\c_backslash_str begin { \__scontents_env_name_tl } #1 } {#2}
639       }
640       __scontents_make_control_chars_active:
641       __scontents_file_tl_write_start:V \__scontents_file_name_tl
642     }
643   \cs_new_protected:Nn __scontents_stop_environment:
644     {
645       __scontents_file_write_stop:N \__scontents_processed_body_lines_tl
646       \bool_lazy_and:nnT
647         { \l__scontents_storing_bool }
648         { \tl_if_empty_p:N \__scontents_processed_body_lines_tl }
649       {
650         \msg_warning:nne { scontents } { empty-environment } { \__scontents_env_name_tl }
651       }
652     }

```

(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.

```

653   \cs_new_protected:Npn __scontents_file_tl_write_start:n #1
654     {
655       \group_begin:

```

```

656     \_scontents_file_if_writable:nTF {#1}
657     {
658         \bool_set_true:N \_scontents_writable_bool
659         \iow_open:Nn \_scontents_file_write_iow {#1}
660     }
661     { \bool_set_false:N \_scontents_writable_bool }
662 \tl_clear:N \_scontents_save_every_body_lines_tl
663 \seq_map_function:NN \_char_special_seq \char_set_catcode_other:N
664 \int_step_function:nnnN { 128 } { 1 } { 255 } \char_set_catcode_letter:n
665 \cs_set_protected:Npe \_scontents_ret:w ##1 ^^M
666     {
667         \exp_not:N \_scontents_verb_processor_iterate:w
668         ##1 \c__scontents_end_env_tl
669         \c__scontents_end_env_tl
670         \exp_not:N \q__scontents_stop
671     }
672 \_scontents_make_control_chars_active:
673 \_scontents_ret:w
674 }
675 \cs_new:Nn \_scontents_setup_verb_processor:
676 {
677     \use:e
678     {
679         \cs_set:Npn \exp_not:N \_scontents_verb_processor_iterate:w
680             ##1 \c__scontents_end_env_tl
681             ##2 \c__scontents_end_env_tl
682             ##3 \exp_not:N \q__scontents_stop
683     } { \_scontents_verb_processor_iterate:nnn {##1} {##2} {##3} }
684 }
685 \cs_new:Npn \_scontents_verb_processor_iterate:nnn #1 #2 #3
686 {
687     \tl_if_blank:nTF {#3}
688     {
689         \_scontents_analyse_nesting:n {#1}
690         \_scontents_verb_processor_output:n {#1}
691     }
692     {
693         \_scontents_if_nested:TF
694         {
695             \_scontents_nesting_decr:
696             \_scontents_verb_processor_output:e
697             { \exp_not:n {#1} \c__scontents_end_env_tl \exp_not:n {#2} }
698         }
699         {
700             \tl_if_blank:nF {#1}
701             { \_scontents_verb_processor_output:n {#1} }
702             \cs_set_protected:Npe \_scontents_ret:w
703             {
704                 \_scontents_env_end_function:
705                 \bool_lazy_or:nnF
706                 { \tl_if_blank_p:n {#2} }
707                 { \str_if_eq_p:ee {#2} { \c_percent_str } }
708                 {
709                     \str_if_eq:VnF \c__scontents_hidden_space_str {#2}
710                     {
711                         \msg_warning:nnnn { scontents } { rescanning-text }
712                         {#2} { \tl_use:N \_scontents_env_name_tl }
713                     }
714                     \_scontents_rescan_tokens:n {#2}
715                 }
716             }
717             \char_set_active_eq:NN ^^M \_scontents_ret:w
718         }
719     }
720     ^^M
721 }
722 \cs_new:Nn \_scontents_env_end_function:
723 {
724     \_scontents_format_case:nnn
725     { \exp_not:N \end { \if_false: } \fi: }
726     { \exp_after:wN \exp_not:N \cs:w end }

```

```

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

```

768 \cs_new_protected:Npn \__scontents_verb_processor_output:n #1
769 {
770   \bool_if:NT \l__scontents_writable_bool
771   { \iow_now:Nn \l__scontents_file_write_iow {#1} }
772   \bool_if:NT \l__scontents_storing_bool
773   { \tl_put_right:Nn \l__scontents_save_every_body_lines_tl { ^^J #1 } }
774 }
775 \group_end:
776 \cs_generate_variant:Nn \__scontents_verb_processor_output:n { e }
777 \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{scontents}` 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{scontents}` is found.

```

778 \cs_new_protected:Npn \__scontents_analyse_nesting:n #1
779 {
780   \int_zero:N \l__scontents_nesting_aux_int
781   \__scontents_analyse_nesting_format:n {#1}
782   \int_compare:nNnT { \l__scontents_nesting_aux_int } > { 1 }

```

```

783     { \msg_warning:nn { scontents } { multiple-begin } }
784   }
785 \cs_new_protected:Nn \__scontents_nesting_incr:
786   {
787     \int_incr:N \__scontents_nesting_env_int
788     \int_incr:N \__scontents_nesting_aux_int
789   }
790 \cs_new_protected:Nn \__scontents_nesting_decr:
791   {
792     \int_decr:N \__scontents_nesting_env_int
793   }
794 \prg_new_protected_conditional:Npnn \__scontents_if_nested: { TF }
795   {
796     \int_compare:nNnTF { \__scontents_nesting_env_int } > { \c_zero_int }
797     { \prg_return_true: }
798     { \prg_return_false: }
799   }

```

- 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.

```

800 \use:e
801   {
802     \cs_new_protected:Npn \exp_not:N \__scontents_analyse_nesting_latex:w #1
803     \c_backslash_str begin \c_left_brace_str #2 \c_right_brace_str
804   } {
805     \__scontents_tl_if_head_is_q_mark:nTF {#2}
806     { \__scontents_use_none_delimit_by_q_stop:w }
807     {
808       \str_if_eq:VnT \__scontents_env_name_tl {#2}
809       { \__scontents_nesting_incr: }
810       \__scontents_analyse_nesting_latex:w
811     }
812   }
813 \cs_new_protected:Npe \__scontents_analyse_nesting_latex:n #1
814   {
815     \__scontents_analyse_nesting_latex:w #1
816     \c_backslash_str begin
817     \c_left_brace_str \exp_not:N \q__scontents_mark \c_right_brace_str
818     \exp_not:N \q__scontents_stop
819   }

```

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`).

```

820 \cs_new_protected:Npn \__scontents_analyse_nesting_generic_process:nn #1 #2
821   {
822     \tl_if_head_is_N_type:nTF {#2}
823     {
824       \__scontents_tl_if_head_is_q_mark:nF {#2}
825       {
826         \__scontents_nesting_incr:
827         \__scontents_analyse_nesting_generic:w #2 \q__scontents_stop
828       }
829     }
830     { \__scontents_analyse_nesting_generic:w #2 \q__scontents_stop }
831   }
832 \cs_new_protected:Npn \__scontents_analyse_nesting_generic:nn #1 #2
833   {
834     \__scontents_define_generic_nesting_function:n {#1}
835     \use:e
836     {
837       \exp_not:N \__scontents_analyse_nesting_generic:w #2
838       \c_backslash_str #1 \tl_use:N \__scontents_env_name_tl
839       \exp_not:N \q__scontents_mark \exp_not:N \q__scontents_stop
840     }
841   }
842 \cs_new_protected:Npn \__scontents_define_generic_nesting_function:n #1
843   {
844     \use:e
845     {

```

```

846     \cs_set_protected:Npn \exp_not:N \__scontents_analyse_nesting_generic:w ##1
847     \c_backslash_str #1 \tl_use:N \l__scontents_env_name_tl
848     ##2 \exp_not:N \q__scontents_stop
849   } { \__scontents_analyse_nesting_generic_process:nn {##1} {##2} }
850 }
851 </core>

```

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

```

852 <*loader>
853 <latex>\cs_new_eq:NN \__scontents_analyse_nesting_format:n
854 <latex> \__scontents_analyse_nesting_latex:n
855 <latex>\cs_new_protected:Npn \__scontents_analyse_nesting_format:n
856 <plain> { \__scontents_analyse_nesting_generic:nn { } }
857 <context> { \__scontents_analyse_nesting_generic:nn { start } }
858 </loader>

```

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

## 12.9 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 (§12.8).

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

```

859 <*core>
860 \cs_new_protected:Npn \__scontents_generic_begin:
861 {
862   \char_set_catcode_active:N \^^M
863   \__scontents_start_environment:w
864 }

```

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.

```

865 \cs_new_protected:Npn \__scontents_generic_end:
866 {
867   \__scontents_stop_environment:
868   \__scontents_finish_storing:NNN
869   \l__scontents_processed_body_lines_tl
870   \l__scontents_name_seq_env_tl
871   \l__scontents_print_env_bool
872 }

```

(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.

```

873 \cs_new_protected:Npn \__scontents_setting_env:nn #1 #2
874 {
875   \cs_new_protected:cpn { __scontents_#1_begin: }
876   {
877     \tl_set:Nn \l__scontents_env_name_tl {#1}
878     \keys_set:nn { scontents } {#2}
879     \__scontents_setup_verb_processor:
880     \__scontents_generic_begin:
881   }
882   \cs_new_protected:cpn { __scontents_#1_end: }
883   { \__scontents_generic_end: }
884   \exp_args:Nooo \__scontents_define_env:nnn % http://nooooooooooooooooo.com :) jeje
885   { \tl_to_str:n {#1} }
886   { \cs:w __scontents_#1_begin: \cs_end: }
887   { \cs:w __scontents_#1_end: \cs_end: }
888 }
889 </core>

```

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

```

890 <*loader>
891 \cs_new_protected:Npn \_scontents_define_env:nnn #1 #2 #3
892 {
893   <latex|plain> \NewDocumentEnvironment {#1} { }
894   <context> \cs_new_protected:cpn { start #1 }
895   {
896     <!latex> \group_begin:
897             #2
898   }
899   <context> \cs_new_protected:cpn { stop #1 }
900   {
901     #3
902     <!latex> \group_end:
903   }
904 }

```

(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  $\ConTeXt$ .

```

905 \NewDocumentCommand \newenvsc { m O{} }
906 {
907   <latex|plain> \cs_if_exist:cTF { #1 }
908   <context> \cs_if_exist:cTF { start #1 }
909   { \msg_error:nnn { scontents } { env-already-defined } {#1} }
910   { \_scontents_setting_env:nn {#1} {#2} }
911 }
912 </loader>

```

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

## 12.10 The environment scontents

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

```

\sccontents
\endscontents
\startsccontents
\stopsccontents
913 <*loader>
914 \newenvsc{scontents}
915 </loader>

```

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

## 12.11 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`.

```

916 <!latex>
917 \cs_if_exist:NF \dospecials
918 {
919   \cs_new:Npn \dospecials
920     { \seq_map_function:NN \l_char_special_seq \do }
921 }
922 </!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  $\ConTeXt$ .

```

\endverbatimsc
\endverbatimsc
\stopverbatimsc
923 <*loader>
924 <*!context>
925 \use:e
926 {
927   \cs_new_protected:Npn \exp_not:N \_scontents_xverb:w
928     #1 \g__scontents_end_verbatimsc_tl
929   <latex> { #1 \exp_not:N \end{verbatimsc} }

```

©2019–2025 by Pablo González L

```

930 ⟨plain⟩      { #1 \exp_not:N \endverbatimsc }
931 ⟨context⟩    { #1 \exp_not:N \stopverbatimsc }
932 }
933 ⟨!context⟩
934 ⟨/loader⟩

```

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

### 12.11.1 plain T<sub>E</sub>X version off `verbatimsc`

```

\verbatimsc In plain TEX we emulate LATEX's verbatim environment.
\endverbatimsc
\__scontents_verbatimsc_aux: 935 ⟨*plain⟩
\__scontents_vobeyspaces: 936 \cs_new_protected:Npn \verbatimsc
  \__scontents_xverb: 937 {
    \__scontents_xverb: 938 \group_begin:
\__scontents_nolig_list: 939 \__scontents_verbatimsc_aux: \frenchspacing \__scontents_vobeyspaces:
\__scontents_xobeysp: 940 \__scontents_xverb:
941 }
942 \cs_new_protected:Npn \endverbatimsc
943 { \group_end: }
944 \cs_new_protected:Nn \__scontents_verbatimsc_aux:
945 {
946 \skip_vertical:N \parskip
947 \dim_zero:N \parindent
948 \skip_set:Nn \parfillskip { 0pt plus 1fil }
949 \skip_set:Nn \parskip { 0pt plus 0pt minus 0pt }
950 \tex_par:D
951 \bool_set_false:N \__scontents_plain_bool
952 \cs_set:Npn \par
953 {
954 \bool_if:NTF \__scontents_plain_bool
955 {
956 \mode_leave_vertical:
957 \null
958 \tex_par:D
959 \penalty \interlinepenalty
960 }
961 {
962 \bool_set_true:N \__scontents_plain_bool
963 \mode_if_horizontal:T
964 { \tex_par:D \penalty \interlinepenalty }
965 }
966 }
967 \cs_set_eq:NN \do \char_set_catcode_other:N
968 \dospecials \obeylines
969 \tl_use:N \__scontents_verb_font_tl
970 \cs_set_eq:NN \do \__scontents_do_noligs:N
971 \__scontents_nolig_list:
972 \tex_everypar:D \exp_after:wN
973 { \tex_the:D \tex_everypar:D \tex_unpenalty:D }
974 }
975 \cs_new_protected:Nn \__scontents_nolig_list:
976 { \do\` \do\< \do\> \do\, \do\' \do\ - }
977 \cs_new_protected:Nn \__scontents_vobeyspaces:
978 { \__scontents_set_active_eq:NN \ \__scontents_xobeysp: }
979 \cs_new_protected:Nn \__scontents_xobeysp:
980 { \mode_leave_vertical: \nobreak \ }
981 ⟨/plain⟩

```

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

### 12.11.2 ConT<sub>E</sub>Xt version off `verbatimsc`

```

\startverbatimsc In ConTEXt we use our own tool \definetyping.
\stopverbatimsc
982 ⟨*loader⟩
983 ⟨context⟩\definetyping[verbatimsc]
984 ⟨/loader⟩

```

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



### 12.11.3 $\LaTeX$ 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  $\LaTeX$  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)*.

`verbatimsc`

```

985 <*loader>
986 <*latex>
987 \cs_new_protected:Nn __scontents_verbatimsc_instance:
988   {
989     \DeclareInstance{blockenv}{verbatimsc}{display}
990     {
991       env-name      = verbatimsc,
992       tag-name      = verbatim,
993       tag-class     = ,
994       tagging-recipe = standard,
995       inner-level-counter = ,
996       level-increase = false,
997       setup-code    = ,
998       block-instance = displayblock,
999       inner-instance = ,
1000      final-code    = \legacyverbatimsetup \tag_tool:n {paratag=codeline},
1001      para-flattened = true
1002    }
1003  }
1004 \NewDocumentEnvironment { verbatimsc } { }
1005   {
1006     \IfDocumentMetadataTF
1007     {
1008       __scontents_verbatimsc_instance:
1009       \UseInstance{blockenv}{verbatimsc}{}
1010       \@setupverbinvisiblespace\frenchspacing\@vobeyspaces
1011       __scontents_xverb:
1012     }
1013     {
1014       \cs_set_eq:cN { @xverbatim } __scontents_xverb:
1015       \verbatim
1016     }
1017   }

```

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

```

1018   {
1019     \IfDocumentMetadataTF
1020     {
1021       \endblockenv
1022     }
1023     { \endverbatim }
1024   }
1025 </latex>
1026 </loader>

```

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

## 12.12 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 Latex2e to expl3*.

`__scontents_Scontents_code:n`

`__scontents_Scontents_norm_arg:n`

`__scontents_Scontents_verb_arg:w`

```

1027 <*core>
1028 \NewDocumentCommand \Scontents { !s !0{} }
1029   {
1030     \tl_set:Nn \__scontents_cmd_name_tl { Scontents }
1031     __scontents_Scontents_code:n {#1} {#2}
1032   }

```

The internal function `__scontents_Scontents_code:n` 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`.

```

1033 \cs_new_protected:Npn __scontents_Scontents_code:n #1 #2

```

```

1034 {
1035   \__scontents_bsphack:
1036   \group_begin:
1037     \tl_if_novalue:nF {#2}
1038     { \keys_set:nn { scontents / Scontents } {#2} }
1039     \char_set_catcode_active:n { 9 }
1040     \bool_if:NTF #1
1041     { \__scontents_Scontents_verb_arg:w }
1042     { \__scontents_Scontents_norm_arg:n }
1043 }

```

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  $\LaTeX$  release 2024-06-01 by `\^^M` 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.

```

1044 \NewDocumentCommand \__scontents_Scontents_verb_arg:w { +v }
1045 {
1046   \tl_set:Nn \__scontents_Scontents_arg_tl {#1}
1047   \cs_if_exist:NT \obeyedline
1048   {
1049     \RenewDocumentCommand \obeyedline { } { \iow_char:N \^^M }
1050     \tl_replace_all:Nee \__scontents_Scontents_arg_tl { \obeyedline } { \iow_char:N \^^M }
1051   }
1052   \tl_replace_all:Nee \__scontents_Scontents_arg_tl { \iow_char:N \^^M } { \iow_char:N \^^J }
1053   \__scontents_Scontents_finish:
1054 }

```

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:.`

```

1055 \cs_new_protected:Npn \__scontents_Scontents_norm_arg:n #1
1056 {
1057   \tl_set:Nn \__scontents_Scontents_arg_tl {#1}
1058   \__scontents_Scontents_finish:
1059 }

```

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 `\__scontents_Scontents_arg_tl` variable in the *sequence* `\__scontents_name_seq_cmd_tl` and print it from this *sequence* according to the state of the `\__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.

```

1060 \cs_new_protected:Nn \__scontents_Scontents_finish:
1061 {
1062   \__scontents_file_write_cmd:VV \__scontents_file_name_tl \__scontents_Scontents_arg_tl
1063   \__scontents_finish_storing:NNN
1064   \__scontents_Scontents_arg_tl \__scontents_name_seq_cmd_tl \__scontents_print_cmd_bool
1065   \use:e
1066   {
1067     \group_end:
1068     \bool_if:NF \__scontents_print_cmd_bool { \__scontents_esphack: }
1069   }
1070 }

```

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

### 12.13 The command `\getstored`

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

```

\__scontents_getstored:nn
1071 \NewDocumentCommand \getstored { 0{-1} m }
1072 {
1073   \__scontents_getstored:nn {#1} {#2}
1074 }

```

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

```

1075 \cs_new_protected:Npn \__scontents_getstored:nn #1 #2

```

```

1076 {
1077   \group_begin:
1078     \int_set:Nn \tex_newlinechar:D { `^^^J }
1079     \__scontents_rescan_tokens:e
1080     {
1081     \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
1082     \__scontents_getfrom_seq:nn {#1} {#2}
1083     }
1084 }

```

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

## 12.14 The command `\foreachsc`

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

```

\__scontents_foreachsc:nn
\__scontents_foreach_add_body:n
1085 \NewDocumentCommand \foreachsc { o m }
1086 {
1087   \tl_set:Nn \l__scontents_cmd_name_tl { foreachsc }
1088   \__scontents_foreachsc:nn {#1} {#2}
1089 }
1090 \cs_new_protected:Npn \__scontents_foreachsc:nn #1 #2
1091 {
1092   \group_begin:
1093     \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / foreachsc } {#1} }
1094     \tl_set:Nn \l__scontents_foreachsc_arg_tl {#2}
1095     \seq_clear:N \l__scontents_foreach_print_seq
1096     \bool_if:NF \l__scontents_foreach_stop_bool
1097     {
1098       \int_set:Nn \l__scontents_foreach_stop_int
1099       { \seq_count:c { g__scontents_name_#2_seq } }
1100     }
1101     \int_step_function:nnnN
1102     { \l__scontents_foreach_start_int }
1103     { \l__scontents_foreach_step_int }
1104     { \l__scontents_foreach_stop_int }
1105     \__scontents_foreach_add_body:n
1106     \tl_gset:Ne \g__scontents_foreach_exec_tl
1107     {
1108       \exp_args:NNV \seq_use:Nn
1109       \l__scontents_foreach_print_seq \l__scontents_foreach_sep_tl
1110     }
1111     \group_end:
1112     \exp_after:wN \tl_gclear:N
1113     \exp_after:wN \g__scontents_foreach_exec_tl
1114     \g__scontents_foreach_exec_tl
1115 }
1116 \cs_new_protected:Npn \__scontents_foreach_add_body:n #1
1117 {
1118   \seq_put_right:Ne \l__scontents_foreach_print_seq
1119   {
1120     \bool_if:NT \l__scontents_foreach_before_bool
1121     { \exp_not:V \l__scontents_foreach_before_tl }
1122     \bool_if:NTF \l__scontents_foreach_wrapper_bool
1123     { \__scontents_foreach_wrapper:n }
1124     { \use:n }
1125     { \getstored [#1] { \tl_use:N \l__scontents_foreachsc_arg_tl } }
1126     \bool_if:NT \l__scontents_foreach_after_bool
1127     { \exp_not:V \l__scontents_foreach_after_tl }
1128   }
1129 }

```

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

## 12.15 The command `\tpestored`

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

```

\__scontents_tpestored:nn
\__scontents_tpestored:N
1130 \NewDocumentCommand \tpestored { o m }
1131 {
1132   \tl_set:Nn \l__scontents_cmd_name_tl { tpestored }

```

```

1133   \_scontents_tpestored:nn {#1} {#2}
1134 }
1135 \cs_new_protected:Npn \_scontents_tpestored:nn #1 #2
1136 {
1137   \_scontents_bspack:
1138   \group_begin:
1139     \seq_clear:N \_scontents_seq_item_seq
1140     \str_set:Ne \_scontents_current_seq_name_str {#2}
1141     \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / typemeaning } {#1} }
1142     \seq_if_empty:NT \_scontents_seq_item_seq
1143       { \seq_set_from_clist:Nn \_scontents_seq_item_seq { 1 } }
1144     \tl_set:Ne \_scontents_tpestored_arg_tl
1145       { \_scontents_getfrom_seq:Nn \_scontents_seq_item_seq {#2} }
1146     \_scontents_remove_trailing_eol:N \_scontents_tpestored_arg_tl
1147     \tl_replace_all:Nen \_scontents_tpestored_arg_tl { \c__scontents_hidden_space_str } { ^^J }
1148     \tl_log:N \_scontents_tpestored_arg_tl
1149     \tl_if_empty:NF \_scontents_tpestored_arg_tl
1150     {
1151       \bool_if:NT \_scontents_print_verb_style_bool
1152       {
1153         \_scontents_tpestored:N \_scontents_tpestored_arg_tl
1154       }
1155     }
1156     \_scontents_file_write_cmd:VV \_scontents_file_name_tl \_scontents_tpestored_arg_tl
1157     \use:e
1158     {
1159   \group_end:
1160   \bool_if:NF \_scontents_print_verb_style_bool { \_scontents_espack: }
1161   }
1162 }

```

The `\_scontents_tpestored: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 `\_scontents_rescan_tokens:e`.

```

1163 \group_begin:
1164   \char_set_catcode_active:N ^^M
1165   \cs_new_protected:Npn \_scontents_tpestored:N #1
1166   {
1167     \tl_if_blank:VT #1
1168     { \msg_error:nnn { scontents } { empty-variable } {#1} }
1169     \cs_set_eq:NN \_scontents_verb_print_EOL: ^^M
1170     \cs_set_eq:NN ^^M \scan_stop:
1171     \cs_set_eq:cN { do@noligs } \_scontents_do_noligs:N
1172     \int_set:Nn \tex_newlinechar:D { `^^J }
1173     \_scontents_rescan_tokens:e
1174     {
1175       \_scontents_format_case:nnn
1176       { \exp_not:N \begin{verbatimsc} } % LaTeX
1177       { \verbatimsc } % Plain/Generic
1178       { \startverbatimsc } % ConTeXt
1179       ^^M
1180       \exp_not:V #1 ^^M
1181       \g__scontents_end_verbatimsc_tl
1182     }
1183     \cs_set_eq:NN ^^M \_scontents_verb_print_EOL:
1184   }
1185 \group_end:
1186 \cs_new_protected:Nn \_scontents_xverb:
1187 {
1188   \char_set_catcode_active:n { 9 }
1189   \char_set_active_eq:nN { 9 } \_scontents_tabs_to_spaces:
1190   \_scontents_xverb:w
1191 }

```

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

## 12.16 The command `\meaningsc`

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

```

\__scontents_meaningsc:n
\__scontents_meaningsc:n
1192 \NewDocumentCommand \meaningsc { o m }
1193 {

```

```

1194 \tl_set:Nn \l__scontents_cmd_name_tl { meaningsc }
1195 \__scontents_meaningsc:nn {#1} {#2}
1196 }
1197 \cs_new_protected:Npn \l__scontents_meaningsc:n #1 #2
1198 {
1199   \__scontents_bsphack:
1200   \group_begin:
1201     \seq_clear:N \l__scontents_seq_item_seq
1202     \str_set:Ne \l__scontents_current_seq_name_str {#2}
1203     \tl_if_noalue:nF {#1} { \keys_set:nn { scontents / typemeaning } {#1} }
1204     \seq_if_empty:NT \l__scontents_seq_item_seq
1205       { \seq_set_from_clist:Nn \l__scontents_seq_item_seq { 1 } }
1206     \__scontents_meaningsc:n {#2}
1207     \use:e
1208     {
1209       \group_end:
1210       \bool_if:NF \l__scontents_print_verb_style_bool { \__scontents_esphack: }
1211     }
1212   }
1213   \group_begin:
1214   \char_set_catcode_active:N ^^I
1215   \cs_new_protected:Npn \l__scontents_meaningsc:n #1
1216   {
1217     \tl_set:Ne \l__scontents_meaningsc_arg_tl
1218     { \__scontents_getfrom_seq:Nn \l__scontents_seq_item_seq {#1} }
1219     \tl_replace_all:Nen \l__scontents_meaningsc_arg_tl { \iow_char:N ^^J } { ~ }
1220     \tl_replace_all:Nen \l__scontents_meaningsc_arg_tl { \c__scontents_hidden_space_str } { ~ }
1221     \tl_log:N \l__scontents_meaningsc_arg_tl
1222     \tl_use:N \l__scontents_verb_font_tl
1223     \tl_replace_all:Nne \l__scontents_meaningsc_arg_tl { ^^I } { \__scontents_tabs_to_spaces: }
1224     \tl_if_empty:NF \l__scontents_meaningsc_arg_tl
1225     {
1226       \bool_if:NT \l__scontents_print_verb_style_bool
1227       {
1228         \cs_replacement_spec:N \l__scontents_meaningsc_arg_tl
1229       }
1230     }
1231     \__scontents_file_write_cmd:VV \l__scontents_file_name_tl \l__scontents_meaningsc_arg_tl
1232   }
1233   \group_end:

```

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

## 12.17 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
\__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

```

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.

```

1234 \NewDocumentCommand \mergesc { o m }
1235 {
1236   \tl_set:Nn \l__scontents_cmd_name_tl { mergesc }
1237   \__scontents_mergesc_code:nn {#1} {#2}
1238 }
1239 \cs_new_protected:Npn \l__scontents_mergesc_code:nn #1 #2
1240 {
1241   \group_begin:
1242     \tl_clear:N \l__scontents_mergesc_keys_tl
1243     \tl_if_noalue:nF {#1}
1244     {
1245       \keys_set_known:nn { scontents / mergesc } {#1} \l__scontents_mergesc_keys_tl
1246     }
1247     \seq_gclear:c { g__scontents_name_sc!internal_seq }
1248     \__scontents_mergesc_parse_list:n {#2}
1249     \exp_args:Ne \l__scontents_mergesc_cmd:nn
1250     { 1-end, \exp_not:V \l__scontents_mergesc_keys_tl } { sc!internal }
1251   \group_end:
1252 }

```

```

1253 \cs_new_protected:Npn \__scontents_mergesc_parse_list:n #1
1254 {
1255   \clist_map_inline:nn {#1} { \__scontents_parse_mergesc:nw ##1 \s__scontents_stop }
1256   \seq_gpop_right:cN { g__scontents_name_sc!internal_seq } \l__scontents_mergesc_arg_tl
1257   \__scontents_remove_trailing_eol:N \l__scontents_mergesc_arg_tl
1258   \seq_gput_right:cV { g__scontents_name_sc!internal_seq } \l__scontents_mergesc_arg_tl
1259 }
1260 \cs_new_protected:Npe \__scontents_remove_trailing_eol:N #1
1261 {
1262   \exp_not:N \exp_after:wN \exp_not:N \__scontents_remove_trailing_eol:w
1263   #1 \s__scontents_stop \c__scontents_hidden_space_str \s__scontents_stop \s__scontents_mark #1
1264 }
1265 \use:e
1266 {
1267   \cs_new_protected:Npn \exp_not:N \__scontents_remove_trailing_eol:w #1
1268     \c__scontents_hidden_space_str \s__scontents_stop #2 \s__scontents_mark #3
1269 } {
1270   \tl_set:Ne #3
1271   {
1272     \tl_if_empty:nTF {#2}
1273     { \exp_not:o { \__scontents_use_delimit_by_s_stop:nw #1 } }
1274     { \exp_not:n {#1} }
1275   }
1276 }
1277 \cs_new_protected:Npn \__scontents_parse_mergesc:nw #1
1278 {
1279   \peek_remove_spaces:n
1280   {
1281     \peek_charcode:NTF [ % ]
1282     { \__scontents_parse_mergesc_aux:nw {#1} }
1283     { \__scontents_parse_mergesc_aux:nw {#1} [ 1-\seq_count:c { g__scontents_name_#1_seq } ] }
1284   }
1285 }
1286 \cs_new_protected:Npn \__scontents_parse_mergesc_aux:nw #1 [#2]
1287 {
1288   \seq_clear:N \l__scontents_seq_item_seq
1289   \clist_map_inline:nn {#2}
1290   { \__scontents_parse_mergesc_range:nw {#1} ##1 - \q__scontents_mark - \s__scontents_mark }
1291   \seq_map_inline:Nn \l__scontents_seq_item_seq
1292   {
1293     \seq_gput_right:ce { g__scontents_name_sc!internal_seq }
1294     { \seq_item:cn { g__scontents_name_#1_seq } {#1} }
1295   }
1296   \__scontents_use_none_delimit_by_s_stop:w
1297 }
1298 \cs_new_protected:Npn \__scontents_parse_mergesc_range:nw #1 #2 - #3 - #4 \s__scontents_mark
1299 {
1300   \cs_set_protected:Npn \__scontents_tmp:w ##1
1301   {
1302     \msg_error:nnee { scontents } { index-out-of-range }
1303     {##1} {#1} { \seq_count:c { g__scontents_name_#1_seq } }
1304   }
1305   \__scontents_range_parser:nne {#2} {#3}
1306   { \seq_count:c { g__scontents_name_#1_seq } }
1307   { \__scontents_tmp:w }
1308 }

```

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

## 12.18 The command `\setupsc`

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

```

1309 \NewDocumentCommand \setupsc { +m }
1310 {
1311   \keys_set:nn { scontents } {#1}
1312 }

```

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

## 12.19 The command `\countsc`

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

```

1313 \NewExpandableDocumentCommand \countsc { m }
1314 {
1315   \seq_count:c { g__scontents_name_#1_seq }
1316 }

```

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

## 12.20 The command `\cleanseqsc`

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

```

1317 \NewDocumentCommand \cleanseqsc { m }
1318 {
1319   \seq_gc_clear_new:c { g__scontents_name_#1_seq }
1320 }

```

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

## 12.21 Warning and error messages

Warning and error messages used throughout the package.

```

1321 \msg_new:nnn { scontents } { junk-after-begin }
1322 {
1323   Junk~characters~#1~\msg_line_context: :
1324   \\ \\
1325   #2
1326 }
1327 \msg_new:nnnn { scontents } { env-already-defined }
1328 { Environment~'#1'~already~defined! }
1329 {
1330   You~have~used~\newenvsc
1331   with~an~environment~that~already~has~a~definition. \\ \\
1332   The~existing~definition~of~'#1'~will~not~be~altered.
1333 }
1334 \msg_new:nnn { scontents } { empty-stored-content }
1335 { Empty~value~for~key~'getstored'~\msg_line_context:. }
1336 \msg_new:nnn { scontents } { empty-variable }
1337 { Variable~'#1'~empty~\msg_line_context:. }
1338 \msg_new:nnn { scontents } { overwrite-file }
1339 { Overwriting~file~'#1'. }
1340 \msg_new:nnn { scontents } { writing-file }
1341 { Writing~file~'#1'. }
1342 \msg_new:nnn { scontents } { not-writing }
1343 { File~'#1'~already~exists.~Not~writing. }
1344 \msg_new:nnn { scontents } { rescanning-text }
1345 { Rescanning~text~'#1'~after~\c_backslash_str end{#2}~\msg_line_context:. }
1346 \msg_new:nnn { scontents } { multiple-begin }
1347 { Multiple~\c_backslash_str begin{ \l__scontents_env_name_tl }~\msg_line_context:. }
1348 \msg_new:nnn { scontents } { undefined-storage }
1349 { Storage~named~'#1'~is~not~defined. }
1350 \msg_new:nnnn { scontents } { mergesc-missing-key }
1351 {
1352   Need~mandatory~key~'tpestored'~or~'meaningsc'~for~\
1353   command~\c_backslash_str mergesc~\msg_line_context:.
1354 }
1355 {
1356   The~command~\c_backslash_str mergesc~need~a~mandatory~key~tpestored~or~meaningsc.\\
1357   Check~that~you~have~spelled~the~key~name~correctly.
1358 }
1359 \msg_new:nnn { scontents } { index-out-of-range }
1360 {
1361   \int_compare:nNnTF {#1} = { 0 }
1362   { Index~of~sequence~cannot~be~zero. }
1363   {
1364     Index~'#1'~out~of~range~for~'#2'.~
1365     \int_compare:nNnTF {#1} > { 0 }
1366     { Max = } { Min = - } #3.
1367   }

```

```

1368 }
1369 \msg_new:nnnn { scontents } { env-key-unknown }
1370 {
1371   The~key~'#1'~is~unknown~by~environment~
1372   '\l__scontents_env_name_tl'~and~is~being~ignored.
1373 }
1374 {
1375   The~environment~'\l__scontents_env_name_tl'~does~not~have~a~key~called~'#1'.\\
1376   Check~that~you~have~spelled~the~key~name~correctly.
1377 }
1378 \msg_new:nnnn { scontents } { env-key-value-unknown }
1379 {
1380   The~key~'#1=#2'~is~unknown~by~environment~
1381   '\l__scontents_env_name_tl'~and~is~being~ignored.
1382 }
1383 {
1384   The~environment~'\l__scontents_env_name_tl'~does~not~have~a~key~called~'#1'.\\
1385   Check~that~you~have~spelled~the~key~name~correctly.
1386 }
1387 \msg_new:nnnn { scontents } { cmd-key-unknown }
1388 {
1389   The~key~'#1'~is~unknown~by~command~\c_backslash_str \l__scontents_cmd_name_tl \c_space_tl
1390   and~is~being~ignored~ \msg_line_context:.
1391 }
1392 {
1393   The~command~\c_backslash_str \l__scontents_cmd_name_tl \c_space_tl
1394   does~not~have~a~key~called~'#1'.\\
1395   Check~that~you~have~spelled~the~key~name~correctly.
1396 }
1397 \msg_new:nnnn { scontents } { cmd-key-value-unknown }
1398 {
1399   The~key~'#1=#2'~is~unknown~by~command~\c_backslash_str \l__scontents_cmd_name_tl \c_space_tl
1400   and~is~being~ignored~ \msg_line_context:.
1401 }
1402 {
1403   The~command~\c_backslash_str \l__scontents_cmd_name_tl \c_space_tl
1404   does~not~have~a~key~called~'#1'.\\
1405   Check~that~you~have~spelled~the~key~name~correctly.
1406 }
1407 \msg_new:nnn { scontents } { empty-environment }
1408 { environment~'#1'~empty~\msg_line_context:. }
1409 \msg_new:nnnn { scontents } { verbatim-newline }
1410 { Verbatim~argument~of~#1~ended~by~end~of~line. }
1411 {
1412   The~verbatim~argument~of~the~#1~cannot~contain~more~than~one~line,~
1413   but~the~end~
1414   of~the~current~line~has~been~reached.~You~may~have~forgotten~the~
1415   closing~delimiter.
1416   \\ \\
1417   LaTeX~will~ignore~'#2'.
1418 }
1419 \msg_new:nnnn { scontents } { verbatim-tokenized }
1420 { The~verbatim~#1~cannot~be~used~inside~an~argument. }
1421 {
1422   The~#1~takes~a~verbatim~argument.~
1423   It~may~not~appear~within~the~argument~of~another~function.~
1424   It~received~an~illegal~token \tl_if_empty:nF {#3} { ~'#3' } .
1425   \\ \\
1426   LaTeX~will~ignore~'#2'.
1427 }
1428 </core>

```

## 12.22 Finish package

Finish package implementation.

```

1429 <plain|context>\ExplSyntaxOff
1430 <plain|context>\endinput
1431 <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.

|                            |   |
|----------------------------|---|
| <b>Symbols</b>             |   |
| \'                         | 976   |
| \*                         | 293, 298, 309   |
| \,                         | 976   |
| \-                         | 976   |
| \<                         | 976   |
| \>                         | 976   |
| \\                         | 25, 67, 1081, 1324, 1331, 1352, 1356, 1375, 1384, 1394, 1404, 1416, 1425  |
| \`                         | 976   |
| <b>A</b>                   |   |
| after                      | 406   |
| <b>B</b>                   |   |
| before                     | 406   |
| \beginngroup               | 95, 100   |
| bool commands:             |   |
| \bool_if:NTF               | 46, 239, 243, 542, 544, 547, 736, 741, 770, 772, 954, 1040, 1068, 1096, 1120, 1122, 1126, 1151, 1160, 1210, 1226  |
| \bool_lazy_and:nnTF        | 512, 646  |
| \bool_lazy_or:nnTF         | 579, 705  |
| \bool_new:N                | 190, 191, 193, 197, 199, 201, 202, 216  |
| \bool_set_false:N          | 360, 391, 452, 661, 951   |
| \bool_set_true:N           | 192, 355, 356, 361, 386, 387, 392, 409, 414, 422, 430, 453, 658, 962  |
| <b>C</b>                   |   |
| \catcode                   | 96, 1081  |
| char commands:             |   |
| \char_set_active_eq:NN     | 283, 288, 717   |
| \char_set_active_eq:nN     | 1189  |
| \char_set_catcode:nn       | 61  |
| \char_set_catcode_active:N | 276, 287, 309, 616, 617, 618, 862, 1164, 1214   |
| \char_set_catcode_active:n | 293, 1039, 1188   |
| \char_set_catcode_letter:N | 59  |
| \char_set_catcode_letter:n | 664   |
| \char_set_catcode_other:N  | 67, 69, 70, 663, 967  |
| \char_set_lccode:nn        | 298   |
| \l_char_special_seq        | 38, 663, 920  |
| \char_value_catcode:n      | 58  |
| \cleanseqsc                | 8, 46, 1317   |
| clist commands:            |   |
| \clist_map_inline:nn       | 1255, 1289  |
| \contextlmtxmode           | 46  |
| \countsc                   | 7, 46, 1313   |
| cs commands:               |   |
| \cs:w                      | 726, 727, 886, 887  |
| \cs_end:                   | 731, 732, 886, 887  |
| \cs_generate_variant:Nn    | 220, 269, 502, 534, 596, 776, 777   |
| \cs_gset_eq:NN             | 348, 349  |
| \cs_if_exist:NTF           | 20, 34, 907, 908, 917, 1047   |
| \cs_new:Nn                 | 270, 271, 272, 675, 722   |
| \cs_new:Npn                | 89, 222, 223, 224, 225, 550, 559, 563, 567, 577, 685, 751, 761, 767, 919  |
| \cs_new_eq:NN              | 853   |
| \cs_new_protected:Nn       | 294, 310, 320, 329, 643, 785, 790, 944, 975, 977, 979, 987, 1060, 1186  |
| \cs_new_protected:Npe      | 813, 1260   |
| \cs_new_protected:Npn      | 219, 260, 274, 285, 375, 377, 440, 442, 477, 481, 483, 489, 495, 503, 526, 535, 540, 585, 603, 619, 621, 631, 633, 653, 734, 768, 778, 802, 820, 832, 842, 855, 860, 865, 873, 875, 882, 891, 894, 899, 927, 936, 942, 1033, 1055, 1075, 1090, 1116, 1135, 1165, 1197, 1215, 1239, 1253, 1267, 1277, 1286, 1298 |
| \cs_replacement_spec:N     | 1228  |
| \cs_set:Npe                | 277   |
| \cs_set:Npn                | 679, 952  |
| \cs_set_eq:NN              | 468, 473, 967, 970, 1014, 1169, 1170, 1171, 1183  |
| \cs_set_protected:Npe      | 665, 702  |
| \cs_set_protected:Npn      | 431, 846, 1300  |
| \csname                    | 99, 109   |
| <b>D</b>                   |   |
| \DeclareInstance           | 989   |
| \def                       | 2, 3, 5, 6, 98, 101, 102, 110, 123  |
| \definetying               | 983   |
| dim commands:              |   |
| \dim_compare:nNnTF         | 335   |
| \dim_zero:N                | 947   |
| \c_zero_dim                | 280   |
| \do                        | 920, 967, 970, 976  |
| \dospecials                | 916, 968  |
| <b>E</b>                   |   |
| \else                      | 120, 122  |
| else commands:             |   |
| \else:                     | 233   |
| \end                       | 73, 725, 929  |
| \endblockenv               | 1021  |
| \endcsname                 | 99, 109   |
| \endgroup                  | 98, 101, 104, 117, 131, 1081  |
| \endinput                  | 118, 132, 1430  |
| \endlinechar               | 97  |
| \endscontents              | 4, 913  |
| \endverbatim               | 1023  |
| \endverbatimsc             | 74, 923, 935  |
| \end{verbatimsc}           | 923   |
| Environments:              |   |
| verbatim                   | 38, 39  |
| \errhelp                   | 105   |
| \errmessage                | 106   |
| exp commands:              |   |
| \exp_after:wN              | 229, 230, 726, 727, 972, 1112, 1113, 1262   |
| \exp_args:Ne               | 1249  |
| \exp_args:Nee              | 497   |
| \exp_args:Nf               | 554, 571, 755   |
| \exp_args:Nnf              | 565   |
| \exp_args:NNV              | 1108  |
| \exp_args:Noo              | 884   |
| \exp_args:NV               | 376, 441, 482, 745  |
| \exp_not:N                 | 83, 281, 667, 670, 679, 682, 725, 726, 727, 743, 802, 817, 818, 837, 839, 846, 848, 927, 929, 930, 931, 1176, 1262, 1267  |

|                                    |   |
|------------------------------------|---|
| \exp_not:n                         | 498, 499, 538, 697, 759, 764, 765, 1121, 1127, 1180, 1250, 1273, 1274             |
| \expandafter                       | 99, 109   |
| \ExplSyntaxOff                     | 29, 41, 1429  |
| \ExplSyntaxOn                      | 16  |
| <b>F</b>                           |   |
| \fi                                | 108, 134, 135   |
| fi commands:                       |   |
| \fi:                               | 235, 725, 730   |
| file commands:                     |   |
| \file_if_exist:nTF                 | 241   |
| \file_input:n                      | 60, 88  |
| \file_input_stop:                  | 30, 42, 1431  |
| force-eol                          | 137, 351, 383   |
| \foreachsc                         | 6, 28, 29, 42, 1085   |
| \frenchspacing                     | 939, 1010   |
| <b>G</b>                           |   |
| \getstored                         | 6, 41, 1071, 1125   |
| group commands:                    |   |
| \group_begin:                      | 292, 297, 308, 615, 655, 896, 938, 1036, 1077, 1092, 1138, 1163, 1200, 1213, 1241 |
| \group_end:                        | 299, 305, 317, 740, 775, 902, 943, 1067, 1111, 1159, 1185, 1209, 1233, 1251       |
| \group_insert_after:N              | 591, 592, 593, 594  |
| <b>I</b>                           |   |
| if commands:                       |   |
| \if_false:                         | 725, 730  |
| \if_meaning:w                      | 229   |
| \IfDocumentMetadataTF              | 1006, 1019  |
| \ifx                               | 99, 109, 121  |
| \input                             | 15  |
| int commands:                      |   |
| \int_abs:n                         | 581   |
| \int_compare:nNnTF                 | 516, 782, 796, 1361, 1365   |
| \int_compare_p:nNn                 | 580, 581  |
| \int_decr:N                        | 792   |
| \int_eval:n                        | 508, 565  |
| \int_incr:N                        | 787, 788  |
| \int_new:N                         | 57, 178, 179, 200, 208  |
| \int_set:Nn                        | 58, 423, 1078, 1098, 1172   |
| \int_set_eq:NN                     | 326, 334  |
| \int_step_function:nnnN            | 664, 1101   |
| \int_step_inline:nnnn              | 517, 518  |
| \int_step_tokens:nn                | 20  |
| \int_to_roman:n                    | 507, 513, 514   |
| \int_zero:N                        | 780   |
| \c_zero_int                        | 796   |
| \interlinepenalty                  | 959, 964  |
| iow commands:                      |   |
| \iow_char:N                        | 1049, 1050, 1052, 1219  |
| \iow_close:N                       | 266, 737  |
| \iow_log:n                         | 19  |
| \iow_new:N                         | 189   |
| \iow_now:Nn                        | 265, 771  |
| \iow_open:Nn                       | 264, 659  |
| <b>K</b>                           |   |
| Keys provide by <b>SCONTENTS</b> : |   |
| meaningsc                          | 29  |
| tpestored                          | 29  |
| keys commands:                     |   |
| \keys_define:nn                    | 137, 168, 352, 383, 406, 448, 464   |
| \l_keys_key_str                    | 376, 441, 482   |
| \keys_set:nn                       | 609, 878, 1038, 1093, 1141, 1203, 1311  |
| \keys_set_known:nnN                | 1245  |
| <b>L</b>                           |   |
| \legacyverbatimsetup               | 1000  |
| <b>M</b>                           |   |
| \meaningsc                         | 6, 29, 30, 43, 1192   |
| meaningsc                          | 464   |
| \mergesc                           | 7, 29, 30, 44, 1234   |
| mode commands:                     |   |
| \mode_if_horizontal:TF             | 323, 332, 963   |
| \mode_leave_vertical:              | 279, 956, 980   |
| msg commands:                      |   |
| \msg_error:nn                      | 52, 479, 529  |
| \msg_error:nnn                     | 380, 445, 493, 909, 1168  |
| \msg_error:nnnn                    | 381, 446, 487, 637  |
| \msg_error:nnnnn                   | 1302  |
| \msg_expandable_error:nnn          | 557, 575  |
| \msg_expandable_error:nnnn         | 582   |
| \msg_fatal:nn                      | 28  |
| \msg_line_context:                 | 38, 1323, 1335, 1337, 1345, 1347, 1353, 1390, 1400, 1408                          |
| \msg_new:nnn                       | 22, 36, 48, 1321, 1334, 1336, 1338, 1340, 1342, 1344, 1346, 1348, 1359, 1407      |
| \msg_new:nnnn                      | 1327, 1350, 1369, 1378, 1387, 1397, 1409, 1419                                    |
| \msg_warning:nn                    | 40, 783   |
| \msg_warning:nnn                   | 245, 249, 254, 650  |
| \msg_warning:nnnn                  | 711   |
| <b>N</b>                           |   |
| \NeedsTeXFormat                    | 8   |
| \NewDocumentCommand                | 599, 905, 1028, 1044, 1071, 1085, 1130, 1192, 1234, 1309, 1317                    |
| \NewDocumentEnvironment            | 893, 1004   |
| \newenvsc                          | 5, 37, 905, 914, 1330   |
| \NewExpandableDocumentCommand      | 1313  |
| \next                              | 98, 101, 110, 123, 136  |
| \nobreak                           | 339, 980  |
| \null                              | 957   |
| <b>O</b>                           |   |
| \obeyedline                        | 1047, 1049, 1050  |
| \obeylines                         | 968   |
| overwrite                          | 137, 351, 383, 448  |
| <b>P</b>                           |   |
| \PackageError                      | 102, 112, 125   |
| Packages:                          |   |
| expl3                              | 19, 24  |
| l3keys                             | 30  |
| latex-lab                          | 40  |
| ltxcmd                             | 20, 32, 41  |
| lua-widow-control                  | 20  |
| scontents                          | 19, 21, 24, 30, 38  |
| verbatim                           | 40  |
| xparse                             | 19, 20  |
| xsim                               | 33  |
| xtemplate                          | 40  |
| \par                               | 952   |
| \parfillskip                       | 948   |
| \parindent                         | 947   |
| \parskip                           | 946, 949  |
| peek commands:                     |   |
| \peek_charcode:N                   | 1281  |

`\peek_remove_spaces:n` ..... 1279  
`\penalty` ..... 959, 964  
 prg commands:  
   `\prg_generate_conditional_variant:Nnn` .. 221  
   `\prg_new_protected_conditional:Npnn` . 226, 237, 794  
   `\prg_replicate:nn` ..... 273  
   `\prg_return_false:` ..... 234, 250, 258, 798  
   `\prg_return_true:` ..... 232, 246, 255, 797  
`print-all` ..... 137  
`print-cmd` ..... 137, 383, 448  
`print-env` ..... 137, 351  
`\ProcessKeyOptions` ..... 172  
`\ProvidesExplPackage` ..... 9

## Q

quark commands:  
   `\quark_new:N` ..... 210, 211  
 quark internal commands:  
   `\q_scontents_mark` 25, 210, 231, 486, 817, 839, 1290  
   `\q_scontents_stop` 210, 225, 670, 682, 818, 827, 830, 839, 848

## R

`\relax` ..... 99, 109  
`\RenewDocumentCommand` ..... 1049

## S

scan commands:  
   `\scan_new:N` ..... 212, 213  
   `\scan_stop:` ..... 300, 322, 331, 1170  
 scan internal commands:  
   `\s_scontents_mark` 210, 486, 489, 1263, 1268, 1290, 1298  
   `\s_scontents_stop` .. 210, 222, 223, 224, 231, 1255, 1263, 1268  
`\Scontents` ..... 5, 28, 29, 40, 1027  
`\scontents` ..... 4, 913  
`scontents` ..... 4, 913  
 scontents internal commands:  
   `\__scontents_analyse_nesting:n` 35, 689, 778, 778  
   `\__scontents_analyse_nesting:w` ..... 778  
   `\__scontents_analyse_nesting_format:n` 37, 781, 853, 855  
   `\__scontents_analyse_nesting_generic:nn` . 832, 856, 857  
   `\__scontents_analyse_nesting_generic:w` .. 827, 830, 837, 846  
   `\__scontents_analyse_nesting_generic-process:nn` ..... 820, 849  
   `\__scontents_analyse_nesting_latex:n` 813, 854  
   `\__scontents_analyse_nesting_latex:w` 802, 810, 815  
   `\__scontents_append_contents:nn` 30, 525, 526, 534, 538  
   `\__scontents_bsphack:` .. 24, 40, 319, 320, 348, 1035, 1137, 1199  
   `\l_scontents_char_value_int` ..... 56  
   `\__scontents_check_line_process:nn` 33, 614, 627, 629, 632, 633  
   `\l_scontents_cmd_name_tl` 23, 180, 1030, 1087, 1132, 1194, 1236, 1389, 1393, 1399, 1403  
   `\l_scontents_current_seq_name_str` 23, 185, 492, 1140, 1202  
   `\__scontents_define_env:nnn` . 37, 38, 873, 884, 891

`\__scontents_define_generic_nesting-function:n` ..... 834, 842  
`\__scontents_do_noligs:N` . 26, 274, 274, 970, 1171  
`\c_scontents_end_env_tl` 20, 63, 668, 669, 680, 681, 697  
`\g_scontents_end_verbatimsc_tl` 20, 63, 928, 1181  
`\__scontents_env_end_function:` ..... 704, 722  
`\l_scontents_env_name_tl` 20, 37, 63, 638, 650, 712, 728, 808, 838, 847, 877, 1347, 1372, 1375, 1381, 1384  
`\l_scontents_environment_keys_tl` . 22, 174, 607, 608, 609  
`\__scontents_esphack:` .. 24, 41, 319, 329, 349, 1068, 1160, 1210  
`\__scontents_file_if_writable:n` .. 25, 237, 237  
`\__scontents_file_if_writable:nTF` . 23, 237, 262, 656  
`\l_scontents_file_name_tl` . 23, 188, 357, 362, 388, 393, 454, 641, 1062, 1156, 1231  
`\__scontents_file_tl_write_start:n` 33, 641, 653, 653, 777  
`\__scontents_file_write_cmd:nn` . 25, 41, 260, 260, 269, 1062, 1156, 1231  
`\l_scontents_file_write_iow` .. 23, 188, 264, 265, 266, 659, 737, 771  
`\__scontents_file_write_stop:N` 33, 645, 653, 734  
`\__scontents_finish_storing:NNN` 31, 37, 41, 540, 540, 868, 1063  
`\l_scontents_forced_eol_bool` .... 31, 153, 544  
`\__scontents_foreach_add_body:n` 1085, 1105, 1116  
`\l_scontents_foreach_after_bool` 194, 414, 1126  
`\l_scontents_foreach_after_tl` . 194, 415, 1127  
`\l_scontents_foreach_before_bool` 194, 409, 1120  
`\l_scontents_foreach_before_tl` 194, 410, 1121  
`\g_scontents_foreach_exec_tl` . 194, 1106, 1113, 1114  
`\l_scontents_foreach_print_seq` 194, 1095, 1109, 1118  
`\l_scontents_foreach_sep_tl` ..... 435, 1109  
`\l_scontents_foreach_start_int` ... 418, 1102  
`\l_scontents_foreach_step_int` ... 426, 1103  
`\l_scontents_foreach_stop_bool` 201, 422, 1096  
`\l_scontents_foreach_stop_int` . 194, 423, 1098, 1104  
`\__scontents_foreach_wrapper:n` .... 432, 1123  
`\l_scontents_foreach_wrapper_bool` .. 194, 430, 1122  
`\__scontents_foreachsc:nn` ..... 1085, 1088, 1090  
`\l_scontents_foreachsc_arg_tl` 23, 181, 1094, 1125  
`\__scontents_format_case:nnn` 89, 89, 724, 729, 1175  
`\__scontents_generic_begin:` ... 37, 859, 860, 880  
`\__scontents_generic_end:` ..... 37, 859, 865, 883  
`\__scontents_getfrom_seq:Nn` .. 31, 550, 550, 1145, 1218  
`\__scontents_getfrom_seq:nn` .. 41, 550, 567, 1082  
`\__scontents_getfrom_seq:nNn` .... 550, 554, 559  
`\__scontents_getfrom_seq:nnn` . 550, 565, 571, 577  
`\__scontents_getfrom_seq_aux:nnn` . 550, 561, 563  
`\__scontents_getstored:nn` ... 41, 1071, 1073, 1075  
`\__scontents_grab_opt_arg:n` ..... 598, 601, 603  
`\__scontents_grab_opt_arg:w` . 32, 33, 598, 599, 626  
`\c_scontents_hidden_space_str` . 24, 31, 206, 545, 709, 1147, 1220, 1263, 1268  
`\__scontents_if_nested:` ..... 35, 794  
`\__scontents_if_nested:TF` ..... 693, 778

`\g__scontents_last_stored_tl` 24, 205, 587, 592, 594  
`\g__scontents_last_tl` . . . . . 32  
`\__scontents_lastfrom_seq:n` . 24, 31, 32, 547, 585, 585, 596  
`\__scontents_make_control_chars_active:` . 285, 310, 640, 672  
`\__scontents_meaningsc:n` . . . . . 1192, 1206, 1215  
`\__scontents_meaningsc:nn` . . 473, 1192, 1195, 1197  
`\l__scontents_meaningsc_arg_tl` . . 23, 181, 1217, 1219, 1220, 1221, 1223, 1224, 1228, 1231  
`\l__scontents_mergesc_arg_tl` . 23, 181, 185, 1256, 1257, 1258  
`\__scontents_mergesc_cmd:nn` 29, 464, 468, 473, 477, 1249  
`\__scontents_mergesc_code:nn` . . 1234, 1237, 1239  
`\l__scontents_mergesc_keys_tl` 23, 185, 1242, 1245, 1250  
`\__scontents_mergesc_parse_list:n` . . 1234, 1248, 1253  
`\g__scontents_name_sc!internal_seq` . . . 23, 203  
`\l__scontents_name_seq_cmd_tl` . . . 41, 142, 1064  
`\l__scontents_name_seq_env_tl` . . . . . 139, 870  
`\l__scontents_nesting_aux_int` . 22, 174, 780, 782, 788  
`\__scontents_nesting_decr:` . . . . 35, 695, 778, 790  
`\l__scontents_nesting_env_int` . . 22, 35, 174, 787, 792, 796  
`\__scontents_nesting_incr:` . . . . . 785, 809, 826  
`\__scontents_nolig_list:` . . . . . 935, 971, 975  
`\__scontents_normalise_line_ends:N` 32, 608, 619  
`\l__scontents_overwrite_bool` . . . . . 156, 243  
`\__scontents_par:` . . . . . 270, 271, 314  
`\__scontents_parse_environment_keys:n` . . . 27  
`\__scontents_parse_mergesc:nw` . 1234, 1255, 1277  
`\__scontents_parse_mergesc_aux:nw` . . 1234, 1282, 1283, 1286  
`\__scontents_parse_mergesc_range:nw` 1234, 1290, 1298  
`\__scontents_parse_type_meaning_key:n` 23, 462, 481, 481  
`\__scontents_parse_type_meaning_key:nn` . . 481, 482, 483  
`\__scontents_parse_type_meaning_range:w` . 481, 486, 489  
`\__scontents_parse_typemeaning_key:n` . . . . 30  
`\l__scontents_plain_bool` . . 24, 215, 951, 954, 962  
`\__scontents_plain_disable_outer_par:` 285, 294, 312  
`\l__scontents_print_cmd_bool` 41, 150, 1064, 1068  
`\l__scontents_print_env_bool` . . . . . 147, 871  
`\l__scontents_print_verb_style_bool` 459, 1151, 1160, 1210, 1226  
`\l__scontents_processed_body_lines_tl` 22, 174, 645, 648, 869  
`\__scontents_range_parser:nnnn` 481, 491, 495, 502, 1305  
`\__scontents_range_parser_aux:nnn` 481, 497, 503  
`\__scontents_remove_leading_nl:n` . . 33, 653, 746, 751  
`\__scontents_remove_leading_nl:nn` . . . 756, 761  
`\__scontents_remove_leading_nl:w` . 653, 764, 767  
`\__scontents_remove_trailing_eol:N` . 1146, 1234, 1257, 1260  
`\__scontents_remove_trailing_eol:w` . 1234, 1262, 1267  
`\__scontents_rescan_tokens:n` . 32, 41, 43, 34, 218, 219, 220, 591, 714, 1079, 1173  
`\__scontents_ret:w` . . . . 33, 315, 665, 673, 702, 717  
`\l__scontents_save_every_body_lines_tl` 22, 174, 662, 746, 773  
`\l__scontents_save_sf_int` . . . . . 208, 326, 334  
`\l__scontents_save_skip` . . . . . 208, 325, 335  
`\l__scontents_Scontents_arg_tl` 23, 41, 181, 1046, 1050, 1052, 1057, 1062, 1064  
`\__scontents_Scontents_code:nn` . 40, 1027, 1031, 1033  
`\__scontents_Scontents_finish:` . 41, 1053, 1058, 1060  
`\__scontents_Scontents_norm_arg:n` . 40, 41, 1027, 1042, 1055  
`\__scontents_Scontents_verb_arg:w` . 40, 41, 1027, 1041, 1044  
`\l__scontents_seq_item_seq` 23, 203, 508, 519, 1139, 1142, 1143, 1145, 1201, 1204, 1205, 1218, 1288, 1291  
`\__scontents_set_active_eq:NN` 285, 285, 313, 314, 315, 978  
`\__scontents_setting_env:nn` . . . 37, 873, 873, 910  
`\__scontents_setup_verb_processor:` 653, 675, 879  
`\__scontents_start_after_option:w` . 33, 611, 614, 631  
`\__scontents_start_environment:w` . . 37, 614, 621, 863  
`\__scontents_stop_environment:` 37, 614, 643, 867  
`\__scontents_store_to_seq:NN` . . 31, 535, 535, 546  
`\l__scontents_storing_bool` . 23, 35, 191, 192, 355, 360, 386, 391, 452, 542, 647, 741, 772  
`\l__scontents_storing_bool` . . . . . 188  
`\__scontents_tab:` . . . . . 270, 270, 313  
`\l__scontents_tab_width_int` . . . . . 159, 273  
`\__scontents_tabs_to_spaces:` . 25, 272, 272, 1189, 1223  
`\__scontents_tl_if_head_is_q_mark:n` . . 25, 226  
`\__scontents_tl_if_head_is_q_mark:nTF` 226, 505, 805, 824  
`\__scontents_tmp:w` . . . . . 1300, 1307  
`\__scontents_typerstored:N` 42, 43, 1130, 1153, 1165  
`\__scontents_typerstored:nn` . 468, 1130, 1133, 1135  
`\l__scontents_typerstored_arg_tl` . 23, 181, 1144, 1146, 1147, 1148, 1149, 1153, 1156  
`\__scontents_unknown_keys_cmd:n` 29, 404, 438, 440, 440  
`\__scontents_unknown_keys_cmd:nn` . 440, 441, 442  
`\__scontents_unknown_keys_env:n` . 373, 375, 375  
`\__scontents_unknown_keys_env:nn` . 375, 376, 377  
`\__scontents_use_delimit_by_s_stop:nw` 222, 222, 1273  
`\__scontents_use_i_delimit_by_s_stop:nw` . 222, 223, 231  
`\__scontents_use_none_delimit_by_q_stop:w` 222, 225, 778, 806  
`\__scontents_use_none_delimit_by_s_stop:w` 222, 224, 1296  
`\l__scontents_verb_font_tl` . . . . . 145, 969, 1222  
`\__scontents_verb_print_EOL:` . . . . . 1169, 1183  
`\__scontents_verb_processor_iterate:nnn` . 653, 683, 685  
`\__scontents_verb_processor_iterate:w` 653, 667, 679

|   |  |
|---|--|
| \_scontents_verb_processor_output:n     | 35, 690, 696, 701, <u>768</u> , 768, 776   |
| \_scontents_verbatimisc_aux:            | ... 935, 939, 944  |
| \_scontents_verbatimisc_instance:       | 985, 987, 1008   |
| \_scontents_vobeyspaces:                | ... 935, 939, 977  |
| \l_scontents_writable_bool              | 23, <u>188</u> , 658, 661, 736, 770  |
| \l_scontents_writing_bool               | 23, 35, <u>188</u> , 239, 356, 361, 387, 392, 453  |
| \_scontents_xobeysp:                    | ... 935, 978, 979  |
| \_scontents_xverb:                      | ... 935, 940, 1011, 1014, 1186   |
| \_scontents_xverb:w                     | ... 923, 927, <u>1130</u> , 1190   |
| \ScontentsCoreFileDate                  | ... 3, 121   |
| \ScontentsFileDate                      | ... 2, 10, 17, 121   |
| \ScontentsFileDescription               | ... 6, 10, 18  |
| \ScontentsFileVersion                   | ... 5, 10, 13, 18  |
| sep                                     | ... <u>406</u>   |
| seq commands:                           |  |
| \seq_clear:N                            | ... 1095, 1139, 1201, 1288   |
| \seq_count:N                            | 492, 555, 572, 1099, 1283, 1303, 1306, 1315  |
| \seq_gclear:N                           | ... 1247   |
| \seq_gclear_new:N                       | ... 1319   |
| \seq_gpop_right:NN                      | ... 1256   |
| \seq_gput_right:Nn                      | ... 532, 1258, 1293  |
| \seq_if_empty:NTF                       | ... 1142, 1204   |
| \seq_if_exist:NTF                       | ... 530, 552, 569  |
| \seq_item:Nn                            | ... 583, 589, 1294   |
| \seq_map_function:NN                    | ... 663, 920   |
| \seq_map_inline:Nn                      | ... 1291   |
| \seq_map_tokens:Nn                      | ... 561  |
| \seq_new:N                              | ... 194, 203, 204, 531   |
| \seq_put_right:Nn                       | ... 508, 519, 1118   |
| \seq_set_from_clist:Nn                  | ... 1143, 1205   |
| \seq_use:Nn                             | ... 1108   |
| \setupsc                                | ... 3, 45, <u>1309</u>   |
| skip commands:                          |  |
| \skip_horizontal:n                      | ... 340  |
| \skip_if_eq:nnTF                        | ... 337  |
| \skip_new:N                             | ... 209  |
| \skip_set:Nn                            | ... 948, 949   |
| \skip_set_eq:NN                         | ... 325  |
| \skip_vertical:N                        | ... 946  |
| \c_zero_skip                            | ... 335, 337, 340  |
| \space                                  | ... 17, 18   |
| start                                   | ... <u>406</u>   |
| \startscontents                         | ... 4, <u>913</u>  |
| \startverbatimsc                        | ... <u>982</u> , 1178  |
| step                                    | ... <u>406</u>   |
| stop                                    | ... <u>406</u>   |
| \stopscontents                          | ... 4, <u>913</u>  |
| \stopverbatimsc                         | ... 75, <u>923</u> , <u>982</u>  |
| store-all                               | ... <u>137</u>   |
| store-cmd                               | ... <u>137</u> , <u>383</u>  |
| store-env                               | ... <u>137</u> , <u>351</u>  |
| str commands:                           |  |
| \c_backslash_str                        | ... 79, 638, 803, 816, 838, 847, 1345, 1347, 1353, 1356, 1389, 1393, 1399, 1403                                    |
| \c_circumflex_str                       | ... 207  |
| \c_left_brace_str                       | ... 82, 803, 817   |
| \c_percent_str                          | ... 207, 707   |
| \c_right_brace_str                      | ... 84, 803, 817   |
| \str_const:Nn                           | ... 206  |
| \str_if_eq:nnTF                         | ... 498, 499, 625, 709, 808  |
| \str_if_eq_p:nn                         | ... 707  |
| \str_new:N                              | ... 187  |
| \str_set:Nn                             | ... 1140, 1202   |
| <b>T</b>                                |  |
| tag commands:                           |  |
| \tag_tool:n                             | ... 1000   |
| TeX and $\TeX$ $2_{\epsilon}$ commands: |  |
| \@                                      | ... 58, 59, 61   |
| \@bsphack                               | ... 26, 348  |
| \@esphack                               | ... 26, 349  |
| \@setupverbinvisiblespace               | ... 1010   |
| \@vobeyspaces                           | ... 1010   |
| tex commands:                           |  |
| \tex_char:D                             | ... 281  |
| \tex_everypar:D                         | ... 972, 973   |
| \tex_ignorespaces:D                     | ... 342  |
| \tex_kern:D                             | ... 280  |
| \tex_lastskip:D                         | ... 325, 337   |
| \tex_let:D                              | ... 300  |
| \tex_lowercase:D                        | ... 299  |
| \tex_newlinechar:D                      | ... 1078, 1172   |
| \tex_par:D                              | ... 950, 958, 964  |
| \tex_scantokens:D                       | ... 219  |
| \tex_spacefactor:D                      | ... 326, 334   |
| \tex_the:D                              | ... 973  |
| \tex_unpenalty:D                        | ... 973  |
| tl commands:                            |  |
| \c_space_tl                             | ... 270, 1389, 1393, 1399, 1403  |
| \tl_clear:N                             | ... 662, 1242  |
| \tl_const:Nn                            | ... 77   |
| \tl_gclear:N                            | ... 593, 1112  |
| \tl_gset:Nn                             | ... 17, 587, 1106  |
| \tl_gset_rescan:Nnn                     | ... 65   |
| \tl_head:n                              | ... 625, 757   |
| \tl_if_blank:nTF                        | ... 379, 444, 485, 528, 635, 687, 700, 1167  |
| \tl_if_blank_p:n                        | ... 706  |
| \tl_if_empty:n                          | ... 221  |
| \tl_if_empty:NTF                        | ... 1149, 1224   |
| \tl_if_empty:nTF                        | ... <u>221</u> , 507, 1272, 1424   |
| \tl_if_empty_p:N                        | ... 648  |
| \tl_if_empty_p:n                        | ... 513, 514   |
| \tl_if_head_is_N_type:nTF               | ... 623, 753, 822  |
| \tl_if_novalue:nTF                      | 605, 1037, 1093, 1141, 1203, 1243  |
| \tl_log:N                               | ... 537, 1148, 1221  |
| \tl_new:N                               | 64, 76, 175, 176, 177, 180, 181, 182, 183, 184, 185, 186, 188, 195, 196, 198, 205                                  |
| \tl_put_right:Nn                        | ... 545, 773   |
| \tl_replace_all:Nnn                     | ... 620, 1050, 1052, 1147, 1219, 1220, 1223  |
| \tl_set:Nn                              | 357, 362, 388, 393, 410, 415, 454, 607, 743, 877, 1030, 1046, 1057, 1087, 1094, 1132, 1144, 1194, 1217, 1236, 1270 |
| \tl_to_str:n                            | ... 885  |
| \tl_use:N                               | ... 712, 728, 838, 847, 969, 1125, 1222  |
| token commands:                         |  |
| \token_if_eq_meaning:NNTF               | ... 763  |
| \token_to_str:N                         | ... 277, 283   |
| \tt                                     | ... 170  |
| \ttfamily                               | ... 169  |
| \tystored                               | ... 6, 29, 30, 42, <u>1130</u>   |
| tystored                                | ... <u>464</u>   |
| <b>U</b>                                |  |
| unknown                                 | ... <u>351</u> , <u>383</u> , <u>406</u> , <u>448</u>  |
| \unprotect                              | ... 14   |

use commands:

\use:N ..... 19

\use:n 565, 677, 738, 800, 835, 844, 925, 1065, 1124, 1157,  
1207, 1265

\UseInstance ..... 1009

**V**

verb-font ..... 137

\verbatim ..... 1015

\verbatimsc ..... 935, 1177

verbatimsc ..... 7, 985

**W**

width-tab ..... 137, 448

wrapper ..... 406

write-cmd ..... 383

write-env ..... 351

write-out ..... 351, 383, 448

\writestatus ..... 13