

SCONTENTS

Stores \LaTeX contents

v1.4 — 2019/10/03*

©2019 by Pablo González †

CTAN: <http://www.ctan.org/pkg/scontents>
GIT: <https://github.com/pablgonz/scontents>

Abstract

The `scontents` package stores valid \LaTeX code in `(sequences)` using the `l3seq` module of `expl3`. The `(stored content)`, including *verbatim*, can be used as many times as desired in the document, additionally can be written to `(external files)`.

Contents

1	Motivation and Acknowledgments	1	5	Other commands provided	5
2	License and Requirements	1	5.1	The command <code>\meaningsc</code>	5
3	The <code>scontents</code> package	2	5.2	The command <code>\countsc</code>	5
3.1	Description of the package and load	2	5.3	The command <code>\cleanseqsc</code>	5
3.2	The TAB character	2	6	The <code>scontents</code> package in action	6
3.3	Configuration of the options	2	7	Examples	7
3.4	Options Overview	3	7.1	From answers package	7
4	User interface	3	7.2	From filecontentsdef package	7
4.1	The environment <code>scontents</code>	3	7.3	From TeX-SX	8
4.2	The command <code>\Scontents</code>	4	7.4	Customization of <code>verbatimsc</code>	9
4.3	The command <code>\getstored</code>	4	8	Change history	11
4.4	The command <code>\typestored</code>	4	9	Index of Documentation	12
4.5	The environment <code>verbatimsc</code>	5	10	Implementation	13
			11	Index of Implementation	27

1 Motivation and Acknowledgments

In \LaTeX there is no direct way to record content for later use, although you can do this using `\macros`, recording `(verbatim content)` is a problem, usually you can avoid this by creating external files or boxes. The general idea of this package is to try to imitate this implementation *buffers* that has ConTeXt which allows you to save content in memory, including *verbatim*, to be used later. The package `filecontentsdef` solves this problem and since `expl3` 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. Also a special thanks to Phelype Oleinik who has collaborated and adapted a large part of the code and all \LaTeX 3 team for their great work and to the different members of the TeX-SX community who have provided great answers and ideas. Here a note of the main ones:

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

2 License and Requirements

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

The `scontents` package loads `expl3`, `xparse` and `l3keys2e`. This package can be used with `xelatex`, `lualatex`, `pdflatex` and the classical workflow `latex»dvips»ps2pdf`.

*This file describes a documentation for v1.4, last revised 2019/10/03.

†E-mail: pablgonz@educarchile.cl

3 The scontents package

3.1 Description of the package and load

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>`.

The package uses a `[<key = val>]` system for the options and is loaded in the usual way:

```
\usepackage{scontents}
```

or

```
\usepackage[<key = val>]{scontents}
```

3.2 The TAB character

Some users use horizontal TABs “`␣`” from keyboard to indented the source code of the document and depending on the text editor used, some will use real TABs (“hard tabs”), others with “soft tabs”(`␣` or `␣␣␣`) or both.

At first glance it may seem the same, but the way in which TABs (“hard tabs”) are processed according to the context in which they are found within a file, both in `<reading>`¹ and `<writing>`² are different and may have adverse consequences.

In a standard \TeX document, the character TAB “`␣`” are treated as explicit spaces (in most contexts) and is the behavior when `<stored contents>`, but when `<writing files>` these are preserved.

With a \TeX Live distribution, the TAB character is “printable” for `latex`, `pdflatex` and `lualatex`, but if you use `xelatex` you must add the `-8bit` option on the command line, otherwise you will get \TeX -TAB (`^^I`) in the `<output file>`.

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

3.3 Configuration of the options

Most of the options can be passed directly to the package or using the command `\setupsc`. All boolean keys can be passed using the equal sign “=” or just the name of the key, all unknown keys will return an error. In this section are described some of the options, a summary of all options is shown in section 3.4.

```
\setupsc {<keyval list>}
```

The command `\setupsc` sets the `<keys>` in a global way, it can be used both in the preamble and in the body of the document as many times as desired.

```
verb-font = {<font family>} (default: \ttfamily)
```

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

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

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

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

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

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

Sets if the end of line for the `<stored content>` is hidden or not. This key is necessary only if the last line is the closing of some environment defined by the `fancyvrb` package as `\end{Verbatim}` or another environment that does not support a comments “%” after closing `\end{...}%`. This key is available for the `scontents` environment and the `\Scontents` command.

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

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

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

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

3.4 Options Overview

Summary of available options.

key	package	\setupsc	scontents	\Scontents	\Scontents*	\tpestored	\meaningsc
store-env	✓	✓	✓	✗	✗	✗	✗
store-cmd	✓	✓	✗	✓	✓	✗	✗
print-env	✓	✓	✓	✗	✗	✗	✗
print-cmd	✓	✓	✗	✓	✓	✗	✗
print-all	✓	✓	✗	✗	✗	✗	✗
store-all	✓	✓	✗	✗	✗	✗	✗
write-env	✗	✗	✓	✗	✗	✗	✗
write-out	✗	✗	✓	✗	✗	✗	✗
width-tab	✓	✓	✗	✗	✗	✓	✓
force-eol	✓	✓	✓	✗	✓	✗	✗
verb-font	✓	✓	✗	✗	✗	✗	✗

4 User interface

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

4.1 The environment scontents

```
scontents \begin{scontents}[<keyval list>]
           <env contents>
           \end{scontents}
```

The `scontents` environment allows you to `<store>` and `<write>` content, 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 `<keys>` must be passed in a `<single line>` separated by commas “without vertical spaces” and “without separation” of the start of the environment.

Comments “%” or “any character” after `\begin{scontents}` or `[<keyval list>]` 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, content stored in the `<inner>` sequence is only available after content stored in the `<outer>` sequence one has been retrieved, either by using the key `print-env` or `\getstored` command.

It is advisable to store content within sequences with different names, so as not to get lost in the order in which content is stored.

Options for environment

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

`store-env = {<seq name>}` (default: *contents*)

Sets the name of the `<sequence>` in which the contents will be stored. If the sequence does not exist, it will be created globally.

`print-env = {<true | false>}` (default: *false*)

Sets if the `<stored content>` is displayed or not at the time of running the environment. The content is extracted from the `<sequence>` in which it is stored.

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

Sets the name of the *external file* in which the *contents* of the environment will be written. The *file.ext* will be created in the working directory, if *file.ext* exists it will be overwritten, relative or absolute paths are not supported. The characters TABs will be written in *file.ext* and the *contents* will be stored in the sequence established at that time. X₃LT₃EX users using the TAB character must add `-8bit` at the command line, otherwise you will get T_EX-TAB ($\wedge^{\wedge}\text{I}$) in *file.ext*.

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

Sets the name of the *external file* in which the *contents* of the environment will be written. The *file.ext* will be created in the working directory, if *file.ext* exists it will be overwritten, relative or absolute paths are not supported. The characters TABs will be written in *file.ext*, the rest of the *keys* will not be available and the *contents* will NOT be stored in any sequence. X₃LT₃EX users using the TAB character must add `-8bit` at the command line, otherwise you will get T_EX-TAB ($\wedge^{\wedge}\text{I}$) in *file.ext*.

4.2 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. It is not possible to pass environments such as *verbatim*, but it is possible to use the implementation of `\Verb` provided by the `fvextra` package for contents on one line and `\lstinline` from `listings` package, but it is preferable to use the starred version.

The `\Scontents*` command reads the {⟨argument⟩} under *verbatim* category code regimen. If its first delimiter is a brace, it will be assumed that the {⟨argument⟩} is nested into braces. Otherwise it will be assumed that the ending of that ⟨argument⟩ is delimited by that first delimiter ⟨del⟩ like command `\verb`. Blank lines are preserved, escaped braces “\{” and “\}” must also be balanced if the argument is used with braces and TABs characters typed from the keyboard are converted into spaces.

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⟩]. The key `force-eol` is available for this command.

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

Sets the name of the *sequence* in which the contents will be 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 content is extracted from the *sequence* in which it is stored.

4.3 The command `\getstored`

`\getstored` `\getstored[⟨index⟩]{⟨seq name⟩}`

The command `\getstored` gets the content stored in *seq name* according to the *index* in which it was stored. The command is robust and can be used as an ⟨argument⟩ for another command. If the optional argument is not passed it defaults to the first element stored in the *seq name*.

4.4 The command `\typestored`

`\typestored` `\typestored[⟨index, width-tab = number⟩]{⟨seq name⟩}`

The command `\typestored` internally places the content stored in the *seq name* into the `verbatimsc` environment. The *index* number corresponds to the position in which the content is stored in the *seq name*.

If the optional argument is not passed it defaults to the first element stored in the *seq name*. The key `width-tab` is available for this command.

The `verbatim` package is not compatible with the implementation of the `verbatimsc` environment used by this command, if you are a user of that package consider customizing the `verbatimsc` environment.

4.5 The environment `verbatimsc`

`verbatimsc`

Internal environment used by `\typestored` to display *verbatim style* contents.

One consideration to keep in mind is that this is a *representation* of the *stored content* in a *verbatim* environment and not a real *verbatim* environment.

The `verbatimsc` environment can be customized in the following ways:

Using the package `fancyvrb`:

```
\makeatletter
\let\verbatimsc\@undefined
\let\endverbatimsc\@undefined
\makeatother
\DefineVerbatimEnvironment{verbatimsc}{Verbatim}{numbers=left}
```

Using the package `minted`:

```
\makeatletter
\let\verbatimsc\@undefined
\let\endverbatimsc\@undefined
\makeatother
\usepackage{minted}
\newminted{tex}{linenos}
\newenvironment{verbatimsc}{\VerbatimEnvironment\begin{texcode}}{\end{texcode}}
```

Using the package `listings`:

```
\makeatletter
\let\verbatimsc\@undefined
\let\endverbatimsc\@undefined
\makeatother
\usepackage{listings}
\lstnewenvironment{verbatimsc}
{
  \lstset{
    basicstyle=\small\ttfamily,
    columns=fullflexible,
    language=[LaTeX]TeX,
    numbers=left,
    numberstyle=\tiny\color{gray},
    keywordstyle=\color{red}
  }
}{}
}
```

5 Other commands provided

5.1 The command `\meaningsc`

`\meaningsc`

`\meaningsc` [*index*, *width-tab = number*] {*seq name*}

The command `\meaningsc` executes `\meaning` on the content stored in *seq name*. The *index* number corresponds to the position in which the content is stored in the *seq name*.

If the optional argument is not passed it defaults to the first element stored in the *seq name*. The key *width-tab* is available for this command.

5.2 The command `\countsc`

`\countsc`

`\countsc` {*seq name*}

The command `\countsc` count a number of contents stored in *seq name*.

5.3 The command `\cleanseqsc`

`\cleanseqsc`

`\cleanseqsc` {*seq name*}

The command `\cleanseqsc` remove all contents stored in *seq name*.

6 The `scontents` package in action

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

Abstract

The `scontents` package stores valid \TeX code in `<sequences>` using the `l3seq` module of `expl3`. The `<stored content>`, including *verbatim*, can be used as many times as desired in the document, additionally can be written to `<external files>`.

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}
The \mypkg*{scontents} package stores valid \hologo{(La)TeX} code in
\mymeta{sequences} using the \mypkg{l3seq} module of \mypkg{expl3}.
The \mymeta{stored content}, including \emph{verbatim}, can be used
as many times as desired in the document, additionally can be written
to \mymeta{external files}.
\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} \enquote{\emph{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 The \mypkg*{scontents} package stores valid \hologo{(La)TeX} code in
4 \mymeta{sequences} using the \mypkg{l3seq} module of \mypkg{expl3}.
5 The \mymeta{stored content}, including \emph{verbatim}, can be used
6 as many times as desired in the document, additionally can be written
7 to \mymeta{external files}.
8 \end{abstract}
9 \end{scontents}
```

and

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


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

7 Examples

These are some (adapted) examples that have served as inspiration for the creation of this package.

7.1 From answers package

Example 1

Adaptation of example 1 (ansexam1) of the package answers .


```

1 \documentclass[12pt,a4paper]{article}
2 \usepackage[store-cmd=solutions]{scontents}
3 \usepackage{pgffor}
4 \newtheorem{ex}{Exercise}
5 \begin{document}
6 \section{Problems}
7 \begin{ex}
8 First exercise
9 \Scontents{
10   First solution.
11 }
12 \end{ex}
13
14 \begin{ex}
15 Second exercise
16 \Scontents{
17   Second solution.
18 }
19 \end{ex}
20
21 \section{Solutions}
22 \foreach \i in {1,...,\countsc{solutions}} {
23   \noindent\textbf{\i} \getstored[\i]{solutions}\par
24 }
25 \end{document}

```

7.2 From filecontentsdef package

Example 2

Adaptation of example from package filecontentsdef .

```

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

```

7.3 From TeX-SX

Example 3

Adapted from [LaTeX equivalent of ConTeXt buffers](#) .

```

1 \documentclass{article}
2 \usepackage[store-cmd=tikz]{scontents}
3 \usepackage{tikz}
4 \pagestyle{empty}
5 \Scontents*\matrix{ \node (a) {$a$} ; & \node (b) {$b$} ; \\ } ;}
6 \Scontents*\matrix[ampersand replacement=\&]
7 { \node (a) {$a$} ; \& \node (b) {$b$} ; \\ } ;}
8 \Scontents*\matrix{\node (a) {$a$} ; & \node (b) {$b$} ; \\ } ; }
9 \begin{document}
10 \section{tikzpicture}
11 \begin{tikzpicture}
12 \getstored[1]{tikz}
13 \end{tikzpicture}
14
15 \begin{tikzpicture}
16 \getstored[2]{tikz}
17 \end{tikzpicture}
18
19 \begin{tikzpicture}
20 \getstored[3]{tikz}
21 \end{tikzpicture}
22
23 \section{source}
24 \typestored[1]{tikz}
25 \typestored[2]{tikz}
26 \typestored[3]{tikz}
27 \end{document}

```

Example 4

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

```

1 \documentclass{article}
2 \usepackage{scontents}
3 \usepackage{pgffor}
4 \pagestyle{empty}
5 \begin{document}
6 \begin{scontents}[store-env=a]
7 Something for a
8 \end{scontents}
9
10 \begin{scontents}[store-env=a]
11 Something for b
12 \end{scontents}
13
14 \begin{scontents}[store-env=a]
15 Something with no label
16 \end{scontents}
17
18 \textbf{Let's print them}
19
20 This is a: \getstored[1]{a}\par
21 This is b: \getstored[2]{a}
22
23 \textbf{Print all of them}
24
25 \foreach \i in {1,...,\countsc{a}} {\getstored[\i]{a}\par}
26 \end{document}

```

Example 5

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

```

1 \documentclass{article}
2 \usepackage{scontents}

```




```

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

```

7.4 Customization of verbatimsc

Example 6

Customization of `verbatimsc` using the `fancyvrb` and `tcolorbox` package .

```

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

```

```

45 \section{Test \texttt{\textbackslash typestored}}
46 \typestored[1]{contents}
47 \typestored[2]{contents}
48 \end{document}

```

Example 7

Customization of `verbatimsc` using the listings package .

```

1 \documentclass{article}
2 \usepackage{scontents}
3 \makeatletter
4 \let\verbatimsc@undefined
5 \let\endverbatimsc@undefined
6 \makeatother
7 \usepackage{xcolor}
8 \usepackage{listings}
9 \lstnewenvironment{verbatimsc}
10 {
11   \lstset{
12     basicstyle=\small\ttfamily,
13     breaklines=true,
14     columns=fullflexible,
15     language=[LaTeX]TeX,
16     numbers=left,
17     numbersep=1em,
18     numberstyle=\tiny\color{gray},
19     keywordstyle=\color{red}
20   }
21 }{}
22 \setlength{\parindent}{0pt}
23 \pagestyle{empty}
24 \begin{document}
25 \section{Test \texttt{\textbackslash begin\{scontents\}} whit \texttt{listings}}
26 Test \verb+{scontents}+ \par
27
28 \begin{scontents}
29 Using \verb+scontents+ env no \verb+[key=val]+, save in seq \verb+contents+ with index 1.\par
30
31 Prove \lstinline[basicstyle=\ttfamily]| \lstinline | and environment \verb+Verbatim*+
32 \begin{verbatim}
33   verbatim environment
34 \end{verbatim}
35 \end{scontents}
36
37 \section{Test \texttt{\textbackslash Scontents*} whit \texttt{listings}}
38
39 \Scontents*+ We have coded this in \lstinline[basicstyle=\ttfamily]|\LaTeX:  $E=mc^2$ 
40 and more.+
41
42 \section{Test \texttt{\textbackslash getstored}}
43 \getstored[2]{contents}\par
44 \getstored[1]{contents}
45
46 \section{Test \texttt{\textbackslash typestored}}
47 \typestored[1]{contents}
48 \typestored[2]{contents}
49 \end{document}

```

Example 8

Customization of `verbatimsc` using the minted package .

```

1 % need ---shell-escape
2 \documentclass{article}
3 \usepackage{scontents}
4 \makeatletter
5 \let\verbatimsc@undefined
6 \let\endverbatimsc@undefined
7 \makeatother
8 \usepackage{minted}

```

```

9 \newminted{tex}{linenos}
10 \newenvironment{verbatimsc}{\VerbatimEnvironment\begin{texcode}}{\end{texcode}}
11 \pagestyle{empty}
12 \begin{document}
13 \section{Test \texttt{\textbackslash begin\{scontents\}} whit \texttt{minted}}
14 Test \verb+{scontents}+ \par
15
16 \begin{scontents}[write-env=usingtab.tex,force-eol=true]
17 Using \verb+scontents+ env no \verb+[key=val]+, save in seq \verb+contents+
18 with index 1.\par
19
20 Prove new \Verb*{ new fvextra whit braces } and environment \verb+Verbatim*+
21 % Real TABs here :)
22 \begin{Verbatim}[obeytabs, showtabs, tab=\rightarrowfill, tabcolor=red,showspaces, spacecolor=blue]
23 No tab
24         One real tab
25             Two real Tab plus         one tab
26 \end{Verbatim}
27 \end{scontents}
28
29 \section{Test \texttt{\textbackslash Scontents} whit \texttt{minted}}
30
31 \Scontents{ We have coded \par this in \LaTeX:  $E=mc^2$ .}
32
33 \section{Test \texttt{\textbackslash getstored}}
34 \getstored[1]{contents}\par
35 \getstored[2]{contents}
36
37 \section{Test \texttt{\textbackslash typestored}}
38 \typestored[1]{contents}
39 \typestored[2]{contents}
40 \end{document}

```

8 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` package to the current version with only `expl3`.

- 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 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 `\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:D`.
- v1.1 (ctan), 2019/08/12**
 - Extension of documentation.
 - Replace `\tex_endinput:D` by `\file_input_stop:`.
- v1.0 (ctan), 2019/07/30**
 - First public release,

9 Index of Documentation

C

Commands provide by `scontents`

<code>\Scontents*</code>	3, 4
<code>\Scontents</code>	2–4
<code>\cleanseqsc</code>	5
<code>\countsc</code>	5
<code>\getstored</code>	3, 4
<code>\meaningsc</code>	2, 3, 5
<code>\setupsc</code>	2–4
<code>\typestored</code>	2–5

E

Environment provide by `scontents`:

<code>scontents</code>	2, 3
<code>verbatimsc</code>	4, 5, 9, 10

Environments

<code>filecontentsdefmacro</code>	1
-----------------------------------	---

K

Keys

<code>force-eol</code>	2
<code>print-all</code>	2
<code>print-cmd</code>	4
<code>print-env</code>	3
<code>store-all</code>	2
<code>store-cmd</code>	4
<code>store-env</code>	3
<code>verb-font</code>	2
<code>width-tab</code>	2
<code>write-env</code>	4

<code>write-out</code>	4
------------------------	---

L

<code>\lstinline</code>	4
-------------------------	---

M

<code>\meaning</code>	5
-----------------------	---

P

Packages

<code>answers</code>	7
<code>expl3</code>	1, 6, 11
<code>fancyvrb</code>	2, 5, 9
<code>filecontentsdef</code>	1, 2, 6, 7, 11
<code>fvextra</code>	4
<code>l3keys2e</code>	1
<code>l3seq</code>	1, 6
<code>listings</code>	2, 4, 5, 10
<code>minted</code>	5, 10
<code>scontents</code>	1, 2, 6, 11
<code>tcolorbox</code>	9
<code>verbatim</code>	4
<code>xparse</code>	1

S

<code>\setupsc</code>	3
-----------------------	---

V

<code>\Verb</code>	4
<code>\verb</code>	4

10 Implementation

10.1 Declaration of the package

First we set up the module name for l3doc:

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

Then, we can give the traditional declaration of a package written with expl3 and the necessary packages for its operation.

```
2 \RequirePackage{l3keys2e}
3 \RequirePackage{xparse}[2019/05/28]
4 \ProvidesExplPackage{scontents}{2019/10/03}{1.4}
5   {Stores LaTeX contents in memory or files}
```

A check to make sure that expl3 is not too old

```
6 \ifpackagelater { expl3 } { 2019/09/19 }
7   { }
8   {
9     \PackageError { scontents } { Support-package~expl3~too-old }
10    {
11      You~need~to~update~your~installation~of~the~bundles~
12      'l3kernel'~and~'l3packages'.\MessageBreak
13      Loading~scontents~will~abort!
14    }
15    \file_input_stop:
16  }
```

10.2 Definition of common keys

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.

```
17 \keys_define:nn { scontents }
18   {
19     store-env .tl_set:N      = \l__scontents_name_seq_env_tl,
20     store-env .initial:n     = contents,
21     store-env .value_required:n = true,
22     store-cmd .tl_set:N      = \l__scontents_name_seq_cmd_tl,
23     store-cmd .initial:n     = contents,
24     store-cmd .value_required:n = true,
25     verb-font .tl_set:N      = \l__scontents_verb_font_tl,
26     verb-font .initial:n     = \ttfamily,
27     verb-font .value_required:n = true,
28     print-env .bool_set:N     = \l__scontents_print_env_bool,
29     print-env .initial:n     = false,
30     print-env .default:n     = true,
31     print-cmd .bool_set:N     = \l__scontents_print_cmd_bool,
32     print-cmd .initial:n     = false,
33     print-cmd .default:n     = true,
34     force-eol .bool_set:N     = \l__scontents_forced_eol_bool,
35     force-eol .initial:n     = false,
36     force-eol .default:n     = true,
37     width-tab .int_set:N      = \l__scontents_tab_width_int,
38     width-tab .initial:n     = 1,
39     width-tab .value_required:n = true,
40     print-all .meta:n        = { print-env = #1 , print-cmd = #1 },
41     print-all .default:n     = true,
42     store-all .meta:n        = { store-env = #1 , store-cmd = #1 },
43     store-all .value_required:n = true
44   }
```

We process the *<keys>* as options passed on to the package, the package l3keys2e will verify the *<keys>* and will return an error when they are *unknown*.

```
45 \ProcessKeysOptions { scontents }
```

10.3 Internal variables

Now we declare the internal variables we will use.

`\l__scontents_macro_tmp_tl` `\l__scontents_macro_tmp_tl` is a temporary token list to hold the contents of the macro/environment, `\l__scontents_fname_out_tl` is used as the name of the output file, when there's one, `\l__scontents_file_tl` holds the contents of an environment as it's being read, and `\l__scontents_temp_tl` and `\g__scontents_temp_tl` are generic temporary token lists.

```

46 \tl_new:N \l__scontents_macro_tmp_tl
47 \tl_new:N \l__scontents_fname_out_tl
48 \tl_new:N \l__scontents_temp_tl
49 \tl_new:N \l__scontents_file_tl
50 \tl_new:N \g__scontents_temp_tl

```

(End definition for `\l__scontents_macro_tmp_tl` and others.)

`\l__scontents_seq_item_int` `\l__scontents_seq_item_int` stores the index in the sequence of the item requested to `\typestored` or `\meaningsc`. `\l__scontents_env_nesting_int` stores the current nesting level of the scontents environment.

```

51 \int_new:N \l__scontents_seq_item_int
52 \int_new:N \l__scontents_env_nesting_int
53 \int_new:N \l__scontents_tmpa_int

```

(End definition for `\l__scontents_seq_item_int`, `\l__scontents_env_nesting_int`, and `\l__scontents_tmpa_int`.)

`\l__scontents_writing_bool` The boolean `\l__scontents_writing_bool` keeps track of whether we should write to a file, and `\l__scontents_storing_bool` determines whether it is in write-only mode when the `write-out` option is used.

```

54 \bool_new:N \l__scontents_writing_bool
55 \bool_set_false:N \l__scontents_writing_bool
56 \bool_new:N \l__scontents_storing_bool
57 \bool_set_true:N \l__scontents_storing_bool

```

(End definition for `\l__scontents_writing_bool` and `\l__scontents_storing_bool`.)

`\g__scontents_end_verbatimsc_tl` A token list to match when ending `verbatimsc` and `scontents` environments.

```

58 \tl_new:N \g__scontents_end_verbatimsc_tl
59 \tl_gset_rescan:Nnn
60   \g__scontents_end_verbatimsc_tl
61   {
62     \char_set_catcode_other:N \
63     \char_set_catcode_other:N \{
64     \char_set_catcode_other:N \}
65   }
66   { \end{verbatimsc} }
67 \tl_const:Nx \c__scontents_end_env_tl
68   { \c_backslash_str end \c_left_brace_str scontents \c_right_brace_str }

```

(End definition for `\g__scontents_end_verbatimsc_tl` and `\c__scontents_end_env_tl`.)

`\c__scontents_hidden_space_str` `\c__scontents_hidden_space_str` is a constant *string* to used to hide the *forced space* added by `TeX` when recording content in a macro. This *string* contains the *reserved phrase* “`%^^Ascheol%`” which is added to the end of the argument stored in `seq` when the key `force-eol` is false.

```

69 \str_const:Nx \c__scontents_hidden_space_str
70   { \c_percent_str \c_circumflex_str \c_circumflex_str A scheol \c_percent_str }

```

(End definition for `\c__scontents_hidden_space_str`.)

`\q__scontents_stop` Some quarks used along the code as macro delimiters.

```

71 \quark_new:N \q__scontents_stop
72 \quark_new:N \q__scontents_mark

```

(End definition for `\q__scontents_stop` and `\q__scontents_mark`.)

`\l__scontents_file_iow` An output stream for saving the contents of an environment to a file.

```

73 \iow_new:N \l__scontents_file_iow

```

(End definition for `\l__scontents_file_iow`.)

`__scontents_rescan_tokens:n` `\tl_rescan:n` doesn't fit the needs of this package because it does not allow catcode changes inside the argument, so verbatim commands used inside one of `scontents`'s commands/environments will not work. Here we create a private copy of `\tex_scantokens:D` which will serve our purposes.

```
74 \cs_new_protected:Npn \__scontents_rescan_tokens:n #1 { \tex_scantokens:D {#1} }
75 \cs_generate_variant:Nn \__scontents_rescan_tokens:n { V, x }
```

(End definition for `__scontents_rescan_tokens:n`.)

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

```
76 \cs_new:Npx \__scontents_tab: { \c_space_tl }
77 \cs_new:Npn \__scontents_par: { ^^J ^^J }
```

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

`\tl_remove_once:N` Some nonstandard variants.

```
\tl_replace_all:Nxx
\tl_replace_all:Nxn
\tl_replace_all:Nnx
\tl_if_empty:fTF
78 \cs_generate_variant:Nn \tl_remove_once:Nn { NV }
79 \cs_generate_variant:Nn \tl_replace_all:Nnn { Nx, Nxx, Nnx }
80 \prg_generate_conditional_variant:Nnn \tl_if_empty:n { f } { TF }
```

(End definition for `\tl_remove_once:N`, `\tl_replace_all:Nxx`, and `\tl_if_empty:fTF`.)

10.4 Defining keys for the environment and commands

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

10.4.1 Keys for environment `scontents`

We define a set of `<keys>` for environment `scontents`.

```
81 \keys_define:nn { scontents / scontents }
82 {
83   write-env .code:n = {
84     \bool_set_true:N \l__scontents_writing_bool
85     \tl_set:Nn \l__scontents_fname_out_tl {#1}
86   },
87   write-out .code:n = {
88     \bool_set_false:N \l__scontents_storing_bool
89     \bool_set_true:N \l__scontents_writing_bool
90     \tl_set:Nn \l__scontents_fname_out_tl {#1}
91   },
92   write-env .value_required:n = true,
93   write-out .value_required:n = true,
94   print-env .meta:nn = { scontents } { print-env = #1 },
95   print-env .default:n = true,
96   store-env .meta:nn = { scontents } { store-env = #1 },
97   force-eol .meta:nn = { scontents } { force-eol = #1 },
98   force-eol .default:n = true,
99   unknown .code:n = { \__scontents_parse_environment_keys:n {#1} }
100 }
```

10.4.2 Keys for command `\Scontents`

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

```
101 \keys_define:nn { scontents / Scontents }
102 {
103   print-cmd .meta:nn = { scontents } { print-cmd = #1 },
104   print-cmd .default:n = true,
105   store-cmd .meta:nn = { scontents } { store-cmd = #1 },
106   force-eol .meta:nn = { scontents } { force-eol = #1 },
107   force-eol .default:n = true,
108   unknown .code:n = { \__scontents_parse_command_keys:n {#1} }
109 }
```

10.4.3 Key for commands \typestored and \meaningsc

We define a *key* for command `\typestored` and `\meaningsc`. Both commands accept the same type of optional arguments, just define a common *key*.

```

110 \keys_define:nn { scontents / typemeaning }
111 {
112   width-tab .meta:nn = { scontents } { width-tab = #1 },
113   unknown   .code:n  = { \__scontents_parse_type_meaning_key:n {#1} }
114 }

```

10.5 Handling undefined keys

The *keys* are stored in the token list variable `\l_keys_key_tl`, and the value (if any) is passed as an argument to each *function*.

10.5.1 Undefined keys for environment scontents

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

```

115 \cs_new_protected:Npn \__scontents_parse_environment_keys:n #1
116 { \exp_args:NV \__scontents_parse_environment_keys:nn \l_keys_key_tl {#1} }
117 \cs_new_protected:Npn \__scontents_parse_environment_keys:nn #1#2
118 {
119   \tl_if_blank:nTF {#2}
120   { \msg_error:nnn { scontents } { env-key-unknown } {#1} }
121   { \msg_error:nnnn { scontents } { env-key-value-unknown } {#1} {#2} }
122 }

```

(End definition for `__scontents_parse_environment_keys:n` and `__scontents_parse_environment_keys:nn`.)

10.5.2 Undefined keys for \Scontents and \Scontents*

We check the *keys* passed to commands `\Scontents` or `\Scontents*` and process it with `__scontents_parse_command_keys:n` if the *key* is *unknown* we return an error message.

```

123 \cs_new_protected:Npn \__scontents_parse_command_keys:n #1
124 { \exp_args:NV \__scontents_parse_command_keys:nn \l_keys_key_tl {#1} }
125 \cs_new_protected:Npn \__scontents_parse_command_keys:nn #1#2
126 {
127   \tl_if_blank:nTF {#2}
128   { \msg_error:nnn { scontents } { cmd-key-unknown } {#1} }
129   { \msg_error:nnnn { scontents } { cmd-key-value-unknown } {#1} {#2} }
130 }

```

(End definition for `__scontents_parse_command_keys:n` and `__scontents_parse_command_keys:nn`.)

10.5.3 Undefined keys for \typestored and \meaningsc

The commands `\typestored` and `\meaningsc` accept an optional argument for setting the `width-tab` to print the stored contents. However their optional argument also contains the number of the item to retrieve from the stored sequence. To avoid the awkward `\typestored[][\options]{...}` syntax, we'll make the commands have a single optional argument which is processed by `l3keys`, and the unknown keys are brought here to `__scontents_parse_type_meaning_key:n` to process.

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

```

131 \cs_new_protected:Npn \__scontents_parse_type_meaning_key:n #1
132 { \exp_args:NV \__scontents_parse_type_meaning_key:nn \l_keys_key_tl {#1} }
133 \cs_new_protected:Npn \__scontents_parse_type_meaning_key:nn #1#2
134 {
135   \tl_if_empty:FTF { \int_to_roman:n { -0 #1 } }
136   {
137     \tl_if_blank:nTF {#2}
138     { \int_set:Nn \__scontents_seq_item_int {#1} }
139     { \msg_error:nnnn { scontents } { type-key-value-unknown } {#1} {#2} }
140   }
141   {
142     \tl_if_blank:nTF {#2}

```



```

143         { \msg_error:nnn { scontents } { type-key-unknown } {#1} }
144         { \msg_error:nnnn { scontents } { type-key-value-unknown } {#1} {#2} }
145     }
146 }

```

(End definition for `__scontents_parse_type_meaning_key:n` and `__scontents_parse_type_meaning_key:nn`.)

10.6 Programming of the sequences

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

`__scontents_append_contents:nn` `__scontents_append_contents:nn` creates a seq variable if one didn't exist and appends the contents in the argument to the right of the sequence.

```

147 \cs_new_protected:Npn \__scontents_append_contents:nn #1#2
148 {
149     \tl_if_blank:nT {#1}
150     { \msg_error:nn { scontents } { empty-store-cmd } }
151     \seq_if_exist:cF { g__scontents_name_#1_seq }
152     { \seq_new:c { g__scontents_name_#1_seq } }
153     \seq_gput_right:cn { g__scontents_name_#1_seq } {#2}
154 }
155 \cs_generate_variant:Nn \__scontents_append_contents:nn { Vx }

```

(End definition for `__scontents_append_contents:nn`.)

`__scontents_getfrom_seq:nn` `__scontents_getfrom_seq:nn` retrieves the saved item from the sequence.

```

156 \cs_new:Npn \__scontents_getfrom_seq:nn #1#2
157 {
158     \seq_if_exist:cTF { g__scontents_name_#2_seq }
159     {
160         \exp_args:Nf \__scontents_getfrom_seq:nnn
161         { \seq_count:c { g__scontents_name_#2_seq } }
162         {#1} {#2}
163     }
164     { \msg_expandable_error:nnn { scontents } { undefined-storage } {#2} }
165 }
166 \cs_new:Npn \__scontents_getfrom_seq:nnn #1#2#3
167 {
168     \bool_lazy_or:nnTF
169     { \int_compare_p:nNn {#2} = { 0 } }
170     { \int_compare_p:nNn { \int_abs:n {#2} } > {#1} }
171     { \msg_expandable_error:nnnn { scontents } { index-out-of-range } {#2} {#3} {#1} }
172     { \seq_item:cn { g__scontents_name_#3_seq } {#2} }
173 }

```

(End definition for `__scontents_getfrom_seq:nn` and `__scontents_getfrom_seq:nnn`.)

`__scontents_lastfrom_seq:n` `__scontents_lastfrom_seq:n` retrieves the last saved item from the sequence when `\l__scontents__print_env_bool` or `\l__scontents__print_cmd_bool` is true.

```

174 \cs_new_protected:Npn \__scontents_lastfrom_seq:n #1
175 {
176     \tl_gset:Nx \g__scontents_temp_tl { \seq_item:cn { g__scontents_name_#1_seq } {-1} }
177     \group_insert_after:N \__scontents_rescan_tokens:V
178     \group_insert_after:N \g__scontents_temp_tl
179     \group_insert_after:N \tl_gclear:N
180     \group_insert_after:N \g__scontents_temp_tl
181 }

```

(End definition for `__scontents_lastfrom_seq:n`.)

`__scontents_store_to_seq:NN` The `__scontents_store_to_seq:NN` writes the recorded contents in #1 to the log and stores it in #2.

```

182 \cs_new_protected:Npn \__scontents_store_to_seq:NN #1#2
183 {
184     \tl_log:N #1
185     \__scontents_append_contents:Vx #2 { \exp_not:V #1 }
186 }

```

(End definition for `__scontents_store_to_seq:NN`.)

10.7 Construction of environment scontents

We define the environment scontents, next to the system `[(key = val)]`. The environment is divided into three parts. This implementation is taken from answer by Enrico Gregorio in <https://tex.stackexchange.com/a/487746/7832>.

`scontents` This is the main environment used in the document.

```

187 \NewDocumentEnvironment { scontents } { }
188 {
189     \char_set_catcode_active:N ^^M
190     \__scontents_start_environment:w
191 }
192 {
193     \__scontents_stop_environment:
194     \__scontents_atend_environment:
195 }
```

(End definition for scontents. This function is documented on page 3.)

10.7.1 key val for environment

Define a `[(key = val)]` for environment scontents

`__scontents_environment_inline:w` The macro `__scontents_environment_inline:w` is called from the scontents environment with the tokens following the `\begin{scontents}`. If the immediate next token (ignoring spaces) is a `[`, then we look for an optional argument delimited by a `]`. All the remaining tokens are treated as junk and an error is raised if they are non-blank.

```

196 \cs_new_protected:Npn \__scontents_environment_inline:w
197 {
198     \peek_charcode_ignore_spaces:NTF [ % ]
199     { \__scontents_environment_keys:w }
200     {
201         \__scontents_environment_junk:xw
202         { after~\c_backslash_str begin{scontents} }
203     }
204 }
205 \cs_new_protected:Npn \__scontents_environment_keys:w [ #1 ]
206 {
207     \keys_set:nn { scontents / scontents } {#1}
208     \__scontents_environment_junk:xw
209     { after~optional~argument~to~\c_backslash_str begin{scontents} }
210 }
211 \cs_new_protected:Npn \__scontents_environment_junk:nw #1 #2 \q__scontents_mark
212 {
213     \tl_if_blank:nF {#2}
214     { \msg_error:nnnn { scontents } { junk-after-begin } {#1} {#2} }
215 }
216 \cs_generate_variant:Nn \__scontents_environment_junk:nw { x }
```

(End definition for `__scontents_environment_inline:w`, `__scontents_environment_keys:w`, and `__scontents_environment_junk:nw`.)

10.7.2 The environment itself

`__scontents_start_environment:w` 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 line directly after `\begin{scontents}` contains an optional argument enclosed in `[...]` or other tokens. The trailing tokens are treated as junk and an error is raised. The `__scontents_environment_inline:w` macro checks for those cases.

`__scontents_start_environment:w` calls the `__scontents_file_tl_write_start:V` function, 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, `__scontents_file_write_stop:N` does the cleanup and the read token list is smuggled out of the verbatim group. This part of the code is inspired and adapted from the code of the package `xsmverb` by Clemens Niederberger.

```

217 \group_begin:
218     \char_set_catcode_active:N ^^I
```

```

219 \char_set_catcode_active:N ^^L
220 \char_set_catcode_active:N ^^M
221 \cs_new_protected:Npn \__scontents_start_environment:w #1 ^^M
222 {
223   \__scontents_environment_inline:w #1 \q__scontents_mark
224   \__scontents_make_control_chars_active:
225   \group_begin:
226     \__scontents_file_tl_write_start:V \l__scontents_fname_out_tl
227 }
228 \cs_new_protected:Npn \__scontents_stop_environment:
229 {
230   \__scontents_file_write_stop:N \l__scontents_macro_tmp_tl
231   \exp_args:NNNV
232   \group_end:
233   \tl_set:Nn \l__scontents_macro_tmp_tl \l__scontents_macro_tmp_tl
234 }

```

(End definition for `__scontents_start_environment:w` and `__scontents_stop_environment:.`)

```

\__scontents_file_tl_write_start:n
\__scontents_file_tl_write_start:V
\__scontents_verb_processor_iterate:w
\__scontents_file_write_stop:N
\__scontents_remove_leading_nl:n
\__scontents_remove_leading_nl:w

```

This is the main macro to collect the contents of a verbatim environment. The macro starts a group, opens the output file, if necessary, sets verbatim catcodes, and then issues `^^M` (set equal to `__scontents_ret:w`) to read 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).

```

235 \cs_new_protected:Npn \__scontents_file_tl_write_start:n #1
236 {
237   \group_begin:
238   \bool_if:NT \l__scontents_writing_bool
239   {
240     \file_if_exist:nTF {#1}
241     { \msg_warning:nxx { scontents } { rewriting-file } {#1} }
242     { \msg_warning:nxx { scontents } { writing-file } {#1} }
243     \iow_open:Nn \l__scontents_file_iow {#1}
244   }
245   \tl_clear:N \l__scontents_file_tl
246   \seq_map_function:NN \l_char_special_seq \char_set_catcode_other:N
247   \int_step_function:nnN { 128 } { 255 } \char_set_catcode_letter:n
248   \cs_set_protected:Npx \__scontents_ret:w ##1 ^^M
249   {
250     \exp_not:N \__scontents_verb_processor_iterate:w
251     ##1 \c__scontents_end_env_tl
252     \c__scontents_end_env_tl
253     \exp_not:N \q__scontents_stop
254   }
255   \__scontents_make_control_chars_active:
256   \__scontents_ret:w
257 }
258 \use:x
259 {
260   \cs_new:Npn \exp_not:N \__scontents_verb_processor_iterate:w
261   ##1 \c__scontents_end_env_tl
262   ##2 \c__scontents_end_env_tl
263   ##3 \exp_not:N \q__scontents_stop
264 } {
265   \tl_if_blank:nTF {#3}
266   {
267     \__scontents_analyse_nesting:n {#1}
268     \__scontents_verb_processor_output:n {#1}
269   }
270   {
271     \__scontents_if_nested:TF
272     {
273       \__scontents_nesting_decr:
274       \__scontents_verb_processor_output:x
275       { \exp_not:n {#1} \c__scontents_end_env_tl \exp_not:n {#2} }

```

```

276         }
277     {
278         \tl_if_blank:nF {#1}
279         { \__scontents_verb_processor_output:n {#1} }
280         \cs_set_protected:Npx \__scontents_ret:w
281         {
282             \exp_not:N \end{scontents}
283             \bool_lazy_or:nnF
284             { \tl_if_blank_p:n {#2} }
285             { \str_if_eq_p:ee {#2} { \c_percent_str } }
286             {
287                 \msg_warning:nnn { scontents } { rescanning-text } {#2}
288                 \__scontents_rescan_tokens:n {#2}
289             }
290         }
291         \char_set_active_eq:NN ^^M \__scontents_ret:w
292     }
293 }
294 ^^M
295 }
296 \cs_new_protected:Npn \__scontents_file_write_stop:N #1
297 {
298     \bool_if:NT \l__scontents_writing_bool
299     { \iow_close:N \l__scontents_file_iow }
300     \use:x
301     {
302         \group_end:
303         \bool_if:NT \l__scontents_storing_bool
304         {
305             \tl_set:Nn \exp_not:N #1
306             { \exp_args:NV \__scontents_remove_leading_nl:n \l__scontents_file_tl }
307         }
308     }
309 }
310 \cs_new:Npn \__scontents_remove_leading_nl:n #1
311 { \exp_not:o { \__scontents_remove_leading_nl:w #1 } }
312 \cs_new:Npn \__scontents_remove_leading_nl:w ^^J { }

```

(End definition for __scontents_file_tl_write_start:n and others.)

__scontents_verb_processor_output:n __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.

__scontents_analyse_nesting:n __scontents_analyse_nesting:n looks for nested \begin{scontents} and adds to a \l__scontents_env_nesting_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.

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

```

313 \cs_new_protected:Npn \__scontents_verb_processor_output:n #1
314 {
315     \bool_if:NT \l__scontents_writing_bool
316     { \iow_now:Nn \l__scontents_file_iow {#1} }
317     \bool_if:NT \l__scontents_storing_bool
318     { \tl_put_right:Nn \l__scontents_file_tl { ^^J #1 } }
319 }
320 \cs_generate_variant:Nn \__scontents_verb_processor_output:n { x }
321 \cs_new_protected:Npx \__scontents_analyse_nesting:n #1
322 {
323     \int_zero:N \l__scontents_tmpa_int
324     \exp_not:N \__scontents_analyse_nesting:w #1
325     \c_backslash_str begin
326     \c_left_brace_str \exp_not:N \q__scontents_mark \c_right_brace_str
327     \exp_not:N \q__scontents_stop
328     \int_compare:nNnT { \l__scontents_tmpa_int } > { 1 }
329     { \msg_warning:nn { scontents } { multiple-begin } }
330 }
331 \use:x
332 {
333     \cs_new_protected:Npn \exp_not:N \__scontents_analyse_nesting:w ##1
334     \c_backslash_str begin \c_left_brace_str ##2 \c_right_brace_str

```

```

335   } {
336       \if_meaning:w \q__scontents_mark #2
337       \exp_after:wN \use_i:nn
338   \else:
339       \exp_after:wN \use_ii:nn
340   \fi:
341       { \__scontents_use_none_delimit_by_q_stop:w }
342       {
343           \str_if_eq:eeT {#2} {scontents}
344           {
345               \int_incr:N \l__scontents_env_nesting_int
346               \int_incr:N \l__scontents_tmpa_int
347               \__scontents_analyse_nesting:w
348           }
349           \__scontents_analyse_nesting:w
350       }
351   }
352   \cs_new_protected:Npn \__scontents_nesting_decr:
353   { \int_decr:N \l__scontents_env_nesting_int }
354   \prg_new_protected_conditional:Npnn \__scontents_if_nested: { TF }
355   {
356       \int_compare:nNnTF { \l__scontents_env_nesting_int } > { \c_zero_int }
357       { \prg_return_true: }
358       { \prg_return_false: }
359   }
360   \cs_new:Npn \__scontents_use_none_delimit_by_q_stop:w #1 \q__scontents_stop { }
361   \group_end:
362   \cs_generate_variant:Nn \__scontents_file_tl_write_start:n { V }

```

(End definition for `__scontents_verb_processor_output:n` and others.)

10.7.3 Recording of the content in the sequence

`__scontents_atend_environment:` Finishes the environment by optionally calling `__scontents_store_to_seq:` and then clearing the temporary token list.

```

363 \cs_new_protected:Npn \__scontents_atend_environment:
364 {
365     \bool_if:NT \l__scontents_storing_bool
366     {
367         \bool_if:NF \l__scontents_forced_eol_bool
368         { \tl_put_right:Nx \l__scontents_macro_tmp_tl { \c__scontents_hidden_space_str } }
369         \__scontents_store_to_seq:NN \l__scontents_macro_tmp_tl \l__scontents_name_seq_env_tl
370         \bool_if:NT \l__scontents_print_env_bool
371         { \__scontents_lastfrom_seq:n \l__scontents_name_seq_env_tl }
372     }
373 }

```

(End definition for `__scontents_atend_environment:.`)

10.8 The `\Scontents` command

User command to *⟨stored content⟩*, adapted from <https://tex.stackexchange.com/a/500281/7832>.

`\Scontents` The `\Scontents` macro starts by parsing an optional argument and then delegates to `__scontents_verb_arg:w` or `__scontents_norm_arg:n` depending whether a star (*) argument is present. `__scontents_norm_arg:n` grabs a normal argument, adds it to the seq variable, and optionally prints it.

`__scontents_verb_arg:w` grabs a verbatim argument using xparse's +v argument parser.

```

374 \NewDocumentCommand { \Scontents }{ !s !0{ } }
375 {
376     \group_begin:
377     \IfNoValueF {#2}
378     { \keys_set:nn { scontents / Scontents } {#2} }
379     \char_set_catcode_active:n { 9 }
380     \IfBooleanTF {#1}
381     { \__scontents_verb_arg:w }
382     { \__scontents_norm_arg:n }
383 }

```

```

384 \cs_new_protected:Npn \__scontents_norm_arg:n #1
385 {
386     \tl_set:Nx \l__scontents_temp_tl { \exp_not:n {#1} }
387     \tl_put_right:Nx \l__scontents_temp_tl { \c__scontents_hidden_space_str }
388     \__scontents_store_to_seq:NN \l__scontents_temp_tl \l__scontents_name_seq_cmd_tl
389     \bool_if:NT \l__scontents_print_cmd_bool
390     { \__scontents_lastfrom_seq:n \l__scontents_name_seq_cmd_tl }
391     \group_end:
392 }
393 \NewDocumentCommand { \__scontents_verb_arg:w } { +v }
394 {
395     \tl_set:Nx \l__scontents_temp_tl { \exp_not:n {#1} }
396     \tl_replace_all:Nxx \l__scontents_temp_tl { \iow_char:N ^^M } { \iow_char:N ^^J }
397     \bool_if:NF \l__scontents_forced_eol_bool
398     { \tl_put_right:Nx \l__scontents_temp_tl { \c__scontents_hidden_space_str } }
399     \__scontents_store_to_seq:NN \l__scontents_temp_tl \l__scontents_name_seq_cmd_tl
400     \bool_if:NT \l__scontents_print_cmd_bool
401     { \__scontents_lastfrom_seq:n \l__scontents_name_seq_cmd_tl }
402     \group_end:
403 }

```

(End definition for `\Scontents`, `__scontents_norm_arg:n`, and `__scontents_verb_arg:w`. This function is documented on page 4.)

10.9 The command `\getstored`

`\getstored` User command `\getstored` to extract *stored content* in seq (robust).

```

404 \NewDocumentCommand { \getstored } { 0{1} m }
405 {
406     \group_begin:
407     \__scontents_rescan_tokens:x
408     { \__scontents_getfrom_seq:nn {#1} {#2} }
409     \group_end:
410 }

```

(End definition for `\getstored`. This function is documented on page 4.)

10.10 The `\typestored` command

This implementation is an adaptation taken from answer by Phelype Oleinik in (<https://tex.stackexchange.com/a/497651/7832>).

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

`__scontents_verb_print:N`
`__scontents_xverb:w`
`verbatimsc`

```

411 \NewDocumentCommand { \typestored } { o m }
412 {
413     \group_begin:
414     \int_set:Nn \l__scontents_seq_item_int { 1 }
415     \IfValueT {#1} { \keys_set:nn { scontents / typemeaning } {#1} }
416     \tl_set:Nx \l__scontents_temp_tl
417     {
418         \exp_args:NV \__scontents_getfrom_seq:nn \l__scontents_seq_item_int {#2}
419     }
420     \tl_remove_once:NV \l__scontents_temp_tl \c__scontents_hidden_space_str
421     \tl_log:N \l__scontents_temp_tl
422     \tl_if_empty:NF \l__scontents_temp_tl
423     { \__scontents_verb_print:N \l__scontents_temp_tl }
424     \group_end:
425 }

```

The `__scontents_verb_print:N` macro is defined with active carriage return (ASCII 13) characters to mimick an actual verbatim environment “on the loose”. The contents of the environment are placed in a `verbatimsc` environment and rescanned using `__scontents_rescan_tokens:x`.

```

426 \group_begin:
427     \char_set_catcode_active:N ^^M
428     \cs_new_protected:Npn \__scontents_verb_print:N #1
429     {
430         \tl_if_blank:VT #1

```

```

431     { \msg_error:nnn { scontents } { empty-variable } {#1} }
432     \cs_set_eq:NN \__scontents_verb_print_EOL: ^^M
433     \cs_set_eq:NN ^^M \scan_stop:
434     \cs_set_eq:cN { do@noligs } \__scontents_do_noligs:N
435     \__scontents_rescan_tokens:x
436     {
437         \exp_not:N \begin{verbatim} ^^M
438         \exp_not:N #1 ^^M
439         \g__scontents_end_verbatim_sc_tl
440     }
441     \cs_set_eq:NN ^^M \__scontents_verb_print_EOL:
442 }
443 \group_end:

```

Finally, the `verbatim` environment is defined.

```

444 \cs_new_protected:Npn \__scontents_xverb:
445 {
446     \char_set_catcode_active:n { 9 }
447     \char_set_active_eq:nN { 9 } \__scontents_tabs_to_spaces:
448     \__scontents_xverb:w
449 }
450 \use:x
451 {
452     \cs_new_protected:Npn \exp_not:N \__scontents_xverb:w
453     ##1 \g__scontents_end_verbatim_sc_tl
454 }
455 { #1 \end{verbatim} }
456 \NewDocumentEnvironment { verbatim } { }
457 {
458     \cs_set_eq:cN { @xverbatim } \__scontents_xverb:
459     \verbatim
460 }
461 { }

```

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

10.11 Some auxiliaries

`__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.

```

462 \cs_new:Npn \__scontents_tabs_to_spaces:
463 { \prg_replicate:nn { \l__scontents_tab_width_int } { ~ } }

```

(End definition for `__scontents_tabs_to_spaces:`)

`__scontents_do_noligs:N` `__scontents_do_noligs:N` is an alternative definition for $\text{\LaTeX}_{2\epsilon}$'s `\do@noligs` which makes sure to not consume following space tokens. The $\text{\LaTeX}_{2\epsilon}$ version ends with `\char`#1`, which leaves \TeX still looking for an *optional space*. This version uses `\char_generate:nn` to ensure that doesn't happen.

```

464 \cs_new:Npn \__scontents_do_noligs:N #1
465 {
466     \char_set_catcode_active:N #1
467     \char_set_active_eq:Nc #1 { \__scontents_active_char_ \token_to_str:N #1 : }
468     \cs_set:cpx { \__scontents_active_char_ \token_to_str:N #1 : }
469     {
470         \mode_leave_vertical:
471         \tex_kern:D \c_zero_dim
472         \char_generate:nn { `#1 } { 12 }
473     }
474 }

```

(End definition for `__scontents_do_noligs:N`)

`__scontents_set_active_eq:NN` `__scontents_make_control_chars_active:` Shortcut definitions for common catcode changes.

```

475 \cs_new_protected:Npn \__scontents_set_active_eq:NN #1
476 {
477     \char_set_catcode_active:N #1

```

```

478     \char_set_active_eq:NN #1
479   }
480   \cs_new_protected:Npn \__scontents_make_control_chars_active:
481   {
482     \__scontents_set_active_eq:NN \^^I \__scontents_tab:
483     \__scontents_set_active_eq:NN \^^L \__scontents_par:
484     \__scontents_set_active_eq:NN \^^M \__scontents_ret:w
485   }

```

(End definition for __scontents_set_active_eq:NN and __scontents_make_control_chars_active:.)

10.12 The command \setupsc

User command \setupsc to setup module.

`\setupsc` A user-level wrapper for \keys_set:nn{ scontents }.

```

486 \NewDocumentCommand { \setupsc } { +m }
487 { \keys_set:nn { scontents } {#1} }

```

(End definition for \setupsc. This function is documented on page 2.)

10.13 The command meaningscs

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

```

488 \NewDocumentCommand { \meaningscs } { o m }
489 {
490   \group_begin:
491     \int_set:Nn \l__scontents_seq_item_int { 1 }
492     \IfValueT {#1} { \keys_set:nn { scontents / typemeaning } {#1} }
493     \__scontents_meaningscs:n {#2}
494   \group_end:
495 }
496 \group_begin:
497   \char_set_catcode_active:N \^^I
498   \cs_new_protected:Npn \__scontents_meaningscs:n #1
499   {
500     \tl_set:Nx \l__scontents_temp_tl
501     {
502       \exp_args:NV \__scontents_getfrom_seq:nn \l__scontents_seq_item_int {#1}
503     }
504     \tl_replace_all:Nxn \l__scontents_temp_tl { \iow_char:N \^^J } { ~ }
505     \tl_remove_once:NV \l__scontents_temp_tl \c__scontents_hidden_space_str
506     \tl_log:N \l__scontents_temp_tl
507     \tl_use:N \l__scontents_verb_font_tl
508     \tl_replace_all:Nxn \l__scontents_temp_tl { ^^I } { \__scontents_tabs_to_spaces: }
509     \cs_replacement_spec:N \l__scontents_temp_tl
510   }
511 \group_end:

```

(End definition for \meaningscs. This function is documented on page 5.)

10.14 The command \countsc

`\countsc` User command \countsc to count number of contents stored in seq.

```

512 \NewExpandableDocumentCommand { \countsc } { m }
513 { \seq_count:c { g__scontents_name_#1_seq } }

```

(End definition for \countsc. This function is documented on page 5.)

10.15 The command \cleanseqsc

`\cleanseqsc` A user command \cleanseqsc to clear (remove) a defined seq.

```

514 \NewDocumentCommand { \cleanseqsc } { m }
515 { \seq_clear_new:c { g__scontents_name_#1_seq } }

```

(End definition for \cleanseqsc. This function is documented on page 5.)

10.16 Warning and error messages

Warning and error messages used throughout the package.

```

516 \msg_new:nnn { scontents } { junk-after-begin }
517 {
518   Junk~characters~#1~\msg_line_context: :
519   \\ \\
520   #2
521 }
522 \msg_new:nnn { scontents } { empty-stored-content }
523 { Empty~value~for~key~'getstored'~\msg_line_context:. }
524 \msg_new:nnn { scontents } { empty-variable }
525 { Variable~'#1'~empty~\msg_line_context:. }
526 \msg_new:nnn { scontents } { rewriting-file }
527 { Overwriting ~ file ~ '#1' }
528 \msg_new:nnn { scontents } { writing-file }
529 { Writing ~ file ~ '#1' }
530 \msg_new:nnn { scontents } { rescanning-text }
531 { Rescanning~text~'#1'~after~\c_backslash_str end{scontents}~\msg_line_context:.}
532 \msg_new:nnn { scontents } { multiple-begin }
533 { Multiple~\c_backslash_str begin{scontents}~\msg_line_context:.}
534 \msg_new:nnn { scontents } { undefined-storage }
535 { Storage~named~'#1'~is~not~defined. }
536 \msg_new:nnn { scontents } { index-out-of-range }
537 {
538   \int_compare:nNnTF {#1} = { 0 }
539   { Index~of~sequence~cannot~be~zero. }
540   {
541     Index~'#1'~out~of~range~for~'#2'.~
542     \int_compare:nNnTF {#1} > { 0 }
543     { Max = } { Min = - } #3.
544   }
545 }
546 \msg_new:nnnn { scontents } { env-key-unknown }
547 { The~key~'#1'~is~unknown~by~environment~'scontents'~and~is~being~ignored.}
548 {
549   The~environment~'scontents'~does~not~have~a~key~called~'#1'.\\
550   Check~that~you~have~spelled~the~key~name~correctly.
551 }
552 \msg_new:nnnn { scontents } { env-key-value-unknown }
553 { The~key~'#1=#2'~is~unknown~by~environment~'scontents'~and~is~being~ignored. }
554 {
555   The~environment~'scontents'~does~not~have~a~key~called~'#1'.\\
556   Check~that~you~have~spelled~the~key~name~correctly.
557 }
558 \msg_new:nnnn { scontents } { cmd-key-unknown }
559 { The~key~'#1'~is~unknown~by~'\c_backslash_str Scontents'~and~is~being~ignored.}
560 {
561   The~command~'\c_backslash_str Scontents'~does~not~have~a~key~called~'#1'.\\
562   Check~that~you~have~spelled~the~key~name~correctly.
563 }
564 \msg_new:nnnn { scontents } { cmd-key-value-unknown }
565 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str Scontents'~and~is~being~ignored. }
566 {
567   The~command~'\c_backslash_str Scontents'~does~not~have~a~key~called~'#1'.\\
568   Check~that~you~have~spelled~the~key~name~correctly.
569 }
570 \msg_new:nnnn { scontents } { type-key-unknown }
571 { The~key~'#1'~is~unknown~and~is~being~ignored. }
572 {
573   This~command~does~not~have~a~key~called~'#1'.\\
574   This~command~only~accepts~the~key~'width-tab'.
575 }
576 \msg_new:nnnn { scontents } { type-key-value-unknown }
577 { The~key~'#1'~to~which~you~passed~'#2'~is~unknown~and~is~being~ignored. }
578 {
579   This~command~does~not~have~a~key~called~'#1'.\\
580   This~command~only~accepts~the~key~'width-tab'.
581 }

```

10.17 Finish package

Finish package implementation.

```
582 \file_input_stop:
```

11 Index of Implementation

Symbols

\\ 62, 519, 549, 555, 561, 567, 573, 579

B

bool commands:

\bool_if:NTF . 238, 298, 303, 315, 317, 365, 367, 370, 389, 397, 400
 \bool_lazy_or:nnTF 168, 283
 \bool_new:N 54, 56
 \bool_set_false:N 55, 88
 \bool_set_true:N 57, 84, 89

C

\char 23

char commands:

\char_generate:nn 23, 472
 \char_set_active_eq:NN 291, 467, 478
 \char_set_active_eq:nN 447
 \char_set_catcode_active:N 189, 218, 219, 220, 427, 466, 477, 497
 \char_set_catcode_active:n 379, 446
 \char_set_catcode_letter:n 247
 \char_set_catcode_other:N 62, 63, 64, 246
 \l_char_special_seq 246

\cleanseqsc 1, 5, 5, 24, 514

\countsc 1, 5, 5, 24, 512

cs commands:

\cs_generate_variant:Nn . 75, 78, 79, 155, 216, 320, 362
 \cs_new:Npn 77, 156, 166, 260, 310, 312, 360, 462, 464
 \cs_new:Npx 76
 \cs_new_protected:Npn . 74, 115, 117, 123, 125, 131, 133, 147, 174, 182, 196, 205, 211, 221, 228, 235, 296, 313, 333, 352, 363, 384, 428, 444, 452, 475, 480, 498
 \cs_new_protected:Npx 321
 \cs_replacement_spec:N 509
 \cs_set:Npx 468
 \cs_set_eq:NN 432, 433, 434, 441, 458
 \cs_set_protected:Npx 248, 280

D

dim commands:

\c_zero_dim 471

E

else commands:

\else: 338

\end 66, 282, 455

exp commands:

\exp_after:wN 337, 339
 \exp_args:Nf 160
 \exp_args:NNNV 231
 \exp_args:NV 116, 124, 132, 306, 418, 502
 \exp_not:N 250, 253, 260, 263, 282, 305, 324, 326, 327, 333, 437, 452
 \exp_not:n 185, 275, 311, 386, 395, 438

F

fi commands:

\fi: 340

file commands:

\file_if_exist:nTF 240
 \file_input_stop: 15, 582

G

\getstored 1, 4, 4, 22, 404

group commands:

\group_begin: . 217, 225, 237, 376, 406, 413, 426, 490, 496
 \group_end: 232, 302, 361, 391, 402, 409, 424, 443, 494, 511
 \group_insert_after:N 177, 178, 179, 180

I

if commands:

\if_meaning:w 336
 \IfBooleanTF 380
 \IfNoValueF 377
 \IfValueT 415, 492

int commands:

\int_abs:n 170
 \int_compare:nNnTF 328, 356, 538, 542
 \int_compare_p:nNn 169, 170
 \int_decr:N 353
 \int_incr:N 345, 346
 \int_new:N 51, 52, 53
 \int_set:Nn 138, 414, 491
 \int_step_function:nnN 247
 \int_to_roman:n 16, 135
 \int_zero:N 323
 \c_zero_int 356

iow commands:

\iow_char:N 396, 504
 \iow_close:N 299
 \iow_new:N 73
 \iow_now:Nn 316
 \iow_open:Nn 243

K

keys commands:

\keys_define:nn 17, 81, 101, 110
 \l_keys_key_tl 16, 116, 124, 132
 \keys_set:nn 24, 207, 378, 415, 487, 492

L

left commands:

\c_left_brace_str 68, 326, 334

M

\meaningsc 1, 5, 5, 14, 16, 16, 488

\MessageBreak 12

mode commands:

\mode_leave_vertical: 470

msg commands:

\msg_error:nn 150
 \msg_error:nnn 120, 128, 143, 431
 \msg_error:nnnn 121, 129, 139, 144, 214
 \msg_expandable_error:nnn 164
 \msg_expandable_error:nnnn 171
 \msg_line_context: 518, 523, 525, 531, 533
 \msg_new:nnn 516, 522, 524, 526, 528, 530, 532, 534, 536
 \msg_new:nnnn 546, 552, 558, 564, 570, 576
 \msg_warning:nn 329
 \msg_warning:nnn 241, 242, 287

N

\NewDocumentCommand ... 374, 393, 404, 411, 486, 488, 514

\NewDocumentEnvironment 187, 456

\NewExpandableDocumentCommand 512

P

\PackageError 9

peek commands:

\peek_charcode_ignore_spaces:NTF 198

prg commands:

\prg_generate_conditional_variant:Nnn ... 80

\prg_new_protected_conditional:Npnn 354

\prg_replicate:nn 463

\prg_return_false: 358

\prg_return_true: 357

\ProcessKeysOptions 45

\ProvidesExplPackage 4

Q

quark commands:

\quark_new:N 71, 72

quark internal commands:

\q__scontents_mark 71, 211, 223, 326, 336

\q__scontents_stop 71, 253, 263, 327, 360

R

\RequirePackage 2, 3

right commands:

\c_right_brace_str 68, 326, 334

S

scan commands:

\scan_stop: 433

\Scontents 1, 4, 4, 15, 16, 21, 374

scontents 3, 187

scontents internal commands:

\l__scontents__print_cmd_bool 17

\l__scontents__print_env_bool 17

__scontents_analyse_nesting:n ... 20, 267, 313

__scontents_analyse_nesting:w 313

__scontents_append_contents:nn ... 17, 147, 185

__scontents_atend_environment: 194, 363

__scontents_do_noligs:N 23, 434, 464

\c__scontents_end_env_tl 58, 251, 252, 261, 262, 275

\g__scontents_end_verbatimsc_tl .. 58, 439, 453

\l__scontents_env_nesting_int 14, 20, 51, 345, 353, 356

__scontents_environment_inline:w .. 18, 18, 196, 223

__scontents_environment_junk:nw 196

__scontents_environment_keys:w 196

\l__scontents_file_iow 73, 243, 299, 316

\l__scontents_file_tl 14, 46, 245, 306, 318

__scontents_file_tl_write_start:n 18, 226, 235, 362

__scontents_file_write_stop:N ... 18, 230, 235

\l__scontents_fname_out_tl ... 14, 46, 85, 90, 226

\l__scontents_forced_eol_bool 34, 367, 397

__scontents_getfrom_seq:nn 17, 156, 408, 418, 502

__scontents_getfrom_seq:nnn 156

\c__scontents_hidden_space_str 14, 69, 368, 387, 398, 420, 505

__scontents_if_nested: 20

__scontents_if_nested:TF 271, 313

__scontents_lastfrom_seq:n 17, 174, 371, 390, 401

\l__scontents_macro_tmp_tl . 14, 46, 230, 233, 368, 369

__scontents_make_control_chars_active: . 224, 255, 475

__scontents_meaningsc:n 493, 498

\l__scontents_name_seq_cmd_tl . 22, 388, 390, 399, 401

\l__scontents_name_seq_env_tl 19, 369, 371

__scontents_nesting_decr: 20, 273, 313

__scontents_norm_arg:n 21, 374

__scontents_par: 76, 483

__scontents_parse_command_keys:n . 16, 108, 123

__scontents_parse_command_keys:nn 123

__scontents_parse_environment_keys:n . 16, 99, 115

__scontents_parse_environment_keys:nn .. 115

__scontents_parse_type_meaning_key:n 113, 131

__scontents_parse_type_meaning_key:nn .. 131

__scontents_parse_typemeaning_key:n 16

\l__scontents_print_cmd_bool 31, 389, 400

\l__scontents_print_env_bool 28, 370

__scontents_remove_leading_nl:n 19, 235

__scontents_remove_leading_nl:w 235

__scontents_rescan_tokens:n 22, 74, 177, 288, 407, 435

__scontents_ret:w 19, 248, 256, 280, 291, 484

\l__scontents_seq_item_int . 14, 51, 138, 414, 418, 491, 502

__scontents_set_active_eq:NN 475

__scontents_start_environment:w .. 18, 190, 217

__scontents_stop_environment: 193, 217

__scontents_store_to_seq: 21

__scontents_store_to_seq:NN 17, 182, 369, 388, 399

\l__scontents_storing_bool 14, 20, 54, 88, 303, 317, 365

__scontents_tab: 76, 482

\l__scontents_tab_width_int 37, 463

__scontents_tabs_to_spaces: .. 23, 447, 462, 508

\g__scontents_temp_tl 14, 46, 176, 178, 180

\l__scontents_temp_tl 14, 46, 386, 387, 388, 395, 396, 398, 399, 416, 420, 421, 422, 423, 500, 504, 505, 506, 508, 509

\l__scontents_tmpa_int 51, 323, 328, 346

__scontents_use_none_delimit_by_q_stop:w 313

__scontents_verb_arg:w 21, 374

\l__scontents_verb_font_tl 25, 507

__scontents_verb_print:N 22, 22, 411

__scontents_verb_print_EOL: 432, 441

__scontents_verb_processor_iterate:w .. 235

__scontents_verb_processor_output:n . 20, 268, 274, 279, 313

\l__scontents_writing_bool . 14, 20, 54, 84, 89, 238, 298, 315

__scontents_xverb: 444, 458

__scontents_xverb:w 411

\Scontents* 16

seq commands:

\seq_clear_new:N 515

\seq_count:N 161, 513

\seq_gput_right:Nn 153

\seq_if_exist:NTF 151, 158

\seq_item:Nn 172, 176

\seq_map_function:NN 246

\seq_new:N 152

`\setupsc` 2, 24, [486](#)
 str commands:
 `\c_backslash_str` 68, 202, 209, 325, 334, 531, 533, 559,
 [561](#), [565](#), [567](#)
 `\c_circumflex_str` 70
 `\c_percent_str` 70, [285](#)
 `\str_const:Nn` 69
 `\str_if_eq:nnTF` 343
 `\str_if_eq_p:nn` 285

T

TeX and $\TeX_{2\epsilon}$ commands:

`\@ifpackagelater` 6
 `\do@noligs` 23

tex commands:

`\tex_kern:D` 471
 `\tex_scantokens:D` 15, 74

tl commands:

`\c_space_tl` 76
 `\tl_clear:N` 245
 `\tl_const:Nn` 67
 `\tl_gclear:N` 179
 `\tl_gset:Nn` 176
 `\tl_gset_rescan:Nnn` 59
 `\tl_if_blank:nTF` .. 119, 127, 137, 142, 149, 213, 265,
 [278](#), [430](#)

`\tl_if_blank_p:n` 284
 `\tl_if_empty:n` 80
 `\tl_if_empty:NTF` 422
 `\tl_if_empty:nTF` [78](#), 135
 `\tl_log:N` 184, [421](#), 506
 `\tl_new:N` 46, 47, 48, 49, 50, 58
 `\tl_put_right:Nn` 318, 368, 387, 398
 `\tl_remove_once:Nn` [78](#), 78, 420, 505
 `\tl_replace_all:Nnn` [78](#), 79, 396, 504, 508
 `\tl_rescan:nn` 15
 `\tl_set:Nn` 85, 90, 233, 305, 386, 395, 416, 500
 `\tl_use:N` 507

token commands:

`\token_to_str:N` 467, 468
 `\ttfamily` 26
 `\typestored` 1, 4, 4, 14, 16, 16, 22, [411](#)

U

use commands:

`\use:n` 258, 300, 331, 450
 `\use_i:nn` 337
 `\use_ii:nn` 339

V

`\verbatim` 459
`verbatimsc` 5, [411](#)