

SCONTENTS

STORES L^AT_EX CONTENTS

V2.6 2025-11-20*

©2019–2025 by Pablo González L[†]

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

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

Abstract

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

1 Description of the package

The `SCONTENTS` package allows to \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 ConT_EXt “*buffers*” by making use \langle *sequences* \rangle .


2 Motivation and Acknowledgments

In L^AT_EX there is no direct way to record content for later use, although you can do this using `\macro`, recording \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 ConT_EXt which allows you to save content in memory, including *verbatim*, to be used later. The `filecontentsdef`[2] package solves this problem and since `expl3`[1] has an excellent way to manage data, ideas were combined giving rise to this package.

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

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

 Starting with version 2.3 the `SCONTENTS` package is fully compatible with *tagged* PDF and will have L^AT_EX release 2025-06-01 (T_EX Live 2025) and `expl3` 2025-07-08 as a minimum requirement.

*This file describes a documentation for v2.6, last revised 2025-11-20.

[†]E-mail: pablgonz@educarchile.cl.

3 License and Requirements

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

The package `SCONTENTS` requires an updated version of \LaTeX to work (minimum required to \LaTeX release 2025-11-01 and `expl3` 2025-07-08). This package can be used with `plain`, `context`, `xelatex`, `lualatex`, `pdflatex` and the classical workflow `latex»dvips»ps2pdf`.

4 The scontents package

4.1 Installation

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

<code>scontents.tex</code>	»	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 $\text{Con}\text{\TeX}$ t users

```
\usemodule{scontents}
```

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

4.3 The TAB character

Some users use the horizontal TAB ‘`␣`’ from keyboard to indent the source code of the document and depending on the text editor used, some will use real TAB (*hard tabs*), others with *soft tabs* (`_` 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 \langle reading \rangle ¹ and \langle writing \rangle ² are different and may have adverse consequences.

In a standard \LaTeX document, the character TAB ‘`␣`’ are treated as explicit spaces (in most contexts) and is the behavior when \langle stored content \rangle , but when \langle writing files \rangle 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 \langle output file \rangle .

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

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

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

4.4 Configuration of the options

Most of the options can be passed directly to the package or using the command `\setupsc`. All boolean keys can be passed using the equal sign ‘=’ or just the name of the key, all unknown $\langle keys \rangle$ 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 spaces* ‘`␣`’ of the start of the environment.

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

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

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

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

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

Notes for plain T_EX and ConT_EXt users

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

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

ConT_EXt 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. X₃TeX 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 within *braces*. Otherwise it will be assumed that `{(argument)}` is delimited by that first delimiter `` like command `\verb`. Blank lines are preserved, escaped braces ‘\{’ and ‘\}’ must also be balanced if the `{(argument)}` is used with *braces* and character TAB typed from the keyboard are converted into spaces.

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

Options for command

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

```
store-cmd = {(seq name)} default: contents
```

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

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

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

Options only for starred version

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

```
write-cmd = {(file.ext)} default: not used
```

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

```
write-out = {(file.ext)} default: not used
```

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

- In the keys `write-cmd` and `write-out` the character TAB will be written in `(file.ext)`, relative or absolute paths are not supported. X₃TeX users using character TAB must add `-8bit` at the command line, otherwise you will get 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` iterates through and executes the command `\getstored` on the *stored content* in the *sequence* `{<seq name>}`. If the *optional argument* is not passed run `\getstored` on all *stored content* in the *sequence* `{<seq name>}`.

Options for command

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.6 The command `\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. X₃LaTeX 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 <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 allows joining *stored content* *without rewriting* the file over and over again, by design \TeX does not have an *append mode* for writing files, this effectively allows you to write chunks of code and then merge them into a single file.

5.9 The environment `verbatimsc`

```
verbatimsc
```

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

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

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

Using the package `minted`[8]:

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

Using the package `listings`[6]:

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

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

6 Other commands provided

6.1 The command `\countsc`

```
\countsc <seq name>
```

The command `\countsc` counts a number of *stored content* in the sequence `{<seq name>}`.

6.2 The command `\cleanseqsc`

```
\cleanseqsc <cleanseqsc>{<seq name>}
```

The command `\cleanseqsc` removes 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 \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`³ tool to compile them.

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

8.1 From answers package

Example 1

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


```

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

```

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

9 Change history

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

- v2.6 (ctan), 2025-11-20**
 - Internal environment `verbatimsc` update for *tagged* PDF.
 - Update requirement to \LaTeX release 2025-11-01.
- v2.5 (ctan), 2025-11-01**
 - Replacing `\scantokens` (primitive) with `\tl_retokenize:n`.
 - Cleanup warnings and details returned by `expltools`.
 - Update minimum required of `expl3` to 2025-07-08.
- 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.

- v1.0 (ctan), 2019-07-30**
 - Replace `\tex_endinput:D` by `\file_input_stop:`.
 - First public release.

10 Index of Documentation

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

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

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 \LaTeX Project. “The \LaTeX 2 ϵ sources”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2025.
- [11] The \LaTeX Project. “ \LaTeX for authors current version”. Available from CTAN, <https://ctan.org/pkg/latex-base>, 2025.
- [12] FISCHER, ULRIKE. “tagpdf – \LaTeX kernel code for PDF tagging”. Available from CTAN, <https://www.ctan.org/pkg/tagpdf>, 2025.
- [13] The \LaTeX Project. “latex-lab – \LaTeX laboratory”. Available from CTAN, <https://www.ctan.org/pkg/latex-lab>, 2025.
- [14] F. STURM, THOMAS. “tcolorbox – Coloured boxes, for \LaTeX examples and theorems, etc”. Available from CTAN, <https://ctan.org/pkg/tcolorbox>, 2025.
- [15] The \LaTeX Project. “verbatim – Reimplementation of and extensions to \LaTeX verbatim”. Available from CTAN, <https://www.ctan.org/pkg/verbatim>, 2023.
- [16] WRIGHT, JOSEPH. “Programming key–value in expl3”. Available from TUGBOAT, <https://www.tug.org/TUGboat/tb31-1/tb97wright-l3keys.pdf>, 2010.
- [17] WRIGHT, JOSEPH. “answers – Setting questions (or exercises) and answers”. Available from CTAN, <https://ctan.org/pkg/answers>, 2014.

12 Implementation

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

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

12.1 Declaration of the package

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

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

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

```
2 <*loader>
3 \def\ScontentsFileDate{2025-11-20}%
4 \def\ScontentsFileVersion{2.6}%
5 \def\ScontentsFileDescription{Stores LaTeX contents in memory or files}%
```

The \LaTeX loader is quite simple, we just need to make sure of the minimum version for correct operation and then set interfaces up. The choice of \LaTeX release 2025-11-01 is the latest available in \TeX Live 2025 and is necessary to be able to implement the package's full compatibility with *tagged* PDF.

```
6 <*latex>
7 \NeedsTeXFormat{LaTeX2e}[2025-11-01]
8 \ProvidesExplPackage
9   {scontents} {\ScontentsFileDate} {\ScontentsFileVersion} {\ScontentsFileDescription}
10 </latex>
```

12.1.1 Load expl3-generic and xparse-generic in plain \TeX and Con \TeX t

```
\c__scontents_version_str
\l__scontents_char_value_int
```

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 string variable `\c__scontents_version_str` with version info (just in case) and add `\ExplSyntaxOn` to be able to load the frozen `xparse[3]`.

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

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

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

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

12.1.2 Checking expl3 version

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

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

```
36 }
```

12.1.3 Preventing double loading in plain T_EX

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

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

12.1.4 Checking proper loader

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

```
50 ⟨*core⟩
51 \def\ScontentsCoreFileDate{2025-11-20}%
52 \begingroup
53   \catcode32=10
54   \endlinechar=32
55   \def\next{\endgroup}%
56   \expandafter\ifx\csname PackageError\endcsname\relax
57     \begingroup
58       \def\next{\endgroup\endgroup}%
59       \def\PackageError#1#2#3%
60         {%
61           \endgroup
62           \errhelp{#3}%
63           \errmessage{#1 Error: #2!}%
64         }%
65     \fi
66     \expandafter\ifx\csname ScontentsFileDate\endcsname\relax
67       \def\next
68         {%
69           \PackageError{scontents}{No scontents loader detected}
70           {%
71             You have attempted to use the scontents code directly rather than using
72             the correct loader. Loading of scontents will abort.
73           }%
74           \endgroup
75           \endinput
76         }%
77     \else
78       \ifx\ScontentsFileDate\ScontentsCoreFileDate
79         \else
80           \def\next
81             {%
82               \PackageError{scontents}{Mismatched scontents files detected}
83               {%
84                 You have attempted to load scontents with mismatched files:
85                 probably you have one or more files 'locally installed' which
86                 are in conflict. Loading of scontents will abort.
87               }%
88             \endgroup
89             \endinput
90           }%
91         \fi
92       \fi
93     \next
94 ⟨/core⟩
```

12.2 Variables and functions by format

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

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

```

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

```

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

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

```

100 ⟨!context⟩
101 \tl_const:Nc \c__scontents_newline_tl { \iow_char:N ^^J }
102 ⟨!context⟩

```

(End of definition for `\c__scontents_newline_tl`.)

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

```

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

```

Por alguna razón desconocida para mi, no funciona `^^J` bajo LMTX, Max me dio esto en el chat...preguntar que es lo que hace `\syst_obeyed_line` y donde se encuentra documentado. <https://chat.stackexchange.com/tra>

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

```

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

```

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

```

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

```

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

12.3 Loading the package core

Now we load the core `scontents` code:

```
139 \file_input:n { scontents-code.tex }
140 </loader>
```

12.4 Keys for the package

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

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

Set default value for `verb-font` key.

```
173 <loader>\keys_define:nn { scontents }
174 <latex> { verb-font .initial:n = \ttfamily }
175 <plain|context> { verb-font .initial:n = \tt }
```

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

```
176 <*latex>
177 \ProcessKeyOptions [ scontents ]
178 </latex>
```

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

12.5 Internal variables

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

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

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

```
179 <*core>
180 \tl_new:N \l__scontents_save_every_body_lines_tl
181 \tl_new:N \l__scontents_processed_body_lines_tl
```

```

182 \tl_new:N \l__scontents_environment_keys_tl
183 \int_new:N \l__scontents_nesting_env_int
184 \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`.

```

185 \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` (§12.14), `\foreachsc` (§12.16), `\typestored` (§12.17) and `\meaningsc` (§12.18) commands.

```

186 \tl_new:N \l__scontents_Scontents_arg_tl
187 \tl_new:N \l__scontents_foreachsc_arg_tl
188 \tl_new:N \l__scontents_typedstored_arg_tl
189 \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` (§12.19) command.

`\l__scontents_current_seq_name_str` 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 `__scontents_parse_type_meaning_key:n`.

```

190 \tl_new:N \l__scontents_mergesc_arg_tl
191 \tl_new:N \l__scontents_mergesc_keys_tl
192 \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` (§12.7).

`\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 `__scontents_file_if_writable:nTF` (see 12.10.2).

```

193 \tl_new:N \l__scontents_file_name_tl
194 \iow_new:N \l__scontents_file_write_iow
195 \bool_new:N \l__scontents_writing_bool
196 \bool_new:N \l__scontents_storing_bool
197 \bool_set_true:N \l__scontents_storing_bool
198 \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⟩` (§12.8.2) and implementation of `\foreachsc` command (§12.16).

```

199 \seq_new:N \l__scontents_foreach_print_seq
200 \tl_new:N \g__scontents_foreach_exec_tl
201 \tl_new:N \l__scontents_foreach_before_tl
202 \bool_new:N \l__scontents_foreach_before_bool
203 \tl_new:N \l__scontents_foreach_after_tl
204 \bool_new:N \l__scontents_foreach_after_bool
205 \int_new:N \l__scontents_foreach_stop_int
206 \bool_new:N \l__scontents_foreach_stop_bool
207 \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.

```
208 \seq_new:N \l__scontents_seq_item_seq
209 \seq_new:c { g__scontents_name_sc!internal_seq }
```

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

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

```
210 \tl_new:N \g__scontents_last_stored_tl
```

(End of definition for `\g__scontents_last_stored_tl`.)

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

```
211 \str_const:Ne \c__scontents_hidden_space_str
212 { \c_percent_str \c_circumflex_str \c_circumflex_str A scheol \c_percent_str }
```

(End of definition for `\c__scontents_hidden_space_str`.)

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

```
\l__scontents_save_skip
213 \int_new:N \l__scontents_save_sf_int
214 \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
215 \quark_new:N \q__scontents_stop
216 \quark_new:N \q__scontents_mark
217 \scan_new:N \l__scontents_stop
218 \scan_new:N \l__scontents_mark
219 ⟨/core⟩
```

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

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

```
220 ⟨*plain⟩
221 \bool_new:N \l__scontents_plain_bool
222 ⟨/plain⟩
```

(End of definition for `\l__scontents_plain_bool`.)

12.6 Utility functions

`\tl_if_empty:FTF` Some nonstandard kernel variant.

```
\tl_replace_all:NeV
\l__scontents_use_delimit_by_s_stop:nw
223 ⟨*core⟩
224 \prg_generate_conditional_variant:Nnn \tl_if_empty:n { f } { p, TF }
225 \cs_generate_variant:Nn \tl_replace_all:Nnn { NeV }
226 \cs_generate_variant:Nn \tl_retokenize:n { e }
```

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

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

```
\__scontents_use_i_delimit_by_s_stop:nw
\__scontents_use_none_delimit_by_s_stop:w
\__scontents_use_none_delimit_by_q_stop:w
227 \cs_new:Npn \__scontents_use_delimit_by_s_stop:nw #1 \s__scontents_stop {#1}
228 \cs_new:Npn \__scontents_use_i_delimit_by_s_stop:nw #1 #2 \s__scontents_stop {#1}
229 \cs_new:Npn \__scontents_use_none_delimit_by_s_stop:w #1 \s__scontents_stop { }
230 \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`.

```

231 \prg_new_protected_conditional:Npnn \__scontents_tl_if_head_is_q_mark:n #1
232 { T, F, TF }
233 {
234   \exp_after:wN \if_meaning:w
235     \exp_after:wN
236     \q__scontents_mark \__scontents_use_i_delimit_by_s_stop:nw #1 ? \__scontents_stop
237     \prg_return_true:
238   \else:
239     \prg_return_false:
240   \fi:
241 }

```

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

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

```

242 \prg_new_protected_conditional:Npnn \__scontents_file_if_writable:n #1 { T, F, TF }
243 {

```

```

244   \bool_if:NTF \l__scontents_writing_bool
245   {
246     \file_if_exist:nTF {#1}
247     {
248       \bool_if:NTF \l__scontents_overwrite_bool
249       {
250         \msg_warning:nne { scontents } { overwrite-file } {#1}
251         \prg_return_true:
252       }
253       {
254         \msg_warning:nne { scontents } { not-writing } {#1}
255         \prg_return_false:
256       }
257     }
258     {
259       \msg_warning:nne { scontents } { writing-file } {#1}
260       \prg_return_true:
261     }
262   }
263   { \prg_return_false: }
264 }

```

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

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

```

265 \cs_new_protected:Npn \__scontents_file_write_cmd:nn #1#2
266 {
267   \__scontents_file_if_writable:nT {#1}
268   {
269     \iow_open:Nn \l__scontents_file_write_iow {#1}
270     \iow_now:Nn \l__scontents_file_write_iow {#2}
271     \iow_close:N \l__scontents_file_write_iow
272   }
273 }
274 \cs_generate_variant:Nn \__scontents_file_write_cmd:nn { VV }

```

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

12.6.1 Functions for TAB and verbatimsc

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

```

\__scontents_par:
275 \cs_new:Nc \__scontents_tab: { \c_space_tl }
276 \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.

```

277 \cs_new:Nn \__scontents_tabs_to_spaces:
278   { \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 \TeX still looking for an *optional space*.

```

279 \cs_new_protected:Npn \__scontents_do_noligs:N #1
280   {
281     \char_set_catcode_active:N #1
282     \cs_set:cpe { __scontents_active_char_ \token_to_str:N #1 : }
283     {
284       \mode_leave_vertical:
285       \tex_kern:D \c_zero_dim
286       \tex_char:D \exp_not:N #1
287     }
288     \char_set_active_eq:Nc #1 { __scontents_active_char_ \token_to_str:N #1 : }
289   }

```

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

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

```

290 \cs_new_protected:Npn \__scontents_set_active_eq:NN #1
291   {
292     \char_set_catcode_active:N #1
293     \char_set_active_eq:NN #1
294   }
295 </core>
296 <*loader>
297 \group_begin:
298 <plain> \char_set_catcode_active:n { \* }
299 \cs_new_protected:Nn \__scontents_plain_disable_outer_par:
300 <*plain>
301   {
302     \group_begin:
303     \char_set_lccode:nn { \* } { ^^L }
304     \tex_lowercase:D { \group_end:
305       \tex_let:D * \scan_stop:
306     }
307   }
308 </plain>
309 <latex|context> { }
310 \group_end:
311 </loader>
312 <*core>
313 \group_begin:
314 \char_set_catcode_active:N \*
315 \cs_new_protected:Nn \__scontents_make_control_chars_active:
316   {
317     \__scontents_plain_disable_outer_par:
318     \__scontents_set_active_eq:NN ^^I \__scontents_tab:
319     \__scontents_set_active_eq:NN ^^L \__scontents_par:
320     \__scontents_set_active_eq:NN ^^M \__scontents_ret:w
321   }
322 \group_end:
323 </core>

```

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

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

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

```

324 <*core>
325 \cs_new_protected:Nn \__scontents_bsphack:
326   {

```

```

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

```

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

12.7 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
356 <core>
357 \keys_define:nn { scontents / scontents }
358 {
359     write-env .code:n          = {
360         \bool_set_true: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-out .code:n         = {
365         \bool_set_false:N \l__scontents_storing_bool
366         \bool_set_true:N \l__scontents_writing_bool
367         \tl_set:Nn \l__scontents_file_name_tl {#1}
368     },
369     write-env .value_required:n = true,
370     write-out .value_required:n = true,
371     print-env .meta:nn        = { scontents } { print-env = #1 },
372     print-env .default:n      = true,
373     store-env .meta:nn        = { scontents } { store-env = #1 },
374     force-eol .meta:nn        = { scontents } { force-eol = #1 },
375     force-eol .default:n      = true,
376     overwrite .meta:nn        = { scontents } { overwrite = #1 },
377     overwrite .default:n      = true,
378     unknown .code:n           = { \l__scontents_unknown_keys_env:n {#1} },
379 }

```

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

12.7.1 Handling unknown keys for environment `scontents`

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

`\l__scontents_unknown_keys_env:n` 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.

```

380 \cs_new_protected:Npn \l__scontents_unknown_keys_env:n #1

```

```

381 { \exp_args:NV \__scontents_unknown_keys_env:nn \l_keys_key_str {#1} }
382 \cs_new_protected:Npn \__scontents_unknown_keys_env:nn #1#2
383 {
384   \tl_if_blank:nTF {#2}
385     { \msg_error:nnn { scontents } { env-key-unknown } {#1} }
386     { \msg_error:nnnn { scontents } { env-key-value-unknown } {#1} {#2} }
387 }

```

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

12.8 Keys for commands

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

12.8.1 Keys for command `\Scontents`

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

```

write-cmd We define a set of <keys> for commands \Scontents and \Scontents*.
write-out
print-cmd
store-cmd
force-eol
overwrite
unknown
388 \keys_define:nn { scontents / Scontents }
389 {
390   write-cmd .code:n          = {
391     \bool_set_true:N \__scontents_storing_bool
392     \bool_set_true:N \__scontents_writing_bool
393     \tl_set:Nn \__scontents_file_name_tl {#1}
394   },
395   write-out .code:n         = {
396     \bool_set_false:N \__scontents_storing_bool
397     \bool_set_true:N \__scontents_writing_bool
398     \tl_set:Nn \__scontents_file_name_tl {#1}
399   },
400   write-cmd .value_required:n = true,
401   write-out .value_required:n = true,
402   print-cmd .meta:nn        = { scontents } { print-cmd = #1 },
403   print-cmd .default:n      = true,
404   store-cmd .meta:nn        = { scontents } { store-cmd = #1 },
405   force-eol .meta:nn        = { scontents } { force-eol = #1 },
406   force-eol .default:n      = true,
407   overwrite .meta:nn        = { scontents } { overwrite = #1 },
408   overwrite .default:n      = true,
409   unknown .code:n           = { \__scontents_unknown_keys_cmd:n {#1} },
410 }

```

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

12.8.2 Keys for command `\foreachsc`

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

```

before We define a set of <keys> for command \foreachsc.
after
start
stop
step
wrapper
sep
unknown
411 \keys_define:nn { scontents / foreachsc }
412 {
413   before .code:n          = {
414     \bool_set_true:N \__scontents_foreach_before_bool
415     \tl_set:Nn \__scontents_foreach_before_tl {#1}
416   },
417   before .value_required:n = true,
418   after .code:n           = {
419     \bool_set_true:N \__scontents_foreach_after_bool
420     \tl_set:Nn \__scontents_foreach_after_tl {#1}
421   },
422   after .value_required:n = true,
423   start .int_set:N        = \__scontents_foreach_start_int,
424   start .value_required:n = true,
425   start .initial:n        = 1,
426   stop .code:n            = {
427     \bool_set_true:N \__scontents_foreach_stop_bool
428     \int_set:Nn \__scontents_foreach_stop_int {#1}
429   },
430   stop .value_required:n = true,
431   step .int_set:N         = \__scontents_foreach_step_int,
432   step .value_required:n = true,
433   step .initial:n         = 1,
434   wrapper .code:n         = {
435     \bool_set_true:N \__scontents_foreach_wrapper_bool

```

```

436         \cs_set_protected:Npn
437         \__scontents_foreach_wrapper:n #1 {#1}
438         },
439     wrapper .value_required:n = true,
440     sep     .tl_set:N         = \__scontents_foreach_sep_tl,
441     sep     .initial:n       = { },
442     sep     .value_required:n = true,
443     unknown .code:n         = { \__scontents_unknown_keys_cmd:n {#1} },
444 }

```

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

12.8.3 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.

```

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

```

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

12.8.4 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 (§12.19).

```

453 \keys_define:nn { scontents / typemeaning }
454 {
455     width-tab .meta:nn     = { scontents } { width-tab = #1 },
456     write-out .code:n      = {
457         \bool_set_false:N \__scontents_storing_bool
458         \bool_set_true:N  \__scontents_writing_bool
459         \tl_set:Nn \__scontents_file_name_tl {#1}
460     },
461     write-out .value_required:n = true,
462     overwrite .meta:nn        = { scontents } { overwrite = #1 },
463     overwrite .default:n     = true,
464     print-cmd .bool_set:N    = \__scontents_print_verb_style_bool,
465     print-cmd .initial:n     = true,
466     print-cmd .default:n     = true,
467     unknown   .code:n        = { \__scontents_parse_type_meaning_key:n {#1} },
468 }

```

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

12.8.5 Keys for command `\mergesc`

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

```

469 \keys_define:nn { scontents / mergesc }
470 {
471     typestored .code:n =
472     {
473         \cs_set_eq:NN \__scontents_mergesc_cmd:nn \__scontents_typestored:nn
474     },
475     typestored .value_forbidden:n = true,
476     meanings   .code:n =
477     {
478         \cs_set_eq:NN \__scontents_mergesc_cmd:nn \__scontents_meaningsc:nn
479     },
480     meanings   .value_forbidden:n = true,
481 }
482 \cs_new_protected:Npn \__scontents_mergesc_cmd:nn #1 #2
483 {
484     \msg_error:nn { scontents } { mergesc-missing-key }

```

```
485 }
```

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

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

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

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

To avoid the awkward `\typestored[...][\langle keys \rangle]{...}` syntax, we'll make the commands have a single *optional argument* which is processed by `!3keys[9]`, and the *unknown* $\langle keys \rangle$ are brought here to `__scontents_parse_type_meaning_key:n` to process.

First we check if the $\langle key \rangle$ is an integer using `\int_to_roman:n`. If it is, we check that the value passed to the $\langle key \rangle$ 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 $\langle index \rangle$, otherwise raise an error about an *unknown* option.

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

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

12.9 Functions for sequences

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

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

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

```

530 (*core)
531 \cs_new_protected:Npn \__scontents_append_contents:nn #1#2
532 {
533   \tl_if_blank:nT {#1}
534     { \msg_error:nn { scontents } { empty-store-cmd } }
535   \seq_if_exist:cF { g__scontents_name_#1_seq }
536     { \seq_new:c { g__scontents_name_#1_seq } }
537   \seq_gput_right:cn { g__scontents_name_#1_seq } {#2}
538 }
539 \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`.

```

540 \cs_new_protected:Npn \__scontents_store_to_seq:NN #1#2
541 {
542   \tl_log:N #1
543   \__scontents_append_contents:Ve #2 { \exp_not:V #1 }
544 }

```

(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.

```

545 \cs_new_protected:Npn \__scontents_finish_storing:NNN #1 #2 #3
546 {
547   \bool_if:NT \l__scontents_storing_bool
548     {
549       \bool_if:NF \l__scontents_forced_eol_bool
550         { \tl_put_right:Ne #1 { \c__scontents_hidden_space_str } }
551       \__scontents_store_to_seq:NN #1 #2
552       \bool_if:NT #3 { \__scontents_lastfrom_seq:V #2 }
553     }
554 }

```

(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
555 \cs_new:Npn \__scontents_getfrom_seq:Nn #1#2
556 {
557   \seq_if_exist:cTF { g__scontents_name_#2_seq }
558     {
559       \exp_args:Nf \__scontents_getfrom_seq:nNn
560         { \seq_count:c { g__scontents_name_#2_seq } } #1 {#2}
561     }
562     { \msg_expandable_error:nnn { scontents } { undefined-storage } {#2} }
563 }
564 \cs_new:Npn \__scontents_getfrom_seq:nNn #1 #2 #3
565 {
566   \seq_map_tokens:Nn #2 { \__scontents_getfrom_seq_aux:nnn {#1} {#3} }
567 }
568 \cs_new:Npn \__scontents_getfrom_seq_aux:nnn #1 #2 #3
569 {
570   \exp_args:Nnf \use:n { \__scontents_getfrom_seq:nnn {#1} } { \int_eval:n {#3} } {#2}
571 }
572 \cs_new:Npn \__scontents_getfrom_seq:nn #1#2
573 {
574   \seq_if_exist:cTF { g__scontents_name_#2_seq }
575     {

```



```

576     \exp_args:Nf \__scontents_getfrom_seq:nnn
577     { \seq_count:c { g__scontents_name_#2_seq } }
578     {#1} {#2}
579   }
580   { \msg_expandable_error:nnn { scontents } { undefined-storage } {#2} }
581 }
582 \cs_new:Npn \__scontents_getfrom_seq:nnn #1#2#3
583 {
584   \bool_lazy_or:nnTF
585   { \int_compare_p:nNn {#2} = { 0 } }
586   { \int_compare_p:nNn { \int_abs:n {#2} } > {#1} }
587   { \msg_expandable_error:nnnn { scontents } { index-out-of-range } {#2} {#3} {#1} }
588   { \seq_item:cn { g__scontents_name_#3_seq } {#2} }
589 }

```

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

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

```

590 \cs_new_protected:Npn \__scontents_lastfrom_seq:n #1
591 {
592   \tl_gset:Nx \g__scontents_last_stored_tl
593   {
594     \seq_item:cn { g__scontents_name_#1_seq } { -1 }
595   }
596   \group_insert_after:N \tl_retokenize:V
597   \group_insert_after:N \g__scontents_last_stored_tl
598   \group_insert_after:N \tl_gclear:N
599   \group_insert_after:N \g__scontents_last_stored_tl
600 }
601 \cs_generate_variant:Nn \__scontents_lastfrom_seq:n { V }
602 </core>

```

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

12.10 Base functions for environments

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

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

12.10.1 Functions for keyval of environment

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

```

603 <*core>
604 \NewDocumentCommand \__scontents_grab_opt_arg:w { r[] }
605 {
606   \__scontents_grab_opt_arg:n {#1}
607 }

```

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.

```

608 \cs_new_protected:Npn \__scontents_grab_opt_arg:n #1
609 {
610   \tl_if_novalue:nF {#1}
611   {
612     \tl_set:Nn \l__scontents_environment_keys_tl {#1}
613     \__scontents_normalise_line_ends:N \l__scontents_environment_keys_tl
614     \keys_set:nV { scontents / scontents } \l__scontents_environment_keys_tl
615   }
616   \__scontents_start_after_option:w
617 }
618 </core>

```

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

12.10.2 Functions for save the body of environment

```

__scontents_start_environment:w
__scontents_start_after_option:w
__scontents_check_line_process:en
__scontents_stop_environment:

```

Here we make ‘`^^I`’, ‘`^^L`’ and ‘`^^M`’ active characters so that the end of line can be “seen” to be used as a delimiter, and \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.

```

619 <*core>
620 \group_begin:
621   \char_set_catcode_active:N ^^I
622   \char_set_catcode_active:N ^^L
623   \char_set_catcode_active:N ^^M
624   \cs_new_protected:Npn __scontents_normalise_line_ends:N #1
625     { \tl_replace_all:Nnn #1 { ^^M } { ~ } }
626   \cs_new_protected:Npn __scontents_start_environment:w #1 ^^M
627     {
628       \tl_if_head_is_N_type:nTF {#1}
629         {
630           \str_if_eq:eeTF { \tl_head:n {#1} } { [ ]
631             { __scontents_grab_opt_arg:w #1 ^^M }
632             { __scontents_check_line_process:en { } {#1} }
633           }
634           { __scontents_check_line_process:en { } {#1} }
635         }
636   \cs_new_protected:Npn __scontents_start_after_option:w #1 ^^M
637     { __scontents_check_line_process:en { [...] } {#1} }
638   \cs_new_protected:Npn __scontents_check_line_process:en #1 #2
639     {
640       \tl_if_blank:nF {#2}
641       {
642         \msg_error:nnee { scontents } { junk-after-begin }
643         { after~\c_backslash_str begin { \__scontents_env_name_tl } #1 } {#2}
644       }
645       \__scontents_make_control_chars_active:
646       \__scontents_file_tl_write_start:V \__scontents_file_name_tl
647     }
648   \cs_new_protected:Nn __scontents_stop_environment:
649     {
650       \__scontents_file_write_stop:N \__scontents_processed_body_lines_tl
651       \bool_lazy_and:nnT
652         { \l__scontents_storing_bool }
653         { \tl_if_empty_p:N \__scontents_processed_body_lines_tl }
654       {
655         \msg_warning:nne { scontents } { empty-environment } { \__scontents_env_name_tl }
656       }
657     }

```

(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.

```

658   \cs_new_protected:Npn __scontents_file_tl_write_start:n #1
659     {
660       \group_begin:

```

```

661     \__scontents_file_if_writable:nTF {#1}
662     {
663         \bool_set_true:N \__scontents_writable_bool
664         \iow_open:Nn \__scontents_file_write_iow {#1}
665     }
666     { \bool_set_false:N \__scontents_writable_bool }
667 \tl_clear:N \__scontents_save_every_body_lines_tl
668 \seq_map_function:NN \l_char_special_seq \char_set_catcode_other:N
669 \int_step_function:nnnN { 128 } { 1 } { 255 } \char_set_catcode_letter:n
670 \cs_set_protected:Npe \__scontents_ret:w ##1 ^^M
671     {
672         \exp_not:N \__scontents_verb_processor_iterate:w
673         ##1 \c__scontents_end_env_tl
674         \c__scontents_end_env_tl
675         \exp_not:N \q__scontents_stop
676     }
677 \__scontents_make_control_chars_active:
678 \__scontents_ret:w
679 }
680 \cs_new:Nn \__scontents_setup_verb_processor:
681 {
682     \use:e
683     {
684         \cs_set:Npn \exp_not:N \__scontents_verb_processor_iterate:w
685         ##1 \c__scontents_end_env_tl
686         ##2 \c__scontents_end_env_tl
687         ##3 \exp_not:N \q__scontents_stop
688     } { \__scontents_verb_processor_iterate:nnn {##1} {##2} {##3} }
689 }
690 \cs_new:Npn \__scontents_verb_processor_iterate:nnn #1 #2 #3
691 {
692     \tl_if_blank:nTF {#3}
693     {
694         \__scontents_analyse_nesting:n {#1}
695         \__scontents_verb_processor_output:n {#1}
696     }
697     {
698         \__scontents_if_nested:TF
699         {
700             \__scontents_nesting_decr:
701             \__scontents_verb_processor_output:e
702             { \exp_not:n {#1} \c__scontents_end_env_tl \exp_not:n {#2} }
703         }
704         {
705             \tl_if_blank:nF {#1}
706             { \__scontents_verb_processor_output:n {#1} }
707             \cs_set_protected:Npe \__scontents_ret:w
708             {
709                 \__scontents_env_end_function:
710                 \bool_lazy_or:nnF
711                 { \tl_if_blank_p:n {#2} }
712                 { \str_if_eq_p:ee {#2} { \c_percent_str } }
713                 {
714                     \str_if_eq:VnF \c__scontents_hidden_space_str {#2}
715                     {
716                         \msg_warning:nnnn { scontents } { rescanning-text }
717                         {#2} { \tl_use:N \__scontents_env_name_tl }
718                     }
719                     \tl_retokenize:n {#2}
720                 }
721             }
722             \char_set_active_eq:NN ^^M \__scontents_ret:w
723         }
724     }
725     ^^M
726 }
727 \cs_new:Nn \__scontents_env_end_function:
728 {
729     \__scontents_format_case:nnn
730     { \exp_not:N \end { \if_false: } \fi: }
731     { \exp_after:wN \exp_not:N \cs:w end }

```

```

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

```

773 \cs_new_protected:Npn \__scontents_verb_processor_output:n #1
774 {
775   \bool_if:NT \l__scontents_writable_bool
776   { \iow_now:Nn \l__scontents_file_write_iow {#1} }
777   \bool_if:NT \l__scontents_storing_bool
778   { \tl_put_right:Nn \l__scontents_save_every_body_lines_tl { ^^J #1 } }
779 }
780 \group_end:
781 \cs_generate_variant:Nn \__scontents_verb_processor_output:n { e }
782 \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.

```

783 \cs_new_protected:Npn \__scontents_analyse_nesting:n #1
784 {
785   \int_zero:N \l__scontents_nesting_aux_int
786   \__scontents_analyse_nesting_format:n {#1}
787   \int_compare:nNnT { \l__scontents_nesting_aux_int } > { 1 }

```

```

788     { \msg_warning:nn { scontents } { multiple-begin } }
789   }
790 \cs_new_protected:Nn \__scontents_nesting_incr:
791   {
792     \int_incr:N \__scontents_nesting_env_int
793     \int_incr:N \__scontents_nesting_aux_int
794   }
795 \cs_new_protected:Nn \__scontents_nesting_decr:
796   {
797     \int_decr:N \__scontents_nesting_env_int
798   }
799 \prg_new_protected_conditional:Npnn \__scontents_if_nested: { TF }
800   {
801     \int_compare:nNnTF { \__scontents_nesting_env_int } > { \c_zero_int }
802     { \prg_return_true: }
803     { \prg_return_false: }
804   }

```

- 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.

```

805 \use:e
806   {
807     \cs_new_protected:Npn \exp_not:N \__scontents_analyse_nesting_latex:w #1
808     { \c_backslash_str begin \c_left_brace_str #2 \c_right_brace_str
809     } {
810       \__scontents_tl_if_head_is_q_mark:nTF {#2}
811       { \__scontents_use_none_delimit_by_q_stop:w }
812       {
813         \str_if_eq:VnT \__scontents_env_name_tl {#2}
814         { \__scontents_nesting_incr: }
815         \__scontents_analyse_nesting_latex:w
816       }
817     }
818 \cs_new_protected:Npe \__scontents_analyse_nesting_latex:n #1
819   {
820     \__scontents_analyse_nesting_latex:w #1
821     \c_backslash_str begin
822     \c_left_brace_str \exp_not:N \q__scontents_mark \c_right_brace_str
823     \exp_not:N \q__scontents_stop
824   }

```

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`).

```

825 \cs_new_protected:Npn \__scontents_analyse_nesting_generic_process:nn #1 #2
826   {
827     \tl_if_head_is_N_type:nTF {#2}
828     {
829       \__scontents_tl_if_head_is_q_mark:nF {#2}
830       {
831         \__scontents_nesting_incr:
832         \__scontents_analyse_nesting_generic:w #2 \q__scontents_stop
833       }
834     }
835     { \__scontents_analyse_nesting_generic:w #2 \q__scontents_stop }
836   }
837 \cs_new_protected:Npn \__scontents_analyse_nesting_generic:nn #1 #2
838   {
839     \__scontents_define_generic_nesting_function:n {#1}
840     \use:e
841     {
842       \exp_not:N \__scontents_analyse_nesting_generic:w #2
843       \c_backslash_str #1 \tl_use:N \__scontents_env_name_tl
844       \exp_not:N \q__scontents_mark \exp_not:N \q__scontents_stop
845     }
846   }
847 \cs_new_protected:Npn \__scontents_define_generic_nesting_function:n #1
848   {
849     \use:e
850     {

```

```

851     \cs_set_protected:Npn \exp_not:N \__scontents_analyse_nesting_generic:w ##1
852     \c_backslash_str #1 \tl_use:N \l__scontents_env_name_tl
853     ##2 \exp_not:N \q__scontents_stop
854   } { \__scontents_analyse_nesting_generic_process:nn {##1} {##2} }
855 }
856 </core>

```

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

```

857 <*loader>
858 <latex>\cs_new_eq:NN \__scontents_analyse_nesting_format:n
859 <latex> \__scontents_analyse_nesting_latex:n
860 <latex>\cs_new_protected:Npn \__scontents_analyse_nesting_format:n
861 <plain> { \__scontents_analyse_nesting_generic:nn { } }
862 <context> { \__scontents_analyse_nesting_generic:nn { start } }
863 </loader>

```

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

12.11 The command `\newenvsc`

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

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

```

864 <*core>
865 \cs_new_protected:Npn \__scontents_generic_begin:
866 {
867   \char_set_catcode_active:N ^^M
868   \__scontents_start_environment:w
869 }

```

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.

```

870 \cs_new_protected:Npn \__scontents_generic_end:
871 {
872   \__scontents_stop_environment:
873   \__scontents_finish_storing:NNN
874   \l__scontents_processed_body_lines_tl
875   \l__scontents_name_seq_env_tl
876   \l__scontents_print_env_bool
877 }

```

(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.

```

878 \cs_new_protected:Npn \__scontents_setting_env:nn #1 #2
879 {
880   \cs_new_protected:cpn { __scontents_#1_begin: }
881   {
882     \tl_set:Nn \l__scontents_env_name_tl {#1}
883     \keys_set:nn { scontents } {#2}
884     \__scontents_setup_verb_processor:
885     \__scontents_generic_begin:
886   }
887   \cs_new_protected:cpn { __scontents_#1_end: }
888   { \__scontents_generic_end: }
889   \exp_args:Nooo \__scontents_define_env:nnn % http://nooooooooooooooooo.com :) jeje
890   { \tl_to_str:n {#1} }
891   { \cs:w __scontents_#1_begin: \cs_end: }
892   { \cs:w __scontents_#1_end: \cs_end: }
893 }
894 </core>

```

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

```

895 <*loader>
896 \cs_new_protected:Npn \_scontents_define_env:nnn #1 #2 #3
897   {
898   <latex|plain>   \NewDocumentEnvironment {#1} { }
899   <context>      \cs_new_protected:cpn { start #1 }
900     {
901   <!latex>       \group_begin:
902                 #2
903     }
904   <context>      \cs_new_protected:cpn { stop #1 }
905     {
906                 #3
907   <!latex>       \group_end:
908     }
909   }

```

(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 .

```

910 \NewDocumentCommand \newenvsc { m O{} }
911   {
912   <latex|plain>   \cs_if_exist:cTF { #1 }
913   <context>      \cs_if_exist:cTF { start #1 }
914     { \msg_error:nnn { scontents } { env-already-defined } {#1} }
915     { \_scontents_setting_env:nn {#1} {#2} }
916   }
917 </loader>

```

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

12.12 The environment scontents

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

```

\contents
\endscontents
\startcontents
\stopcontents

```

```

918 <*loader>
919 \newenvsc{scontents}
920 </loader>

```

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

12.13 The environment verbatimsc

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

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

```

921 <!latex>
922 \cs_if_exist:NF \dospecials
923   {
924     \cs_new:Npn \dospecials
925       { \seq_map_function:NN \l_char_special_seq \do }
926   }
927 </!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

```

```

928 <*loader>
929 <*!context>
930 \use:e
931   {
932     \cs_new_protected:Npn \exp_not:N \_scontents_xverb:w
933       #1 \g_--scontents_end_verbatimsc_tl
934   <latex>     { #1 \exp_not:N \end{verbatimsc} }

```

```

935 ⟨plain⟩      { #1 \exp_not:N \endverbatimsc }
936 ⟨context⟩    { #1 \exp_not:N \stopverbatimsc }
937 }
938 ⟨!context⟩
939 ⟨/loader⟩

```

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

12.13.1 plain T_EX version off `verbatimsc`

```

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

```

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

12.13.2 ConT_EXt version off `verbatimsc`

```

\startverbatimsc In ConTEXt we use our own tool \definetyping.
\stopverbatimsc
987 ⟨*loader⟩
988 ⟨context⟩\definetyping[verbatimsc]
989 ⟨/loader⟩

```

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

12.13.3 L^AT_EX version off verbatimsc

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

`verbatimsc`

```

990 <*loader>
991 <*latex>
992 \cs_new_protected:Nn __scontents_verbatimsc_instance:
993 {
994   \DeclareInstance{blockenv}{verbatimsc}{display}
995   {
996     name           = verbatimsc,
997     tag-name       = verbatim,
998     tagging-recipe = standard,
999     increment-level = false,
1000    tagging-suppress-paras = true,
1001    setup-code      = ,
1002    block-instance  = verbatimblock ,
1003    para-instance   = justify ,
1004    inner-instance  = ,
1005    final-code      = \legacyverbatimsetup,%\tag_tool:n {paratag=codeline},
1006  }
1007 }
1008 \NewDocumentEnvironment { verbatimsc } { }
1009 {
1010   \IfDocumentMetadataTF
1011   {
1012     __scontents_verbatimsc_instance:
1013     \UseInstance{blockenv}{verbatimsc}{}
1014     \@setupverbinvisiblespace\frenchspacing\@vobeyspaces
1015     __scontents_xverb:
1016   }
1017   {
1018     \cs_set_eq:cN { @xverbatim } __scontents_xverb:
1019     \verbatim
1020   }
1021 }

```

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

```

1022 {
1023   \IfDocumentMetadataTF
1024   {
1025     \endblockenv
1026   }
1027   { \endverbatim }
1028 }
1029 </latex>
1030 </loader>

```

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

12.14 The command `\Scontents`

`\Scontents`

`__scontents_Scontents_code:nn`

`__scontents_Scontents_norm_arg:n`

`__scontents_Scontents_verb_arg:w`

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 `@sircusa` in *Convert a macro from Latexze to expl3*.

```

1031 <*core>
1032 \NewDocumentCommand \Scontents { !s !0{} }
1033 {
1034   \tl_set:Nn \__scontents_cmd_name_tl { Scontents }
1035   __scontents_Scontents_code:nn {#1} {#2}
1036 }

```

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

```

1037 \cs_new_protected:Npn __scontents_Scontents_code:nn #1 #2
1038 {

```

```

1039   \_scontents_espback:
1040   \group_begin:
1041     \tl_if_novalue:nF {#2}
1042     { \keys_set:nn { scontents / Scontents } {#2} }
1043     \char_set_catcode_active:n { 9 }
1044     \bool_if:NTF #1
1045     { \_scontents_Scontents_verb_arg:w }
1046     { \_scontents_Scontents_norm_arg:n }
1047   }

```

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 `_scontents_Scontents_arg_tl` and replaces all `'^^M'` and `\obeyedline` added in L^AT_EX release 2024-06-01 by `'^^J'` and then calls the function `_scontents_Scontents_finish:`. Here we will apply `\RenewDocumentCommand` since `\obeyedline` can be modified by the user and if so the code would return a low-level error.

```

1048 \NewDocumentCommand \_scontents_Scontents_verb_arg:w { +v }
1049 {
1050   \tl_set:Nn \_scontents_Scontents_arg_tl {#1}
1051   \tl_if_empty:NT \_enumext_ref_key_arg_tl
1052   {
1053     \msg_warning:nne { scontents } { empty-argument-command } { \_scontents_Scontents_arg_tl }
1054   }
1055   \cs_if_exist:NT \obeyedline
1056   {
1057     \RenewDocumentCommand \obeyedline { } { \iow_char:N ^^M }
1058     \tl_replace_all:Nee \_scontents_Scontents_arg_tl { \obeyedline } { \iow_char:N ^^M }
1059   }
1060   \tl_replace_all:NeV \_scontents_Scontents_arg_tl { \iow_char:N ^^M } \c_scontents_newline_tl
1061   \_scontents_Scontents_finish:
1062 }

```

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:`.

```

1063 \cs_new_protected:Npn \_scontents_Scontents_norm_arg:n #1
1064 {
1065   \tl_set:Nn \_scontents_Scontents_arg_tl {#1}
1066   \_scontents_Scontents_finish:
1067 }

```

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_espback:` if the `print-cmd` key is not active.

```

1068 \cs_new_protected:Nn \_scontents_Scontents_finish:
1069 {
1070   \_scontents_file_write_cmd:VV \_scontents_file_name_tl \_scontents_Scontents_arg_tl
1071   \_scontents_finish_storing:NNN
1072   \_scontents_Scontents_arg_tl \_scontents_name_seq_cmd_tl \_scontents_print_cmd_bool
1073   \use:e
1074   {
1075     \group_end:
1076     \bool_if:NF \_scontents_print_cmd_bool { \_scontents_espback: }
1077   }
1078 }

```

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

12.15 The command `\getstored`

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

```

1079 \NewDocumentCommand \getstored { 0{-1} m }
1080 {
1081   \_scontents_getstored:nn {#1} {#2}
1082 }

```

The internal function `_scontents_getstored:nn` will set the end of line then apply the function `\tl_retokenize: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`.

```

1083 \cs_new_protected:Npn \__scontents_getstored:nn #1 #2
1084 {
1085   \tl_retokenize:e
1086   {
1087     \__scontents_getfrom_seq:nn {#1} {#2}
1088   }
1089 }

```

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

12.16 The command `\foreachsc`

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

```

\__scontents_foreachsc:nn
\__scontents_foreach_add_body:n
1090 \NewDocumentCommand \foreachsc { o m }
1091 {
1092   \tl_set:Nn \__scontents_cmd_name_tl { foreachsc }
1093   \__scontents_foreachsc:nn {#1} {#2}
1094 }
1095 \cs_new_protected:Npn \__scontents_foreachsc:nn #1 #2
1096 {
1097   \group_begin:
1098   \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / foreachsc } {#1} }
1099   \tl_set:Nn \__scontents_foreachsc_arg_tl {#2}
1100   \seq_clear:N \__scontents_foreach_print_seq
1101   \bool_if:NF \__scontents_foreach_stop_bool
1102   {
1103     \int_set:Nn \__scontents_foreach_stop_int
1104     { \seq_count:c { g__scontents_name_#2_seq } }
1105   }
1106   \int_step_function:nnnN
1107   { \__scontents_foreach_start_int }
1108   { \__scontents_foreach_step_int }
1109   { \__scontents_foreach_stop_int }
1110   \__scontents_foreach_add_body:n
1111   \tl_gset:Ne \g__scontents_foreach_exec_tl
1112   {
1113     \exp_args:NNV \seq_use:Nn
1114     \__scontents_foreach_print_seq \__scontents_foreach_sep_tl
1115   }
1116   \group_end:
1117   \exp_after:wN \tl_gclear:N
1118   \exp_after:wN \g__scontents_foreach_exec_tl
1119   \g__scontents_foreach_exec_tl
1120 }
1121 \cs_new_protected:Npn \__scontents_foreach_add_body:n #1
1122 {
1123   \seq_put_right:Ne \__scontents_foreach_print_seq
1124   {
1125     \bool_if:NT \__scontents_foreach_before_bool
1126     { \exp_not:V \__scontents_foreach_before_tl }
1127     \bool_if:NTF \__scontents_foreach_wrapper_bool
1128     { \__scontents_foreach_wrapper:n }
1129     { \use:n }
1130     { \getstored [#1] { \tl_use:N \__scontents_foreachsc_arg_tl } }
1131     \bool_if:NT \__scontents_foreach_after_bool
1132     { \exp_not:V \__scontents_foreach_after_tl }
1133   }
1134 }

```

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

12.17 The command `\tpestored`

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

```

\__scontents_tpestored:nn
\__scontents_tpestored:N
\__scontents_xverb:w
1135 \NewDocumentCommand \tpestored { o m }
1136 {
1137   \tl_set:Nn \__scontents_cmd_name_tl { tpestored }

```

```

1138   \__scontents_tpestored:nn {#1} {#2}
1139 }
1140 \cs_new_protected:Npn \__scontents_tpestored:nn #1 #2
1141 {
1142   \__scontents_bspack:
1143   \group_begin:
1144     \seq_clear:N \__scontents_seq_item_seq
1145     \str_set:Ne \__scontents_current_seq_name_str {#2}
1146     \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / typemeaning } {#1} }
1147     \seq_if_empty:NT \__scontents_seq_item_seq
1148       { \seq_set_from_clist:Nn \__scontents_seq_item_seq { 1 } }
1149     \tl_set:Ne \__scontents_tpestored_arg_tl
1150       { \__scontents_getfrom_seq:Nn \__scontents_seq_item_seq {#2} }
1151     \__scontents_remove_trailing_eol:N \__scontents_tpestored_arg_tl
1152     \tl_replace_all:NeV \__scontents_tpestored_arg_tl { \c__scontents_hidden_space_str } \c__scontents_hidden_space_str
1153     \tl_log:N \__scontents_tpestored_arg_tl
1154     \tl_if_empty:NF \__scontents_tpestored_arg_tl
1155       {
1156         \bool_if:NT \__scontents_print_verb_style_bool
1157           {
1158             \__scontents_tpestored:N \__scontents_tpestored_arg_tl
1159           }
1160       }
1161     \__scontents_file_write_cmd:VV \__scontents_file_name_tl \__scontents_tpestored_arg_tl
1162     \use:e
1163     {
1164   \group_end:
1165   \bool_if:NF \__scontents_print_verb_style_bool { \__scontents_espack: }
1166   }
1167 }

```

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 `\tl_retokenize:e`.

```

1168 \group_begin:
1169   \char_set_catcode_active:N ^^M
1170   \cs_new_protected:Npn \__scontents_tpestored:N #1
1171   {
1172     \tl_if_blank:VT #1
1173       { \msg_error:nnn { scontents } { empty-variable } {#1} }
1174     \cs_set_eq:NN \__scontents_verb_print_EOL: ^^M
1175     \cs_set_eq:NN ^^M \scan_stop:
1176     \cs_set_eq:cN { do@noligs } \__scontents_do_noligs:N
1177     \int_set:Nn \tex_newlinechar:D { `^^J }
1178     \tl_retokenize:e
1179     {
1180       \__scontents_format_case:nnn
1181         { \exp_not:N \begin{verbatimsc} } % LaTeX
1182         { \verbatimsc } % Plain/Generic
1183         { \startverbatimsc } % ConTeXt
1184         ^^M
1185       \exp_not:V #1 ^^M
1186       \g__scontents_end_verbatimsc_tl
1187     }
1188     \cs_set_eq:NN ^^M \__scontents_verb_print_EOL:
1189   }
1190 \group_end:
1191 \cs_new_protected:Nn \__scontents_xverb:
1192 {
1193   \char_set_catcode_active:n { 9 }
1194   \char_set_active_eq:nN { 9 } \__scontents_tabs_to_spaces:
1195   \__scontents_xverb:w
1196 }

```

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

12.18 The command `\meaningsc`

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

```

\__scontents_meaningsc:n
\__scontents_meaningsc:n
1197 \NewDocumentCommand \meaningsc { o m }
1198 {

```

```

1199 \tl_set:Nn \__scontents_cmd_name_tl { meaningsc }
1200 \__scontents_meaningsc:nn {#1} {#2}
1201 }
1202 \cs_new_protected:Npn \__scontents_meaningsc:n #1 #2
1203 {
1204 \__scontents_bsphack:
1205 \group_begin:
1206 \seq_clear:N \__scontents_seq_item_seq
1207 \str_set:Ne \__scontents_current_seq_name_str {#2}
1208 \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / typemeaning } {#1} }
1209 \seq_if_empty:NT \__scontents_seq_item_seq
1210 { \seq_set_from_clist:Nn \__scontents_seq_item_seq { 1 } }
1211 \__scontents_meaningsc:n {#2}
1212 \use:e
1213 {
1214 \group_end:
1215 \bool_if:NF \__scontents_print_verb_style_bool { \__scontents_esphack: }
1216 }
1217 }
1218 \group_begin:
1219 \char_set_catcode_active:N ^^I
1220 \cs_new_protected:Npn \__scontents_meaningsc:n #1
1221 {
1222 \tl_set:Ne \__scontents_meaningsc_arg_tl
1223 { \__scontents_getfrom_seq:Nn \__scontents_seq_item_seq {#1} }
1224 \tl_replace_all:Nen \__scontents_meaningsc_arg_tl { \iow_char:N ^^J } { ~ }
1225 \tl_replace_all:Nen \__scontents_meaningsc_arg_tl { \c__scontents_hidden_space_str } { ~ }
1226 \tl_log:N \__scontents_meaningsc_arg_tl
1227 \tl_use:N \__scontents_verb_font_tl
1228 \tl_replace_all:Nne \__scontents_meaningsc_arg_tl { ^^I } { \__scontents_tabs_to_spaces: }
1229 \tl_if_empty:NF \__scontents_meaningsc_arg_tl
1230 {
1231 \bool_if:NT \__scontents_print_verb_style_bool
1232 {
1233 \cs_replacement_spec:N \__scontents_meaningsc_arg_tl
1234 }
1235 }
1236 \__scontents_file_write_cmd:VV \__scontents_file_name_tl \__scontents_meaningsc_arg_tl
1237 }
1238 \group_end:

```

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

12.19 The command `\mergesc`

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

```

\mergesc
\__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.

```

1239 \NewDocumentCommand \mergesc { o m }
1240 {
1241 \tl_set:Nn \__scontents_cmd_name_tl { mergesc }
1242 \__scontents_mergesc_code:nn {#1} {#2}
1243 }
1244 \cs_new_protected:Npn \__scontents_mergesc_code:nn #1 #2
1245 {
1246 \group_begin:
1247 \tl_clear:N \__scontents_mergesc_keys_tl
1248 \tl_if_novalue:nF {#1}
1249 {
1250 \keys_set_known:nn { scontents / mergesc } {#1} \__scontents_mergesc_keys_tl
1251 }
1252 \seq_gclear:c { g__scontents_name_sc!internal_seq }
1253 \__scontents_mergesc_parse_list:n {#2}
1254 \exp_args:Ne \__scontents_mergesc_cmd:nn
1255 { 1-end, \exp_not:V \__scontents_mergesc_keys_tl } { sc!internal }
1256 \group_end:
1257 }

```

```

1258 \cs_new_protected:Npn \__scontents_mergesc_parse_list:n #1
1259 {
1260   \clist_map_inline:nn {#1} { \__scontents_parse_mergesc:nw ##1 \s__scontents_stop }
1261   \seq_gpop_right:cN { g__scontents_name_sc!internal_seq } \l__scontents_mergesc_arg_tl
1262   \__scontents_remove_trailing_eol:N \l__scontents_mergesc_arg_tl
1263   \seq_gput_right:cV { g__scontents_name_sc!internal_seq } \l__scontents_mergesc_arg_tl
1264 }
1265 \cs_new_protected:Npe \__scontents_remove_trailing_eol:N #1
1266 {
1267   \exp_not:N \exp_after:wN \exp_not:N \__scontents_remove_trailing_eol:w
1268   #1 \s__scontents_stop \c__scontents_hidden_space_str \s__scontents_stop \s__scontents_mark #1
1269 }
1270 \use:e
1271 {
1272   \cs_new_protected:Npn \exp_not:N \__scontents_remove_trailing_eol:w #1
1273     \c__scontents_hidden_space_str \s__scontents_stop #2 \s__scontents_mark #3
1274 } {
1275   \tl_set:Ne #3
1276   {
1277     \tl_if_empty:nTF {#2}
1278     { \exp_not:o { \__scontents_use_delimit_by_s_stop:nw #1 } }
1279     { \exp_not:n {#1} }
1280   }
1281 }
1282 \cs_new_protected:Npn \__scontents_parse_mergesc:nw #1
1283 {
1284   \peek_remove_spaces:n
1285   {
1286     \peek_charcode:NTF [ % ]
1287     { \__scontents_parse_mergesc_aux:nw {#1} }
1288     { \__scontents_parse_mergesc_aux:nw {#1} [ 1-\seq_count:c { g__scontents_name_#1_seq } ] }
1289   }
1290 }
1291 \cs_new_protected:Npn \__scontents_parse_mergesc_aux:nw #1 [#2]
1292 {
1293   \seq_clear:N \l__scontents_seq_item_seq
1294   \clist_map_inline:nn {#2}
1295     { \__scontents_parse_mergesc_range:nw {#1} ##1 - \q__scontents_mark - \s__scontents_mark }
1296   \seq_map_inline:Nn \l__scontents_seq_item_seq
1297     {
1298       \seq_gput_right:ce { g__scontents_name_sc!internal_seq }
1299       { \seq_item:cn { g__scontents_name_#1_seq } {##1} }
1300     }
1301   \__scontents_use_none_delimit_by_s_stop:w
1302 }
1303 \cs_new_protected:Npn \__scontents_parse_mergesc_range:nw #1 #2 - #3 - #4 \s__scontents_mark
1304 {
1305   \cs_set_protected:Npn \__scontents_tmp:w ##1
1306   {
1307     \msg_error:nnee { scontents } { index-out-of-range }
1308     {##1} {#1} { \seq_count:c { g__scontents_name_#1_seq } }
1309   }
1310   \__scontents_range_parser:nne {#2} {#3}
1311   { \seq_count:c { g__scontents_name_#1_seq } }
1312   { \__scontents_tmp:w }
1313 }

```

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

12.20 The command `\setupsc`

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

```

1314 \NewDocumentCommand \setupsc { +m }
1315 {
1316   \keys_set:nn { scontents } {#1}
1317 }

```

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

12.21 The command `\countsc`

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

```

1318 \NewExpandableDocumentCommand \countsc { m }
1319 {
1320   \seq_count:c { g__scontents_name_#1_seq }
1321 }

```

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

12.22 The command `\cleanseqsc`

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

```

1322 \NewDocumentCommand \cleanseqsc { m }
1323 {
1324   \seq_gc_clear_new:c { g__scontents_name_#1_seq }
1325 }

```

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

12.23 Warning and error messages

Warning and error messages used throughout the package.

```

1326 \msg_new:nnn { scontents } { junk-after-begin }
1327 {
1328   Junk~characters~#1~\msg_line_context: :
1329   \ \ \
1330   #2
1331 }
1332 \msg_new:nnnn { scontents } { env-already-defined }
1333 { Environment~'#1'~already~defined! }
1334 {
1335   You~have~used~\newenvsc
1336   with~an~environment~that~already~has~a~definition. \ \ \
1337   The~existing~definition~of~'#1'~will~not~be~altered.
1338 }
1339 \msg_new:nnn { scontents } { empty-argument-command }
1340 { Empty~argument~for~'Scontents'~\msg_line_context:. }
1341 \msg_new:nnn { scontents } { empty-environment }
1342 { environment~'#1'~empty~\msg_line_context:. }
1343 \msg_new:nnn { scontents } { empty-variable }
1344 { Variable~'#1'~empty~\msg_line_context:. }
1345 \msg_new:nnn { scontents } { overwrite-file }
1346 { Overwriting~file~'#1'. }
1347 \msg_new:nnn { scontents } { writing-file }
1348 { Writing~file~'#1'. }
1349 \msg_new:nnn { scontents } { not-writing }
1350 { File~'#1'~already~exists.~Not~writing. }
1351 \msg_new:nnn { scontents } { rescanning-text }
1352 { Rescanning~text~'#1'~after~\c_backslash_str end{#2}~\msg_line_context:. }
1353 \msg_new:nnn { scontents } { multiple-begin }
1354 { Multiple~\c_backslash_str begin{ \_scontents_env_name_tl }~\msg_line_context:. }
1355 \msg_new:nnn { scontents } { undefined-storage }
1356 { Storage~named~'#1'~is~not~defined. }
1357 \msg_new:nnnn { scontents } { mergesc-missing-key }
1358 {
1359   Need~mandatory~key~'typestored'~or~'meaningsc'~for~\
1360   command~\c_backslash_str mergesc~\msg_line_context:.
1361 }
1362 {
1363   The~command~\c_backslash_str mergesc~need~a~mandatory~key~typestored~or~meaningsc.\ \
1364   Check~that~you~have~spelled~the~key~name~correctly.
1365 }
1366 \msg_new:nnn { scontents } { index-out-of-range }
1367 {
1368   \int_compare:nNnTF {#1} = { 0 }
1369   { Index~of~sequence~cannot~be~zero. }
1370   {
1371     Index~'#1'~out~of~range~for~'#2'.~
1372     \int_compare:nNnTF {#1} > { 0 }

```

```

1373         { Max = } { Min = -} #3.
1374     }
1375 }
1376 \msg_new:nnnn { scontents } { env-key-unknown }
1377 {
1378     The~key~'#1'~is~unknown~by~environment~
1379     '\l__scontents_env_name_tl'~and~is~being~ignored.
1380 }
1381 {
1382     The~environment~'\l__scontents_env_name_tl'~does~not~have~a~key~called~'#1'.\\
1383     Check~that~you~have~spelled~the~key~name~correctly.
1384 }
1385 \msg_new:nnnn { scontents } { env-key-value-unknown }
1386 {
1387     The~key~'#1=#2'~is~unknown~by~environment~
1388     '\l__scontents_env_name_tl'~and~is~being~ignored.
1389 }
1390 {
1391     The~environment~'\l__scontents_env_name_tl'~does~not~have~a~key~called~'#1'.\\
1392     Check~that~you~have~spelled~the~key~name~correctly.
1393 }
1394 \msg_new:nnnn { scontents } { cmd-key-unknown }
1395 {
1396     The~key~'#1'~is~unknown~by~command~\c_backslash_str \l__scontents_cmd_name_tl \c_space_tl
1397     and~is~being~ignored~ \msg_line_context:.
1398 }
1399 {
1400     The~command~\c_backslash_str \l__scontents_cmd_name_tl \c_space_tl
1401     does~not~have~a~key~called~'#1'.\\
1402     Check~that~you~have~spelled~the~key~name~correctly.
1403 }
1404 \msg_new:nnnn { scontents } { cmd-key-value-unknown }
1405 {
1406     The~key~'#1=#2'~is~unknown~by~command~\c_backslash_str \l__scontents_cmd_name_tl \c_space_tl
1407     and~is~being~ignored~ \msg_line_context:.
1408 }
1409 {
1410     The~command~\c_backslash_str \l__scontents_cmd_name_tl \c_space_tl
1411     does~not~have~a~key~called~'#1'.\\
1412     Check~that~you~have~spelled~the~key~name~correctly.
1413 }
1414 </core>

```

12.24 Finish package

Finish package implementation.

```

1415 <plain|context>\ExplSyntaxOff
1416 <plain|context>\endinput
1417 <latex|core>\file_input_stop:

```


13 Index of Implementation

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

Symbols	
'	981
*	298, 303, 314
\,	981
\-	981
\<	981
\>	981
\\	30, 120, 1329, 1336, 1359, 1363, 1382, 1391, 1401, 1411
\`	981
A	
after	<u>411</u>
B	
before	<u>411</u>
\beginngroup	52, 57
bool commands:	
\bool_if:N _{TF}	104, 244, 248, 547, 549, 552, 741, 746, 775, 777, 959, 1044, 1076, 1101, 1125, 1127, 1131, 1156, 1165, 1215, 1231
\bool_lazy_and:nn _{TF}	517, 651
\bool_lazy_or:nn _{TF}	584, 710
\bool_new:N	195, 196, 198, 202, 204, 206, 207, 221
\bool_set_false:N	365, 396, 457, 666, 956
\bool_set_true:N	197, 360, 361, 366, 391, 392, 397, 414, 419, 427, 435, 458, 663, 967
C	
\catcode	53
char commands:	
\char_set_active_eq:NN	288, 293, 722
\char_set_active_eq:nN	1194
\char_set_catcode:nn	23
\char_set_catcode_active:N	281, 292, 314, 621, 622, 623, 867, 1169, 1219
\char_set_catcode_active:n	298, 1043, 1193
\char_set_catcode_letter:N	21
\char_set_catcode_letter:n	669
\char_set_catcode_other:N	120, 122, 123, 668, 972
\char_set_lccode:nn	303
\l_char_special_seq	39, 668, 925
\char_value_catcode:n	20
\cleanseqsc	8, 47, <u>1322</u>
clist commands:	
\clist_map_inline:nn	1260, 1294
\contextlmtxmode	104
\countsc	7, 47, <u>1318</u>
cs commands:	
\cs:w	731, 732, 891, 892
\cs_end:	736, 737, 891, 892
\cs_generate_variant:Nn	225, 226, 274, 507, 539, 601, 781, 782
\cs_gset_eq:NN	353, 354
\cs_if_exist:N _{TF}	25, 912, 913, 922, 1055
\cs_new:Nn	275, 276, 277, 680, 727
\cs_new:Npn	96, 227, 228, 229, 230, 555, 564, 568, 572, 582, 690, 756, 766, 772, 924
\cs_new_eq:NN	858
\cs_new_protected:Nn	299, 315, 325, 334, 648, 790, 795, 949, 980, 982, 984, 992, 1068, 1191
\cs_new_protected:Npe	818, 1265
\cs_new_protected:Npn	265, 279, 290, 380, 382, 445, 447, 482, 486, 488, 494, 500, 508, 531, 540, 545, 590, 608, 624, 626, 636, 638, 658, 739, 773, 783, 807, 825, 837, 847, 860, 865, 870, 878, 880, 887, 896, 899, 904, 932, 941, 947, 1037, 1063, 1083, 1095, 1121, 1140, 1170, 1202, 1220, 1244, 1258, 1272, 1282, 1291, 1303
\cs_replacement_spec:N	1233
\cs_set:Npe	282
\cs_set:Npn	684, 957
\cs_set_eq:NN	473, 478, 972, 975, 1018, 1174, 1175, 1176, 1188
\cs_set_protected:Npe	670, 707
\cs_set_protected:Npn	436, 851, 1305
\csname	56, 66
D	
\DeclareInstance	994
\def	3, 4, 5, 51, 55, 58, 59, 67, 80
\definotyping	988
dim commands:	
\dim_compare:nNn _{TF}	340
\dim_zero:N	952
\c_zero_dim	285
\do	925, 972, 975, 981
\dospecials	<u>921</u> , 973
E	
\else	77, 79
else commands:	
\else:	238
\end	126, 730, 934
\endblockenv	1025
\endcsname	56, 66
\endgroup	55, 58, 61, 74, 88
\endinput	75, 89, 1416
\endlinechar	54
\endscontents	4, <u>918</u>
\endverbatim	1027
\endverbatimsc	127, 928, <u>940</u>
\end{verbatimsc}	<u>928</u>
enumext internal commands:	
\l_enumext_ref_key_arg_tl	1051
Environments:	
verbatim	39, 40
\errhelp	62
\errmessage	63
exp commands:	
\exp_after:wN	234, 235, 731, 732, 977, 1117, 1118, 1267
\exp_args:Ne	1254
\exp_args:Nee	502
\exp_args:Nf	559, 576, 760
\exp_args:Nnf	570
\exp_args:NNV	1113
\exp_args:Noo	889
\exp_args:NV	381, 446, 487, 750
\exp_not:N	136, 286, 672, 675, 684, 687, 730, 731, 732, 748, 807, 822, 823, 842, 844, 851, 853, 932, 934, 935, 936, 1181, 1267, 1272

\exp_not:n	503, 504, 543, 702, 764, 769, 770, 1126, 1132, 1185, 1255, 1278, 1279
\expandafter	56, 66
\ExplSyntaxOff	34, 45, 1415
\ExplSyntaxOn	15
F	
\fi	65, 91, 92
fi commands:	
\fi:	240, 730, 735
file commands:	
\file_if_exist:nTF	246
\file_input:n	22, 139
\file_input_stop:	35, 46, 1417
force-eol	141, 356, 388
\foreachsc	6, 29, 30, 43, 1090
\frenchspacing	944, 1014
G	
\getstored	6, 42, 1079, 1130
group commands:	
\group_begin: .	297, 302, 313, 620, 660, 901, 943, 1040, 1097, 1143, 1168, 1205, 1218, 1246
\group_end: . .	304, 310, 322, 745, 780, 907, 948, 1075, 1116, 1164, 1190, 1214, 1238, 1256
\group_insert_after:N	596, 597, 598, 599
I	
if commands:	
\if_false:	730, 735
\if_meaning:w	234
\IfDocumentMetadataTF	1010, 1023
\ifx	56, 66, 78
\input	14
int commands:	
\int_abs:n	586
\int_compare:nNnTF	521, 787, 801, 1368, 1372
\int_compare_p:nNn	585, 586
\int_decr:N	797
\int_eval:n	513, 570
\int_incr:N	792, 793
\int_new:N	19, 183, 184, 205, 213
\int_set:Nn	20, 428, 1103, 1177
\int_set_eq:NN	331, 339
\int_step_function:nnnN	669, 1106
\int_step_inline:nnnn	522, 523
\int_to_roman:n	512, 518, 519
\int_zero:N	785
\c_zero_int	801
\interlinepenalty	964, 969
iow commands:	
\iow_char:N	101, 114, 1057, 1058, 1060, 1224
\iow_close:N	271, 742
\iow_log:n	18
\iow_new:N	194
\iow_now:Nn	270, 776
\iow_open:Nn	269, 664
K	
Keys provide by SCONTENTS :	
meaningsc	30
tpestored	30
keys commands:	
\keys_define:nn	142, 173, 357, 388, 411, 453, 469
\l_keys_key_str	381, 446, 487
\keys_set:nn	614, 883, 1042, 1098, 1146, 1208, 1316
\keys_set_known:nnN	1250
L	
\legacyverbatimsetup	1005
M	
\meaningsc	6, 30, 31, 44, 1197
meaningsc	469
\mergesc	7, 30, 31, 45, 1239
mode commands:	
\mode_if_horizontal:TF	328, 337, 968
\mode_leave_vertical:	284, 961, 985
msg commands:	
\msg_error:nn	110, 484, 534
\msg_error:nnn	385, 450, 498, 914, 1173
\msg_error:nnnn	386, 451, 492, 642
\msg_error:nnnnn	1307
\msg_expandable_error:nnn	562, 580
\msg_expandable_error:nnnnn	587
\msg_fatal:nn	33
\msg_line_context:	42, 1328, 1340, 1342, 1344, 1352, 1354, 1360, 1397, 1407
\msg_new:nnn	27, 40, 106, 1326, 1339, 1341, 1343, 1345, 1347, 1349, 1351, 1353, 1355, 1366
\msg_new:nnnn	1332, 1357, 1376, 1385, 1394, 1404
\msg_warning:nn	44, 788
\msg_warning:nnn	250, 254, 259, 655, 1053
\msg_warning:nnnn	716
N	
\NeedsTeXFormat	7
\NewDocumentCommand	604, 910, 1032, 1048, 1079, 1090, 1135, 1197, 1239, 1314, 1322
\NewDocumentEnvironment	898, 1008
\newenvsc	5, 38, 910, 919, 1335
\NewExpandableDocumentCommand	1318
\next	55, 58, 67, 80, 93
\nobreak	344, 985
\null	962
O	
\obeyedline	1055, 1057, 1058
\obeylines	973
overwrite	141, 356, 388, 453
P	
\PackageError	59, 69, 82
Packages:	
expl3	20
l3keys	31
latex-lab	41
ltxcmd	20, 33, 42
lua-widow-control	22
scontents	20, 23, 31, 39
verbatim	41
xparse	20
xsim	34
xtemplate	41
\par	957
\parfillskip	953
\parindent	952
\parskip	951, 954
peek commands:	
\peek_charcode:N	1286
\peek_remove_spaces:n	1284
\penalty	964, 969

prg commands:

`\prg_generate_conditional_variant:Nnn` . . . 224
`\prg_new_protected_conditional:Npnn` . . . 231, 242, 799
`\prg_replicate:nn` 278
`\prg_return_false:` 239, 255, 263, 803
`\prg_return_true:` 237, 251, 260, 802
print-all 141
print-cmd 141, 388, 453
print-env 141, 356
`\ProcessKeyOptions` 177
`\ProvidesExplPackage` 8

Q

quark commands:

`\quark_new:N` 215, 216

quark internal commands:

`\q__scontents_mark` 26, 215, 236, 491, 822, 844, 1295
`\q__scontents_stop` 215, 230, 675, 687, 823, 832, 835, 844, 853

R

`\relax` 56, 66
`\RenewDocumentCommand` 1057

S

scan commands:

`\scan_new:N` 217, 218
`\scan_stop:` 305, 327, 336, 1175

scan internal commands:

`\s__scontents_mark` 215, 491, 494, 1268, 1273, 1295, 1303
`\s__scontents_stop` . . . 215, 227, 228, 229, 236, 1260, 1268, 1273

`\Scontents` 5, 29, 30, 41, 1031

`\scontents` 4, 918

`scontents` 4, 918

scontents internal commands:

`__scontents_analyse_nesting:n` 36, 694, 783, 783
`__scontents_analyse_nesting:w` 783
`__scontents_analyse_nesting_format:n` 38, 786, 858, 860
`__scontents_analyse_nesting_generic:nn` . . . 837, 861, 862
`__scontents_analyse_nesting_generic:w` . . . 832, 835, 842, 851
`__scontents_analyse_nesting_generic_process:nn` 825, 854
`__scontents_analyse_nesting_latex:n` 818, 859
`__scontents_analyse_nesting_latex:w` 807, 815, 820
`__scontents_append_contents:nn` 31, 530, 531, 539, 543
`__scontents_bsphack:` . . . 25, 41, 324, 325, 353, 1039, 1142, 1204
`\l__scontents_char_value_int` 11
`__scontents_check_line_process:nn` 34, 619, 632, 634, 637, 638
`\l__scontents_cmd_name_tl` . . . 24, 38, 185, 1034, 1092, 1137, 1199, 1241, 1396, 1400, 1406, 1410
`\l__scontents_current_seq_name_str` 24, 190, 497, 1145, 1207
`__scontents_define_env:nnn` . . . 38, 39, 878, 889, 896
`__scontents_define_generic_nesting_function:n` 839, 847

`__scontents_do_noligs:N` . . . 27, 279, 279, 975, 1176
`\c__scontents_end_env_tl` 22, 117, 673, 674, 685, 686, 702
`\g__scontents_end_verbatimsc_tl` . . . 22, 117, 933, 1186
`__scontents_env_end_function:` 709, 727
`\l__scontents_env_name_tl` 22, 38, 117, 643, 655, 717, 733, 813, 843, 852, 882, 1354, 1379, 1382, 1388, 1391
`\l__scontents_environment_keys_tl` . . . 23, 179, 612, 613, 614
`__scontents_esphack:` . . . 25, 42, 324, 334, 354, 1076, 1165, 1215
`__scontents_file_if_writable:n` . . . 26, 242, 242
`__scontents_file_if_writable:nTF` . . . 24, 242, 267, 661
`\l__scontents_file_name_tl` . . . 24, 193, 362, 367, 393, 398, 459, 646, 1070, 1161, 1236
`__scontents_file_tl_write_start:n` 34, 646, 658, 658, 782
`__scontents_file_write_cmd:nn` . . . 26, 42, 265, 265, 274, 1070, 1161, 1236
`\l__scontents_file_write_iow` . . . 24, 193, 269, 270, 271, 664, 742, 776
`__scontents_file_write_stop:N` 34, 650, 658, 739
`__scontents_finish_storing:NNN` 32, 38, 42, 545, 545, 873, 1071
`\l__scontents_forced_eol_bool` 32, 158, 549
`__scontents_foreach_add_body:n` 1090, 1110, 1121
`\l__scontents_foreach_after_bool` 199, 419, 1131
`\l__scontents_foreach_after_tl` . . . 199, 420, 1132
`\l__scontents_foreach_before_bool` 199, 414, 1125
`\l__scontents_foreach_before_tl` 199, 415, 1126
`\g__scontents_foreach_exec_tl` . . . 199, 1111, 1118, 1119
`\l__scontents_foreach_print_seq` 199, 1100, 1114, 1123
`\l__scontents_foreach_sep_tl` 440, 1114
`\l__scontents_foreach_start_int` 423, 1107
`\l__scontents_foreach_step_int` 431, 1108
`\l__scontents_foreach_stop_bool` 206, 427, 1101
`\l__scontents_foreach_stop_int` . . . 199, 428, 1103, 1109
`__scontents_foreach_wrapper:n` 437, 1128
`\l__scontents_foreach_wrapper_bool` . . . 199, 435, 1127
`__scontents_foreachsc:nn` 1090, 1093, 1095
`\l__scontents_foreachsc_arg_tl` 24, 186, 1099, 1130
`__scontents_format_case:nnn` 95, 96, 729, 734, 1180
`__scontents_generic_begin:` 38, 864, 865, 885
`__scontents_generic_end:` 38, 864, 870, 888
`__scontents_getfrom_seq:Nn` . . . 32, 555, 555, 1150, 1223
`__scontents_getfrom_seq:nn` . . . 43, 555, 572, 1087
`__scontents_getfrom_seq:nNn` 555, 559, 564
`__scontents_getfrom_seq:nnn` 555, 570, 576, 582
`__scontents_getfrom_seq_aux:nnn` 555, 566, 568
`__scontents_getstored:nn` 42, 1079, 1081, 1083
`__scontents_grab_opt_arg:n` 603, 606, 608
`__scontents_grab_opt_arg:w` . . . 33, 34, 603, 604, 631
`\c__scontents_hidden_space_str` . . . 25, 32, 211, 550, 714, 1152, 1225, 1268, 1273
`__scontents_if_nested:` 36, 799
`__scontents_if_nested:TF` 698, 783
`\g__scontents_last_stored_tl` 25, 210, 592, 597, 599

`\g__scontents_last_tl` 33
`__scontents_lastfrom_seq:n` . 25, 32, 33, 552, 590, 590, 601
`__scontents_make_control_chars_active:` . 290, 315, 645, 677
`__scontents_meaningsc:n` 1197, 1211, 1220
`__scontents_meaningsc:nn` . . 478, 1197, 1200, 1202
`\l__scontents_meaningsc_arg_tl` . . 24, 186, 1222, 1224, 1225, 1226, 1228, 1229, 1233, 1236
`\l__scontents_mergesc_arg_tl` . 24, 186, 190, 1261, 1262, 1263
`__scontents_mergesc_cmd:nn` 30, 469, 473, 478, 482, 1254
`__scontents_mergesc_code:nn` . . 1239, 1242, 1244
`\l__scontents_mergesc_keys_tl` 24, 190, 1247, 1250, 1255
`__scontents_mergesc_parse_list:n` . . 1239, 1253, 1258
`\g__scontents_name_sc!internal_seq` . . . 25, 208
`\l__scontents_name_seq_cmd_tl` . . . 42, 147, 1072
`\l__scontents_name_seq_env_tl` 144, 875
`\l__scontents_nesting_aux_int` . 23, 179, 785, 787, 793
`__scontents_nesting_decr:` 36, 700, 783, 795
`\l__scontents_nesting_env_int` . . 23, 36, 179, 792, 797, 801
`__scontents_nesting_incr:` 790, 814, 831
`\c__scontents_newline_tl` . 22, 100, 111, 114, 1060, 1152
`__scontents_nolig_list:` 940, 976, 980
`__scontents_normalise_line_ends:N` 33, 613, 624
`\l__scontents_overwrite_bool` 161, 248
`__scontents_par:` 275, 276, 319
`__scontents_parse_environment_keys:n` . . . 28
`__scontents_parse_mergesc:nw` . 1239, 1260, 1282
`__scontents_parse_mergesc_aux:` . . 1239, 1287, 1288, 1291
`__scontents_parse_mergesc_range:nw` 1239, 1295, 1303
`__scontents_parse_type_meaning_key:n` 24, 467, 486, 486
`__scontents_parse_type_meaning_key:nn` . . 486, 487, 488
`__scontents_parse_type_meaning_range:w` . 486, 491, 494
`__scontents_parse_typemeaning_key:n` 31
`\l__scontents_plain_bool` . . 25, 220, 956, 959, 967
`__scontents_plain_disable_outer_par:` 290, 299, 317
`\l__scontents_print_cmd_bool` 42, 155, 1072, 1076
`\l__scontents_print_env_bool` 152, 876
`\l__scontents_print_verb_style_bool` 464, 1156, 1165, 1215, 1231
`\l__scontents_processed_body_lines_tl` 23, 179, 650, 653, 874
`__scontents_range_parser:nnnn` 486, 496, 500, 507, 1310
`__scontents_range_parser_aux:nnn` 486, 502, 508
`__scontents_remove_leading_nl:n` . . 34, 658, 751, 756
`__scontents_remove_leading_nl:nn` . . . 761, 766
`__scontents_remove_leading_nl:w` . 658, 769, 772
`__scontents_remove_trailing_eol:N` . 1151, 1239, 1262, 1265
`__scontents_remove_trailing_eol:w` . 1239, 1267, 1272
`__scontents_ret:w` 34, 320, 670, 678, 707, 722
`\l__scontents_save_every_body_lines_tl` 23, 179, 667, 751, 778
`\l__scontents_save_sf_int` 213, 331, 339
`\l__scontents_save_skip` 213, 330, 340
`\l__scontents_Scontents_arg_tl` 24, 42, 186, 1050, 1053, 1058, 1060, 1065, 1070, 1072
`__scontents_Scontents_code:nn` . 41, 1031, 1035, 1037
`__scontents_Scontents_finish:` . 42, 1061, 1066, 1068
`__scontents_Scontents_norm_arg:n` . 41, 42, 1031, 1046, 1063
`__scontents_Scontents_verb_arg:w` . 41, 42, 1031, 1045, 1048
`\l__scontents_seq_item_seq` 25, 208, 513, 524, 1144, 1147, 1148, 1150, 1206, 1209, 1210, 1223, 1293, 1296
`__scontents_set_active_eq:NN` 290, 290, 318, 319, 320, 983
`__scontents_setting_env:nn` . . . 38, 878, 878, 915
`__scontents_setup_verb_processor:` 658, 680, 884
`__scontents_start_after_option:w` . 34, 616, 619, 636
`__scontents_start_environment:w` . . 38, 619, 626, 868
`__scontents_stop_environment:` 38, 619, 648, 872
`__scontents_store_to_seq:NN` . . 32, 540, 540, 551
`\l__scontents_storing_bool` . 24, 36, 196, 197, 360, 365, 391, 396, 457, 547, 652, 746, 777
`\l__scontents_storing_bool` 193
`__scontents_tab:` 275, 275, 318
`\l__scontents_tab_width_int` 164, 278
`__scontents_tabs_to_spaces:` . 26, 277, 277, 1194, 1228
`__scontents_tl_if_head_is_q_mark:n` . . 26, 231
`__scontents_tl_if_head_is_q_mark:nTF` 231, 510, 810, 829
`__scontents_tmp:w` 1305, 1312
`__scontents_typerstored:N` 43, 44, 1135, 1158, 1170
`__scontents_typerstored:nn` . 473, 1135, 1138, 1140
`\l__scontents_typerstored_arg_tl` . 24, 186, 1149, 1151, 1152, 1153, 1154, 1158, 1161
`__scontents_unknown_keys_cmd:n` 30, 409, 443, 445, 445
`__scontents_unknown_keys_cmd:nn` . 445, 446, 447
`__scontents_unknown_keys_env:n` . 378, 380, 380
`__scontents_unknown_keys_env:nn` . 380, 381, 382
`__scontents_use_delimit_by_s_stop:nw` 227, 227, 1278
`__scontents_use_i_delimit_by_s_stop:nw` . 227, 228, 236
`__scontents_use_none_delimit_by_q_stop:w` 227, 230, 783, 811
`__scontents_use_none_delimit_by_s_stop:w` 227, 229, 1301
`\l__scontents_verb_font_tl` 150, 974, 1227
`__scontents_verb_print_EOL:` 1174, 1188
`__scontents_verb_processor_iterate:nnn` . 658, 688, 690
`__scontents_verb_processor_iterate:w` 658, 672, 684
`__scontents_verb_processor_output:n` . 36, 695,

701, 706, 773, 773, 781	
_scontents_verbatimsc_aux:	940, 944, 949
_scontents_verbatimsc_instance:	990, 992, 1012
\c_scontents_version_str	20, 11
_scontents_vobeyspaces:	940, 944, 982
\l_scontents_writable_bool 24, 193, 663, 666, 741, 775	
\l_scontents_writing_bool . 24, 36, 193, 244, 361, 366, 392, 397, 458	
_scontents_xobeysp:	940, 983, 984
_scontents_xverb:	940, 945, 1015, 1018, 1191
_scontents_xverb:w	928, 932, 1135, 1195
\ScontentsCoreFileDate	51, 78
\ScontentsFileDate	3, 9, 16, 78
\ScontentsFileDescription	5, 9, 17
\ScontentsFileVersion	4, 9, 12, 17
sep	411
seq commands:	
\seq_clear:N	1100, 1144, 1206, 1293
\seq_count:N 497, 560, 577, 1104, 1288, 1308, 1311, 1320	
\seq_gclear:N	1252
\seq_gclear_new:N	1324
\seq_gpop_right:NN	1261
\seq_gput_right:Nn	537, 1263, 1298
\seq_if_empty:NTF	1147, 1209
\seq_if_exist:NTF	535, 557, 574
\seq_item:Nn	588, 594, 1299
\seq_map_function:NN	668, 925
\seq_map_inline:Nn	1296
\seq_map_tokens:Nn	566
\seq_new:N	199, 208, 209, 536
\seq_put_right:Nn	513, 524, 1123
\seq_set_from_clist:Nn	1148, 1210
\seq_use:Nn	1113
\setupsc	3, 46, 1314
skip commands:	
\skip_horizontal:n	345
\skip_if_eq:nnTF	342
\skip_new:N	214
\skip_set:Nn	953, 954
\skip_set_eq:NN	330
\skip_vertical:N	951
\c_zero_skip	340, 342, 345
\space	16, 17
start	411
\startscontents	4, 918
\startverbatimsc	987, 1183
step	411
stop	411
\stopscontents	4, 918
\stopverbatimsc	128, 928, 987
store-all	141
store-cmd	141, 388
store-env	141, 356
str commands:	
\c_backslash_str . 132, 643, 808, 821, 843, 852, 1352, 1354, 1360, 1363, 1396, 1400, 1406, 1410	
\c_circumflex_str	212
\c_left_brace_str	135, 808, 822
\c_percent_str	212, 712
\c_right_brace_str	137, 808, 822
\str_const:Nn	16, 211
\str_if_eq:nnTF	503, 504, 630, 714, 813
\str_if_eq_p:nn	712
\str_new:N	192
\str_set:Nn	1145, 1207
\str_use:N	18
sys commands:	
\sys_obeyed_line	111
T	
tag commands:	
\tag_tool:n	1005
T_EX and L^AT_EX 2_ε commands:	
\@	20, 21, 23
\@bsphack	27, 353
\@esphack	27, 354
\@setupverbinvisiblespace	1014
\@vobeyspaces	1014
tex commands:	
\tex_char:D	286
\tex_everypar:D	977, 978
\tex_ignorespaces:D	347
\tex_kern:D	285
\tex_lastskip:D	330, 342
\tex_let:D	305
\tex_lowercase:D	304
\tex_newlinechar:D	1177
\tex_par:D	955, 963, 969
\tex_spacefactor:D	331, 339
\tex_the:D	978
\tex_unpenalty:D	978
tl commands:	
\c_space_tl	275, 1396, 1400, 1406, 1410
\tl_clear:N	667, 1247
\tl_const:Nn	101, 111, 114, 130
\tl_gclear:N	598, 1117
\tl_gset:Nn	592, 1111
\tl_gset_rescan:Nnn	118
\tl_head:n	630, 762
\tl_if_blank:nTF . . 384, 449, 490, 533, 640, 692, 705, 1172	
\tl_if_blank_p:n	711
\tl_if_empty:n	224
\tl_if_empty:NTF	1051, 1154, 1229
\tl_if_empty:nTF	223, 512, 1277
\tl_if_empty_p:N	653
\tl_if_empty_p:n	518, 519
\tl_if_exist:NTF	38
\tl_if_head_is_N_type:nTF	628, 758, 827
\tl_if_noalue:nTF 610, 1041, 1098, 1146, 1208, 1248	
\tl_log:N	542, 1153, 1226
\tl_new:N . 117, 129, 180, 181, 182, 185, 186, 187, 188, 189, 190, 191, 193, 200, 201, 203, 210	
\tl_put_right:Nn	550, 778
\tl_replace_all:Nnn 223, 225, 625, 1058, 1060, 1152, 1224, 1225, 1228	
\tl_retokenize:n . 25, 223, 226, 596, 719, 1085, 1178	
\tl_set:Nn 362, 367, 393, 398, 415, 420, 459, 612, 748, 882, 1034, 1050, 1065, 1092, 1099, 1137, 1149, 1199, 1222, 1241, 1275	
\tl_to_str:n	890
\tl_use:N	717, 733, 843, 852, 974, 1130, 1227
token commands:	
\token_if_eq_meaning:NNTF	768
\token_to_str:N	282, 288
\tt	175
\ttfamily	174
\typestored	6, 30, 31, 43, 1135

tpestored	469	\verbatim	1019
U		\verbatimsc	940, 1182
unknown	356, 388, 411, 453	verbatimsc	7, 990
\unprotect	13	W	
use commands:		width-tab	141, 453
\use:n	570, 682, 743, 805, 840, 849, 930, 1073, 1129, 1162, 1212, 1270	wrapper	411
\UseInstance	1013	write-cmd	388
V		write-env	356
verb-font	141	write-out	356, 388, 453
		\writestatus	12