

SCONTENTS

Stores \LaTeX CONTENTS

v1.8 — 2019-11-18*

©2019 by Pablo González†

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

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

Abstract

This package allows to store \LaTeX code, including “*verbatim*”, in *(sequences)* using the `l3seq` module of `expl3`. The *(stored content)* can be used as many times as desired in the document, additionally you can write to *(external files)* or show it in *(verbatim style)*.

Contents

1	Motivation and Acknowledgments . . .	1	5	Other commands provided	6
2	License and Requirements	1	5.1	The command <code>\meaningsc</code>	6
3	The <code>scontents</code> package	2	5.2	The command <code>\countsc</code>	6
3.1	Description of the package and load . .	2	5.3	The command <code>\cleanseqsc</code>	6
3.2	The TAB character	2	6	The <code>scontents</code> package in action . . .	7
3.3	Configuration of the options	2	7	Examples	7
3.4	Options Overview	3	7.1	From <code>answers</code> package	7
4	User interface	3	7.2	From <code>filecontentsdef</code> package . . .	8
4.1	The environment <code>scontents</code>	3	7.3	From <code>TeX-SX</code>	8
4.2	The command <code>\newenvsc</code>	4	7.4	Customization of <code>verbatimsc</code>	10
4.3	The command <code>\Scontents</code>	4	8	Change history	13
4.4	The command <code>\getstored</code>	5	9	Index of Documentation	14
4.5	The command <code>\foreachsc</code>	5	10	Implementation	15
4.6	The command <code>\tpestored</code>	5	11	Index of Implementation	38
4.7	The environment <code>verbatimsc</code>	6			

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 (lpp1), version 1.3 or later (<http://www.latex-project.org/lpp1.txt>). The software has the status “maintained”.

The `SCONTENTS` package loads `expl3`, `xparse` and `l3keys2e`. This package can be used with `plain`, `context`, `xelatex`, `lualatex`, `pdflatex` and the classical workflow `latex>dvi>ps>pdf`.

*This file describes a documentation for v1.8, last revised 2019-11-18.

†E-mail: pablgonz@educarchile.cl.

3 The scontents package

3.1 Description of the package and load

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

The package is loaded in the usual way:

For \LaTeX users

```
\usepackage{scontents}
```

or

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

The package options are not available for plain \TeX and ConTeXt, see 3.3.

For plain \TeX users

```
\input scontents.tex
```

For ConTeXt users

```
\usemodule{scontents}
```

3.2 The TAB character

Some users use horizontal TABs “ $\text{\hspace{1cm}}$ ” from keyboard to indented the source code of the document and depending on the text editor used, some will use real TABs (“hard tabs”), others with “soft tabs” (␣ or ␣␣␣) or both.

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

In a standard \LaTeX document, the character TAB “ $\text{\hspace{1cm}}$ ” are treated as explicit spaces (in most contexts) and is the behavior when *stored contents*, but when *writing files* these are preserved.

With a \TeX Live distribution, the TAB character is “printable” for `latex`, `pdflatex` and `lualatex`, but if you use `xelatex` you must add the `-8bit` option on the command line, otherwise you will get \TeX -TAB ($\text{\^{\hspace{1cm}}}$) in the *output file*.

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

3.3 Configuration of the options

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

```
\setupsc {keyval list}
```

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

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

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

```
store-all = {seq name} default: not used
```

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

```
overwrite = {true | false} default: false
```

Sets whether the *files* generated by `write-out` and `write-env` from the `scontents` environment will be rewritten. This key is available as a package option or using `\setupsc` or `scontents` environment.

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](#).

print-all = { *<true|false>* } default: false

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

force-eol = { *<true|false>* } default: false

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

width-tab = { *<integer>* } default: 1

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

3.4 Options Overview

Summary of available options:

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

4 User interface

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

4.1 The environment scontents

scontents

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

The `scontents` environment allows you to *<store>* and *<write>* content, including *verbatim* material. After the package has been loaded, the environment can be used both in the preamble and in the body of the document.

For the correct operation `\begin{scontents}` and `\end{scontents}` must be in different lines, all *<keys>* must be passed separated by commas and “without separation” of the start of the environment.

Comments “%” or “any character” after `\begin{scontents}` or [*<keyval list>*] on the same line are not supported, the package will return an “error” message if this happens. In a similar way comments “%” or “any character” after `\end{scontents}` on the same line the package will return a “warning” message.

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

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

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

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

Notes for plain T_EX and ConT_EXt users

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

<code>\scontents</code>	<code>\scontents[⟨keyval list⟩]</code>
<code>\endscontents</code>	<code>⟨env contents⟩</code> <code>\endscontents</code>

ConT_EXt users should use `\startscntents` and `\stopscntents`.

<code>\startscntents</code>	<code>\startscntents[⟨keyval list⟩]</code>
<code>\stopscntents</code>	<code>⟨env contents⟩</code> <code>\stopscntents</code>

Options for environment

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

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

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

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

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

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

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

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

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

4.2 The command `\newenvsc`

<code>\newenvsc</code>	<code>\newenvsc{⟨env name⟩}[⟨initial keys⟩]</code>
------------------------	--

The command `\newenvsc` allows you to create `⟨new environments⟩` based on the same characteristics of the `scontents` environment. The values entered in `[⟨initial keys⟩]` will be considered as the default values for this new environment and the valid `⟨keys⟩` are the same as those of the `scontents` environment. For example:

```
\newenvsc{myenvstore}[store-env=myseq,print-env=false]
```

created the `myenvstore` environment that stored the content in the `myseq` sequence and will not display the content when it is executed.

4.3 The command `\Scontents`

<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.

The `\Scontents*` command reads the `{⟨argument⟩}` under *verbatim* category code regimen. If its first delimiter is a brace, it will be assumed that the `{⟨argument⟩}` is nested into braces. Otherwise it will be assumed that the ending of that `⟨argument⟩` is delimited by that first delimiter `⟨del⟩` like command `\verb`.

Blank lines are preserved, escaped braces “\{” and “\}” must also be balanced if the argument is used with braces and TABs characters typed from the keyboard are converted into spaces. The starred argument (*) and [*key = val*] must not be separated by horizontal spaces between them and the command.

Both versions can be used anywhere in the document and cannot be used as an *argument* for other command.

Options for command

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

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

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

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

Sets if the *stored content* is displayed or not at the time of running the command. The content is extracted from the *sequence* in which it is stored.

4.4 The command `\getstored`

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

The command `\getstored` gets the content stored in *{⟨seq name⟩}* according to the *⟨index⟩* in which it was stored. The command is robust and can be used as an *argument* for another command. If the optional argument is not passed, the default value is the “last element” stored in *{⟨seq name⟩}*.

4.5 The command `\foreachsc`

`\foreachsc` `\foreachsc[⟨key = val⟩]{⟨seq name⟩}`

The command `\foreachsc` goes through and executes the command `\getstored` on the contents stored in *{⟨seq name⟩}*. If you pass without options run `\getstored` on all contents stored in *{⟨seq name⟩}*.

Options for command

`sep = {⟨code⟩}` default: *empty*

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

`step = {⟨integer⟩}` default: *1*

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

`start = {⟨integer⟩}` default: *1*

Sets the *⟨index⟩* number of the *{⟨seq name⟩}* from which execution will start. The value must be a *positive integer*.

`stop = {⟨integer⟩}` default: *total*

Sets the *⟨index⟩* number of the *{⟨seq name⟩}* from which execution it will finish executing. The value must be a *positive integer*.

`before = {⟨code⟩}` default: *empty*

Sets the *{⟨code⟩}* that will be executed *before* each content stored in *{⟨seq name⟩}*. The *{⟨code⟩}* must be passed between braces.

`after = {⟨code⟩}` default: *empty*

Sets the *{⟨code⟩}* that will be executed *after* each content stored in *{⟨seq name⟩}*. The *{⟨code⟩}* must be passed between braces.

`wrapper = {⟨code⟩ {#1} more code}` default: *empty*

Wraps the content stored in *{⟨seq name⟩}* referenced by *{#1}*. The *{⟨code⟩}* must be passed between braces. For example `\foreachsc[wrapper={\makebox[1em][l]{#1}}]{contents}`.

4.6 The command `\typetored`

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

The command `\typetored` internally places the content stored in the *{⟨seq name⟩}* into the `verbatimsc` environment. The *⟨index⟩* corresponds to the position in which the content is stored in the *{⟨seq name⟩}*.

If the optional argument is not passed it defaults to the first element stored in the $\{\langle seq name \rangle\}$. The key `width-tab` is available for this command.

4.7 The environment `verbatimsc`

`verbatimsc`

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

One consideration to keep in mind is that this is a *representation* of the $\langle stored content \rangle$ in a *verbatim* environment and not a real *verbatim* environment. The `verbatim` package is not compatible with the implementation of the `verbatimsc` 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}
  }
}{}
\end{verbatimsc}
```

5 Other commands provided

5.1 The command `\meaningsc`

`\meaningsc`

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

The command `\meaningsc` executes `\meaning` on the content stored in $\{\langle seq name \rangle\}$. The $\langle index \rangle$ corresponds to the position in which the content is stored in the $\{\langle seq name \rangle\}$.

If the optional argument is not passed it defaults to the first element stored in the $\{\langle seq name \rangle\}$. The key `width-tab` is available for this command.

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 `\cleanseqsc`

`\cleanseqsc`

`\cleanseqsc` $\{\langle seq name \rangle\}$

The command `\cleanseqsc` remove all contents stored in $\{\langle seq name \rangle\}$.

6 The **SCONTENTS** package in action

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

Abstract

This package allows to store \LaTeX code, including “*verbatim*”, in *sequences* using the `l3seq` module of `expl3`. The *stored content* can be used as many times as desired in the document, additionally you can write to *external files* or show it in *verbatim style*.

And the description of the package?

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

I’ve only written:

```
\begin{abstract}
This package allows to store \hologo{LaTeX} code, including \enquote{\emph{verbatim}},
in \mymeta{sequences} using the \mypkg{l3seq} module of \mypkg{expl3}. The \mymeta{stored
content} can be used as many times as desired in the document, additionally you can write
to \mymeta{external files} or show it in \mymeta{verbatim style}.
\end{abstract}
```

and

The `\mypkg{scontents}` package allows to `\mymeta{store contents}` in `\mymeta{sequences}` or `\mymeta{external files}`. In some ways it is similar to the `\mypkg{filecontentsdef}` package, with the difference in which the `\mymeta{content}` is stored. The idea behind this package is to get an approach to $\text{\hologo{ConTeXt}}$ `\enquote{\emph{buffers}}` by making use `\mymeta{sequences}`.

Of course, I didn’t copy and paste. The real code they were written with is:

```
1 \begin{scontents}[store-env=abstract,print-env=true]
2 \begin{abstract}
3 This package allows to store \hologo{LaTeX} code, including \enquote{\emph{verbatim}},
4 in \mymeta{sequences} using the \mypkg{l3seq} module of \mypkg{expl3}. The \mymeta{stored
5 content} can be used as many times as desired in the document, additionally you can write
6 to \mymeta{external files} or show it in \mymeta{verbatim style}.
7 \end{abstract}
8 \end{scontents}
```

and

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

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

7 Examples


These are some adapted examples that have served as inspiration for the creation of this package. The examples are attached to this documentation and can be extracted from your PDF viewer or from the command line by running:

```
$ pdftdetach -saveall scontents.pdf
```

and then you can use the excellent [arara](https://www.ctan.org/pkg/arara)³ tool to compile them.

7.1 From answers package

Example 1

Adaptation of example 1 of the package `answers` .

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



```

1 % arara: pdflatex
2 % arara: clean: { extensions: [ aux, log] }
3 \documentclass{article}
4 \usepackage[store-cmd=solutions]{scontents}
5 \newtheorem{ex}{Exercise}
6 \begin{document}
7 \section{Problems}
8 \begin{ex}
9 First exercise
10 \Scontents{First solution.}
11 \end{ex}
12
13 \begin{ex}
14 Second exercise
15 \Scontents{Second solution.}
16 \end{ex}
17
18 \section{Solutions}
19 \foreachsc[sep={\\[10pt]}]{solutions}
20 \end{document}

```

7.2 From filecontentsdef package

Example 2

Adaptation of example from package filecontentsdef .

```

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

```

7.3 From TeX-SX

Example 3

Adapted from LaTeX equivalent of ConTeXt buffers .

```

1 % arara: pdflatex
2 % arara: clean: { extensions: [ aux, log] }
3 \documentclass{article}
4 \usepackage[store-cmd=tikz]{scontents}
5 \usepackage{tikz}
6 \setlength{\parindent}{0pt}
7 \pagestyle{empty}
8 \Scontents*{\matrix{\node (a) {$a$} ; & \node (b) {$b$} ; \\ } ;}
9 \Scontents*{\matrix[ampersand replacement=\&]
10 { \node (a) {$a$} ; \& \node (b) {$b$} ; \\ } ;}
11 \Scontents*{\matrix{\node (a) {$a$} ; & \node (b) {$b$} ; \\ } ;}

```



```

12 \begin{document}
13 \section{tikzpicture}
14 \begin{tikzpicture}
15 \getstored[1]{tikz}
16 \end{tikzpicture}
17
18 \begin{tikzpicture}
19 \getstored[2]{tikz}
20 \end{tikzpicture}
21
22 \begin{tikzpicture}
23 \getstored[3]{tikz}
24 \end{tikzpicture}
25
26 \begin{scontents}[store-env=buffer]
27 Hello World!
28
29 This is a \verb*|fake poor man's buffer :)|.
30 \end{scontents}
31
32 \section{source tikz}
33 \typestored[1]{tikz}
34 \typestored[2]{tikz}
35 \typestored[3]{tikz}
36
37 \section{fake buffer}
38 \subsection{real content}
39 \getstored[1]{buffer}
40 \subsection{verbatim style}
41 \typestored[1]{buffer}
42 \subsection{meaning}
43 \meaningsc[1]{buffer}
44
45 \section{tikz again}
46 \foreachsc[before={\begin{tikzpicture}},after={\end{tikzpicture}},sep={\\[10pt]}]{tikz}
47 \end{document}

```

Example 4

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

```

1 % arara: pdflatex
2 % arara: clean: { extensions: [ aux, log ] }
3 \documentclass{article}
4 \usepackage{scontents}
5 \pagestyle{empty}
6 \begin{document}
7 \begin{scontents}[store-env=main]
8 Something for main A.
9 \end{scontents}
10
11 \begin{scontents}[store-env=main]
12 Something for \verb|main B|.
13 \end{scontents}
14
15 \begin{scontents}[store-env=other]
16 Something for \verb|other|.
17 \end{scontents}
18
19 \textbf{Let's print them}
20
21 This is first stored in main: \getstored[1]{main}\par
22 This is second stored in main: \getstored[2]{main}\par
23 This is stored in other: \getstored[1]{other}
24
25 \textbf{Print all of stored in main}\par
26 \foreachsc[sep={\\[10pt]}]{a}
27 \end{document}

```

Example 5

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

```

1 % arara: pdflatex
2 % arara: clean: { extensions: [ aux, log ] }
3 \documentclass{article}
4 \usepackage{scontents}
5 \pagestyle{empty}
6 \setlength{\parindent}{0pt}
7 \begin{document}
8 \section{Problem stated the first time}
9 \begin{scontents}[print-env=true,store-env=problem]
10 This is normal text.
11 \verb|This is from the verb command.|
12 \verb*|This is from the verb* command.|
13 This is normal text.
14 \begin{verbatim}
15 This is from the verbatim environment:
16 &{%}~
17 \end{verbatim}
18 \end{scontents}
19 \section{Problem restated}
20 \getstored[1]{problem}
21 \section{Problem restated once more}
22 \getstored[1]{problem}
23 \end{document}

```

Example 6

Adapted from [Environment hiding its content](#) .


```

1 % arara: pdflatex
2 % arara: clean: { extensions: [ aux, log ] }
3 \documentclass[10pt]{article}
4 \usepackage{scontents}
5 \newenv{forshort}[store-env=forshort,print-env=false]
6 \begin{document}
7
8 Something in the whole course.
9
10 \begin{forshort}
11     Just a summary...
12 \end{forshort}
13
14 \end{document}

```

7.4 Customization of verbatimsc

Example 7

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

```

1 \documentclass{article}
2 % arara: pdflatex
3 % arara: clean: { extensions: [ aux, log ] }
4 \usepackage{scontents}
5 \makeatletter
6 \let\verbatimsc\@undefined
7 \let\endverbatimsc\@undefined
8 \makeatother
9 \usepackage{fvextra}
10 \usepackage{xcolor}
11 \definecolor{mygray}{gray}{0.9}
12 \usepackage{tcolorbox}
13 \newenvironment{verbatimsc}%
14 {\VerbatimEnvironment
15 \begin{tcolorbox}[colback=mygray, boxsep=0pt, arc=0pt, boxrule=0pt]
16 \begin{Verbatim}[fontsize=\scriptsize, breaklines, breakafter=*, breaksymbolsep=0.5em,
17 breakaftersymbolpre={\,\tiny\ensuremath{\lfloor}}}%
18 {\end{Verbatim}}%
19 \end{tcolorbox}}
20 \setlength{\parindent}{0pt}
21 \pagestyle{empty}
22 \begin{document}
23 \section{Test \texttt{\textbackslash begin\{scontents\}} whit \texttt{fancyvrb}}

```

```

24 Test \verb+{scontents}+ \par
25
26 \begin{scontents}
27 Using \verb+scontents+ env no \verb+[key=val]+, save in seq \verb+contents+
28 with index 1.
29
30 Prove new \Verb*{ fancyvrb whit braces } and environment \verb+Verbatim*+
31 \begin{verbatim}
32 verbatim environment
33 \end{verbatim}
34 \end{scontents}
35
36 \section[Test \texttt{\textbackslash Scontents} whit \texttt{fancyvrb}]
37 \Scontents{ We have coded this in \LaTeX: $E=mc^2$}.
38
39 \section[Test \texttt{\textbackslash getstored}]
40 \getstored[1]{contents}\par
41 \getstored[2]{contents}
42
43 \section[Test \texttt{\textbackslash meaningsc}]
44 \meaningsc[1]{contents}\par
45 \meaningsc[2]{contents}
46
47 \section[Test \texttt{\textbackslash typestored}]
48 \typestored[1]{contents}
49 \typestored[2]{contents}
50 \end{document}

```

Example 8

Customization of `verbatimsc` using the listings package .

```

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

```

```

41 \Scontents*+ We have coded this in \lstinline[basicstyle=\ttfamily]|\LaTeX: $E=mc^2$|
42 and more.+
43
44 \section{Test \texttt{\textbackslash getstored}}
45 \getstored[2]{contents}\par
46 \getstored[1]{contents}
47
48 \section{Test \texttt{\textbackslash typestored}}
49 \typestored[1]{contents}
50 \typestored[2]{contents}
51 \end{document}

```

Example 9

Customization of `verbatimsc` using the minted package .

```

1 % arara: xelatex : {shell: true, options: [-8bit]}
2 % arara: clean: { extensions: [ aux, log ] }
3 \documentclass{article}
4 \usepackage{scontents}
5 \makeatletter
6 \let\verbatimsc\@undefined
7 \let\endverbatimsc\@undefined
8 \makeatother
9 \usepackage{minted}
10 \newminted{tex}{linenos}
11 \newenvironment{verbatimsc}{\VerbatimEnvironment\begin{texcode}}{\end{texcode}}
12 \pagestyle{empty}
13 \begin{document}
14 \section{Test \texttt{\textbackslash begin\{scontents\}} whit \texttt{minted}}
15 Test \verb+{scontents}+ \par
16
17 \begin{scontents}[overwrite,write-env=\jobname.tsc,force-eol=true]
18 Using \verb+scontents+ env no \verb+[key=val]+, save in seq \verb+contents+
19 with index 1.\par
20
21 Prove new \Verb*{ new fvextra whit braces } and environment \verb+Verbatim*+
22 \begin{Verbatim}[obeytabs, showtabs, tab=\rightarrowfill, tabcolor=red]
23 No tab
24         One real tab
25             Two real Tab plus         one tab
26 \end{Verbatim}
27 \end{scontents}
28
29 \section{See \Verb{\jobname.tsc}}
30 Read \Verb{\jobname.tsc} (shows TABs as red arrows):
31 \VerbatimInput[obeytabs, showtabs, tab=\rightarrowfill, tabcolor=red]{\jobname.tsc}
32
33 \section{Test \texttt{\textbackslash Scontents} whit \texttt{minted}}
34
35 \Scontents{ We have coded \par this in \LaTeX: $E=mc^2$_.}
36
37 \section{Test \texttt{\textbackslash getstored}}
38 \getstored[1]{contents}\par
39 \getstored[2]{contents}
40
41 \section{Test \texttt{\textbackslash typestored}}
42 \typestored[1]{contents}
43 \typestored[2]{contents}
44 \end{document}

```

8 Change history

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

- v1.8 (ctan), 2019-11-18**
 - Add `\newenvsc` command.
 - Fix nested environment in plain \TeX and \ConTeXt .
 - Modified default value in `\getstored`.
 - Add `overwrite` key to reduce I/O operations.
 - Deleted an unnecessary group in the code.
- v1.7 (ctan), 2019-10-29**
 - The `verbatimsc` environment was rewritten.
 - Minor adjustments in documentation.
- v1.6 (ctan), 2019-10-26**
 - The internal behavior of `\getstored` has been modified.
 - The internal behavior of `\foreachsc` has been modified.
 - Corrected file extension for \ConTeXt .
 - Remove spurious warning.
- v1.5 (ctan), 2019-10-24**
 - Add support for plain \TeX and \ConTeXt .
 - Split internal code for optimization.
 - Add support for vertical spaces in `key=val`.
 - Add `\foreachsc` command.
 - Check if `verbatim` package is loaded.
- v1.4 (ctan), 2019-10-03**
 - Add `store-all` key.
 - Messages and keys were separated.
 - Restructuring of documentation.
 - Now the version of `expl3` is checked instead of `xparse`.
 - The internal behavior of `force-eol` has been modified.
- v1.3 (ctan), 2019-09-24**
 - The environment 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:`.
- v1.1 (ctan), 2019-08-12**
 - Extension of documentation.
 - Replace `\tex_endinput:D` by `\file_input_stop:`.
- v1.0 (ctan), 2019-07-30**
 - First public release.

9 Index of Documentation

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

C	
Commands provide by SCONTENTS:	
\Sccontents*	3, 4
\Sccontents	2-4
\cleanseqsc	6
\countsc	6
\endscontents	4
\foreachsc	5
\getstored	3, 5
\meaningsc	2, 3, 6
\newenvsc	4
\scontents	4
\setupsc	2-5
\startscontents	4
\stopscontents	4
\typestored	2, 3, 5, 6
E	
Environment provide by SCONTENTS:	
scontents	2-4
verbatimsc	5, 6, 10-12
Environments	
Verbatim	2
filecontentsdefmacro	1
lstlisting	2
K	
Keys	
after	5
before	5
force-eol	3-5
overwrite	2, 3
print-all	3
print-cmd	3, 5
print-env	3, 4
sep	5
L	
start	5
step	5
stop	5
store-all	2, 3
store-cmd	2, 3, 5
store-env	2-4
verb-font	2, 3
width-tab	3, 6
wrapper	5
write-env	2-4
write-out	2-4
L	
\lstinline	4
M	
\meaning	6
P	
Packages	
answers	7
expl3	1, 7, 13
fancyvrb	2, 3, 6, 10
filecontentsdef	1, 2, 7, 8, 13
fvextra	4
l3keys2e	1
l3seq	1, 7
listings	2, 4, 6, 11
minted	6, 12
scontents	1, 2, 7, 13
tcolorbox	10
verbatim	6
xparse	1
V	
\Verb	4
\verb	4

10 Implementation

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

10.1 Declaration of the package

First we set up the module name for `l3doc`:

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

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

```
2 <loader>\def\ScontentsFileDate{2019-11-18}%
3 <core>\def\ScontentsCoreFileDate{2019-11-18}%
4 <*loader>
5 \def\ScontentsFileVersion{1.8}%
6 \def\ScontentsFileDescription{Stores LaTeX contents in memory or files}%
```

The `LaTeX` loader is fairly simple: just load the dependencies, load the core code, and then set interfaces up.

We also check if the verbatim package is loaded and show a compatibility warning.

```
7 <*latex>
8 \RequirePackage{expl3,xparse,l3keys2e}[2019/05/28]
9 \ProvidesExplPackage
10   {scontents} {\ScontentsFileDate} {\ScontentsFileVersion} {\ScontentsFileDescription}
11 \ifpackageloaded { verbatim }
12 {
13   \msg_set:nnn { scontents } { unsupported-verbatim }
14   {
15     The~implementation~of~the~'verbatimsc'~environment~used~by~
16     \iow_char:N \typestored-is-not-compatible-with-package~'verbatim'.~
17     Review~the~documentation~and~redefine~the~'verbatimsc'~environment.
18   }
19   \msg_warning:nn { scontents } { unsupported-verbatim }
20 } { }
21 </latex>
```

The Plain and `ConTeXt` loaders are similar (probably because I don't know how to make a proper `ConTeXt` module :-). We define a `LaTeX`-style `\ver@scontents.sty` macro with version info (just in case):

```
22 <*!latex>
23 <context>\writestatus{loading}{User Module scontents v\ScontentsFileVersion}
24 <context>\unprotect
25 \input expl3-generic.tex
26 \ExplSyntaxOn
27 \tl_gset:cx { ver @ scontents . sty } { \ScontentsFileDate\space
28   v\ScontentsFileVersion\space \ScontentsFileDescription }
29 \iow_log:x { Package: ~ scontents ~ \use:c { ver @ scontents . sty } }
30 </!latex>
```

In Plain, check that the package isn't being loaded twice (`LaTeX` and `ConTeXt` already defend against that):

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

```
\g__scontents_end_verbatimsc_tl
\c__scontents_end_env_tl
```

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

```
41 \tl_new:N \g__scontents_end_verbatimsc_tl
42 \tl_gset_rescan:Nnn
43   \g__scontents_end_verbatimsc_tl
44   {
45     \char_set_catcode_other:N \
46 <*latex>
```



```

47   \char_set_catcode_other:N \{
48   \char_set_catcode_other:N \}
49 \</latex>
50 }
51 \<latex> { \end{verbatim} }
52 \<plain> { \endverbatim }
53 \<context> { \stopverbatim }
54 \tl_const:Nx \c__scontents_end_env_tl
55 {
56   \c_backslash_str
57 \<latex | plain> end
58 \<context> stop
59 \<latex> \c_left_brace_str
60   \exp_not:N \l__scontents_env_name_tl
61 \<latex> \c_right_brace_str
62 }

```

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

Now we load the core `SCONTENTS` code:

```

63 \file_input:n { scontents-code.tex }

```

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

```

64 \cs_new:Npn \__scontents_format_case:nnn #1 #2 #3
65 \<latex> {#1} % LaTeX
66 \<plain> {#2} % Plain/Generic
67 \<context> {#3} % ConTeXt

```

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

```

68 \</loader>
69 \<*core>
70 \begingroup
71   \def\next{\endgroup}%
72   \expandafter\ifx\csname PackageError\endcsname\relax
73     \begingroup
74       \def\next{\endgroup\endgroup}%
75       \def\PackageError#1#2#3%
76         {%
77           \endgroup
78           \errhelp{#3}%
79           \errmessage{#1 Error: #2!}%
80         }%
81     \fi
82     \expandafter\ifx\csname ScontentsFileDate\endcsname\relax
83       \def\next
84         {%
85           \PackageError{scontents}{No scontents loader detected}
86             {%
87               You have attempted to use the scontents code directly rather than using
88               the correct loader. Loading of scontents will abort.
89             }%
90           \endgroup
91           \endinput
92         }
93     \else
94       \ifx\ScontentsFileDate\ScontentsCoreFileDate
95     \else
96       \def\next
97         {%
98           \PackageError{scontents}{Mismatched~scontents~files~detected}
99             {%
100               You~have~attempted~to~load~scontents~with~mismatched~files:~
101               probably~you~have~one~or~more~files~'locally~installed'~which~
102               are~in~conflict.~Loading~of~scontents~will~abort.
103             }%
104           \endgroup
105           \endinput
106         }%
107       \fi
108     \fi

```

```
109 \next
```

10.2 Definition of common keys

We create some common *keys* that will be used by the options passed to the package as well as by the environments and commands defined.

```
110 \keys_define:nn { scontents }
111 {
112   store-env .tl_set:N      = \l__scontents_name_seq_env_tl,
113   store-env .initial:n     = contents,
114   store-env .value_required:n = true,
115   store-cmd .tl_set:N      = \l__scontents_name_seq_cmd_tl,
116   store-cmd .initial:n     = contents,
117   store-cmd .value_required:n = true,
118   verb-font .tl_set:N      = \l__scontents_verb_font_tl,
119   verb-font .value_required:n = true,
120   print-env .bool_set:N    = \l__scontents_print_env_bool,
121   print-env .initial:n     = false,
122   print-env .default:n     = true,
123   print-cmd .bool_set:N    = \l__scontents_print_cmd_bool,
124   print-cmd .initial:n     = false,
125   print-cmd .default:n     = true,
126   force-eol .bool_set:N    = \l__scontents_forced_eol_bool,
127   force-eol .initial:n     = false,
128   force-eol .default:n     = true,
129   overwrite .bool_set:N    = \l__scontents_overwrite_bool,
130   overwrite .initial:n     = false,
131   overwrite .default:n     = true,
132   width-tab .int_set:N     = \l__scontents_tab_width_int,
133   width-tab .initial:n     = 1,
134   width-tab .value_required:n = true,
135   print-all .meta:n       = { print-env = #1 , print-cmd = #1 },
136   print-all .default:n    = true,
137   store-all .meta:n       = { store-env = #1 , store-cmd = #1 },
138   store-all .value_required:n = true
139 }
140 </core>
141 <loader>\keys_define:nn { scontents }
142 <latex> { verb-font .initial:n = \ttfamily }
143 <plain|context> { verb-font .initial:n = \tt }
```

In \LaTeX mode we load `l3keys2e` process the *keys* as options passed on to the package, the package `l3keys2e` will verify the *keys* and will return an error when they are *unknown*.

```
144 <latex>\ProcessKeysOptions { scontents }
145 <*core>
```

10.3 Internal variables

Now we declare the internal variables we will use.

<code>\l__scontents_macro_tmp_tl</code>	<code>\l__scontents_macro_tmp_tl</code> is a temporary token list to hold the contents of the macro/environment,
<code>\l__scontents_fname_out_tl</code>	<code>\l__scontents_fname_out_tl</code> is used as the name of the output file, when there's one, <code>\l__scontents_file_tl</code> holds the contents of an environment as it's being read, and <code>\l__scontents_temp_tl</code> and <code>\g__scontents_temp_tl</code> are generic temporary token lists.
<code>\l__scontents_file_tl</code>	
<code>\g__scontents_temp_tl</code>	
<code>\l__scontents_foreach_name_seq_tl</code>	<code>\l__scontents_foreach_name_seq_tl</code> is the name assigned to the sequence on which the loop will be made, <code>\l__scontents_foreach_before_tl</code> and <code>\l__scontents_foreach_after_tl</code> are token lists in which the assigned material will be placed before and after the execution of the <code>\foreachsc</code> loop.
<code>\l__scontents_foreach_before_tl</code>	
<code>\l__scontents_foreach_after_tl</code>	

```
146 \tl_new:N \l__scontents_macro_tmp_tl
147 \tl_new:N \l__scontents_fname_out_tl
148 \tl_new:N \l__scontents_temp_tl
149 \tl_new:N \l__scontents_file_tl
150 \tl_new:N \g__scontents_temp_tl
151 \tl_new:N \l__scontents_foreach_name_seq_tl
152 \tl_new:N \l__scontents_foreach_before_tl
153 \tl_new:N \l__scontents_foreach_after_tl
```

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

`\l__scontents_seq_item_int` `\l__scontents_seq_item_int` stores the index in the sequence of the item requested to `\typestored` or `\meaningsc`. `\l__scontents_env_nesting_int` stores the current nesting level of the `scontents` environment. `\l__scontents_foreach_stop_int` will save the value at which the `\foreachsc` loop will stop.

```

154 \int_new:N \l__scontents_foreach_stop_int
155 \int_new:N \l__scontents_seq_item_int
156 \int_new:N \l__scontents_env_nesting_int
157 \int_new:N \l__scontents_tmpa_int

```

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

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

```

158 \bool_new:N \l__scontents_writing_bool
159 \bool_set_false:N \l__scontents_writing_bool
160 \bool_new:N \l__scontents_storing_bool
161 \bool_set_true:N \l__scontents_storing_bool

```

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

`\l__scontents_foreach_before_bool` Boolean variables used by the `\foreachsc` loop.

```

\l__scontents_foreach_after_bool
\l__scontents_foreach_stop_bool
\l__scontents_foreach_wrapper_bool
162 \bool_new:N \l__scontents_foreach_before_bool
163 \bool_set_false:N \l__scontents_foreach_before_bool
164 \bool_new:N \l__scontents_foreach_after_bool
165 \bool_set_false:N \l__scontents_foreach_after_bool
166 \bool_new:N \l__scontents_foreach_stop_bool
167 \bool_set_false:N \l__scontents_foreach_stop_bool
168 \bool_new:N \l__scontents_foreach_wrapper_bool
169 \bool_set_false:N \l__scontents_foreach_wrapper_bool

```

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

`\l__scontents_foreach_print_seq` The `\l__scontents_foreach_print_seq` is the sequence used by `\foreachsc`.

```

170 \seq_new:N \l__scontents_foreach_print_seq

```

(End definition for `\l__scontents_foreach_print_seq`.)

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

```

171 \str_const:Nx \c__scontents_hidden_space_str
172 { \c_percent_str \c_circumflex_str \c_circumflex_str A scheol \c_percent_str }

```

(End definition for `\c__scontents_hidden_space_str`.)

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

```

173 \quark_new:N \q__scontents_stop
174 \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.

```

175 \iow_new:N \l__scontents_file_iow

```

(End definition for `\l__scontents_file_iow`.)

`__scontents_rescan_tokens:n` `\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.

```

176 \cs_new_protected:Npn \__scontents_rescan_tokens:n #1 { \tex_scantokens:D {#1} }
177 \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:
178 \cs_new:Npx \__scontents_tab: { \c_space_tl }
179 \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:TF
180 \cs_generate_variant:Nn \tl_remove_once:Nn { NV }
181 \cs_generate_variant:Nn \tl_replace_all:Nnn { Nx, Nxx, Nnx }
182 \cs_generate_variant:Nn \msg_error:nnnn { nnn }
183 \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:TF`.)

10.4 Defining keys for the environment and commands

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

10.4.1 Keys for environment `scontents`

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

```
184 \keys_define:nn { scontents / scontents }
185 {
186   write-env .code:n = {
187     \bool_set_true:N \__scontents_writing_bool
188     \tl_set:Nn \__scontents_fname_out_tl {#1}
189   },
190   write-out .code:n = {
191     \bool_set_false:N \__scontents_storing_bool
192     \bool_set_true:N \__scontents_writing_bool
193     \tl_set:Nn \__scontents_fname_out_tl {#1}
194   },
195   write-env .value_required:n = true,
196   write-out .value_required:n = true,
197   print-env .meta:nn = { scontents } { print-env = #1 },
198   print-env .default:n = true,
199   store-env .meta:nn = { scontents } { store-env = #1 },
200   force-eol .meta:nn = { scontents } { force-eol = #1 },
201   force-eol .default:n = true,
202   overwrite .meta:nn = { scontents } { overwrite = #1 },
203   overwrite .default:n = true,
204   unknown .code:n = { \__scontents_parse_environment_keys:n {#1} }
205 }
```

10.4.2 Keys for command `\Scontents`

We define a set of `⟨keys⟩` for commands `\Scontents` and `\Scontents*`.

```
206 \keys_define:nn { scontents / Scontents }
207 {
208   print-cmd .meta:nn = { scontents } { print-cmd = #1 },
209   print-cmd .default:n = true,
210   store-cmd .meta:nn = { scontents } { store-cmd = #1 },
211   force-eol .meta:nn = { scontents } { force-eol = #1 },
212   force-eol .default:n = true,
213   unknown .code:n = { \__scontents_parse_command_keys:n {#1} }
214 }
```

10.4.3 Keys for command `\foreachsc`

We define a set of `⟨keys⟩` for command `\foreachsc`.

```
215 \keys_define:nn { scontents / foreachsc }
216 {
217   before .code:n = {
218     \bool_set_true:N \__scontents_foreach_before_bool
219     \tl_set:Nn \__scontents_foreach_before_tl {#1}
220   },
```

```

221   before .value_required:n = true,
222   after  .code:n           = {
223       \bool_set_true:N \l__scontents_foreach_after_bool
224       \tl_set:Nn \l__scontents_foreach_after_tl {#1}
225   },
226   after  .value_required:n = true,
227   start  .int_set:N         = \l__scontents_foreach_start_int,
228   start  .value_required:n = true,
229   start  .initial:n         = 1,
230   stop   .code:n           = {
231       \bool_set_true:N \l__scontents_foreach_stop_bool
232       \int_set:Nn \l__scontents_foreach_stop_int {#1}
233   },
234   stop   .value_required:n = true,
235   step   .int_set:N         = \l__scontents_foreach_step_int,
236   step   .value_required:n = true,
237   step   .initial:n         = 1,
238   wrapper .code:n          = {
239       \bool_set_true:N \l__scontents_foreach_wrapper_bool
240       \cs_set_protected:Npn
241         \l__scontents_foreach_wrapper:n ##1 {#1}
242   },
243   wrapper .value_required:n = true,
244   sep     .tl_set:N         = \l__scontents_foreach_sep_tl,
245   sep     .initial:n        = {},
246   sep     .value_required:n = true,
247   unknown .code:n          = { \l__scontents_parse_foreach_keys:n {#1} }
248 }

```

10.4.4 Key for commands \typestored and \meaningsc

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

```

249 \keys_define:nn { scontents / typemeaning }
250 {
251   width-tab .meta:nn = { scontents } { width-tab = #1 },
252   unknown   .code:n  = { \l__scontents_parse_type_meaning_key:n {#1} }
253 }

```

10.5 Handling undefined keys

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

10.5.1 Undefined keys for environment scontents

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

```

254 \cs_new_protected:Npn \l__scontents_parse_environment_keys:n #1
255 { \exp_args:NV \l__scontents_parse_environment_keys:nn \l_keys_key_tl {#1} }
256 \cs_new_protected:Npn \l__scontents_parse_environment_keys:nn #1#2
257 {
258   \tl_if_blank:nTF {#2}
259   { \msg_error:nnn { scontents } { env-key-unknown } {#1} }
260   { \msg_error:nnnn { scontents } { env-key-value-unknown } {#1} {#2} }
261 }

```

(End definition for `\l__scontents_parse_environment_keys:n` and `\l__scontents_parse_environment_keys:nn`)

10.5.2 Undefined keys for \Scontents and \Scontents*

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

```

262 \cs_new_protected:Npn \l__scontents_parse_command_keys:n #1
263 { \exp_args:NV \l__scontents_parse_command_keys:nn \l_keys_key_tl {#1} }
264 \cs_new_protected:Npn \l__scontents_parse_command_keys:nn #1#2
265 {
266   \tl_if_blank:nTF {#2}
267   { \msg_error:nnn { scontents } { cmd-key-unknown } {#1} }
268   { \msg_error:nnnn { scontents } { cmd-key-value-unknown } {#1} {#2} }

```

```
269 }
```

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

10.5.3 Undefined keys for `\foreachsc`

```
\__scontents_parse_foreach_keys:n
\__scontents_parse_foreach_keys:nn
```

We check the `<keys>` passed to command `\foreachsc` and process it with `__scontents_parse_foreach_keys:n`, if the `<key>` is *unknown* we return an error message.

```
270 \cs_new_protected:Npn \__scontents_parse_foreach_keys:nn #1#2
271 {
272   \tl_if_blank:nTF {#2}
273     { \msg_error:nnn { scontents } { for-key-unknown } {#1} }
274     { \msg_error:nnnn { scontents } { for-key-value-unknown } {#1} {#2} }
275   }
276 \cs_new_protected:Npn \__scontents_parse_foreach_keys:n #1
277 { \exp_args:NV \__scontents_parse_foreach_keys:nn \l_keys_key_tl {#1} }
```

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

10.5.4 Undefined keys for `\typestored` and `\meaningsc`

```
\__scontents_parse_type_meaning_key:n
\__scontents_parse_type_meaning_key:nn
```

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[][\langle options \rangle]{...}` syntax, we'll make the commands have a single optional argument which is processed by `l3keys`, and the unknown keys are brought here to `__scontents_parse_type_meaning_key:n` to process.

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

```
278 \cs_new_protected:Npn \__scontents_parse_type_meaning_key:n #1
279 { \exp_args:NV \__scontents_parse_type_meaning_key:nn \l_keys_key_tl {#1} }
280 \cs_new_protected:Npn \__scontents_parse_type_meaning_key:nn #1#2
281 {
282   \tl_if_empty:ftF { \int_to_roman:n { -0 #1 } }
283   {
284     \tl_if_blank:nTF {#2}
285       { \int_set:Nn \l__scontents_seq_item_int {#1} }
286       { \msg_error:nnnn { scontents } { type-key-value-unknown } {#1} {#2} }
287     }
288   {
289     \tl_if_blank:nTF {#2}
290       { \msg_error:nnn { scontents } { type-key-unknown } {#1} }
291       { \msg_error:nnnn { scontents } { type-key-value-unknown } {#1} {#2} }
292     }
293 }
```

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

10.6 Compatibility layer with Plain

When loading the package outside of \TeX we can't usually use `xparse`. However since `xparse` doesn't actually hold any dependency with \TeX except for package-loading commands, we can emulate those commands (much like in `miniltx`) so that `xparse` is loadable in any format.

The bunch of macros below is adapted from the \TeX kernel (greatly simplified).

```
294 </core>
295 <*loader&!latex>
296 \seq_new:N \l__scontents_compat_seq
297 \cs_new_protected:Npn \__scontents_compat_redefine:Npn #1
298 {
299   \seq_put_right:Nn \l__scontents_compat_seq {#1}
300   \cs_set_eq:cN { __scontents_saved_\cs_to_str:N #1: } #1
301   \cs_new_protected:Npn #1
302   {
303     \cs_new_protected:Npn \__scontents_compat_restore:
304       { \seq_map_function:NN \l__scontents_compat_seq \__scontents_compat_restore:N }
305     \cs_new_protected:Npn \__scontents_compat_restore:N #1
306     {
307       \cs_set_eq:Nc #1 { __scontents_saved_\cs_to_str:N #1: }
```

```

308 \cs_undefine:c { __scontents_saved_\cs_to_str:N #1: }
309 }
310 \cs_generate_variant:Nn __scontents_compat_redefine:Npn { c }
311 \cs_new_protected:Npn __scontents_optarg:nn #1 #2
312 { \peek_charcode_ignore_spaces:NTF [ {#1} {#1[#2]} ] }
313 \cs_new_protected:Npn __scontents_stararg:nn #1 #2
314 { \peek_charcode_remove_ignore_spaces:NTF * {#1} {#2} }
315 __scontents_compat_redefine:Npn \RequirePackage
316 { __scontents_optarg:nn { __scontents_require_auxi:wn } { } }
317 \cs_new_protected:Npn __scontents_require_auxi:wn [#1] #2
318 { __scontents_optarg:nn { __scontents_require_auxii:wnw [{#1}]{#2} } { } }
319 \cs_new:Npn __scontents_zap_space:ww #1~#2
320 {
321   #1 \if_meaning:w #2 \q_mark
322   \exp_after:wN \use_none:n
323   \else:
324     \exp_after:wN __scontents_zap_space:ww
325   \fi: #2
326 }
327 \cs_new_protected:Npn __scontents_require_auxii:wnw [#1] #2 [#3]
328 {
329   \tl_set:Nx \l__scontents_temp_tl { __scontents_zap_space:ww #2 ~ \q_mark }
330   \clist_map_function:NN \l__scontents_temp_tl __scontents_require_auxiii:n
331 }
332 \cs_new_protected:Npn __scontents_require_auxiii:n #1
333 {
334   \str_if_eq:eeF {expl3} {#1}
335   { \msg_error:nnn { scontents } { invalid-package } {#1} }
336 }
337 \msg_new:nnn { scontents } { invalid-package }
338 { Package~'#1'~invalid~in~scontents.~This~is~an~error~in~scontents. }
339 __scontents_compat_redefine:cpn { @ifpackagelater } #1
340 { \exp_args:Nc __scontents_package_later_aux:Nn { ver@#1.sty } }
341 \cs_new_protected:Npn __scontents_package_later_aux:Nn #1 #2
342 {
343   \int_compare:nNnTF
344     { \exp_after:wN __scontents_parse_version:w #1 //00 \q_mark } <
345     { \exp_after:wN __scontents_parse_version:w #2 //00 \q_mark }
346 }
347 \cs_new:Npn __scontents_parse_version:w #1 { __scontents_parse_version_auxi:w 0#1 }
348 \cs_new:Npn __scontents_parse_version_auxi:w #1/#2/#3#4#5 \q_mark
349 { __scontents_parse_version_auxii:w #1-#2-#3#4 \q_mark }
350 \cs_new:Npn __scontents_parse_version_auxii:w #1-#2-#3#4#5 \q_mark
351 { \tl_if_blank:nF {#2} {#1} #2 #3 #4 }
352 __scontents_compat_redefine:Npn \ProvidesExplPackage #1 #2 #3 #4
353 { __scontents_provides_aux:nn {#1} { #2 \tl_if_empty:nF {#3} {#3~} #4 } }
354 \cs_new_protected:Npn __scontents_provides_aux:nn #1 #2
355 {
356   \tl_gset:cx { ver@#1.sty } {#2}
357   \iow_log:n { Package~'#1:~#2 }
358   \ExplSyntaxOn
359 }
360 __scontents_compat_redefine:Npn \DeclareOption
361 { __scontents_stararg:nn { \use_none:n } { \use_none:nn } }
362 __scontents_compat_redefine:Npn \ProcessOptions
363 { __scontents_stararg:nn { } { } }

```

Now that the compatibility layer is defined, we can finally load xparse. xparse expects to be loaded with `\ExplSyntaxOff` (not much harm would be done otherwise, but just to be on the safe side).

Within xparse a `\RequirePackage{expl3}` is done. We can ignore that since we have already loaded `expl3`. Next, a `\@ifpackagelater` test is done: we do that test too to ensure that xparse is compatible with the current running version of `expl3`. The following `\ProvidesExplPackage` simply defines `\ver@xparse.sty` for any other package that might use it, and then does `\ExplSyntaxOn`. At the end of the package, xparse parses (heh) the package options. Since we don't have those in non- \TeX formats, they are ignored. Okay, so load xparse:

```

364 \int_set:Nn \l__scontents_tmpa_int { \char_value_catcode:n { \@ } }
365 \char_set_catcode_letter:N \@
366 \exp_after:wN
367 \ExplSyntaxOff
368 \file_input:n { xparse.sty }

```



```

369 \ExplSyntaxOn
370 \char_set_catcode:nn { \@ } { \__scontents_tmpa_int }
371 \__scontents_compat_restore:
372 </loader&!latex>
373 <*core>

```

(actually we don't need to do `\ExplSyntaxOn` there because we don't have \TeX 's full package loading mechanism, so the `expl3` syntax remains active after `xparse` is loaded, but it doesn't harm either).

10.7 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` `__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.

```

374 \cs_new_protected:Npn \__scontents_append_contents:nn #1#2
375 {
376   \tl_if_blank:nT {#1}
377   { \msg_error:nn { scontents } { empty-store-cmd } }
378   \seq_if_exist:cF { g__scontents_name_#1_seq }
379   { \seq_new:c { g__scontents_name_#1_seq } }
380   \seq_gput_right:cn { g__scontents_name_#1_seq } {#2}
381 }
382 \cs_generate_variant:Nn \__scontents_append_contents:nn { Vx }

```

(End definition for `__scontents_append_contents:nn`.)

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

`__scontents_getfrom_seq:nnn`

```

383 \cs_new:Npn \__scontents_getfrom_seq:nn #1#2
384 {
385   \seq_if_exist:cTF { g__scontents_name_#2_seq }
386   {
387     \exp_args:Nf \__scontents_getfrom_seq:nnn
388     { \seq_count:c { g__scontents_name_#2_seq } }
389     {#1} {#2}
390   }
391   { \msg_expandable_error:nnn { scontents } { undefined-storage } {#2} }
392 }
393 \cs_new:Npn \__scontents_getfrom_seq:nnn #1#2#3
394 {
395   \bool_lazy_or:nnTF
396   { \int_compare_p:nNn {#2} = { 0 } }
397   { \int_compare_p:nNn { \int_abs:n {#2} } > {#1} }
398   { \msg_expandable_error:nnnn { scontents } { index-out-of-range } {#2} {#3} {#1} }
399   { \seq_item:cn { g__scontents_name_#3_seq } {#2} }
400 }

```

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

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

```

401 \cs_new_protected:Npn \__scontents_lastfrom_seq:n #1
402 {
403   \tl_gset:Nx \g__scontents_temp_tl { \seq_item:cn { g__scontents_name_#1_seq } {-1} }
404   \group_insert_after:N \__scontents_rescan_tokens:V
405   \group_insert_after:N \g__scontents_temp_tl
406   \group_insert_after:N \tl_gclear:N
407   \group_insert_after:N \g__scontents_temp_tl
408 }

```

(End definition for `__scontents_lastfrom_seq:n`.)

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

```

409 \cs_new_protected:Npn \__scontents_store_to_seq:NN #1#2
410 {
411   \tl_log:N #1
412   \__scontents_append_contents:Vx #2 { \exp_not:V #1 }
413 }

```

(End definition for `__scontents_store_to_seq:NN`.)

10.8 Construction of environment scontents

In order to be able to define environments that behave similarly to `scontents`, we define a generic environment and make all other environment as wrappers around that one.

10.8.1 The command `\newenvsc`

`\newenvsc` The `\newenvsc` command defines two functions `__scontents_#1_env_begin:` and `__scontents_#1_env_end:`, which set the current environment's default properties and then call the generic `__scontents_env_generic_begin:` and `__scontents_env_generic_end:`.

```

414 \tl_new:N \l__scontents_env_name_tl
415 \cs_new_protected:Npn \__scontents_scontents_setenv:nn #1 #2
416 {
417   \cs_new_protected:cpn { __scontents_#1_env_begin: }
418   {
419     \tl_set:Nn \l__scontents_env_name_tl {#1}
420     \keys_set:nn { scontents } {#2}
421     \__scontents_setup_verb_processor:
422     \__scontents_env_generic_begin:
423   }
424   \cs_new_protected:cpn { __scontents_#1_env_end: }
425   { \__scontents_env_generic_end: }
426   \exp_args:Nooo % http://nooooooooooooooooo.com :) jeje
427   \__scontents_env_define:nnn { \tl_to_str:n {#1} }
428   { \cs:w __scontents_#1_env_begin: \cs_end: }
429   { \cs:w __scontents_#1_env_end: \cs_end: }
430 }
431 </core>
432 <*loader>
433 \NewDocumentCommand { \newenvsc } { m O{} }
434 {
435   <latex | plain> \cs_if_exist:cTF { #1 }
436   <context> \cs_if_exist:cTF { start #1 }
437   { \msg_error:nnn { scontents } { env-already-defined } {#1} }
438   { \__scontents_scontents_setenv:nn {#1} {#2} }
439 }
440 \cs_new_protected:Npn \__scontents_env_define:nnn #1 #2 #3
441 {
442   <latex | plain> \NewDocumentEnvironment {#1} { }
443   <context> \cs_new_protected:cpn { start #1 }
444   {
445     <!!latex> \group_begin:
446     #2
447   }
448   <context> \cs_new_protected:cpn { stop #1 }
449   {
450     #3
451     <!!latex> \group_end:
452   }
453 }
454 </loader>
455 <*core>

```

(End definition for `\newenvsc`, `\l__scontents_env_name_tl`, and `__scontents_scontents_setenv:nn`. This function is documented on page 4.)

10.8.2 Generic definition of the environment

`__scontents_env_generic_begin:` Now we define the generic environment `__scontents_env_generic_begin:`
`__scontents_env_generic_begin:`

```

456 \cs_new_protected:Npn \__scontents_env_generic_begin:
457 {
458   \char_set_catcode_active:N ^^M
459   \__scontents_start_environment:w
460 }
461 \cs_new_protected:Npn \__scontents_env_generic_end:
462 {
463   \__scontents_stop_environment:
464   \__scontents_atend_environment:
465 }

```

(End definition for `__scontents_env_generic_begin:`.)

10.8.3 Definition of the environment scontents

Now defining the `scontents` environment should be easy:

```
scontents
\contents
\endscontents
\startcontents
\stopscontents
```

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

10.8.4 key val for environment

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

The macro `__scontents_grab_optional:w` is called from the `scontents` environment with the tokens following the `\begin{scontents}` when the next character is a `[`. This function is defined using `xparse` to exploit its delimited argument processor.

The function is called from a context where `^^M` is active, so `__scontents_normalise_line_ends:N` is used to replace active `^^M` characters by spaces.

```
469 </core>
470 <*loader>
471 \NewDocumentCommand \__scontents_grab_optional:w { r[] }
472 { \__scontents_grab_optional:n {#1} }
473 </loader>
474 <*core>
475 \cs_new_protected:Npn \__scontents_grab_optional:n #1
476 {
477   \tl_if_novalue:nF {#1}
478   {
479     \tl_set:Nn \__scontents_temp_tl {#1}
480     \__scontents_normalise_line_ends:N \__scontents_temp_tl
481     \keys_set:nV { scontents / scontents } \__scontents_temp_tl
482   }
483   \__scontents_start_after_option:w
484 }
```

(End definition for `__scontents_grab_optional:n` and `__scontents_grab_optional:w`.)

10.8.5 The environment itself

Here we make `^^I`, `^^L` and `^^M` active characters so that the end of line can be “seen” to be used as a delimiter, and `TEX` doesn’t try to eliminate space-like characters.

First we check if the immediate next token after `\begin{scontents}` is a `[`. If it is, then `__scontents_grab_optional:w` is called to do the heavy lifting. `__scontents_grab_optional:w` processes the optional argument and calls `__scontents_start_after_option:w`.

`__scontents_start_after_option:w` also checks for trailing tokens after the optional argument and issues an error if any.

In all cases, `__scontents_check_line_process:xn` checks that everything past `\begin{scontents}` is empty and then process the environment. `__scontents_check_line_process:xn` 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. This part of the code is inspired and adapted from the code of the package `xsimverb` by Clemens Niederberger.

```
485 \group_begin:
486   \char_set_catcode_active:N \^^I
487   \char_set_catcode_active:N \^^L
488   \char_set_catcode_active:N \^^M
489   \cs_new_protected:Npn \__scontents_normalise_line_ends:N #1
490   { \tl_replace_all:Nnn #1 { ^^M } { ~ } }
491   \cs_new_protected:Npn \__scontents_start_environment:w #1 ^^M
492   {
493     \tl_if_head_is_N_type:nTF {#1}
494     {
495       \str_if_eq:eeTF { \tl_head:n {#1} } { [ }
496       { \__scontents_grab_optional:w #1 ^^M }
497       { \__scontents_check_line_process:xn { } {#1} }
498     }
499   }
```

```

499     { \__scontents_check_line_process:xn { } {#1} }
500   }
501   \cs_new_protected:Npn \__scontents_start_after_option:w #1 ^^M
502   { \__scontents_check_line_process:xn { [...] } {#1} }
503   \cs_new_protected:Npn \__scontents_check_line_process:xn #1 #2
504   {
505     \tl_if_blank:nF {#2}
506     {
507       \msg_error:nnxn { scontents } { junk-after-begin }
508       { after~\c_backslash_str begin { \__scontents_env_name_tl } #1 } {#2}
509     }
510     \__scontents_make_control_chars_active:
511     \__scontents_file_tl_write_start:V \__scontents_fname_out_tl
512   }
513   \cs_new_protected:Npn \__scontents_stop_environment:
514   {
515     \__scontents_file_write_stop:N \__scontents_macro_tmp_tl
516     \bool_lazy_and:nnT
517     { \__scontents_storing_bool }
518     { \tl_if_empty_p:N \__scontents_macro_tmp_tl }
519     {
520       \msg_warning:nnx { scontents } { empty-environment }
521       { \__scontents_env_name_tl }
522     }
523   }

```

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

```

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

```

This is the main macro to collect the contents of a verbatim environment. The macro starts a group, opens the *⟨output file⟩*, if necessary, sets verbatim catcodes, and then issues `^^M` (set equal to `__scontents_ret:w`) to read the environment line by line until reaching its end. The output token list will be appended with an active `^^J` character and the line just read, and this line is written to the output file, if any. At the end of the environment the *⟨output file⟩* is closed (if it was open), and the output token list is smuggled out of the verbatim group. A leading `^^J` is removed from the token list using `__scontents_remove_leading_nl:n` (which expects an active `^^J` token at the head of the token list; a low level \TeX error is raised otherwise).

```

524   \cs_new_protected:Npn \__scontents_file_tl_write_start:n #1
525   {
526     \group_begin:
527     \bool_if:nT { \__scontents_writing_bool && \__scontents_overwrite_bool }
528     {
529       \file_if_exist:nTF {#1}
530       { \msg_warning:nnx { scontents } { overwrite-file } {#1} }
531       { \msg_warning:nnx { scontents } { writing-file } {#1} }
532       \iow_open:Nn \__scontents_file_iow {#1}
533     }
534     \bool_if:nT { \__scontents_writing_bool && !\__scontents_overwrite_bool }
535     {
536       \file_if_exist:nF {#1}
537       { \msg_warning:nnx { scontents } { writing-file } {#1} }
538       \iow_open:Nn \__scontents_file_iow {#1}
539     }
540     \tl_clear:N \__scontents_file_tl
541     \seq_map_function:NN \l_char_special_seq \char_set_catcode_other:N
542     \int_step_function:nnnN { 128 } { 1 } { 255 } \char_set_catcode_letter:n
543     \cs_set_protected:Npx \__scontents_ret:w ##1 ^^M
544     {
545       \exp_not:N \__scontents_verb_processor_iterate:w
546       ##1 \c__scontents_end_env_tl
547       \c__scontents_end_env_tl
548       \exp_not:N \q__scontents_stop
549     }
550     \__scontents_make_control_chars_active:
551     \__scontents_ret:w
552   }
553   \cs_new:Npn \__scontents_setup_verb_processor:
554   {
555     \use:x
556     {
557       \cs_set:Npn \exp_not:N \__scontents_verb_processor_iterate:w

```

```

558         ###1 \c__scontents_end_env_tl
559         ###2 \c__scontents_end_env_tl
560         ###3 \exp_not:N \q__scontents_stop
561     } { \__scontents_verb_processor_iterate:nnn {##1} {##2} {##3} }
562 }
563 \cs_new:Npn \__scontents_verb_processor_iterate:nnn #1 #2 #3
564 {
565     \tl_if_blank:nTF {#3}
566     {
567         \__scontents_analyse_nesting:n {#1}
568         \__scontents_verb_processor_output:n {#1}
569     }
570     {
571         \__scontents_if_nested:TF
572         {
573             \__scontents_nesting_decr:
574             \__scontents_verb_processor_output:x
575             { \exp_not:n {#1} \c__scontents_end_env_tl \exp_not:n {#2} }
576         }
577         {
578             \tl_if_blank:nF {#1}
579             { \__scontents_verb_processor_output:n {#1} }
580             \cs_set_protected:Npx \__scontents_ret:w
581             {
582                 \__scontents_env_end_function:
583                 \bool_lazy_or:nnF
584                 { \tl_if_blank_p:n {#2} }
585                 { \str_if_eq_p:ee {#2} { \c_percent_str } }
586                 {
587                     \str_if_eq:VnF \c__scontents_hidden_space_str {#2}
588                     {
589                         \msg_warning:nnnn { scontents } { rescanning-text }
590                         {#2} { \tl_use:N \l__scontents_env_name_tl }
591                     }
592                     \__scontents_rescan_tokens:n {#2}
593                 }
594             }
595             \char_set_active_eq:NN ^^M \__scontents_ret:w
596         }
597     }
598     ^^M
599 }
600 \cs_new:Npn \__scontents_env_end_function:
601 {
602     \__scontents_format_case:nnn
603     { \exp_not:N \end { \if_false: } \fi: }
604     { \exp_after:wN \exp_not:N \cs:w end }
605     { \exp_after:wN \exp_not:N \cs:w stop }
606     \tl_use:N \l__scontents_env_name_tl
607     \__scontents_format_case:nnn
608     { \if_false: { \fi: } }
609     { \cs_end: }
610     { \cs_end: }
611 }
612 \cs_new_protected:Npn \__scontents_file_write_stop:N #1
613 {
614     \bool_if:NT \l__scontents_writing_bool
615     { \iow_close:N \l__scontents_file_iow }
616     \use:x
617     {
618         \group_end:
619         \bool_if:NT \l__scontents_storing_bool
620         {
621             \tl_set:Nn \exp_not:N #1
622             { \exp_args:NV \__scontents_remove_leading_nl:n \l__scontents_file_tl }
623         }
624     }
625 }
626 \cs_new:Npn \__scontents_remove_leading_nl:n #1
627 {
628     \tl_if_head_is_N_type:nTF {#1}

```

```

629     {
630         \exp_args:Nf
631         \__scontents_remove_leading_nl:nn
632         { \tl_head:n {#1} } {#1}
633     }
634     { \exp_not:n {#1} }
635 }
636 \cs_new:Npn \__scontents_remove_leading_nl:nn #1 #2
637 {
638     \token_if_eq_meaning:NNTF ^^J #1
639     { \exp_not:o { \__scontents_remove_leading_nl:w #2 } }
640     { \exp_not:n {#2} }
641 }
642 \cs_new:Npn \__scontents_remove_leading_nl:w ^^J { }

```

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

`__scontents_verb_processor_output:n` `__scontents_verb_processor_output:n` does the output of each line read, to a token list and to a file, depending on the booleans `__scontents_writing_bool` and `__scontents_storing_bool`.

```

643 \cs_new_protected:Npn \__scontents_verb_processor_output:n #1
644 {
645     \bool_if:NT \__scontents_writing_bool
646     { \iow_now:Nn \__scontents_file_iow {#1} }
647     \bool_if:NT \__scontents_storing_bool
648     { \tl_put_right:Nn \__scontents_file_tl { ^^J #1 } }
649 }
650 \group_end:
651 \cs_generate_variant:Nn \__scontents_verb_processor_output:n { x }
652 \cs_generate_variant:Nn \__scontents_file_tl_write_start:n { V }

```

(End definition for `__scontents_verb_processor_output:n`.)

`__scontents_analyse_nesting:n` `__scontents_analyse_nesting:n` scans nested `\begin{scontents}` and steps a `__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...

```

653 \cs_new_protected:Npn \__scontents_analyse_nesting:n #1
654 {
655     \int_zero:N \__scontents_tmpa_int
656     \__scontents_analyse_nesting_format:n {#1}
657     \int_compare:nNtT { \__scontents_tmpa_int } > { 1 }
658     { \msg_warning:nn { scontents } { multiple-begin } }
659 }
660 \cs_new_protected:Npn \__scontents_nesting_incr:
661 {
662     \int_incr:N \__scontents_env_nesting_int
663     \int_incr:N \__scontents_tmpa_int
664 }
665 \cs_new_protected:Npn \__scontents_nesting_decr:
666 { \int_decr:N \__scontents_env_nesting_int }
667 \prg_new_protected_conditional:Npnn \__scontents_if_nested: { TF }
668 {
669     \int_compare:nNtTF { \__scontents_env_nesting_int } > { \c_zero_int }
670     { \prg_return_true: }
671     { \prg_return_false: }
672 }
673 \cs_new:Npn \__scontents_use_none_delimit_by_q_stop:w #1 \__scontents_stop { }

```

In \LaTeX , environments start with `\begin{«env»}`, so checking if a string contains `\begin{scontents}` is straightforward. Since no `}` can appear inside `«env»`, then just a macro delimited by `}` is enough.

```

674 \use:x
675 {
676     \cs_new_protected:Npn \exp_not:N \__scontents_analyse_nesting_latex:w ##1
677     \c_backslash_str begin \c_left_brace_str ##2 \c_right_brace_str
678 } {
679     \__scontents_tl_if_head_is_q_mark:nTF {#2}
680     { \__scontents_use_none_delimit_by_q_stop:w }
681     {
682         \str_if_eq:VnT \__scontents_env_name_tl {#2}

```

```

683         { \_scontents_nesting_incr: }
684         \_scontents_analyse_nesting_latex:w
685     }
686 }
687 \cs_new_protected:Npx \_scontents_analyse_nesting_latex:n #1
688 {
689     \_scontents_analyse_nesting_latex:w #1
690     \c_backslash_str begin
691     \c_left_brace_str \exp_not:N \q_scontents_mark \c_right_brace_str
692     \exp_not:N \q_scontents_stop
693 }

```

In other formats, however, we don't have an “end anchor” to delimit the environment name, so a delimited macro won't help. We have to search for the entire environment command (usually `\scontents` and `\startscontents`).

```

694 \cs_new_protected:Npn \_scontents_analyse_nesting_generic_process:nn #1 #2
695 {
696     \tl_if_head_is_N_type:nTF {#2}
697     {
698         \_scontents_tl_if_head_is_q_mark:nF {#2}
699         {
700             \_scontents_nesting_incr:
701             \_scontents_analyse_nesting_generic:w #2 \q_scontents_stop
702         }
703     }
704     { \_scontents_analyse_nesting_generic:w #2 \q_scontents_stop }
705 }
706 \cs_new_protected:Npn \_scontents_analyse_nesting_generic:nn #1 #2
707 {
708     \_scontents_define_generic_nesting_function:n {#1}
709     \use:x
710     {
711         \exp_not:N \_scontents_analyse_nesting_generic:w #2
712         \c_backslash_str #1 \tl_use:N \l_scontents_env_name_tl
713         \exp_not:N \q_scontents_mark \exp_not:N \q_scontents_stop
714     }
715 }
716 \cs_new_protected:Npn \_scontents_define_generic_nesting_function:n #1
717 {
718     \use:x
719     {
720         \cs_set_protected:Npn \exp_not:N \_scontents_analyse_nesting_generic:w ###1
721         \c_backslash_str #1 \tl_use:N \l_scontents_env_name_tl
722         ###2 \exp_not:N \q_scontents_stop
723     } { \_scontents_analyse_nesting_generic_process:nn {##1} {##2} }
724 }
725 </core>
726 <*loader>
727 <latex>\cs_new_eq:NN \_scontents_analyse_nesting_format:n
728 <latex> \_scontents_analyse_nesting_latex:n
729 <!latex>\cs_new_protected:Npn \_scontents_analyse_nesting_format:n
730 <plain> { \_scontents_analyse_nesting_generic:nn { } }
731 <context> { \_scontents_analyse_nesting_generic:nn { start } }
732 </loader>
733 <*core>

```

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

10.8.6 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.

```

734 \cs_new_protected:Npn \_scontents_atend_environment:
735 {
736     \bool_if:NT \l_scontents_storing_bool
737     {
738         \bool_if:NF \l_scontents_forced_eol_bool
739         {
740             \tl_put_right:Nx \l_scontents_macro_tmp_tl
741             { \c__scontents_hidden_space_str }
742         }
743     }

```



```

743     \__scontents_store_to_seq:NN \l__scontents_macro_tmp_tl
744     \l__scontents_name_seq_env_tl
745     \bool_if:NT \l__scontents_print_env_bool
746     { \__scontents_lastfrom_seq:n \l__scontents_name_seq_env_tl }
747   }
748 }
749 </core>

```

(End definition for `__scontents_atend_environment:`)

```

\verbatimsc In Plain we emulate LATEX's verbatim environment.
\endverbatimsc
750 <*plain>
751 \bool_new:N \l__scontents_temp_bool
752 \cs_new_protected:Npn \verbatimsc
753 {
754   \group_begin:
755     \__scontents_verbatimsc_aux: \frenchspacing \__scontents_vobeyspaces:
756     \__scontents_xverb:
757   }
758 \cs_new_protected:Npn \endverbatimsc
759 { \group_end: }
760 \cs_new_protected:Npn \__scontents_verbatimsc_aux:
761 {
762   \skip_vertical:N \parskip
763   \int_set:Nn \parindent { 0pt }
764   \skip_set:Nn \parfillskip { 0pt plus 1fil }
765   \int_set:Nn \parskip { 0pt plus 0pt minus 0pt }
766   \tex_par:D
767   \bool_set_false:N \l__scontents_temp_bool
768   \cs_set:Npn \par
769   {
770     \bool_if:NTF \l__scontents_temp_bool
771     {
772       \mode_leave_vertical:
773       \null
774       \tex_par:D
775       \penalty \interlinepenalty
776     }
777     {
778       \bool_set_true:N \l__scontents_temp_bool
779       \mode_if_horizontal:T
780       { \tex_par:D \penalty \interlinepenalty }
781     }
782   }
783   \cs_set_eq:NN \do \char_set_catcode_other:N
784   \dospecials \obeylines
785   \tl_use:N \l__scontents_verb_font_tl
786   \cs_set_eq:NN \do \__scontents_do_noligs:N
787   \__scontents_nolig_list:
788   \tex_everypar:D \exp_after:wN
789   { \tex_the:D \tex_everypar:D \tex_unpenalty:D }
790 }
791 \cs_new_protected:Npn \__scontents_nolig_list:
792 { \do\` \do\< \do\> \do\, \do\' \do\~ }
793 \cs_new_protected:Npn \__scontents_vobeyspaces:
794 { \__scontents_set_active_eq:NN \ \__scontents_xobeysp: }
795 \cs_new_protected:Npn \__scontents_xobeysp:
796 { \mode_leave_vertical: \nobreak \ }
797 </plain>

```

(End definition for `\verbatimsc` and `\endverbatimsc`.)

`\dospecials` xparse also requires L^AT_EX's `\dospecials`. In case it doesn't exist (at the time `scontents` is loaded) we define `\dospecials` to use the `\l_char_special_seq`.

```

798 <*!latex>
799 \cs_if_exist:NF \dospecials
800 {
801   \cs_new:Npn \dospecials
802   { \seq_map_function:NN \l_char_special_seq \do }

```

```

803 }
804 </!latex>

```

(End definition for `\dospecials`.)

10.9 The command `\Scontents`

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

```

\Scontents
__scontents_norm_arg:n
__scontents_Scontents_auxi:N
__scontents_Scontents_internal:nn
__scontents_verb_arg:w
__scontents_verb_arg_internal:n

```

The `\Scontents` macro starts by parsing an optional argument and then delegates to `__scontents_verb_arg:w` or `__scontents_norm_arg:n` depending whether a star (*) argument is present.

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

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

```

805 <*loader>
806 \NewDocumentCommand { \Scontents } { !s !O{} }
807 { __scontents_Scontents_internal:nn {#1} {#2} }
808 </loader>
809 <*core>
810 \cs_new_protected:Npn __scontents_Scontents_internal:nn #1 #2
811 {
812   \group_begin:
813   \tl_if_novalue:nF {#2}
814     { \keys_set:nn { scontents / Scontents } {#2} }
815   \char_set_catcode_active:n { 9 }
816   \bool_if:NTF #1
817     { __scontents_verb_arg:w }
818     { __scontents_norm_arg:n }
819 }
820 \cs_new_protected:Npn __scontents_norm_arg:n #1
821 {
822   \tl_set:Nx \l__scontents_temp_tl { \exp_not:n {#1} }
823   \tl_put_right:Nx \l__scontents_temp_tl { \c__scontents_hidden_space_str }
824   __scontents_store_to_seq:NN \l__scontents_temp_tl \l__scontents_name_seq_cmd_tl
825   \bool_if:NT \l__scontents_print_cmd_bool
826     { __scontents_lastfrom_seq:n \l__scontents_name_seq_cmd_tl }
827   \group_end:
828 }
829 </core>
830 <*loader>
831 \NewDocumentCommand { __scontents_verb_arg:w } { +v }
832 { __scontents_verb_arg_internal:n {#1} }
833 </loader>
834 <*core>
835 \cs_new_protected:Npn __scontents_verb_arg_internal:n #1
836 {
837   \tl_set:Nx \l__scontents_temp_tl { \exp_not:n {#1} }
838   \tl_replace_all:Nxx \l__scontents_temp_tl { \iow_char:N ^^M } { \iow_char:N ^^J }
839   \bool_if:NF \l__scontents_forced_eol_bool
840     { \tl_put_right:Nx \l__scontents_temp_tl { \c__scontents_hidden_space_str } }
841   __scontents_store_to_seq:NN \l__scontents_temp_tl \l__scontents_name_seq_cmd_tl
842   \bool_if:NT \l__scontents_print_cmd_bool
843     { __scontents_lastfrom_seq:n \l__scontents_name_seq_cmd_tl }
844   \group_end:
845 }

```

(End definition for `\Scontents` and others. This function is documented on page 4.)

10.10 The command `\getstored`

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

```

846 </core>
847 <*loader>
848 \NewDocumentCommand { \getstored } { O{-1} m }
849 { __scontents_getstored_internal:nn {#1} {#2} }
850 </loader>
851 <*core>
852 \cs_new_protected:Npn __scontents_getstored_internal:nn #1 #2
853 {

```

```

854 \group_begin:
855   \int_set:Nn \tex_newlinechar:D { ``^^J }
856   \__scontents_rescan_tokens:x
857   {
858 \endgroup % This assumes \catcode``=0... Things might go off otherwise.
859   \__scontents_getfrom_seq:nn {#1} {#2}
860   }
861 }

```

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

10.11 The command `\foreachsc`

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

```

862 </core>
863 <*loader>
864 \NewDocumentCommand { \foreachsc } { o m }
865 { \__scontents_foreachsc_internal:nn {#1} {#2} }
866 </loader>
867 <*core>
868 \cs_new_protected:Npn \__scontents_foreachsc_internal:nn #1 #2
869 {
870   \group_begin:
871   \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / foreachsc } {#1} }
872   \tl_set:Nn \__scontents_foreach_name_seq_tl {#2}
873   \seq_clear:N \__scontents_foreach_print_seq
874   \bool_if:NF \__scontents_foreach_stop_bool
875   {
876     \int_set:Nn \__scontents_foreach_stop_int
877     { \seq_count:c { g__scontents_name_#2_seq } }
878   }
879   \int_step_function:nnnN
880   { \__scontents_foreach_start_int }
881   { \__scontents_foreach_step_int }
882   { \__scontents_foreach_stop_int }
883   \__scontents_foreach_add_body:n
884   \tl_gset:Nx \g__scontents_temp_tl
885   {
886     \seq_use:Nn \__scontents_foreach_print_seq
887     { \tl_use:N \__scontents_foreach_sep_tl }
888   }
889   \group_end:
890   \exp_after:wN \tl_gclear:N
891   \exp_after:wN \g__scontents_temp_tl
892   \g__scontents_temp_tl
893 }
894 \cs_new_protected:Npn \__scontents_foreach_add_body:n #1
895 {
896   \seq_put_right:Nx \__scontents_foreach_print_seq
897   {
898     \bool_if:NT \__scontents_foreach_before_bool
899     { \exp_not:V \__scontents_foreach_before_tl }
900     \bool_if:NTF \__scontents_foreach_wrapper_bool
901     { \__scontents_foreach_wrapper:n }
902     { \use:n }
903     { \getstored [#1] { \tl_use:N \__scontents_foreach_name_seq_tl } }
904     \bool_if:NT \__scontents_foreach_after_bool
905     { \exp_not:V \__scontents_foreach_after_tl }
906   }
907 }

```

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

10.12 The command `\typestored`

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

```

\__scontents_verb_print:N
\__scontents_xverb:w
\verbatimsc
908 </core>
909 <*loader>
910 \NewDocumentCommand { \typestored } { o m }

```

```

911 { \_scontents_tpestored_internal:nn {#1} {#2} }
912 </loader>
913 <*core>
914 \cs_new_protected:Npn \_scontents_tpestored_internal:nn #1 #2
915 {
916   \group_begin:
917     \int_set:Nn \l__scontents_seq_item_int { 1 }
918     \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / typemeaning } {#1} }
919     \tl_set:Nx \l__scontents_temp_tl
920       { \exp_args:NV \_scontents_getfrom_seq:nn \l__scontents_seq_item_int {#2} }
921     \tl_remove_once:NV \l__scontents_temp_tl \c__scontents_hidden_space_str
922     \tl_log:N \l__scontents_temp_tl
923     \tl_if_empty:NF \l__scontents_temp_tl
924       { \_scontents_verb_print:N \l__scontents_temp_tl }
925   \group_end:
926 }

```

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`.

```

927 \group_begin:
928   \char_set_catcode_active:N ^^M
929   \cs_new_protected:Npn \_scontents_verb_print:N #1
930   {
931     \tl_if_blank:VT #1
932     { \msg_error:nnn { scontents } { empty-variable } {#1} }
933     \cs_set_eq:NN \_scontents_verb_print_EOL: ^^M
934     \cs_set_eq:NN ^^M \scan_stop:
935     \cs_set_eq:cN { do@noligs } \_scontents_do_noligs:N
936     \int_set:Nn \tex_newlinechar:D { `^^J }
937     \_scontents_rescan_tokens:x
938     {
939       \_scontents_format_case:nnn
940       { \exp_not:N \begin{verbatimsc} } % LaTeX
941       { \verbatimsc } % Plain/Generic
942       { \startverbatimsc } % ConTeXt
943       ^^M
944       \exp_not:V #1 ^^M
945       \g__scontents_end_verbatimsc_tl
946     }
947     \cs_set_eq:NN ^^M \_scontents_verb_print_EOL:
948   }
949 \group_end:

```

Finally, the `verbatimsc` environment is defined.

```

950 \cs_new_protected:Npn \_scontents_xverb:
951 {
952   \char_set_catcode_active:n { 9 }
953   \char_set_active_eq:nN { 9 } \_scontents_tabs_to_spaces:
954   \_scontents_xverb:w
955 }
956 </core>
957 <*loader>
958 <*!context>
959 \use:x
960 {
961   \cs_new_protected:Npn \exp_not:N \_scontents_xverb:w
962     ##1 \g__scontents_end_verbatimsc_tl
963   <latex> { ##1 \exp_not:N \end{verbatimsc} }
964   <plain> { ##1 \exp_not:N \endverbatimsc }
965   <context> { ##1 \exp_not:N \stopverbatimsc }
966 }
967 </!context>
968 <*latex>
969 \NewDocumentEnvironment { verbatimsc } { }
970 {
971   \cs_set_eq:cN { @xverbatim } \_scontents_xverb:
972   \verbatim
973 }
974 { }

```

```

975 </latex>
976 <context>\definetyping[verbatimsc]
977 </loader>
978 <*core>

```

(End definition for \typestored and others. These functions are documented on page 5.)

10.13 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.

```

979 \cs_new:Npn \__scontents_tabs_to_spaces:
980   { \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.

```

981 \cs_new:Npn \__scontents_do_noligs:N #1
982   {
983     \char_set_catcode_active:N #1
984     \char_set_active_eq:Nc #1 { __scontents_active_char_ \token_to_str:N #1 : }
985     \cs_set:cpx { __scontents_active_char_ \token_to_str:N #1 : }
986       {
987         \mode_leave_vertical:
988         \tex_kern:D \c_zero_dim
989         \char_generate:nn { `#1 } { 12 }
990       }
991   }

```

(End definition for `__scontents_do_noligs:N`.)

`__scontents_tl_if_head_is_q_mark:nTF` Tests if the head of the token list is `\q__scontents_mark`.

```

992 \prg_new_protected_conditional:Npnn \__scontents_tl_if_head_is_q_mark:n #1
993   { T, F, TF }
994   {
995     \if_meaning:w \q__scontents_mark #1 \scan_stop:
996     \prg_return_true:
997   \else:
998     \prg_return_false:
999   \fi:
1000 }

```

(End definition for `__scontents_tl_if_head_is_q_mark:nTF`.)

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

```

1001 \cs_new_protected:Npn \__scontents_set_active_eq:NN #1
1002   {
1003     \char_set_catcode_active:N #1
1004     \char_set_active_eq:NN #1
1005   }
1006 </core>
1007 <*loader>
1008 \group_begin:
1009 <plain> \char_set_catcode_active:n { \* }
1010 \cs_new_protected:Npn \__scontents_plain_disable_outer_par:
1011 <*plain>
1012   {
1013     \group_begin:
1014     \char_set_lccode:nn { \* } { \^^L }
1015     \tex_lowercase:D { \group_end:
1016       \tex_let:D * \scan_stop:
1017     }
1018   }
1019 </plain>

```

```

1020 ⟨latex | context⟩    { }
1021 \group_end:
1022 ⟨/loader⟩
1023 ⟨*core⟩
1024 \group_begin:
1025   \char_set_catcode_active:N \*
1026   \cs_new_protected:Npn \__scontents_make_control_chars_active:
1027   {
1028     \__scontents_plain_disable_outer_par:
1029     \__scontents_set_active_eq:NN \^^I \__scontents_tab:
1030     \__scontents_set_active_eq:NN \^^L \__scontents_par:
1031     \__scontents_set_active_eq:NN \^^M \__scontents_ret:w
1032   }
1033 \group_end:

```

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

10.14 The command `\setupsc`

User command `\setupsc` to setup module.

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

```

1034 ⟨/core⟩
1035 ⟨*loader⟩
1036 \NewDocumentCommand { \setupsc } { +m }
1037 { \keys_set:nn { scontents } {#1} }
1038 ⟨/loader⟩
1039 ⟨*core⟩

```

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

10.15 The command `\meaningsc`

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

```

1040 ⟨/core⟩
1041 ⟨*loader⟩
1042 \NewDocumentCommand { \meaningsc } { o m }
1043 { \__scontents_meaningsc_internal:nn {#1} {#2} }
1044 ⟨/loader⟩
1045 ⟨*core⟩
1046 \cs_new_protected:Npn \__scontents_meaningsc_internal:nn #1 #2
1047 {
1048   \group_begin:
1049     \int_set:Nn \__scontents_seq_item_int { 1 }
1050     \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / typemeaning } {#1} }
1051     \__scontents_meaningsc:n {#2}
1052   \group_end:
1053 }
1054 \group_begin:
1055   \char_set_catcode_active:N \^^I
1056   \cs_new_protected:Npn \__scontents_meaningsc:n #1
1057   {
1058     \tl_set:Nx \__scontents_temp_tl
1059     { \exp_args:NV \__scontents_getfrom_seq:nn \__scontents_seq_item_int {#1} }
1060     \tl_replace_all:Nxn \__scontents_temp_tl { \iow_char:N \^^J } { ~ }
1061     \tl_remove_once:NV \__scontents_temp_tl \c__scontents_hidden_space_str
1062     \tl_log:N \__scontents_temp_tl
1063     \tl_use:N \__scontents_verb_font_tl
1064     \tl_replace_all:Nxn \__scontents_temp_tl { ^^I } { \__scontents_tabs_to_spaces: }
1065     \cs_replacement_spec:N \__scontents_temp_tl
1066   }
1067 \group_end:

```

(End definition for `\meaningsc`. This function is documented on page 6.)

10.16 The command `\countsc`

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

```

1068 ⟨/core⟩

```

```

1069  ⟨*loader⟩
1070  \NewExpandableDocumentCommand { \countsc } { m }
1071    { \seq_count:c { g__scontents_name_#1_seq } }
1072  ⟨/loader⟩
1073  ⟨*core⟩

```

(End definition for \countsc. This function is documented on page 6.)

10.17 The command \cleanseqsc

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

```

1074  ⟨/core⟩
1075  ⟨*loader⟩
1076  \NewDocumentCommand { \cleanseqsc } { m }
1077    { \seq_clear_new:c { g__scontents_name_#1_seq } }
1078  ⟨/loader⟩
1079  ⟨*core⟩

```

(End definition for \cleanseqsc. This function is documented on page 6.)

10.18 Warning and error messages

Warning and error messages used throughout the package.

```

1080  \msg_new:nnn { scontents } { junk-after-begin }
1081    {
1082      Junk~characters~#1~\msg_line_context: :
1083      \\ \\
1084      #2
1085    }
1086  \msg_new:nnnn { scontents } { env-already-defined }
1087    { Environment~'#1'~already~defined! }
1088    {
1089      You~have~used~\newenvsc
1090      with~an~environment~that~already~has~a~definition. \\ \\
1091      The~existing~definition~of~'#1'~will~not~be~altered.
1092    }
1093  \msg_new:nnn { scontents } { empty-stored-content }
1094    { Empty~value~for~key~'getstored'~\msg_line_context:. }
1095  \msg_new:nnn { scontents } { empty-variable }
1096    { Variable~'#1'~empty~\msg_line_context:. }
1097  \msg_new:nnn { scontents } { overwrite-file }
1098    { Overwriting~file~'#1'. }
1099  \msg_new:nnn { scontents } { writing-file }
1100    { Writing~file~'#1'. }
1101  \msg_new:nnn { scontents } { rescanning-text }
1102    { Rescanning~text~'#1'~after~\c_backslash_str end{#2}~\msg_line_context:. }
1103  \msg_new:nnn { scontents } { multiple-begin }
1104    { Multiple~\c_backslash_str begin{ \l__scontents_env_name_tl }~\msg_line_context:. }
1105  \msg_new:nnn { scontents } { undefined-storage }
1106    { Storage~named~'#1'~is~not~defined. }
1107  \msg_new:nnn { scontents } { index-out-of-range }
1108    {
1109      \int_compare:nNnTF {#1} = { 0 }
1110      { Index~of~sequence~cannot~be~zero. }
1111      {
1112        Index~'#1'~out~of~range~for~'#2',~
1113        \int_compare:nNnTF {#1} > { 0 }
1114        { Max = } { Min = - } #3.
1115      }
1116    }
1117  \msg_new:nnnn { scontents } { env-key-unknown }
1118    {
1119      The~key~'#1'~is~unknown~by~environment~
1120      '\l__scontents_env_name_tl'~and~is~being~ignored.
1121    }
1122    {
1123      The~environment~'\l__scontents_env_name_tl'~does~not~have~a~key~called~'#1'.\\
1124      Check~that~you~have~spelled~the~key~name~correctly.
1125    }
1126  \msg_new:nnnn { scontents } { env-key-value-unknown }

```



```

1127 {
1128   The~key~'#1=#2'~is~unknown~by~environment~
1129   '\__scontents_env_name_tl'~and~is~being~ignored.
1130 }
1131 {
1132   The~environment~'\__scontents_env_name_tl'~does~not~have~a~key~called~'#1'.\\
1133   Check~that~you~have~spelled~the~key~name~correctly.
1134 }
1135 \msg_new:nnnn { scontents } { cmd-key-unknown }
1136 { The~key~'#1'~is~unknown~by~'\c_backslash_str Scontents'~and~is~being~ignored. }
1137 {
1138   The~command~'\c_backslash_str Scontents'~does~not~have~a~key~called~'#1'.\\
1139   Check~that~you~have~spelled~the~key~name~correctly.
1140 }
1141 \msg_new:nnnn { scontents } { cmd-key-value-unknown }
1142 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str Scontents'~and~is~being~ignored. }
1143 {
1144   The~command~'\c_backslash_str Scontents'~does~not~have~a~key~called~'#1'.\\
1145   Check~that~you~have~spelled~the~key~name~correctly.
1146 }
1147 \msg_new:nnnn { scontents } { for-key-unknown }
1148 { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachsc'~and~is~being~ignored. }
1149 {
1150   The~command~'\c_backslash_str foreachsc'~does~not~have~a~key~called~'#1'.\\
1151   Check~that~you~have~spelled~the~key~name~correctly.
1152 }
1153 \msg_new:nnnn { scontents } { for-key-value-unknown }
1154 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachsc'~and~is~being~ignored. }
1155 {
1156   The~command~'\c_backslash_str foreachsc'~does~not~have~a~key~called~'#1'.\\
1157   Check~that~you~have~spelled~the~key~name~correctly.
1158 }
1159 \msg_new:nnnn { scontents } { type-key-unknown }
1160 { The~key~'#1'~is~unknown~and~is~being~ignored. }
1161 {
1162   This~command~does~not~have~a~key~called~'#1'.\\
1163   This~command~only~accepts~the~key~'width-tab'.
1164 }
1165 \msg_new:nnnn { scontents } { type-key-value-unknown }
1166 { The~key~'#1'~to~which~you~passed~'#2'~is~unknown~and~is~being~ignored. }
1167 {
1168   This~command~does~not~have~a~key~called~'#1'.\\
1169   This~command~only~accepts~the~key~'width-tab'.
1170 }
1171 \msg_new:nnn { scontents } { empty-environment }
1172 { environment~'#1'~empty~\msg_line_context:. }
1173 \msg_new:nnnn { scontents } { verbatim-newline }
1174 { Verbatim~argument~of~#1~ended~by~end~of~line. }
1175 {
1176   The~verbatim~argument~of~the~#1~cannot~contain~more~than~one~line,~
1177   but~the~end~
1178   of~the~current~line~has~been~reached.~You~may~have~forgotten~the~
1179   closing~delimiter.
1180   \\ \\
1181   LaTeX~will~ignore~'#2'.
1182 }
1183 \msg_new:nnnn { scontents } { verbatim-tokenized }
1184 { The~verbatim~#1~cannot~be~used~inside~an~argument. }
1185 {
1186   The~#1~takes~a~verbatim~argument.~
1187   It~may~not~appear~within~the~argument~of~another~function.~
1188   It~received~an~illegal~token \tl_if_empty:nF {#3} { ~'#3' } .
1189   \\ \\
1190   LaTeX~will~ignore~'#2'.
1191 }

```

10.19 Finish package

Finish package implementation.

```

1192 </core>
1193 <plain | context>\ExplSyntaxOff

```

11 Index of Implementation

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

Symbols	
\'	792
*	1009, 1014, 1025
\,	792
\-	792
\<	792
\>	792
\\	16, 45, 858, 1083, 1090, 1123, 1132, 1138, 1144, 1150, 1156, 1162, 1168, 1180, 1189
\`	792
B	
\begingroup	70, 73
bool commands:	
\bool_if:NTF	614, 619, 645, 647, 736, 738, 745, 770, 816, 825, 839, 842, 874, 898, 900, 904
\bool_if:nTF	527, 534
\bool_lazy_and:nnTF	516
\bool_lazy_or:nnTF	395, 583
\bool_new:N	158, 160, 162, 164, 166, 168, 751
\bool_set_false:N	159, 163, 165, 167, 169, 191, 767
\bool_set_true:N	161, 187, 192, 218, 223, 231, 239, 778
C	
\catcode	858
char commands:	
\char_generate:nn	34, 989
\char_set_active_eq:NN	595, 984, 1004
\char_set_active_eq:nN	953
\char_set_catcode:nn	370
\char_set_catcode_active:N	458, 486, 487, 488, 928, 983, 1003, 1025, 1055
\char_set_catcode_active:n	815, 952, 1009
\char_set_catcode_letter:N	365
\char_set_catcode_letter:n	542
\char_set_catcode_other:N	45, 47, 48, 541, 783
\char_set_lccode:nn	1014
\l_char_special_seq	30, 541, 802
\char_value_catcode:n	364
\cleanseqsc	1, 6, 6, 36, <u>1074</u>
clist commands:	
\clist_map_function:NN	330
\countsc	1, 6, 6, 35, <u>1068</u>
cs commands:	
\cs:w	428, 429, 604, 605
\cs_end:	428, 429, 609, 610
\cs_generate_variant:Nn	177, 180, 181, 182, 310, 382, 651, 652
\cs_if_exist:NTF	34, 435, 436, 799
\cs_new:Npn	64, 179, 319, 347, 348, 350, 383, 393, 553, 563, 600, 626, 636, 642, 673, 801, 979, 981
\cs_new:Npx	178
\cs_new_eq:NN	727
\cs_new_protected:Npn	176, 254, 256, 262, 264, 270, 276, 278, 280, 297, 301, 303, 305, 311, 313, 317, 327, 332, 341, 354, 374, 401, 409, 415, 417, 424, 440, 443, 448, 456, 461, 475, 489, 491, 501, 503, 513, 524, 612, 643, 653, 660, 665, 676, 694, 706, 716, 729, 734, 752, 758, 760, 791, 793, 795, 810, 820, 835, 852, 868, 894, 914, 929, 950, 961, 1001, 1010, 1026, 1046, 1056
\cs_new_protected:Npx	687
\cs_replacement_spec:N	1065
\cs_set:Npn	557, 768
\cs_set:Npx	985
\cs_set_eq:NN	300, 307, 783, 786, 933, 934, 935, 947, 971
\cs_set_protected:Npn	240, 720
\cs_set_protected:Npx	543, 580
\cs_to_str:N	300, 307, 308
\cs_undefine:N	308
\csname	72, 82
D	
\DeclareOption	360
\def	2, 3, 5, 6, 71, 74, 75, 83, 96
\definetyping	976
dim commands:	
\c_zero_dim	988
\do	783, 786, 792, 802
\dospecials	784, <u>798</u>
E	
\else	93, 95
else commands:	
\else:	323, 997
\end	51, 603, 963
\endcsname	72, 82
\endgroup	71, 74, 77, 90, 104, 858
\endinput	91, 105
\endscontents	4, <u>466</u>
\endverbatimsc	52, <u>750</u> , 964
Environments	
scontents	24, 25
\errhelp	78
\errmessage	79
exp commands:	
\exp_after:wN	322, 324, 344, 345, 366, 604, 605, 788, 890, 891
\exp_args:Nc	340
\exp_args:Nf	387, 630
\exp_args:Nooo	426
\exp_args:NV	255, 263, 277, 279, 622, 920, 1059
\exp_not:N	60, 545, 548, 557, 560, 603, 604, 605, 621, 676, 691, 692, 711, 713, 720, 722, 940, 961, 963, 964, 965
\exp_not:n	412, 575, 634, 639, 640, 822, 837, 899, 905, 944
\expandafter	72, 82
\ExplSyntaxOff	37, 367, 1193
\ExplSyntaxOn	26, 358, 369
F	
\fi	81, 107, 108
fi commands:	
\fi:	325, 603, 608, 999
file commands:	
\file_if_exist:nTF	529, 536
\file_input:n	63, 368
\file_input_stop:	38
\foreachsc	1, 5, 5, 19, 21, 32, <u>862</u>
\frenchspacing	755

G

\getstored 1, 5, 5, 31, [846](#), 903
 group commands:
 \group_begin: . 445, 485, 526, 754, 812, 854, 870, 916,
 927, 1008, 1013, 1024, 1048, 1054
 \group_end: 451, 618, 650, 759, 827, 844, 889, 925, 949,
 1015, 1021, 1033, 1052, 1067
 \group_insert_after:N 404, 405, 406, 407

I

if commands:
 \if_false: 603, 608
 \if_meaning:w 321, 995
 \ifx 72, 82, 94
 \input 25
 int commands:
 \int_abs:n 397
 \int_compare:nNnTF 343, 657, 669, 1109, 1113
 \int_compare_p:nNn 396, 397
 \int_decr:N 666
 \int_incr:N 662, 663
 \int_new:N 154, 155, 156, 157
 \int_set:Nn 232, 285, 364, 763, 765, 855, 876, 917, 936,
 1049
 \int_step_function:nnnN 542, 879
 \int_to_roman:n 21, 282
 \int_zero:N 655
 \c_zero_int 669
 \interlinepenalty 775, 780
 iow commands:
 \iow_char:N 16, 838, 1060
 \iow_close:N 615
 \iow_log:n 29, 357
 \iow_new:N 175
 \iow_now:Nn 646
 \iow_open:Nn 532, 538

K

keys commands:
 \keys_define:nn 110, 141, 184, 206, 215, 249
 \l_keys_key_tl 20, 255, 263, 277, 279
 \keys_set:nn . 35, 420, 481, 814, 871, 918, 1037, 1050

L

left commands:
 \c_left_brace_str 59, 677, 691

M

\meaningsc 1, 6, 6, 18, 20, 21, 35, [1040](#)
 mode commands:
 \mode_if_horizontal:TF 779
 \mode_leave_vertical: 772, 796, 987
 msg commands:
 \msg_error:nn 377
 \msg_error:nnn 259, 267, 273, 290, 335, 437, 932
 \msg_error:nnnn ... 182, 260, 268, 274, 286, 291, 507
 \msg_expandable_error:nnn 391
 \msg_expandable_error:nnnn 398
 \msg_gset:nnn 32
 \msg_line_context: 33, 1082, 1094, 1096, 1102, 1104,
 1172
 \msg_new:nnn 337, 1080, 1093, 1095, 1097, 1099, 1101,
 1103, 1105, 1107, 1171
 \msg_new:nnnn ... 1086, 1117, 1126, 1135, 1141, 1147,
 1153, 1159, 1165, 1173, 1183
 \msg_set:nnn 13

\msg_warning:nn 19, 36, 658
 \msg_warning:nnn 520, 530, 531, 537
 \msg_warning:nnnn 589

N

\NewDocumentCommand ... 433, 471, 806, 831, 848, 864, 910,
 1036, 1042, 1076
 \NewDocumentEnvironment 442, 969
 \newenvsc 1, 4, 4, 24, 24, [414](#), 467, 1089
 \NewExpandableDocumentCommand 1070
 \next 71, 74, 83, 96, 109
 \nobreak 796
 \null 773

O

\obeylines 784

P

\PackageError 75, 85, 98
 Packages
 \lkeys2e 17
 scontents 15, 16, 18
 xparse 25
 \par 768
 \parfillskip 764
 \parindent 763
 \parskip 762, 765
 peek commands:
 \peek_charcode_ignore_spaces:NTF 312
 \peek_charcode_remove_ignore_spaces:NTF . 314
 \penalty 775, 780
 prg commands:
 \prg_generate_conditional_variant:Nnn .. 183
 \prg_new_protected_conditional:Npnn . 667, 992
 \prg_replicate:nn 980
 \prg_return_false: 671, 998
 \prg_return_true: 670, 996
 \ProcessKeysOptions 144
 \ProcessOptions 362
 \ProvidesExplPackage 9, 352

Q

quark commands:
 \q_mark 321, 329, 344, 345, 348, 349, 350
 \quark_new:N 173, 174
 quark internal commands:
 \q_scontents_mark 34, [173](#), 691, 713, 995
 \q_scontents_stop [173](#), 548, 560, 673, 692, 701, 704,
 713, 722

R

\relax 72, 82
 \RequirePackage 8, 315
 right commands:
 \c_right_brace_str 61, 677, 691

S

scan commands:
 \scan_stop: 934, 995, 1016
 \Scontents 1, 4, 4, 19, 20, 31, [805](#)
 \scontents 4, [466](#)
 scontents 3, [466](#)
 scontents internal commands:
 __scontents_analyse_nesting:n ... 28, 567, [653](#)
 __scontents_analyse_nesting:w [653](#)
 __scontents_analyse_nesting_format:n 656, 727,
 729

_scontents_analyse_nesting_generic:nn . 706,
 730, 731
 _scontents_analyse_nesting_generic:w . 701,
 704, 711, 720
 _scontents_analyse_nesting_generic-
 process:nn 694,
 723
 _scontents_analyse_nesting_latex:n . 687, 728
 _scontents_analyse_nesting_latex:w . 676, 684,
 689
 _scontents_append_contents:nn . . 23, 374, 412
 _scontents_atend_environment: 464, 734
 _scontents_check_line_process:nn . . . 25, 485
 _scontents_compat_redefine:Npn . 297, 310, 315,
 339, 352, 360, 362
 _scontents_compat_restore: 303, 371
 _scontents_compat_restore:N 304, 305
 \l_scontents_compat_seq 296, 299, 304
 _scontents_define_generic_nesting-
 function:n 708,
 716
 _scontents_do_noligs:N 34, 786, 935, 981
 \c_scontents_end_env_tl . 41, 546, 547, 558, 559, 575
 \g_scontents_end_verbatimsc_tl . . 41, 945, 962
 _scontents_env_define:nnn 427, 440
 _scontents_env_end_function: 582, 600
 _scontents_env_generic_begin: . 24, 24, 422, 456
 _scontents_env_generic_end: 24, 425, 461
 \l_scontents_env_name_tl . . 60, 414, 508, 521, 590,
 606, 682, 712, 721, 1104, 1120, 1123, 1129, 1132
 \l_scontents_env_nesting_int . . 18, 28, 154, 662,
 666, 669
 \l_scontents_file_iow . . . 175, 532, 538, 615, 646
 \l_scontents_file_tl 17, 146, 540, 622, 648
 _scontents_file_tl_write_start:n . 25, 511, 524,
 652
 _scontents_file_write_stop:N . . . 25, 515, 524
 \l_scontents_fname_out_tl . 17, 146, 188, 193, 511
 \l_scontents_forced_eol_bool . . . 126, 738, 839
 _scontents_foreach_add_body:n 883, 894
 \l_scontents_foreach_after_bool . 162, 223, 904
 \l_scontents_foreach_after_tl . 17, 146, 224, 905
 \l_scontents_foreach_before_bool . 162, 218, 898
 \l_scontents_foreach_before_tl . 17, 146, 219, 899
 \l_scontents_foreach_name_seq_tl . 17, 146, 872,
 903
 \l_scontents_foreach_print_seq . 18, 170, 873, 886,
 896
 \l_scontents_foreach_sep_tl 244, 887
 \l_scontents_foreach_start_int 227, 880
 \l_scontents_foreach_step_int 235, 881
 \l_scontents_foreach_stop_bool . 162, 231, 874
 \l_scontents_foreach_stop_int . 18, 154, 232, 876,
 882
 _scontents_foreach_wrapper:n 241, 901
 \l_scontents_foreach_wrapper_bool . 162, 239, 900
 _scontents_foreachsc_internal:nn . . 865, 868
 _scontents_format_case:nnn . . 64, 602, 607, 939
 _scontents_getfrom_seq:nn . 23, 383, 859, 920, 1059
 _scontents_getfrom_seq:nnn 383
 _scontents_getstored_internal:nn . . 849, 852
 _scontents_grab_optional:n 469
 _scontents_grab_optional:w . . . 25, 25, 469, 496
 \c_scontents_hidden_space_str . 18, 171, 587, 741,
 823, 840, 921, 1061
 _scontents_if_nested: 28
 _scontents_if_nested:TF 571, 653
 _scontents_lastfrom_seq:n . 23, 401, 746, 826, 843
 \l_scontents_macro_tmp_tl . 17, 146, 515, 518, 740,
 743
 _scontents_make_control_chars_active: . 510,
 550, 1001
 _scontents_meaningsc:n 1051, 1056
 _scontents_meaningsc_internal:nn . 1043, 1046
 \l_scontents_name_seq_cmd_tl . 115, 824, 826, 841,
 843
 \l_scontents_name_seq_env_tl . . . 112, 744, 746
 _scontents_nesting_decr: 28, 573, 653
 _scontents_nesting_incr: 660, 683, 700
 _scontents_nolig_list: 787, 791
 _scontents_norm_arg:n 31, 805
 _scontents_normalise_line_ends:N . 25, 480, 489
 _scontents_optarg:nn 311, 316, 318
 \l_scontents_overwrite_bool . . . 129, 527, 534
 _scontents_package_later_aux:Nn . . . 340, 341
 _scontents_par: 178, 1030
 _scontents_parse_command_keys:n . 20, 213, 262
 _scontents_parse_command_keys:nn 262
 _scontents_parse_environment_keys:n . 20, 204,
 254
 _scontents_parse_environment_keys:nn . . 254
 _scontents_parse_foreach_keys:n . 21, 247, 270
 _scontents_parse_foreach_keys:nn 270
 _scontents_parse_type_meaning_key:n . 252, 278
 _scontents_parse_type_meaning_key:nn . . 278
 _scontents_parse_typemeaning_key:n 21
 _scontents_parse_version:w 344, 345, 347
 _scontents_parse_version_auxi:w . . . 347, 348
 _scontents_parse_version_auxii:w . . 349, 350
 _scontents_plain_disable_outer_par: . 1010,
 1028
 \l_scontents_print_cmd_bool . . 23, 123, 825, 842
 \l_scontents_print_env_bool 23, 120, 745
 _scontents_provides_aux:nn 353, 354
 _scontents_remove_leading_nl:n 26, 524
 _scontents_remove_leading_nl:nn . . . 631, 636
 _scontents_remove_leading_nl:w 524
 _scontents_require_auxi:wn 316, 317
 _scontents_require_auxii:wnw 318, 327
 _scontents_require_auxiii:n 330, 332
 _scontents_rescan_tokens:n . 33, 34, 176, 404, 592,
 856, 937
 _scontents_ret:w . . . 26, 543, 551, 580, 595, 1031
 _scontents_Scontents_auxi:N 805
 _scontents_Scontents_internal:nn 805
 _scontents_scontents_setenv:nn 414
 \l_scontents_seq_item_int . 18, 154, 285, 917, 920,
 1049, 1059
 _scontents_set_active_eq:NN 794, 1001
 _scontents_setup_verb_processor: . . 421, 524
 _scontents_stararg:nn 313, 361, 363
 _scontents_start_after_option:w . 25, 483, 485
 _scontents_start_environment:w 459, 485
 _scontents_stop_environment: 463, 485
 _scontents_store_to_seq: 29
 _scontents_store_to_seq:NN . 23, 409, 743, 824, 841
 \l_scontents_storing_bool . 18, 28, 158, 191, 517,
 619, 647, 736

_scontents_tab:	178, 1029
\l_scontents_tab_width_int	132, 980
_scontents_tabs_to_spaces:	34, 953, 979, 1064
\l_scontents_temp_bool	751, 767, 770, 778
\g_scontents_temp_tl	17, 146, 403, 405, 407, 884, 891, 892
\l_scontents_temp_tl	17, 146, 329, 330, 479, 480, 481, 822, 823, 824, 837, 838, 840, 841, 919, 921, 922, 923, 924, 1058, 1060, 1061, 1062, 1064, 1065
_scontents_tl_if_head_is_q_mark:nTF	679, 698, 992
\l_scontents_tmpa_int	154, 364, 370, 655, 657, 663
_scontents_tystored_internal:nn	911, 914
_scontents_use_none_delimit_by_q_stop:w	653
_scontents_verb_arg:w	31, 805
_scontents_verb_arg_internal:n	805
\l_scontents_verb_font_tl	118, 785, 1063
_scontents_verb_print:N	32, 33, 908
_scontents_verb_print_EOL:	933, 947
_scontents_verb_processor_iterate:nnn	524
_scontents_verb_processor_iterate:w	524
_scontents_verb_processor_output:n	28, 568, 574, 579, 643
_scontents_verbatimsc_aux:	755, 760
_scontents_vobeyspaces:	755, 793
\l_scontents_writing_bool	18, 28, 158, 187, 192, 527, 534, 614, 645
_scontents_xobeysp:	794, 795
_scontents_xverb:	756, 950, 971
_scontents_xverb:w	908
_scontents_zap_space:ww	319, 324, 329
\Scontents*	20
\ScontentsCoreFileDate	3, 94
\ScontentsFileDate	2, 10, 27, 94
\ScontentsFileDescription	6, 10, 28
\ScontentsFileVersion	5, 10, 23, 28
seq commands:	
\seq_clear:N	873
\seq_clear_new:N	1077
\seq_count:N	388, 877, 1071
\seq_gput_right:Nn	380
\seq_if_exists:NTF	378, 385
\seq_item:Nn	399, 403
\seq_map_function:NN	304, 541, 802
\seq_new:N	170, 296, 379
\seq_put_right:Nn	299, 896
\seq_use:Nn	886
\setupsc	2, 35, 1034
skip commands:	
\skip_set:Nn	764
\skip_vertical:N	762
\space	27, 28
\startscontents	4, 466
\startverbatimsc	942
\stopscontents	4, 466
\stopverbatimsc	53, 965
str commands:	
\c_backslash_str	56, 508, 677, 690, 712, 721, 1102, 1104, 1136, 1138, 1142, 1144, 1148, 1150, 1154, 1156
\c_circumflex_str	172
\c_percent_str	172, 585
\str_const:Nn	171
\str_if_eq:nnTF	334, 495, 587, 682
\str_if_eq_p:nn	585
T	
TeX and L ^A T _E X 2 _ε commands:	
\@	364, 365, 370
\@ifpackageloaded	11
tex commands:	
\tex_everypar:D	788, 789
\tex_kern:D	988
\tex_let:D	1016
\tex_lowercase:D	1015
\tex_newlinechar:D	855, 936
\tex_par:D	766, 774, 780
\tex_scantokens:D	18, 176
\tex_the:D	789
\tex_unpenalty:D	789
tl commands:	
\c_space_tl	178
\tl_clear:N	540
\tl_const:Nn	54
\tl_gclear:N	406, 890
\tl_gset:Nn	27, 356, 403, 884
\tl_gset_rescan:Nnn	42
\tl_head:n	495, 632
\tl_if_blank:nTF	258, 266, 272, 284, 289, 351, 376, 505, 565, 578, 931
\tl_if_blank_p:n	584
\tl_if_empty:n	183
\tl_if_empty:NTF	923
\tl_if_empty:nTF	180, 282, 353, 1188
\tl_if_empty_p:N	518
\tl_if_head_is_N_type:nTF	493, 628, 696
\tl_if_novalue:nTF	477, 813, 871, 918, 1050
\tl_log:N	411, 922, 1062
\tl_new:N	41, 146, 147, 148, 149, 150, 151, 152, 153, 414
\tl_put_right:Nn	648, 740, 823, 840
\tl_remove_once:Nn	180, 180, 921, 1061
\tl_replace_all:Nnn	180, 181, 490, 838, 1060, 1064
\tl_rescan:nn	18
\tl_set:Nn	188, 193, 219, 224, 329, 419, 479, 621, 822, 837, 872, 919, 1058
\tl_to_str:n	427
\tl_use:N	590, 606, 712, 721, 785, 887, 903, 1063
token commands:	
\token_if_eq_meaning:NNTF	638
\token_to_str:N	984, 985
\tt	143
\ttfamily	142
\tystored	1, 5, 5, 18, 20, 21, 32, 908
U	
\unprotect	24
use commands:	
\use:N	29
\use:n	555, 616, 674, 709, 718, 902, 959
\use_none:n	322, 361
\use_none:nn	361
V	
\verbatim	972
\verbatimsc	750, 941
verbatimsc	6, 908
W	
\writestatus	23