

SCONTENTS

Stores \LaTeX contents

v1.3 — 2019/09/24*

©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 memory (sequences) using the `l3seq` module of `expl3`. The stored content, including *verbatim* material, 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	2	5.3	The command <code>\cleansc</code>	5
3.2	The TAB character	2	6	Examples	6
3.3	Configuration of the options	2	6.1	From answers package	6
3.4	Options Overview	2	6.2	From filecontentsdef package	6
4	User interface	3	6.3	From TeX-SX	7
4.1	The environment <code>scontents</code>	3	6.4	Customization of <code>verbatimsc</code>	8
4.2	The command <code>\Scontents</code>	4	7	Change history	10
4.3	The command <code>\getstored</code>	4	8	Index of Documentation	11
4.4	The command <code>\tpestored</code>	4	9	Implementation	12
4.5	The environment <code>verbatimsc</code>	4	10	Index of Implementation	24

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 (lpl), version 1.3 or later (<http://www.latex-project.org/lpl.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 v1.3, last revised 2019/09/24.

†E-mail: pablgonz@educarchile.cl

3 The scontents package

3.1 Description

The `scontents` package allows to store contents in memory (sequence) or external files, provides a user interface style [`\key = val`] along with the ability to store contents in sequences for later use in different parts of the document. In some ways it works very much like the `filecontentsdef` package, with the difference that it preserves the TABs characters in the output files (see 3.2).

The package is loaded in the usual way:

```
\usepackage{scontents}
```

or

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

3.2 The TAB character

In a standard \TeX document, horizontal TABs (\tt\tab) typed from the keyboard are treated as explicit spaces in most contexts. Some users use this character to indent the source code of the document and depending on the text editor used, some will use real TABs (sometimes called “hard tabs”), others with “soft tabs” (\tt_ or \tt_) 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 reading¹ and writing², are different and may have adverse consequences.

The behavior of this character for the *stored* content can be configured with the `widht-tab` key, but, these will be preserved by writing an output file, unfortunately, dependent on the \TeX distribution you have and how the formats were initialized.

With a \TeX Live distribution, the formats for `latex`, `pdflatex` and `lualatex` are prepared feeding the option `-translate-file=cp227.tcx`, that makes the TAB character “printable”. To write a literal TAB character using `xelatex` you must add the `-8bit` option on command line, otherwise you will get \TeX -TABs (\tt\^I) in the *⟨output file⟩*.

As a general recommendation “Do not use TABs unless strictly necessary”, for example within a verbatim environment that supports this character such as `Verbatim` or `lstlisting` 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 the command `\setupsc`.

```
\setupsc \setupsc {⟨key=val⟩}
```

The command `\setupsc` configures the options 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 = {⟨number⟩} (default: \ttfamily)
```

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

3.4 Options Overview

Summary table of available options.

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

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 User interface

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

4.1 The environment `scontents`

```
scontents \begin{scontents}[\langle key=val \rangle]
          \langle env contents \rangle
          \end{scontents}
```

The `scontents` environment allows you to save the content to memory or external files in a similar way to the package `filecontentsdef`. This allows you to record content, including *verbatim*, for later reuse.

Some considerations to keep in mind, `\begin{scontents}` and `\end{scontents}` must be on different lines, the `[\langle key=val \rangle]` options must be passed on *one line* right after starting the environment and TABs characters are preserved when writing an *external file*.

The environment can be nested complying with the above considerations and bearing in mind that within it should not be written literally, as a comment or within *verbatim* environments `\begin{scontents}` and `\end{scontents}`. For example:

```
\begin{scontents}[store-env=outer]
This text is in the outermost environment.
\begin{scontents}[store-env=inner]
This text is found in the most inner environment.
\end{scontents}
This text is in the outermost environment.
\end{scontents}
```

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.

`store-env = { \langle seq name \rangle }` (default: contents)

The name of the *sequence* in which the content recorded by the environment was stored.

`print-env = { \langle true|false \rangle }` (default: false)

It will show the current content of the environment.

`force-eol = { \langle true|false \rangle }` (default: false)

This key is only necessary if the last line inside the environment is `\end{Verbatim}` or the `\end{...}` of some environment created with the package `fancyvrb`. This adds an extra line to the `\typestored` output.

`write-env = { \langle file.ext \rangle }` (default: not used)

In addition to storing the content of the environment will write this in an external *file.ext*, if *file.ext* exists it will be overwritten. The *file.ext* will be created in the working directory, relative or absolute paths are not supported, the way the external *file.ext* is recorded is similar to that produced by `filecontents*` environment. If the content within the environment has literal TABs they will be recorded in *file.ext*.

To write a literal TAB character using `xelatex` you must add the `-8bit` option on command line, otherwise you will get `TEX-TABS (^I)` in the *output file*.

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

It will write the contents in an external *file.ext*, but, it will not store the contents of this one. The *file.ext* will be created in the working directory, relative or absolute paths are not supported. The way the external *file.ext* is recorded is similar to that produced by `filecontents*` environment. If the content within the environment has literal TABs they will be recorded in *file.ext*.

To write a literal TAB character using `xelatex` you must add the `-8bit` option on command line, otherwise you will get `TEX-TABS (^I)` in the *output file*.

4.2 The command `\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-like the argument of `\verb`. Blank lines are preserved, escaped `\{` 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 (including starred version) can be configured globally using option in package or the `\setupsc` command and locally using [*<key=val>*].

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

The name of the sequence in which the content recorded by `\Scontents` was stored.

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

It will show the current content of `\Scontents`.

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

This key is only necessary if the last line inside the command is `\end{Verbatim}` or the `\end{...}` of some environment created with the package `fancyvrb`. This adds an extra line to the `\typestored` output.

4.3 The command `\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 last element saved in the *<seq name>*.

4.4 The command `\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 last element saved in the *<seq name>*.

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

Establishes the equivalence in *<spaces>* for the character TAB typed from the keyboard. The value must be a *<positive integer>*.

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 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 `\meaning`

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

The command `\meaning` 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 last element saved in the *⟨seq name⟩*.

width-tab = {*⟨number⟩*} (default: 1)

Establishes the equivalence in *⟨spaces⟩* for the character TAB typed from the keyboard. The value must be a *⟨positive integer⟩*.

5.2 The command `\countsc`

`\countsc` {*⟨seq name⟩*}

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

5.3 The command `\cleansc`

`\cleanseqsc` {*⟨seq name⟩*}


The command `\cleansc` remove all contents stored in *⟨seq name⟩*.

6 Examples

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

6.1 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
8 \begin{ex}
9 First exercise
10 \Scontents{
11   First solution.
12 }
13 \end{ex}
14
15 \begin{ex}
16 Second exercise
17 \Scontents{
18   Second solution.
19 }
20 \end{ex}
21
22 \section{Solutions}
23
24 \foreach \i in {1,...,\countsc{solutions}} {
25   \noindent\textbf{\i} \getstored[\i]{solutions}\par
26 }
27 \end{document}

```

6.2 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
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
12 \Scontents*{Refute the existence of black holes in less than  $140$  characters.}
13
14 \begin{scontents}[write-env=\jobname-3.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
21 \foreach \i in {1,...,3} {
22   \begin{itemize}
23     \item \getstored[\i]{defexercise}
24   \end{itemize}}
25
26 \section{\getstored[2]{defexercise}} % \getstored are robust :)
27 \end{document}

```

6.3 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}
21
22 This is b: \getstored[2]{a}
23
24 \textbf{Print all of them}
25
26 \foreach \i in {1,...,\countsc{a}} {\getstored[\i]{a}\par}
27 \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}

```

6.4 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
36 \Scontents{ We have coded this in \LaTeX: $E=mc^2$..}
37
38 \section{Test \texttt{\textbackslash getstored}}
39
40 \getstored[1]{contents}\par
41 \getstored[2]{contents}
42
43 \section{Test \texttt{\textbackslash meanings}}

```



```

44
45 \meaningsc[1]{contents}\par
46 \meaningsc[2]{contents}
47
48 \section{Test \texttt{\textbackslash typstored}}
49
50 \typstored[1]{contents}
51 \typstored[2]{contents}
52 \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
26 \section{Test \texttt{\textbackslash begin\{scontents\}} whit \texttt{listings}}
27 Test \verb+{scontents}+ \par
28
29 \begin{scontents}
30 Using \verb+scontents+ env no \verb+[key=val]+, save in seq \verb+contents+ with index 1.\par
31
32 Prove \lstinline[basicstyle=\ttfamily]| \lstinline | and environment \verb+Verbatim*+
33 \begin{verbatim}
34   verbatim environment
35 \end{verbatim}
36 \end{scontents}
37
38 \section{Test \texttt{\textbackslash Scontents*} whit \texttt{listings}}
39
40 \Scontents*+ We have coded this in \lstinline[basicstyle=\ttfamily]|\LaTeX: $E=mc^2$|
41 and more.+
42
43 \section{Test \texttt{\textbackslash getstored}}
44
45 \getstored[2]{contents}\par
46 \getstored[1]{contents}
47
48 \section{Test \texttt{\textbackslash typstored}}
49
50 \typstored[1]{contents}
51 \typstored[2]{contents}
52 \end{document}

```

Example 8

Customization of `verbatimsc` using the `minted` package .

```

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

```

7 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`.

- v1.3 (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,

8 Index of Documentation

C	
Commands provide by <code>scontents</code>	
<code>\Scontents*</code>	2-4
<code>\Scontents</code>	2-4
<code>\cleansc</code>	5
<code>\countsc</code>	5
<code>\getstored</code>	3, 4
<code>\meaningsc</code>	2, 5
<code>\setupsc</code>	2-4
<code>\typestored</code>	2-4
E	
Environment provide by <code>scontents</code> :	
<code>scontents</code>	2, 3
<code>verbatimsc</code>	4, 8, 9
Environments	
<code>filecontents*</code>	3
<code>filecontentsdefmacro</code>	1
K	
Keys	
<code>force-eol</code>	3, 4
<code>print-cmd</code>	4
<code>print-env</code>	3
<code>store-cmd</code>	4
<code>store-env</code>	3
<code>verb-font</code>	2
<code>width-tab</code>	4, 5
<code>write-env</code>	3
<code>write-out</code>	3
L	
<code>\lstinline</code>	4
M	
<code>\meaning</code>	5
P	
Packages	
<code>answers</code>	6
<code>expl3</code>	1, 10
<code>fancyvrb</code>	3, 4, 8
<code>filecontentsdef</code>	1-3, 6, 10
<code>fvextra</code>	4
<code>l3keys2e</code>	1
<code>l3seq</code>	1
<code>listings</code>	4, 5, 9
<code>minted</code>	5, 9
<code>scontents</code>	1, 2, 10
<code>tcolorbox</code>	8
<code>xparse</code>	1
S	
<code>\setupsc</code>	2
V	
<code>\Verb</code>	4
<code>\verb</code>	4

9 Implementation

9.1 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}{2019/09/24}{1.3}
5 {Stores LaTeX contents in memory or files}
```

A check to make sure that `xparse` is not too old

```
6 \ifpackagelater { xparse } { 2019/05/03 }
7 { }
8 {
9   \PackageError { scontents } { Support-package~xparse~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 }
```

9.2 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   unknown .code:n         = { \__scontents_parse_unknown_key:n {#1} }
43 }
```

We process the keys as options passed on to the package.

```
44 \ProcessKeysOptions { scontents }
```

9.3 Internal variables

Now we declare the internal variables we will use.

<pre>\l__scontents_macro_tmp_tl \l__scontents_fname_out_tl \l__scontents_temp_tl \l__scontents_file_tl</pre>	<pre>\l__scontents_macro_tmp_tl is a temporary token list to hold the contents of the macro/en- vironment, \l__scontents_fname_out_tl is used as the name of the output file, when there's</pre>
--	--

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.

```
45 \tl_new:N \l__scontents_macro_tmp_tl
46 \tl_new:N \l__scontents_fname_out_tl
47 \tl_new:N \l__scontents_temp_tl
48 \tl_new:N \l__scontents_file_tl
49 \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 contents environment.

```
50 \int_new:N \l__scontents_seq_item_int
51 \int_new:N \l__scontents_env_nesting_int
52 \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` `\l__scontents_storing_bool` determines whether it is in write-only mode when the write-out option is used.

```
53 \bool_new:N \l__scontents_writing_bool
54 \bool_set_false:N \l__scontents_writing_bool
55 \bool_new:N \l__scontents_storing_bool
56 \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.

```
\c__scontents_end_env_tl
57 \tl_new:N \g__scontents_end_verbatimsc_tl
58 \tl_gset_rescan:Nnn
59   \g__scontents_end_verbatimsc_tl
60   {
61     \char_set_catcode_other:N \
62     \char_set_catcode_other:N \{
63     \char_set_catcode_other:N \}
64   }
65   { \end{verbatimsc} }
66 \tl_const:Nx \c__scontents_end_env_tl
67   { \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`.)

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

```
\q__scontents_mark
68 \quark_new:N \q__scontents_stop
69 \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.

```
70 \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.

```
71 \cs_new_protected:Npn \__scontents_rescan_tokens:n #1 { \tex_scantokens:D {#1} }
72 \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:`

```
73 \cs_new:Npx \__scontents_tab: { \c_space_tl }
74 \cs_new:Npn \__scontents_par: { ^^J ^^J }
```

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

`\tl_remove_once:Nv` Some nonstandard variants.

```
\tl_replace_all:Nxx
\tl_replace_all:Nxn
\tl_replace_all:Nnx
\tl_if_empty:FTF
```

```
75 \cs_generate_variant:Nn \tl_remove_once:Nn { NV }
76 \cs_generate_variant:Nn \tl_replace_all:Nnn { Nx, Nxx, Nnx }
77 \prg_generate_conditional_variant:Nnn \tl_if_empty:n { f } { TF }
```

(End definition for `\tl_remove_once:Nv`, `\tl_replace_all:Nxx`, and `\tl_if_empty:FTF`. These functions are documented on page ??.)

9.4 Add keys for environment

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

```
78 \keys_define:nn { scontents }
79 {
80   write-env .code:n      = {
81     \bool_set_true:N \__scontents_writing_bool
82     \tl_set:Nn \__scontents_fname_out_tl {#1}
83   },
84   write-out .code:n      = {
85     \bool_set_false:N \__scontents_storing_bool
86     \bool_set_true:N \__scontents_writing_bool
87     \tl_set:Nn \__scontents_fname_out_tl {#1}
88   },
89   write-env .value_required:n = true,
90   write-out .value_required:n = true
91 }
```

9.5 Define keys for command

A sub/keys for command `\Scontents` and `\Scontents*`

```
92 \keys_define:nn { scontents / Scontents }
93 {
94   print-cmd .meta:nn = { scontents } { print-cmd = #1 },
95   store-cmd .meta:nn = { scontents } { store-cmd = #1 },
96   force-eol .meta:nn = { scontents } { force-eol = #1 }
97 }
```

9.6 Handling undefined keys

`__scontents_parse_unknown_key:n` 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_unknown_key:n` to process.

The key is stored in the token list variable `\l_keys_key_tl`, and the value (if any) is passed as argument to this function.

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.

```
98 \cs_new_protected:Npn \__scontents_parse_unknown_key:n #1
99 { \exp_args:Nv \__scontents_parse_unknown_key:nn \l_keys_key_tl {#1} }
100 \cs_new_protected:Npn \__scontents_parse_unknown_key:nn #1 #2
101 {
102   \tl_if_empty:FTF { \int_to_roman:n { -0 #1 } }
103   {
104     \tl_if_blank:nTF {#2}
```

```

105         { \int_set:Nn \__scontents_seq_item_int {#1} }
106         { \msg_error:nnnn { scontents } { key-value-unknown } {#1} {#2} }
107     }
108     {
109         \tl_if_blank:nTF {#2}
110         { \msg_error:nnn { scontents } { key-unknown } {#1} }
111         { \msg_error:nnnn { scontents } { key-value-unknown } {#1} {#2} }
112     }
113 }

```

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

9.7 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` creates a seq variable if one didn't exist and appends the contents in the argument to the right of the sequence. `__scontents_getfrom_seq:nn` retrieves the saved item from the sequence.

```

114 \cs_new_protected:Npn \__scontents_append_contents:nn #1#2
115 {
116     \tl_if_blank:nT {#1}
117     { \msg_error:nn { scontents } { empty-store-cmd } }
118     \seq_if_exist:cF { g__scontents_seq_name_#1_seq }
119     { \seq_new:c { g__scontents_seq_name_#1_seq } }
120     \seq_gput_right:cn { g__scontents_seq_name_#1_seq } {#2}
121 }
122 \cs_generate_variant:Nn \__scontents_append_contents:nn { Vx }
123 \cs_new:Npn \__scontents_getfrom_seq:nn #1#2
124 {
125     \seq_if_exist:cTF { g__scontents_seq_name_#2_seq }
126     {
127         \exp_args:Nf \__scontents_getfrom_seq:nnn
128         { \seq_count:c { g__scontents_seq_name_#2_seq } }
129         {#1} {#2}
130     }
131     { \msg_expandable_error:nnn { scontents } { undefined-storage } {#2} }
132 }
133 \cs_new:Npn \__scontents_getfrom_seq:nnn #1 #2 #3
134 {
135     \bool_lazy_or:nnTF
136     { \int_compare_p:nNn {#2} = { 0 } }
137     { \int_compare_p:nNn { \int_abs:n {#2} } > {#1} }
138     { \msg_expandable_error:nnnn { scontents } { index-out-of-range } {#2} {#3} {#1} }
139     { \seq_item:cn { g__scontents_seq_name_#3_seq } {#2} }
140 }

```

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

`__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.

```

141 \cs_new_protected:Npn \__scontents_store_to_seq:NN #1 #2
142 {
143     \tl_log:N #1
144     \__scontents_append_contents:Vx #2 { \exp_not:V #1 }
145 }

```

(End definition for `__scontents_store_to_seq:NN`.)

9.8 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.

```

146 \NewDocumentEnvironment { scontents } { }
147 {
148   \char_set_catcode_active:N ^^M
149   \__scontents_start_environment:w
150 }
151 {
152   \__scontents_stop_environment:
153   \__scontents_atend_environment:
154 }

```

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

9.8.1 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.

```

155 \cs_new_protected:Npn \__scontents_environment_inline:w
156 {
157   \peek_charcode_ignore_spaces:NTF [ % ]
158   { \__scontents_environment_keys:w }
159   {
160     \__scontents_environment_junk:xw
161     { after~\c_backslash_str begin{scontents} }
162   }
163 }
164 \cs_new_protected:Npn \__scontents_environment_keys:w [ #1 ]
165 {
166   \keys_set_known:nn { scontents } {#1}
167   \__scontents_environment_junk:xw
168   { after~optional~argument~to~\c_backslash_str begin{scontents} }
169 }
170 \cs_new_protected:Npn \__scontents_environment_junk:nw #1 #2 \q__scontents_mark
171 {
172   \tl_if_blank:nF {#2}
173   { \msg_error:nnnn { scontents } { junk-after-begin } {#1} {#2} }
174 }
175 \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`.)

9.8.2 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 `xsimverb` by Clemens Niederberger.

```

176 \group_begin:
177   \char_set_catcode_active:N ^^I
178   \char_set_catcode_active:N ^^L
179   \char_set_catcode_active:N ^^M
180   \cs_new_protected:Npn \__scontents_start_environment:w #1 ^^M
181   {
182     \__scontents_environment_inline:w #1 \q__scontents_mark
183     \__scontents_make_control_chars_active:
184     \group_begin:

```



```

185     \_scontents_file_tl_write_start:V \_l__scontents_fname_out_tl
186   }
187   \cs_new_protected:Npn \_scontents_stop_environment:
188   {
189     \_scontents_file_write_stop:N \_l__scontents_macro_tmp_tl
190     \exp_args:NNNV
191     \group_end:
192     \tl_set:Nn \_l__scontents_macro_tmp_tl \_l__scontents_macro_tmp_tl
193   }

```

(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 `^^M` is removed from the token list using `_scontents_remove_leading_nl:n` (which expects an active `^^M` token at the head of the token list; a low level TeX error is raised otherwise).

```

194   \cs_new_protected:Npn \_scontents_file_tl_write_start:n #1
195   {
196     \group_begin:
197     \bool_if:NT \_l__scontents_writing_bool
198     {
199       \file_if_exist:nTF {#1}
200       { \msg_warning:nxx { scontents } { rewriting-file } {#1} }
201       { \msg_warning:nxx { scontents } { writing-file } {#1} }
202       \iow_open:Nn \_l__scontents_file_iow {#1}
203     }
204     \tl_clear:N \_l__scontents_file_tl
205     \seq_map_function:NN \_l_char_special_seq \char_set_catcode_other:N
206     \int_step_function:nnN { 128 } { 255 } \char_set_catcode_letter:n
207     \cs_set_protected:Npx \_scontents_ret:w ##1 ^^M
208     {
209       \exp_not:N \_scontents_verb_processor_iterate:w
210       ##1 \_c__scontents_end_env_tl
211       \_c__scontents_end_env_tl
212       \exp_not:N \_q__scontents_stop
213     }
214     \_scontents_make_control_chars_active:
215     \_scontents_ret:w
216   }
217   \use:x
218   {
219     \cs_new:Npn \exp_not:N \_scontents_verb_processor_iterate:w
220     ##1 \_c__scontents_end_env_tl
221     ##2 \_c__scontents_end_env_tl
222     ##3 \exp_not:N \_q__scontents_stop
223   } {
224     \tl_if_blank:nTF {#3}
225     {
226       \_scontents_analyse_nesting:n {#1}
227       \_scontents_verb_processor_output:n {#1}
228     }
229     {
230       \_scontents_if_nested:TF
231       {
232         \_scontents_nesting_decr:
233         \_scontents_verb_processor_output:x
234         { \exp_not:n {#1} \_c__scontents_end_env_tl \exp_not:n {#2} }
235       }
236       {
237         \tl_if_blank:nF {#1}
238         { \_scontents_verb_processor_output:n {#1} }
239         \cs_set_protected:Npx \_scontents_ret:w
240         {
241           \exp_not:N \end{scontents}

```

```

242         \bool_lazy_or:nnF
243         { \tl_if_blank_p:n {#2} }
244         { \str_if_eq_p:ee {#2} { \c_percent_str } }
245         {
246             \msg_warning:nnn { scontents } { rescanning-text } {#2}
247             \__scontents_rescan_tokens:n {#2}
248         }
249     }
250     \char_set_active_eq:NN ^^M \__scontents_ret:w
251 }
252 }
253 ^^M
254 }
255 \cs_new_protected:Npn \__scontents_file_write_stop:N #1
256 {
257     \bool_if:NT \l__scontents_writing_bool
258     { \iow_close:N \l__scontents_file_iow }
259     \use:x
260     {
261         \group_end:
262         \bool_if:NT \l__scontents_storing_bool
263         {
264             \tl_set:Nn \exp_not:N #1
265             {
266                 \exp_args:NV \__scontents_remove_leading_nl:n \l__scontents_file_tl
267                 \bool_if:NT \l__scontents_forced_eol_bool { \exp_not:N ^^J }
268             }
269         }
270     }
271 }
272 \cs_new:Npn \__scontents_remove_leading_nl:n #1
273 { \exp_not:o { \__scontents_remove_leading_nl:w #1 } }
274 \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 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 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...

```

275 \cs_new_protected:Npn \__scontents_verb_processor_output:n #1
276 {
277     \bool_if:NT \l__scontents_writing_bool
278     { \iow_now:Nn \l__scontents_file_iow {#1} }
279     \bool_if:NT \l__scontents_storing_bool
280     { \tl_put_right:Nn \l__scontents_file_tl { ^^J #1 } }
281 }
282 \cs_generate_variant:Nn \__scontents_verb_processor_output:n { x }
283 \cs_new_protected:Npx \__scontents_analyse_nesting:n #1
284 {
285     \int_zero:N \l__scontents_tmpa_int
286     \exp_not:N \__scontents_analyse_nesting:w #1
287     \c_backslash_str begin
288     \c_left_brace_str \exp_not:N \q__scontents_mark \c_right_brace_str
289     \exp_not:N \q__scontents_stop
290     \int_compare:nNnT { \l__scontents_tmpa_int } > { 1 }
291     { \msg_warning:nn { scontents } { multiple-begin } }
292 }
293 \use:x
294 {
295     \cs_new_protected:Npn \exp_not:N \__scontents_analyse_nesting:w ##1
296     \c_backslash_str begin \c_left_brace_str ##2 \c_right_brace_str
297     {
298         \if_meaning:w \q__scontents_mark #2

```

```

299         \exp_after:wN \use_i:nn
300     \else:
301         \exp_after:wN \use_ii:nn
302     \fi:
303     { \__scontents_use_none_delimit_by_q_stop:w }
304     {
305         \str_if_eq:eeT {#2} {scontents}
306         {
307             \int_incr:N \l__scontents_env_nesting_int
308             \int_incr:N \l__scontents_tmpa_int
309             \__scontents_analyse_nesting:w
310         }
311         \__scontents_analyse_nesting:w
312     }
313 }
314 \cs_new_protected:Npn \__scontents_nesting_decr:
315 { \int_decr:N \l__scontents_env_nesting_int }
316 \prg_new_protected_conditional:Npnn \__scontents_if_nested: { TF }
317 {
318     \int_compare:nNnTF { \l__scontents_env_nesting_int } > { \c_zero_int }
319     { \prg_return_true: }
320     { \prg_return_false: }
321 }
322 \cs_new:Npn \__scontents_use_none_delimit_by_q_stop:w #1 \q__scontents_stop { }
323 \group_end:
324 \cs_generate_variant:Nn \__scontents_file_tl_write_start:n { V }

```

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

9.8.3 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.

```

325 \cs_new_protected:Npn \__scontents_atend_environment:
326 {
327     \bool_if:NT \l__scontents_storing_bool
328     {
329         \__scontents_store_to_seq:NN \l__scontents_macro_tmp_tl \l__scontents_name_seq_env_tl
330     }
331     \bool_if:NT \l__scontents_print_env_bool
332     {
333         \tl_gset_eq:NN \g__scontents_temp_tl \l__scontents_macro_tmp_tl
334         \tl_gput_right:NV \g__scontents_temp_tl \c_percent_str
335         \group_insert_after:N \__scontents_rescan_tokens:V
336         \group_insert_after:N \g__scontents_temp_tl
337     }
338     \tl_clear:N \l__scontents_macro_tmp_tl
339 }

```

(End definition for `__scontents_atend_environment:`)

9.9 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_norm_arg:n` or `__scontents_verb_arg:w` 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.

```

340 \NewDocumentCommand { \Scontents } { !s !O{} }
341 {
342     \group_begin:
343     \IfNoValueF {#2}
344     { \keys_set_known:nn { scontents / Scontents } {#2} }
345     \char_set_catcode_active:n { 9 }

```

```

346     \IfBooleanTF {#1}
347     { \_scontents_verb_arg:w }
348     { \_scontents_norm_arg:n }
349   }
350 \cs_new_protected:Npn \_scontents_norm_arg:n #1
351 {
352   \exp_args:NV \_scontents_append_contents:nn \_scontents_name_seq_cmd_tl {#1}
353   \bool_if:NT \_scontents_print_cmd_bool { \_scontents_rescan_tokens:n {#1} }
354   \group_end:
355 }
356 \NewDocumentCommand { \_scontents_verb_arg:w } { +v }
357 {
358   \tl_set:Nx \_scontents_temp_tl
359   {
360     \exp_not:n {#1}
361     \bool_if:NT \_scontents_forced_eol_bool { \exp_not:N ^^J }
362   }
363   \tl_replace_all:Nxx \_scontents_temp_tl { \iow_char:N ^^M } { \iow_char:N ^^J }
364   \_scontents_store_to_seq:NN \_scontents_temp_tl \_scontents_name_seq_cmd_tl
365   \bool_if:NT \_scontents_print_cmd_bool
366   { \_scontents_rescan_tokens:V \_scontents_temp_tl }
367   \group_end:
368 }

```

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

9.10 The command `\getstored`

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

```

369 \NewDocumentCommand { \getstored } { 0{1} m }
370 {
371   \group_begin:
372   \_scontents_rescan_tokens:x
373   {
374     \_scontents_getfrom_seq:nn {#1} {#2}
375     \c_percent_str
376   }
377   \group_end:
378 }

```

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

9.11 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_xverb:w
\__scontents_verb_print:N
\verbatimsc
379 \NewDocumentCommand { \typestored } { o m }
380 {
381   \group_begin:
382   \int_set:Nn \_scontents_seq_item_int { 1 }
383   \IfValueT {#1} { \keys_set:nn { scontents } {#1} }
384   \tl_set:Nx \_scontents_temp_tl { \exp_args:NV \_scontents_getfrom_seq:nn \_scontents_seq_item_int }
385   \tl_log:N \_scontents_temp_tl
386   \tl_if_empty:NF \_scontents_temp_tl
387   { \_scontents_verb_print:N \_scontents_temp_tl }
388   \group_end:
389 }

```

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`.

```

390 \group_begin:
391   \char_set_catcode_active:N ^^M
392   \cs_new_protected:Npn \_scontents_verb_print:N #1

```

```

393 {
394   \tl_if_blank:VT #1
395   { \msg_error:nnn { scontents } { empty-variable } {#1} }
396   \cs_set_eq:NN \__scontents_verb_print_EOL: ^^M
397   \cs_set_eq:NN ^^M \scan_stop:
398   \cs_set_eq:cN { do@noligs } \__scontents_do_noligs:N
399   \__scontents_rescan_tokens:x
400   {
401     \exp_not:N \begin{verbatim} ^^M
402     \exp_not:V #1 ^^M
403     \g__scontents_end_verbatimsc_tl
404   }
405   \cs_set_eq:NN ^^M \__scontents_verb_print_EOL:
406 }
407 \group_end:

```

Finally, the `verbatim` environment is defined.

```

408 \cs_new_protected:Npn \__scontents_xverb:
409 {
410   \char_set_catcode_active:n { 9 }
411   \char_set_active_eq:nN { 9 } \__scontents_tabs_to_spaces:
412   \__scontents_xverb:w
413 }
414 \use:x
415 {
416   \cs_new_protected:Npn \exp_not:N \__scontents_xverb:w
417   ##1 \g__scontents_end_verbatimsc_tl
418 }
419 { #1 \end{verbatim} }
420 \NewDocumentEnvironment { verbatim } { }
421 {
422   \cs_set_eq:cN { @xverbatim } \__scontents_xverb:
423   \verbatim
424 }
425 { }

```

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

9.12 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.

```

426 \cs_new:Npn \__scontents_tabs_to_spaces:
427 { \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 $\text{\TeX}_{2\epsilon}$'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*. This version uses `\char_generate:nn` to ensure that doesn't happen.

```

428 \cs_new:Npn \__scontents_do_noligs:N #1
429 {
430   \char_set_catcode_active:N #1
431   \char_set_active_eq:Nc #1 { __scontents_active_char_ \token_to_str:N #1 : }
432   \cs_set:cpx { __scontents_active_char_ \token_to_str:N #1 : }
433   {
434     \mode_leave_vertical:
435     \tex_kern:D \c_zero_dim
436     \char_generate:nn { `#1 } { 12 }
437   }
438 }

```

(End definition for `__scontents_do_noligs:N.`)

`_scontents_set_active_eq:NN` `_scontents_set_active_eq:NN` Shortcut definitions for common catcode changes.
`_scontents_make_control_chars_active:`

```

439 \cs_new_protected:Npn \__scontents_set_active_eq:NN #1
440 {
441   \char_set_catcode_active:N #1
442   \char_set_active_eq:NN #1
443 }
444 \cs_new_protected:Npn \__scontents_make_control_chars_active:
445 {
446   \__scontents_set_active_eq:NN ^^I \__scontents_tab:
447   \__scontents_set_active_eq:NN ^^L \__scontents_par:
448   \__scontents_set_active_eq:NN ^^M \__scontents_ret:w
449 }

```

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

9.13 The command `\setupsc`

User command `\setupsc` to setup module.

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

```

450 \NewDocumentCommand { \setupsc } { m }
451 { \keys_set:nn { scontents } { #1 } }

```

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

9.14 The command `\meaningsc`

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

```

452 \NewDocumentCommand { \meaningsc } { o m }
453 {
454   \group_begin:
455     \int_set:Nn \l__scontents_seq_item_int { 1 }
456     \IfValueT {#1} { \keys_set:nn { scontents } { #1 } }
457     \__scontents_meaningsc:n {#2}
458   \group_end:
459 }
460 \group_begin:
461   \char_set_catcode_active:N ^^I
462   \cs_new_protected:Npn \__scontents_meaningsc:n #1
463   {
464     \tl_set:Nx \l__scontents_temp_tl { \exp_args:NV \__scontents_getfrom_seq:nn \l__scontents_seq_item_int
465     \tl_replace_all:Nxn \l__scontents_temp_tl { \iow_char:N ^^J } { ~ }
466     \tl_log:N \l__scontents_temp_tl
467     \tl_use:N \l__scontents_verb_font_tl
468     \tl_replace_all:Nnx \l__scontents_temp_tl { ^^I } { \__scontents_tabs_to_spaces: }
469     \cs_replacement_spec:N \l__scontents_temp_tl
470   }
471 \group_end:

```

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

9.15 The command `\countsc`

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

```

472 \NewExpandableDocumentCommand { \countsc } { m }
473 { \seq_count:c { g__scontents_seq_name_#1_seq } }

```

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

9.16 The command `\cleanseqsc`

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

```

474 \NewDocumentCommand { \cleanseqsc } { m }
475 { \seq_clear_new:c { g__scontents_seq_name_#1_seq } }

```

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

9.17 Messages

Messages used throughout the package.

```

476 \msg_new:nnn { scontents } { junk-after-begin }
477 {
478   Junk~characters~#1~\msg_line_context: :
479   \\ \\
480   #2
481 }
482 \msg_new:nnn { scontents } { empty-stored-content }
483 { Empty~value~for~key~\getstored'~\msg_line_context:. }
484 \msg_new:nnn { scontents } { empty-variable }
485 { Variable~`#1'~empty~\msg_line_context:. }
486 \msg_new:nnn { scontents } { rewriting-file }
487 { Overwriting ~ file ~ `#1' }
488 \msg_new:nnn { scontents } { writing-file }
489 { Writing ~ file ~ `#1' }
490 \msg_new:nnn { scontents } { rescanning-text }
491 { Rescanning~text~`#1'~after~\c_backslash_str end{scontents}~\msg_line_context:. }
492 \msg_new:nnn { scontents } { multiple-begin }
493 { Multiple~\c_backslash_str begin{scontents}~\msg_line_context:. }
494 \msg_new:nnn { scontents } { tab-to-space }
495 { Tab~has~been~converted~to~Blank~Space }
496 \msg_new:nnn { scontents } { feed-to-space }
497 { Form~Feed~has~been~converted~to~Blank~Space }
498 \msg_new:nnnn { scontents } { key-unknown }
499 { The~key~`#1'~is~unknown~and~is~being~ignored. }
500 {
501   The~module~'scontents'~does~not~have~a~key~called~`#1'.\\
502   Check~that~you~have~spelled~the~key~name~correctly.
503 }
504 \msg_new:nnnn { scontents } { key-value-unknown }
505 { The~key~`#1'~to~which~you~passed~`#2'~is~unknown~and~is~being~ignored. }
506 {
507   The~module~'scontents'~does~not~have~a~key~called~`#1'.\\
508   Check~that~you~have~spelled~the~key~name~correctly.
509 }
510 \msg_new:nnn { scontents } { undefined-storage }
511 { Storage~named~`#1'~is~not~defined. }
512 \msg_new:nnn { scontents } { index-out-of-range }
513 {
514   \int_compare:nNnTF {#1} = { 0 }
515   { Index~of~sequence~cannot~be~zero. }
516   {
517     Index~`#1'~out~of~range~for~`#2'.~
518     \int_compare:nNnTF {#1} > { 0 }
519     { Max = } { Min = - } #3.
520   }
521 }

```

9.18 Finish package

Finish package

```

522 \file_input_stop:

```

10 Index of Implementation

Symbols

\\ 61, 479, 501, 507

B

\begin 18

bool commands:

\bool_if:NTF . 197, 257, 262, 267, 277, 279, 327,
331, 353, 361, 365
\bool_lazy_or:nnTF 135, 242
\bool_new:N 53, 55
\bool_set_false:N 54, 85
\bool_set_true:N 56, 81, 86

C

\char 21

char commands:

\char_generate:nn 21, 436
\char_set_active_eq:NN 250, 431, 442
\char_set_active_eq:nN 411
\char_set_catcode_active:N 148, 177, 178, 179,
391, 430, 441, 461
\char_set_catcode_active:n 345, 410
\char_set_catcode_letter:n 206
\char_set_catcode_other:N ... 61, 62, 63, 205
\l_char_special_seq 205

\cleansc 1, 5

\cleanseqsc 5, 22, 22, 474

\countsc 1, 5, 5, 22, 22, 472

cs commands:

\cs_generate_variant:Nn . 72, 75, 76, 122, 175,
282, 324
\cs_new:Npn 74, 123, 133, 219, 272, 274, 322, 426,
428
\cs_new:Npx 73
\cs_new_protected:Npn 71, 98, 100, 114, 141, 155,
164, 170, 180, 187, 194, 255, 275, 295, 314, 325,
350, 392, 408, 416, 439, 444, 462
\cs_new_protected:Npx 283
\cs_replacement_spec:N 469
\cs_set:Npx 432
\cs_set_eq:NN 396, 397, 398, 405, 422
\cs_set_protected:Npx 207, 239

D

dim commands:

\c_zero_dim 435

E

else commands:

\else: 300

\end 18, 65, 241, 419

exp commands:

\exp_after:wN 299, 301
\exp_args:Nf 127
\exp_args:NNNV 190
\exp_args:NV 99, 266, 352, 384, 464
\exp_not:N 209, 212, 219, 222, 241, 264, 267, 286,
288, 289, 295, 361, 401, 416
\exp_not:n 144, 234, 273, 360, 402

F

fi commands:

\fi: 302

file commands:

\file_if_exist:nTF 199
\file_input_stop: 15, 522

G

\getstored 1, 4, 4, 20, 20, 369

group commands:

\group_begin: . 176, 184, 196, 342, 371, 381, 390,
454, 460
\group_end: 191, 261, 323, 354, 367, 377, 388, 407,
458, 471
\group_insert_after:N 335, 336

I

if commands:

\if_meaning:w 298
\IfBooleanTF 346
\IfNoValueF 343
\IfValueT 383, 456

int commands:

\int_abs:n 137
\int_compare:nNnTF 290, 318, 514, 518
\int_compare_p:nNn 136, 137
\int_decr:N 315
\int_incr:N 307, 308
\int_new:N 50, 51, 52
\int_set:Nn 105, 382, 455
\int_step_function:nnN 206
\int_to_roman:n 14, 102
\int_zero:N 285
\c_zero_int 318

iow commands:

\iow_char:N 363, 465
\iow_close:N 258
\iow_new:N 70
\iow_now:Nn 278
\iow_open:Nn 202

K

keys commands:

\keys_define:nn 17, 78, 92
\l_keys_key_tl 14, 99
\keys_set:nn 22, 383, 451, 456
\keys_set_known:nn 166, 344

L

left commands:

\c_left_brace_str 67, 288, 296

M

\meaningsc 1, 5, 5, 13, 14, 22, 452

\MessageBreak 12

mode commands:

\mode_leave_vertical: 434

msg commands:

\msg_error:nn 117
\msg_error:nnn 110, 395
\msg_error:nnnn 106, 111, 173
\msg_expandable_error:nnn 131
\msg_expandable_error:nnnnn 138
\msg_line_context: ... 478, 483, 485, 491, 493
\msg_new:nnn . 476, 482, 484, 486, 488, 490, 492,
494, 496, 510, 512
\msg_new:nnnn 498, 504

- \msg_warning:nn 291
 - \msg_warning:nnn 200, 201, 246
- N**
- \NewDocumentCommand 340, 356, 369, 379, 450, 452, 474
 - \NewDocumentEnvironment 146, 420
 - \NewExpandableDocumentCommand 472
- P**
- \PackageError 9
 - peek commands:
 - \peek_charcode_ignore_spaces:NTF 157
 - prg commands:
 - \prg_generate_conditional_variant:Nnn 77
 - \prg_new_protected_conditional:Npnn . 316
 - \prg_replicate:nn 427
 - \prg_return_false: 320
 - \prg_return_true: 319
 - \ProcessKeysOptions 44
 - \ProvidesExplPackage 4
- Q**
- quark commands:
 - \quark_new:N 68, 69
 - quark internal commands:
 - \q__scontents_mark 68, 170, 182, 288, 298
 - \q__scontents_stop 68, 212, 222, 289, 322
- R**
- \RequirePackage 2, 3
 - right commands:
 - \c_right_brace_str 67, 288, 296
- S**
- scan commands:
 - \scan_stop: 397
 - \Scontents 1, 4, 4, 19, 19, 340
 - scontents 3, 146
 - scontents internal commands:
 - __scontents_analyse_nesting:n 18, 226, 275
 - __scontents_analyse_nesting:w 275
 - __scontents_append_contents:nn 15, 114, 144, 352
 - __scontents_atend_environment: . 153, 325
 - __scontents_do_noligs:N 21, 398, 428
 - \c__scontents_end_env_tl . 57, 210, 211, 220, 221, 234
 - \g__scontents_end_verbatimsc_tl 57, 403, 417
 - \l__scontents_env_nesting_int 13, 18, 50, 307, 315, 318
 - __scontents_environment_inline:w .. 16, 16, 155, 182
 - __scontents_environment_junk:nw 155
 - __scontents_environment_keys:w 155
 - \l__scontents_file_iow 70, 202, 258, 278
 - \l__scontents_file_tl .. 13, 45, 204, 266, 280
 - __scontents_file_tl_write_start:n 16, 185, 194, 324
 - __scontents_file_write_stop:N 16, 189, 194
 - \l__scontents_fname_out_tl 12, 45, 82, 87, 185
 - \l__scontents_forced_eol_bool . 34, 267, 361
 - __scontents_getfrom_seq:nn 15, 114, 374, 384, 464
 - __scontents_getfrom_seq:nnn 114
 - __scontents_if_nested: 18
 - __scontents_if_nested:TF 230, 275
 - \l__scontents_macro_tmp_tl . 12, 45, 189, 192, 329, 333, 338
 - __scontents_make_control_chars_-
 - active: 183, 214, 439
 - __scontents_meaningsc:n 457, 462
 - \l__scontents_name_seq_cmd_tl . 22, 352, 364
 - \l__scontents_name_seq_env_tl 19, 329
 - __scontents_nesting_decr: 18, 232, 275
 - __scontents_norm_arg:n 19, 340
 - __scontents_par: 73, 447
 - __scontents_parse_unknown_key:n 14, 42, 98
 - __scontents_parse_unknown_key:nn 98
 - \l__scontents_print_cmd_bool .. 31, 353, 365
 - \l__scontents_print_env_bool 28, 331
 - __scontents_remove_leading_nl:n .. 17, 194
 - __scontents_remove_leading_nl:w 194
 - __scontents_rescan_tokens:n 20, 71, 247, 335, 353, 366, 372, 399
 - __scontents_ret:w 17, 207, 215, 239, 250, 448
 - \l__scontents_seq_item_int . 13, 50, 105, 382, 384, 455, 464
 - __scontents_set_active_eq:NN 439
 - __scontents_start_environment:w .. 16, 149, 176
 - __scontents_stop_environment: .. 152, 176
 - __scontents_store_to_seq: 19
 - __scontents_store_to_seq:NN .. 15, 141, 329, 364
 - \l__scontents_storing_bool 13, 18, 53, 85, 262, 279, 327
 - __scontents_tab: 73, 446
 - \l__scontents_tab_width_int 37, 427
 - __scontents_tabs_to_spaces: .. 21, 411, 426, 468
 - \g__scontents_temp_tl .. 13, 45, 333, 334, 336
 - \l__scontents_temp_tl 13, 45, 358, 363, 364, 366, 384, 385, 386, 387, 464, 465, 466, 468, 469
 - \l__scontents_tmpa_int 50, 285, 290, 308
 - __scontents_use_none_delimit_by_q_-
 - stop:w 275
 - __scontents_verb_arg:w 19, 340
 - \l__scontents_verb_font_tl 25, 467
 - __scontents_verb_print:N 20, 20, 379
 - __scontents_verb_print_EOL: 396, 405
 - __scontents_verb_processor_iterate:w 194
 - __scontents_verb_processor_output:n . 18, 227, 233, 238, 275
 - \l__scontents_writing_bool . 13, 18, 53, 81, 86, 197, 257, 277
 - __scontents_xverb: 408, 422
 - __scontents_xverb:w 379
- seq commands:
- \seq_clear_new:N 475
 - \seq_count:N 128, 473
 - \seq_gput_right:Nn 120
 - \seq_if_exist:NTF 118, 125
 - \seq_item:Nn 139
 - \seq_map_function:NN 205
 - \seq_new:N 119

<code>\setupsc</code>	2, 22, 450	<code>\tl_if_empty:N</code>	386
str commands:		<code>\tl_if_empty:n</code>	75 , 102
<code>\c_backslash_str</code>	67 , 161 , 168 , 287 , 296 , 491 , 493	<code>\tl_log:N</code>	143 , 385 , 466
<code>\c_percent_str</code>	244 , 334 , 375	<code>\tl_new:N</code>	45 , 46 , 47 , 48 , 49 , 57
<code>\str_if_eq:nnTF</code>	305	<code>\tl_put_right:Nn</code>	280
<code>\str_if_eq_p:nn</code>	244	<code>\tl_remove_once:Nn</code>	75 , 75
T		<code>\tl_replace_all:Nnn</code>	75 , 76 , 363 , 465 , 468
TeX and \TeX commands:		<code>\tl_rescan:nn</code>	13
<code>\@ifpackagelater</code>	6	<code>\tl_set:Nn</code>	82 , 87 , 192 , 264 , 358 , 384 , 464
<code>\do@noligs</code>	21	<code>\tl_use:N</code>	467
tex commands:		token commands:	
<code>\tex_kern:D</code>	435	<code>\token_to_str:N</code>	431 , 432
<code>\tex_scantokens:D</code>	13 , 71	<code>\ttfamily</code>	26
tl commands:		<code>\typestored</code>	1 , 4 , 4 , 4 , 13 , 14 , 20 , 20 , 379
<code>\c_space_tl</code>	73		
<code>\tl_clear:N</code>	204 , 338	U	
<code>\tl_const:Nn</code>	66	use commands:	
<code>\tl_gput_right:Nn</code>	334	<code>\use:n</code>	217 , 259 , 293 , 414
<code>\tl_gset_eq:NN</code>	333	<code>\use_i:nn</code>	299
<code>\tl_gset_rescan:Nnn</code>	58	<code>\use_ii:nn</code>	301
<code>\tl_if_blank:nTF</code> .	104 , 109 , 116 , 172 , 224 , 237 , 394	V	
<code>\tl_if_blank_p:n</code>	243	<code>\verbatim</code>	423
<code>\tl_if_empty:n</code>	77	<code>\verbatimsc</code>	4 , 379