

SCONTENTS

Stores \LaTeX contents

v1.2 — 2019/08/28*

©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*) can be used as many times as desired in the document, additionally can be written to external files if desired.

Contents

1	Motivation and Acknowledgments	1	5	Other commands provided	4
2	License and Requirements	1	5.1	The command <code>\meaningsc</code>	4
3	The <code>scontents</code> package	2	5.2	The command <code>\countsc</code>	4
3.1	Description	2	5.3	The command <code>\cleansc</code>	4
3.2	Loading package	2	6	Examples	5
3.3	Configuration of the options	2	6.1	From <code>answers</code> package	5
3.4	Options Overview	2	6.2	From <code>filecontentsdef</code> package	5
4	User interface	2	6.3	From <code>TeX-SX</code>	5
4.1	The environment <code>scontents</code>	2	6.4	Customization of <code>verbatimsc</code>	7
4.2	The command <code>\Scontents</code>	3	7	Change history	9
4.3	The command <code>\getstored</code>	3	8	Index of Documentation	10
4.4	The command <code>\tpestored</code>	3	9	Implementation	11
4.5	The environment <code>verbatimsc</code>	4	10	Index of Implementation	18

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, I decided to combine the best of both.

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 `xparse`, `filecontentsdef` and `l3keys2e`. This package can be used with `xelatex`, `lualatex`, `pdfplatex` and the classical workflow `latex-dvips-ps2pdf`.

*This file describes a documentation for v1.2, last revised 2019/08/28.

†E-mail: pablgonz@educarchile.cl

3 The scontents package

3.1 Description

The `scontents` package encapsulates the `filecontentsdef` package of JEAN FRANÇOIS BURNOL which allows you to save the content in a `\macro` and save it in external files, adding a user interface style `[\langle key = val \rangle]` along with the ability to save content in sequences for later use in different parts of the document.

3.2 Loading package

The package is loaded in the usual way:

```
\usepackage{scontents}
```

or

```
\usepackage[\langle key=val \rangle]{scontents}
```

3.3 Configuration of the options

Most of the options can be passed directly to the package or can be configured by means of the command `\setupsc`.

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

The command `\setupsc` configures the options in a global way, for `\Scontents`, `\Scontents*` and environment `scontents`. it can be used both in the preamble and in the body of the document as many times as desired.

3.4 Options Overview

Summary table of available options.

key	package	\setupsc	scontents	\Scontents	\Scontents*
store-env	✓	✓	✓	✗	✗
store-cmd	✓	✓	✗	✓	✓
print-env	✓	✓	✓	✗	✗
print-cmd	✓	✓	✗	✓	✓
print-all	✓	✓	✓	✗	✗
write-env	✗	✗	✓	✗	✗
write-out	✗	✗	✓	✗	✗

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 *\langle stored content \rangle* 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 encapsulates the `filecontentsdef*` and `filecontentsdefmacro` environments provided by the `filecontentsdef` package. This allows you to record content including verbatim for later reuse.

Some considerations to keep in mind:

1. The environment cannot be nested.
2. Both `\begin` and `\end` must be on different lines.
3. The `[\langle key=val \rangle]` options must be passed on one line right after starting the environment.
4. The content of the environment is treated in the same way as `filecontents*` environment.
5. If you don't want the extra space added by T_EX, you should use `\relax` or `%` at the end of environment.

For more technical information about the environment it is better to read the documentation of the `filecontentsdef` package.

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.

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

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

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

It will show the current content of the environment.

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

In addition to storing the content of the environment will write this in an external file.

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

It will write the contents of the environment in an external file, but, it will not store the contents of this one. It is analogous to the `filecontents*` environment.

4.2 The command `\Scontents`

The command to store content directly in memory, the star version allows to save `⟨verbatim⟩` contents.

<code>\Scontents</code>	<code>\Scontents [⟨key=val⟩] {⟨argument⟩}</code>
	<code>\Scontents* [⟨key=val⟩] {⟨argument⟩}</code>
	<code>\Scontents* [⟨key=val⟩] ⟨del⟩⟨argument⟩⟨del⟩</code>

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.

It can be used anywhere in the document and cannot be used as an `⟨argument⟩` for another command.

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`. Some considerations to keep in mind:

1. Blank lines are preserved.
2. The command cannot be used as an argument for another command.
3. If you don't want the extra space added by \TeX , you should use `\relax` or `%` at the end.

Options for command

The command options (including star 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`.

4.3 The command `\getstored`

<code>\getstored</code>	<code>\getstored [⟨index⟩] {⟨seq name⟩}</code>
-------------------------	--

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 `\tpestored`

<code>\tpestored</code>	<code>\tpestored [⟨index⟩] {⟨seq name⟩}</code>
	<code>\tpestored* [⟨index⟩] {⟨seq name⟩}</code>

The command `\tpestored` shows the content stored in `⟨seq name⟩` in `verbatim` mode. Internally places the content into the `verbatimsc` environment.

The command `\typestored*` must be used for content stored by `\Scontents*` command. Internally places the content into the `verbatimsc` environment.

If the optional argument is not passed it defaults to the last element saved in the $\langle seq name \rangle$.

4.5 The environment `verbatimsc`

`verbatimsc`

Internal environment used by `\typestored` and `\typestored*` to display $\langle verbatim style \rangle$ contents.

One consideration to keep in mind is that this is a *representation* of the content in a $\langle verbatim \rangle$ environment and not a real $\langle verbatim \rangle$ environment, the line ends are not respected. The `verbatimsc` environment can be 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}
  }
}{}
\end{verbatimsc}
```

5 Other commands provided

5.1 The command `\meaningsc`

`\meaningsc`

`\meaningsc` [$\langle index \rangle$] [$\langle seq name \rangle$]

The command `\meaningsc` executes `\meaning` on the content stored in $\langle seq name \rangle$. If the optional argument is not passed it defaults to the last element saved in the $\langle seq name \rangle$.

5.2 The command `\countsc`

`\countsc`

`\countsc` [$\langle seq name \rangle$]

The command `\countsc` count a number of contents stored in $\langle seq name \rangle$.

5.3 The command `\cleansc`

`\cleansc`

`\cleansc` [$\langle seq name \rangle$]

The command `\cleansc` remove all contents stored in $\langle seq name \rangle$.

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` [\[1\]](#).

```

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 \begin{ex}
14   Second exercise
15   \Scontents{
16     Second solution.
17   }
18 \end{ex}
19 \section{Solutions}
20 \foreach \i in {1,...,\countsc{solutions}} {
21   \noindent\textbf{\i} \getstored[\i]{solutions}\par
22 }
23 \end{document}

```

6.2 From filecontentsdef package

Example 2

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

```

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

```

6.3 From TeX-SX

Example 3

Adapted from [LaTeX equivalent of ConTeXt buffers](#) [\[3\]](#).

```

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 \begin{tikzpicture}
15 \getstored[2]{tikz}
16 \end{tikzpicture}
17 \begin{tikzpicture}
18 \getstored[3]{tikz}
19 \end{tikzpicture}
20 \section{source}
21 \foreach \i in {1,...,\countsc{tikz}}{
22 \tpestored*[\i]{tikz}}
23 \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. \verb+This is from the \verb command+. This is normal text.
9 \verb*|This is from the \verb* command|. This is normal text.
10 \begin{verbatim}
11 This is from the verbatim environment:

```


```

12 &%{}
13 \end{verbatim}
14 \end{scontents}
15 \section{Problem restated}
16 \getstored[1]{problem}
17 \section{Problem restated once more}
18 \getstored[1]{problem}
19 \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
22 \section{Test \texttt{\textbackslash begin\{scontents\}} whit \texttt{fancyvrb}}
23 Test \verb+\begin{scontents}+ \par
24
25 \begin{scontents}
26 Using \verb+scontents+ env no \verb+[key=val]+, save in seq \verb+contents+
27 with index 1.
28
29 Prove new \Verb*{ fancyvrb whit braces } and environment \verb+Verbatim*+
30 \begin{verbatim}
31   verbatim environment
32 \end{verbatim}
33 \end{scontents}
34
35 \section{Test \texttt{\textbackslash Scontents} whit \texttt{fancyvrb}}
36
37 \Scontents{ We have coded this in \LaTeX:  $E=mc^2$ .}
38
39 \section{Test \texttt{\textbackslash getstored}}
40
41 \getstored[1]{contents}\par
42 \getstored[2]{contents}
43
44 \section{Test \texttt{\textbackslash meaningsc}}
45
46 \meaningsc[1]{contents}\par
47
48 \meaningsc[2]{contents}
49
50 \section{Test \texttt{\textbackslash typestored}}
51
52 \typestored[1]{contents}
53
54 \typestored*[2]{contents}
55 \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+\begin{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
47 \getstored[1]{contents}
48
49 \section{Test \texttt{\textbackslash typestored}}
50
51 \typestored[1]{contents}
52 \typestored*[2]{contents}
53 \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+\begin{scontents}+ \par
14
15 \begin{scontents}
16 Using \verb+scontents+ env no \verb+[key=val]+, save in seq \verb+contents+ with index 1.\par
17
18 Prove new \Verb*{ new fvxtra whit braces } and environment \verb+Verbatim*+
19 \begin{verbatim}
20     verbatim environment
21 \end{verbatim}
22 \end{scontents}
23
24 \section{Test \texttt{\textbackslash Scontents} whit \texttt{minted}}
25
26 \Scontents{ We have coded \par this in \LaTeX:  $E=mc^2$ .}
27
28 \section{Test \texttt{\textbackslash getstored}}
29 \getstored[2]{contents}\par
30 \getstored[1]{contents}
31
32 \section{Test \texttt{\textbackslash typestored}}
33 \typestored[1]{contents}
34 \end{document}

```

7 Change history

- 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 `scontents`

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

E

Environment provide by `scontents`:

<code>scontents</code>	2
<code>verbatimsc</code>	3, 4, 7, 8

Environments

<code>filecontents*</code>	2, 3
<code>filecontentsdef*</code>	2
<code>filecontentsdefmacro</code>	1, 2

K

Keys

<code>print-cmd</code>	3
<code>print-env</code>	3
<code>store-cmd</code>	3
<code>store-env</code>	3
<code>write-env</code>	3
<code>write-out</code>	3

L

<code>\lstinline</code>	3
-------------------------	---

M

<code>\meaning</code>	4
-----------------------	---

P

Packages

<code>answers</code>	5
<code>expl3</code>	1
<code>fancyvrb</code>	4, 7
<code>filecontentsdef</code>	1, 2, 5
<code>fvextra</code>	3
<code>l3keys2e</code>	1
<code>l3seq</code>	1
<code>listings</code>	3, 4, 8
<code>minted</code>	4, 8
<code>scontents</code>	1, 2
<code>tcolorbox</code>	7
<code>xparse</code>	1

R

<code>\relax</code>	2, 3
---------------------	------

S

<code>\setupsc</code>	2
-----------------------	---

V

<code>\Verb</code>	3
<code>\verb</code>	3

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{filecontentsdef}[2019/04/20]
3 \RequirePackage{l3keys2e}
4 \RequirePackage{xparse}[2019/05/28]
5 \ProvidesExplPackage{scontents}[2019/08/28]{1.2}
6 {Stores LaTeX contents in memory or files}
```

A check to make sure that xparse is not too old

```
7 \ifpackagelater { xparse } { 2019/05/03 }
8 { }
9 {
10   \PackageError { scontents } { Support~package~xparse~too~old }
11   {
12     You~need~to~update~your~installation~of~the~bundles~
13     'l3kernel'~and~'l3packages'.\MessageBreak
14     Loading~scontents~will~abort!
15   }
16   \file_input_stop:
17 }
```

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.

```
18 \keys_define:nn { scontents }
19 {
20   store-env .tl_set:N      = \l__scontents_name_seq_env_tl,
21   store-env .initial:n     = contents,
22   print-env .bool_set:N   = \l__scontents_print_env_bool,
23   print-env .initial:n    = false,
24   store-cmd .tl_set:N     = \l__scontents_name_seq_cmd_tl,
25   store-cmd .initial:n    = contents,
26   print-cmd .bool_set:N   = \l__scontents_print_cmd_bool,
27   print-cmd .initial:n    = false,
28   print-all .meta:n      = { print-env = true , print-cmd = true },
29   store-env .value_required:n = true,
30   store-cmd .value_required:n = true,
31   print-env .value_required:n = true,
32   print-cmd .value_required:n = true,
33   print-all .value_required:n = true
34 }
```

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

```
35 \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</pre>	<pre>\l__scontents_macro_tmp_tl is a temporary token list to hold the contents of the macro/envi- ronment, \l__scontents_fname_out_tl is used as the name of the output file, when there's one, and \l__scontents_temp_tl is a generic temporary token list.</pre>
--	--

```
36 \tl_new:N \l__scontents_macro_tmp_tl
37 \tl_new:N \l__scontents_fname_out_tl
38 \tl_new:N \l__scontents_temp_tl
```

(End definition for \l__scontents_macro_tmp_tl, \l__scontents_fname_out_tl, and \l__scontents_temp_tl.)

<pre>\l__scontents_typeverb_env_bool \l__scontents_writing_bool \l__scontents_storing_bool</pre>	<pre>The boolean \l__scontents_typeverb_env_bool keeps track whether the starred variant of the \typestored macro was used, \l__scontents_writing_bool if we should write to a file, and SCONTENTS © 2019 by Pablo González</pre>
--	---

`\l__scontents_storing_bool` determines whether it is in write-only mode when the `write-out` option is used.

```

39 \bool_new:N \l__scontents_typeverb_env_bool
40 \bool_set_true:N \l__scontents_typeverb_env_bool
41 \bool_new:N \l__scontents_writing_bool
42 \bool_set_false:N \l__scontents_writing_bool
43 \bool_new:N \l__scontents_storing_bool
44 \bool_set_true:N \l__scontents_storing_bool

```

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

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

```

45 \quark_new:N \q__scontents_stop
46 \quark_new:N \q__scontents_mark

```

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

`__scontents_rescan_tokens:w` `\tl_rescan:nn` 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:

```

47 \cs_new_eq:NN \__scontents_rescan_tokens:w \tex_scantokens:D

```

(End definition for `__scontents_rescan_tokens:w`.)

`\g__scontents_end_verbatimsc_tl` A token list to match when ending verbatim environments.

```

48 \tl_new:N \g__scontents_end_verbatimsc_tl
49 \tl_gset_rescan:Nnn
50   \g__scontents_end_verbatimsc_tl
51   {
52     \char_set_catcode_other:N \
53     \char_set_catcode_other:N \{
54     \char_set_catcode_other:N \}
55   }
56   { \end{verbatimsc} }

```

(End definition for `\g__scontents_end_verbatimsc_tl`.)

9.4 Add keys for environment

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

```

57 \keys_define:nn { scontents }
58   {
59     write-env .code:n      = {
60                           \bool_set_true:N \l__scontents_writing_bool
61                           \tl_set:Nn \l__scontents_fname_out_tl {#1}
62                           },
63     write-out .code:n      = {
64                           \bool_set_false:N \l__scontents_storing_bool
65                           \bool_set_true:N \l__scontents_writing_bool
66                           \tl_set:Nn \l__scontents_fname_out_tl {#1}
67                           },
68     write-env .value_required:n = true,
69     write-out .value_required:n = true
70   }

```

9.5 Define keys for command

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

```

71 \keys_define:nn { scontents / Scontents }
72   {
73     print-cmd .meta:nn = { scontents } { print-cmd = #1 },
74     store-cmd .meta:nn = { scontents } { store-cmd = #1 }
75   }

```

9.6 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` 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.

```

76 \cs_new_protected:Npn \__scontents_append_contents:nn #1#2
77 {
78   \tl_if_blank:nT {#1}
79   { \msg_error:nn { scontents } { empty-store-cmd } }
80   \seq_if_exist:cF { g__scontents_seq_name_#1_seq }
81   { \seq_new:c { g__scontents_seq_name_#1_seq } }
82   \seq_gput_right:cn { g__scontents_seq_name_#1_seq } {#2}
83 }
84 \cs_generate_variant:Nn \__scontents_append_contents:nn { Vx }
85 \cs_new:Npn \__scontents_getfrom_seq:nn #1#2
86 { \seq_item:cn { g__scontents_seq_name_#2_seq } {#1} }

```

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

9.7 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.

```

87 \NewDocumentEnvironment { scontents } { }
88 {
89   \char_set_catcode_active:N \^^M
90   \__scontents_start_environment:w
91 }
92 {
93   \__scontents_stop_environment:
94   \__scontents_atend_environment:
95 }

```

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

9.7.1 The environment itself

The environment itself

`__scontents_start_environment:w` Here we make `^^M` an active character so that the end of line can be “seen” to be used as a delimiter. 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.

```

96 \group_begin:
97   \char_set_catcode_active:N \^^M
98   \cs_new_protected:Npn \__scontents_start_environment:w #1 ^^M
99   {
100     \__scontents_environment_inline:w #1 \q__scontents_mark
101     \group_begin:
102     \bool_if:NTF \__scontents_writing_bool
103     {
104       \use:c { filecontentsdef* } { \__scontents_fname_out_tl }
105       { \__scontents_macro_tmp_tl } ^^M
106     }
107     { \filecontentsdefmacro { \__scontents_macro_tmp_tl } ^^M }
108   }
109   \cs_new_protected:Npn \__scontents_stop_environment:
110   {
111     \bool_if:NTF \__scontents_writing_bool
112     { \endfilecontentsdef }
113     { \endfilecontentsdefmacro }
114   }
115   \group_end:
116 \group_end:

```

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

9.7.2 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.

```

117 \cs_new_protected:Npn \__scontents_environment_inline:w
118 {
119   \peek_charcode_ignore_spaces:NTF [ % ]
120   { \__scontents_environment_keys:w }
121   {
122     \__scontents_environment_junk:xw
123     { after~\c_backslash_str begin{scontents} }
124   }
125 }
126 \cs_new_protected:Npn \__scontents_environment_keys:w [ #1 ]
127 {
128   \keys_set_known:nn { scontents } {#1}
129   \__scontents_environment_junk:xw
130   { after~optional~argument~to~\c_backslash_str begin{scontents} }
131 }
132 \cs_new_protected:Npn \__scontents_environment_junk:nw #1 #2 \q__scontents_mark
133 {
134   \tl_if_blank:nF {#2}
135   { \msg_error:nnnn { scontents } { junk-after-begin } {#1} {#2} }
136 }
137 \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`.)

9.7.3 Recording of the content in the sequence

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

The `__scontents_stored_to_seq:` function replaces a carriage return (ASCII 13) by a new line character (ASCII 10) for optionally logging the contents of the current `scontents` environment.

```

138 \cs_new_protected:Npn \__scontents_atend_environment:
139 {
140   \bool_if:NT \l__scontents_storing_bool
141   {
142     \__scontents_stored_to_seq:
143     \bool_if:NT \l__scontents_print_env_bool
144     { \__scontents_getfrom_seq:nn { -1 } { \l__scontents_name_seq_env_tl } }
145   }
146   \tl_clear:N \l__scontents_macro_tmp_tl
147 }
148 \cs_new_protected:Npn \__scontents_stored_to_seq:
149 {
150   \regex_replace_all:nnN { ^M } { ^J } \l__scontents_macro_tmp_tl
151   \tl_log:N \l__scontents_macro_tmp_tl
152   \__scontents_append_contents:Vx \l__scontents_name_seq_env_tl
153   { \exp_not:N \__scontents_rescan_tokens:w { \tl_use:N \l__scontents_macro_tmp_tl } }
154 }

```

(End definition for `__scontents_atend_environment:` and `__scontents_stored_to_seq:.`)

9.8 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_norm:n` or `__scontents_verb:w` depending whether a star argument is present.

`__scontents_norm:n`

`__scontents_verb:w`

`__scontents_norm:n` grabs a normal argument, adds it to the seq variable, and optionally prints it.

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

```

155 \NewDocumentCommand { \Scontents }{ !s !O{} }
156 {
157   \group_begin:
158     \IfNoValueF {#2}
159     { \keys_set_known:nn { scontents / Scontents } {#2} }
160     \IfBooleanTF{#1}
161     { \__scontents_verb:w }
162     { \__scontents_norm:n }
163   }
164   \cs_new_protected:Npn \__scontents_norm:n #1
165   {
166     \exp_args:NV \__scontents_append_contents:nn \l__scontents_name_seq_cmd_tl {#1}
167     \bool_if:NT \l__scontents_print_cmd_bool
168     { \__scontents_getfrom_seq:nn { -1 } { \l__scontents_name_seq_cmd_tl } }
169     \group_end:
170   }
171 \NewDocumentCommand { \__scontents_verb:w } { +v }
172 {
173   \tl_set:Nn \l__scontents_temp_tl {#1}
174   \regex_replace_all:nnN { ^M } { ^J } \l__scontents_temp_tl
175   \tl_log:N \l__scontents_temp_tl
176   \exp_args:NVx \__scontents_append_contents:nn \l__scontents_name_seq_cmd_tl
177   { \exp_not:N \__scontents_rescan_tokens:w { \tl_use:N \l__scontents_temp_tl } }
178   \bool_if:NT \l__scontents_print_cmd_bool
179   { \__scontents_getfrom_seq:nn { -1 } { \l__scontents_name_seq_cmd_tl } }
180   \group_end:
181 }

```

(End definition for `\Scontents`, `__scontents_norm:n`, and `__scontents_verb:w`. This function is documented on page 3.)

9.9 The command `\getstored`

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

```

182 \NewDocumentCommand { \getstored } { O{1} m }
183 { \__scontents_getfrom_seq:nn {#1} {#2} }

```

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

9.10 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 `filecontentsdef's \filecontentsdef@get` macro to read the contents of the token list and pass them to `__scontents_fcdef_print:N`.

`__scontents_xverb:w`
`verbatimsc`

```

184 \NewDocumentCommand { \typestored } { s O{1} m }
185 {
186   \group_begin:
187     \tl_set:Nx \l__scontents_temp_tl { \__scontents_getfrom_seq:nn {#2} {#3} }
188     \tl_log:N \l__scontents_temp_tl
189     \IfBooleanTF {#1}
190     { \bool_set_false:N \l__scontents_typeverb_env_bool }
191     { \bool_set_true:N \l__scontents_typeverb_env_bool }
192     \use:c { filecontentsdef@get } \__scontents_fcdef_print:N \l__scontents_temp_tl
193   \group_end:
194 }

```

The `__scontents_fcdef_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 `\scantokens`.

```

195 \group_begin:
196 \char_set_catcode_active:N ^^M

```

```

197 \cs_new_protected:Npn \__scontents_fcdef_print:N #1
198 {
199   \tl_if_blank:VT #1
200   { \msg_error:nnn { scontents } { empty-variable } {#1} }
201   \cs_set_eq:NN \__scontents_fcdef_saved_EOL: ^^M
202   \cs_set_eq:NN ^^M \scan_stop:
203   \use:x
204   {
205     \exp_not:N \__scontents_rescan_tokens:w
206     {
207       \exp_not:N \begin{verbatim} ^^M
208       \__scontents_strip_scantokens:N #1
209       \bool_if:NF \l__scontents_typeverb_env_bool { ^^M }
210       \g__scontents_end_verbatimsc_tl
211     }
212   }
213   \cs_set_eq:NN ^^M \__scontents_fcdef_saved_EOL:
214 }
215 \group_end:

```

Finally, the verbatimsc environment is defined.

```

216 \use:x
217 { \cs_gset_protected:Npn \exp_not:N \__scontents_xverb:w ##1 \g__scontents_end_verbatimsc_tl }
218 { #1 \end{verbatim} }
219 \NewDocumentEnvironment { verbatimsc } { }
220 {
221   \cs_set_eq:cN { @xverbatim } \__scontents_xverb:w
222   \verbatim
223 }
224 { }

```

(End definition for `\typestored` and others. These functions are documented on page 3.)

`__scontents_strip_scantokens:N` The `__scontents_strip_scantokens:n` (`__scontents_strip_scantokens:N`) macro takes a token list (variable) as argument and examines it. If the argument token list is *exactly* `\scantokens{⟨stuff⟩}`, then the function returns `⟨stuff⟩`, otherwise it returns the input token list without change. The token list is wrapped in `\exp_not:n` to avoid further expansion.

```

225 \cs_new:Npn \__scontents_strip_scantokens:N #1
226 { \exp_args:NV \__scontents_strip_scantokens:n #1 }
227 \cs_new:Npn \__scontents_strip_scantokens:n #1
228 {
229   \tl_if_head_is_N_type:nTF {#1}
230   {
231     \__scontents_if_scantokens:NwTF #1 \q__scontents_mark
232     {
233       \exp_args:No \tl_if_single_token:nTF { \use_none:nn #1 ? }
234       { \exp_not:o { \use_ii:nn #1 } }
235       { \exp_not:n {#1} }
236     }
237     { \exp_not:n {#1} }
238   }
239   { \exp_not:n {#1} }
240 }
241 \prg_new_conditional:Npnn \__scontents_if_scantokens:Nw #1#2 \q__scontents_mark { TF }
242 {
243   \token_if_eq_meaning:NNTF \__scontents_rescan_tokens:w #1
244   { \prg_return_true: }
245   { \prg_return_false: }
246 }

```

(End definition for `__scontents_strip_scantokens:N`, `__scontents_strip_scantokens:n`, and `__scontents_if_scantokens:Nw`.)

9.11 The command `\setupsc`

User command `\setupsc` to setup module.

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

```

247 \NewDocumentCommand { \setupsc } { m }

```



```
248 { \keys_set:nn { scontents } {#1} }
```

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

9.12 The command `\meaningsc`

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

```
249 \NewDocumentCommand { \meaningsc } { 0{1} m }
250 {
251   \group_begin:
252     \tl_set:Nx \l__scontents_temp_tl { \__scontents_getfrom_seq:nn {#1} {#2} }
253     \tl_log:N \l__scontents_temp_tl
254     \tl_set:Nx \l__scontents_temp_tl { \__scontents_strip_scantokens:N \l__scontents_temp_tl }
255     \regex_replace_all:nnN { \v{1,} } { } \l__scontents_temp_tl
256     \ttfamily
257     \cs_replacement_spec:N \l__scontents_temp_tl
258   \group_end:
259 }
```

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

9.13 The command `\countsc`

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

```
260 \NewExpandableDocumentCommand { \countsc } { m }
261 { \seq_count:c { g__scontents_seq_name_#1_seq } }
```

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

9.14 The command `\cleanseqsc`

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

```
262 \NewDocumentCommand { \cleanseqsc } { m }
263 { \seq_clear_new:c { g__scontents_seq_name_#1_seq } }
```

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

9.15 Messages

Messages used throughout the package.

```
264 \msg_new:nnn { scontents } { junk-after-begin }
265 {
266   Junk~characters~#1~\msg_line_context: :
267   \\ \\
268   #2
269 }
270 \msg_new:nnn { scontents } { empty-stored-content }
271 { Empty~value~for~key~`getstored'~\msg_line_context:. }
272 \msg_new:nnn { scontents } { empty-variable }
273 { Variable~`#1'~empty~\msg_line_context:. }
```

9.16 Finish package

Finish package

```
274 \file_input_stop:
```

10 Index of Implementation

Symbols

\\ 52, 267

B

bool commands:

\bool_if:NTF .. 102, 111, 140, 143, 167, 178, 209
 \bool_new:N 39, 41, 43
 \bool_set_false:N 42, 64, 190
 \bool_set_true:N 40, 44, 60, 65, 191

C

char commands:

\char_set_catcode_active:N 89, 97, 196
 \char_set_catcode_other:N 52, 53, 54
 \cleansc 1, 4
 \cleanseqsc 4, 17, 17, 262
 \countsc 1, 4, 4, 17, 17, 260

cs commands:

\cs_generate_variant:Nn 84, 137
 \cs_gset_protected:Npn 217
 \cs_new:Npn 85, 225, 227
 \cs_new_eq:NN 47
 \cs_new_protected:Npn 76, 98, 109, 117, 126, 132,
 138, 148, 164, 197
 \cs_replacement_spec:N 257
 \cs_set_eq:NN 201, 202, 213, 221

E

\end 56, 218
 \endfilecontentsdef 112
 \endfilecontentsdefmacro 113

exp commands:

\exp_args:No 233
 \exp_args:Nv 166, 226
 \exp_args:Nvx 176
 \exp_not:N 153, 177, 205, 207, 217
 \exp_not:n 16, 234, 235, 237, 239

F

file commands:

\file_input_stop: 16, 274
 \filecontentsdefmacro 107

G

\getstored 1, 3, 3, 15, 15, 182

group commands:

\group_begin: 96, 101, 157, 186, 195, 251
 \group_end: ... 114, 116, 169, 180, 193, 215, 258

I

\IfBooleanTF 160, 189
 \IfNoValueF 158

K

keys commands:

\keys_define:nn 18, 57, 71
 \keys_set:nn 16, 248
 \keys_set_known:nn 128, 159

M

\meaningsc 1, 4, 4, 17, 249
 \MessageBreak 13

msg commands:

\msg_error:nn 79
 \msg_error:nnn 200
 \msg_error:nnnn 135
 \msg_line_context: 266, 271, 273
 \msg_new:nnn 264, 270, 272

N

\NewDocumentCommand 155, 171, 182, 184, 247, 249, 262
 \NewDocumentEnvironment 87, 219
 \NewExpandableDocumentCommand 260

P

\PackageError 10

peek commands:

\peek_charcode_ignore_spaces:NTF 119

prg commands:

\prg_new_conditional:Npnn 241
 \prg_return_false: 245
 \prg_return_true: 244
 \ProcessKeysOptions 35
 \ProvidesExplPackage 5

Q

quark commands:

\quark_new:N 45, 46

quark internal commands:

\q__scontents_mark 45, 100, 132, 231, 241
 \q__scontents_stop 45

R

regex commands:

\regex_replace_all:nnN 150, 174, 255
 \RequirePackage 2, 3, 4

S

scan commands:

\scan_stop: 202
 \Scontents 1, 3, 3, 14, 14, 155
 scontents 2, 87

scontents internal commands:

__scontents_append_contents:nn 13, 76, 152,
 166, 176
 __scontents_atend_environment: .. 94, 138
 \g__scontents_end_verbatimsc_tl 48, 210, 217
 __scontents_environment_inline:w .. 13, 14,
 100, 117
 __scontents_environment_junk:nw 117
 __scontents_environment_keys:w 117
 __scontents_fcdef_print:N 15, 15, 184
 __scontents_fcdef_saved_EOL: ... 201, 213
 \l__scontents_fname_out_tl 11, 36, 61, 66, 104
 __scontents_getfrom_seq:nn 13, 76, 144, 168,
 179, 183, 187, 252
 __scontents_if_scantokens:Nw 225
 __scontents_if_scantokens:NwTF 231
 \l__scontents_macro_tmp_tl . 11, 36, 105, 107,
 146, 150, 151, 153
 \l__scontents_name_seq_cmd_tl . 24, 166, 168,
 176, 179
 \l__scontents_name_seq_env_tl . 20, 144, 152
 __scontents_norm:n 14, 15, 155

`\l_scontents_print_cmd_bool` .. 26, 167, 178
`\l_scontents_print_env_bool` 22, 143
`\l_scontents_rescan_tokens:w` .. 47, 153, 177, 205, 243
`\l_scontents_start_environment:w` ... 90, 96
`\l_scontents_stop_environment:` 93, 96
`\l_scontents_stored_to_seq:` 14, 138
`\l_scontents_storing_bool` .. 12, 39, 64, 140
`\l_scontents_strip_scantokens:N` 16, 208, 225, 254
`\l_scontents_strip_scantokens:n` .. 16, 225
`\l_scontents_temp_tl` 11, 36, 173, 174, 175, 177, 187, 188, 192, 252, 253, 254, 255, 257
`\l_scontents_typeverb_env_bool` 11, 39, 190, 191, 209
`\l_scontents_typevrb_env_bool` 40
`\l_scontents_verb:w` 14, 15, 155
`\l_scontents_writing_bool` 11, 39, 60, 65, 102, 111
`\l_scontents_xverb:w` 184
seq commands:
`\seq_clear_new:N` 263
`\seq_count:N` 261
`\seq_gput_right:Nn` 82
`\seq_if_exist:NTF` 80
`\seq_item:Nn` 86
`\seq_new:N` 81
`\setupsc` 2, 16, 247
str commands:
`\c_backslash_str` 123, 130

T

TeX and L^AT_EX 2_ε commands:

`\@ifpackagelater` 7
`\filecontentsdef@get` 15
`\scantokens` 15

tex commands:

`\tex_scantokens:D` 12, 47

tl commands:

`\tl_clear:N` 146
`\tl_gset_rescan:Nnn` 49
`\tl_if_blank:nTF` 78, 134, 199
`\tl_if_head_is_N_type:nTF` 229
`\tl_if_single_token:nTF` 233
`\tl_log:N` 151, 175, 188, 253
`\tl_new:N` 36, 37, 38, 48
`\tl_rescan:nn` 12
`\tl_set:Nn` 61, 66, 173, 187, 252, 254
`\tl_use:N` 153, 177

token commands:

`\token_if_eq_meaning:NNTF` 243
`\ttfamily` 256
`\typestored` 1, 3, 3, 4, 11, 15, 15, 184
`\typestored*` 4

U

use commands:

`\use:N` 104, 192
`\use:n` 203, 216
`\use_ii:nn` 234
`\use_none:nn` 233

V

`\verbatim` 222
`verbatimsc` 4, 184