

SCONTENTS

Stores L^AT_EX CONTENTS

V2.0 2022-04-04*

©2019–2022 by Pablo González†

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

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

Abstract

This package allows to store L^AT_EX code, including “*verbatim*”, in *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	Description of the package	1	5.7	The environment <code>verbatimsc</code>	6
2	Motivation and Acknowledgments	1	6	Other commands provided	7
3	License and Requirements	2	6.1	The command <code>\meaningsc</code>	7
4	The scontents package	2	6.2	The command <code>\countsc</code>	7
4.1	Installation	2	6.3	The command <code>\cleanseqsc</code>	7
4.2	Loading and options	2	7	The scontents package in action	7
4.3	The TAB character	2	8	Examples	8
4.4	Configuration of the options	3	8.1	From answers package	8
4.5	Options Overview	3	8.2	From filecontentsdef package	9
5	User interface	3	8.3	From TeX-SX	9
5.1	The environment <code>scontents</code>	4	8.4	Customization of <code>verbatimsc</code>	11
5.2	The command <code>\newenvsc</code>	5	9	Change history	14
5.3	The command <code>\Scontents</code>	5	10	Index of Documentation	15
5.4	The command <code>\getstored</code>	6	11	References	15
5.5	The command <code>\foreachsc</code>	6	12	Implementation	16
5.6	The command <code>\typestored</code>	6	13	Index of Implementation	40

1 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*.

2 Motivation and Acknowledgments

In L^AT_EX there is no direct way to record content for later use, although you can do this using `\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 L^AT_EX team for their great work and to the different members of the TeX-SX community who have provided great answers and ideas. Here a note of the main ones:

1. Stack datastructure using LaTeX
2. LaTeX equivalent of 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)

*This file describes a documentation for v2.0, last revised 2022-04-04.

†E-mail: pablgonz@educarchile.cl.

3 License and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (lpp1), version 1.3 or later (<http://www.latex-project.org/lpp1.txt>). The software has the status “maintained”.

The `SCONTENTS` package loads `expl3` (minimum version 2020-02-08) and `l3keys2e`. This package can be used with `plain`, `context`, `xelatex`, `lualatex`, `pdflatex` and the classical workflow `latex»dvips»ps2pdf`.

4 The scontents package

4.1 Installation

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

<code>scontents.tex</code>	»	<code>TDS:tex/generic/scontents/</code>
<code>scontents-code.tex</code>	»	<code>TDS:tex/generic/scontents/</code>
<code>scontents.sty</code>	»	<code>TDS:tex/latex/scontents/</code>
<code>t-scontents.mkiv</code>	»	<code>TDS:tex/context/third/scontents/</code>
<code>scontents.pdf</code>	»	<code>TDS:doc/latex/scontents/</code>
<code>README.md</code>	»	<code>TDS:doc/latex/scontents/</code>
<code>scontents.dtx</code>	»	<code>TDS:source/latex/scontents/</code>
<code>scontents.ins</code>	»	<code>TDS:source/latex/scontents/</code>

4.2 Loading and options

The package is loaded in the usual way:

For L^AT_EX users

```
\usepackage{scontents}
```

or

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

The package options are not available for plain TeX and ConTeXt, see 4.4.

For plain TeX users

```
\input scontents.tex
```

For ConTeXt users

```
\usemodule{scontents}
```

4.3 The TAB character

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

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

In a standard L^AT_EX document, the character TAB “`␣`” are treated as explicit spaces (in most contexts) and is the behavior when `⟨stored contents⟩`, but when `⟨writing files⟩` these are preserved.

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

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

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

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

4.4 Configuration of the options

Most of the options can be passed directly to the package or using the command `\setupsc`. All boolean keys can be passed using the equal sign “=” or just the name of the key, all unknown keys will return an error. In this section are described some of the options, a summary of all options is shown in section 4.5.

`\setupsc` `\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. However, options set in the declaration of an environment (with `\newenvsc`) have precedence over options set with `\setupsc`.

Options in the optional arguments of environments and commands have the highest precedence, overriding both options in `\newenvsc`, and in `\setupsc`.

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

Sets the `⟨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, for `\setupsc`, for `\Scontents*` and for the environment `scontents`.

`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.

4.5 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>
<code>store-env</code>	✓	✓	✓	✗	✗	✗	✗
<code>store-cmd</code>	✓	✓	✗	✓	✓	✗	✗
<code>print-env</code>	✓	✓	✓	✗	✗	✗	✗
<code>print-cmd</code>	✓	✓	✗	✓	✓	✗	✗
<code>print-all</code>	✓	✓	✗	✗	✗	✗	✗
<code>store-all</code>	✓	✓	✗	✗	✗	✗	✗
<code>write-env</code>	✗	✗	✓	✗	✗	✗	✗
<code>write-cmd</code>	✗	✗	✗	✗	✓	✗	✗
<code>write-out</code>	✗	✗	✓	✗	✓	✗	✗
<code>overwrite</code>	✓	✓	✓	✗	✓	✗	✗
<code>width-tab</code>	✓	✓	✗	✗	✗	✓	✓
<code>force-eol</code>	✓	✓	✓	✓	✓	✗	✗
<code>verb-font</code>	✓	✓	✗	✗	✗	✗	✗

5 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.

5.1 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`.

```
\scontents \scontents[(keyval list)]
\endscontents <env contents>
\endscontents
```

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

```
\startscntents \startscntents[(keyval list)]
\stopscntents <env contents>
\stopscntents
```

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, relative or absolute paths are not supported. If *file.ext* does not exist, it will be created or overwritten if the `overwrite` key is used.

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 (^[^]I) in *file.ext*.

`write-out = {<file.ext>}` default: *not used*

Sets the name of the *external file* in which the *contents* of the environment will be written. The *file.ext* will be created in the working directory, relative or absolute paths are not supported. If *file.ext* does not exist, it will be created or overwritten if the `overwrite` key is used.

The characters TABs will be written in $\langle file.ext \rangle$, the rest of the $\langle keys \rangle$ will not be available and the $\langle contents \rangle$ will NOT be stored in any $\langle sequence \rangle$. X_YTeX users using the TAB character must add `-8bit` at the command line, otherwise you will get T_EX-TAB ($\wedge\wedge\text{I}$) in $\langle file.ext \rangle$.

5.2 The command `\newenvsc`

```
\newenvsc {newenvsc{env name}[initial keys]}
```

The command `\newenvsc` allows you to create $\langle new environments \rangle$ based on the same characteristics of the `scontents` environment. The values entered in $[\langle initial keys \rangle]$ will be considered as the default values for this new environment and the valid $\langle keys \rangle$ are `store-env` and `print-env`. 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.

5.3 The command `\Scontents`

```
\Scontents [key = val]{argument}
```

```
\Scontents* [key = val]{argument}
```

```
\Scontents* [key = val]{del}{argument}{del}
```

The `\Scontents` command reads the $\langle argument \rangle$ 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 $\langle argument \rangle$ under *verbatim* category code regimen. If its first delimiter is a brace, it will be assumed that the $\langle argument \rangle$ is nested into braces. Otherwise it will be assumed that the ending of that $\langle argument \rangle$ is delimited by that first delimiter $\langle del \rangle$ 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 $[\langle key = val \rangle]$ 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 $\langle argument \rangle$ for other command.

Options for command

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

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

Sets the name of the $\langle sequence \rangle$ 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 $\langle stored content \rangle$ is displayed or not at the time of running the command. The content is extracted from the $\langle sequence \rangle$ in which it is stored.

Options only for the starred version

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

Sets the name of the $\langle external file \rangle$ in which the $\langle contents \rangle$ of the $\langle argument \rangle$ will be written. The $\langle file.ext \rangle$ will be created in the working directory, relative or absolute paths are not supported. If $\langle file.ext \rangle$ does not exist, it will be created or overwritten if the `overwrite` key is used.

The characters TABs will be written in $\langle file.ext \rangle$ and the $\langle contents \rangle$ will be stored in the $\langle sequence \rangle$ established at that time. X_YTeX users using the TAB character must add `-8bit` at the command line, otherwise you will get T_EX-TAB ($\wedge\wedge\text{I}$) in $\langle file.ext \rangle$.

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

Sets the name of the $\langle external file \rangle$ in which the $\langle contents \rangle$ of the $\langle argument \rangle$ will be written. The $\langle file.ext \rangle$ will be created in the working directory, relative or absolute paths are not supported. If $\langle file.ext \rangle$ does not exist, it will be created or overwritten if the `overwrite` key is used.

The characters TABs will be written in $\langle file.ext \rangle$, the rest of the $\langle keys \rangle$ will not be available and the $\langle contents \rangle$ will NOT be stored in any $\langle sequence \rangle$. X_YTeX users using the TAB character must add `-8bit` at the command line, otherwise you will get T_EX-TAB ($\wedge\wedge\text{I}$) in $\langle file.ext \rangle$.

The key `overwrite` is available for this command.

5.4 The command `\getstored`

```
\getstored <index>[<seq name>]
```

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

5.5 The command `\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}`.

5.6 The command `\tpestored`

```
\tpestored <index, width-tab = number>[<seq name>]
```

The command `\tpestored` 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 `{<seq name>}`. The key `width-tab` is available for this command.

5.7 The environment `verbatimsc`

```
verbatimsc
```

Internal environment used by `\tpestored` to display `<verbatim style>` contents.

One consideration to keep in mind is that this is a “*representation*” of the `<stored content>` in a “*verbatim*” environment.

The `verbatimsc` environment can be customized in the following ways after loading the `SCONTENTS` package:

Using the package `fancyvrb`:

```
\makeatletter
\let\verbatimsc@undefined
\let\endverbatimsc@undefined
\makeatother
\usepackage{fancyvrb}
\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}
  }
}{}
}
```

6 Other commands provided

6.1 The command `\meaningsc`

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

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

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

6.2 The command `\countsc`

`\countsc` `\countsc{⟨seq name⟩}`

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

6.3 The command `\cleanseqsc`

`\cleanseqsc` `\cleanseqsc{⟨seq name⟩}`

The command `\cleanseqsc` remove all contents stored in `{⟨seq name⟩}`.

7 The `SCONTENTS` package in action

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

Abstract

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

And the description of the package?

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

I’ve only written:

```
\begin{abstract}
This package allows to store \hologo{LaTeX} code, including \enquote{\emph{verbatim}},
```

in `\mymeta{sequences}` using the `\mypkg{l3seq}` module of `\mypkg{expl3}`. The `\mymeta{stored content}` can be used as many times as desired in the document, additionally you can write to `\mymeta{external files}` or show it in `\mymeta{verbatim style}`.

```
\end{abstract}
```

and

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

8 Examples


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

```
$ pdfdetach -saveall scontents.pdf
```

and then you can use the excellent `arara`³ tool to compile them.

8.1 From answers package

Example 1

Adaptation of example 1 of the package `answers` .

```
1 % arara: pdflatex
2 % arara: clean: { extensions: [ aux, log ] }
3 \documentclass{article}
4 \usepackage[store-cmd=solutions]{scontents}
5 \newtheorem{ex}{Exercise}
6 \setlength{\parindent}{0pt}
7 \pagestyle{empty}
8 \begin{document}
9 \section{Problems}
10 \begin{ex}
11 First exercise
12 \Scontents{First solution.}
13 \end{ex}
14
15 \begin{ex}
16 Second exercise
17 \Scontents{Second solution.}
18 \end{ex}
19
```

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



```

20 \section{Solutions}
21 \foreachc[sep={\\[10pt]}}{solutions}
22 \end{document}

```

8.2 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 \setlength{\parindent}{0pt}
6 \pagestyle{empty}
7 \begin{document}
8 % not starred
9 \Scontents{
10 Prove that  $[x^n+y^n=z^n]$  is not solvable in positive integers if  $n$  is at
11 most  $3$ . \par
12 }
13 % starred
14 \Scontents*|Refute the existence of black holes in less than  $140$  characters.|
15 % write environment to \jobname.txt
16 \begin{scontents}[write-env=\jobname.txt]
17 \def\NSA{NSA}%
18 Prove that factorization is easily done via probabilistic algorithms and
19 advance evidence from knowledge of the names of its employees in the
20 seventies that the \NSA\ has known that for  $40$  years. \par
21 \end{scontents}
22 % see all stored
23 \begin{itemize}
24 \foreachc[before={\item }]{defexercise}
25 \end{itemize}
26 % \getstored are robust :)
27 \section{\getstored[2]{defexercise}}
28 \end{document}

```

8.3 From TeX-SX

Example 3

Adapted from LaTeX equivalent of ConTeXt buffers .

```

1 % arara: pdflatex
2 % arara: clean: { extensions: [ aux, log ] }
3 \documentclass{article}
4 \usepackage[store-cmd=tikz]{scontents}
5 \usepackage{tikz}
6 \setlength{\parindent}{0pt}
7 \pagestyle{empty}
8 \Scontents*{\matrix{ \node (a) {$a$} ; & \node (b) {$b$} ; \\ } ;}
9 \Scontents*{\matrix[ampersand replacement=\&
10 { \node (a) {$a$} ; \& \node (b) {$b$} ; \\ } ;}
11 \Scontents*{\matrix{\node (a) {$a$} ; & \node (b) {$b$} ; \\ } ;}
12 \begin{document}
13 \section{tikzpicture}
14 \begin{tikzpicture}
15 \getstored[1]{tikz}
16 \end{tikzpicture}
17
18 \begin{tikzpicture}
19 \getstored[2]{tikz}
20 \end{tikzpicture}
21
22 \begin{tikzpicture}
23 \getstored{tikz}
24 \end{tikzpicture}
25
26 \begin{scontents}[store-env=buffer]
27 Hello World!

```

```

28
29 This is a \verb*|fake poor man's buffer :)|.
30 \end{scontents}
31
32 \section{source tikz}
33 \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 \setlength{\parindent}{0pt}
6 \pagestyle{empty}
7 \begin{document}
8 \begin{scontents}[store-env=main]
9 Something for main A.
10 \end{scontents}
11
12 \begin{scontents}[store-env=main]
13 Something for \verb|main B|.
14 \end{scontents}
15
16 \begin{scontents}[store-env=other]
17 Something for \verb|other|.
18 \end{scontents}
19
20 \textbf{Let's print them}
21
22 This is first stored in main: \getstored[1]{main}\par
23 This is second stored in main: \getstored{main}\par
24 This is stored in other: \getstored{other}
25
26 \textbf{Print all of stored in main}\par
27 \foreachsc[sep={\[\[10pt]}]{main}
28 \end{document}

```

Example 5

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

```

1 % arara: pdflatex
2 % arara: clean: { extensions: [ aux, log ] }
3 \documentclass{article}
4 \usepackage{scontents}
5 \setlength{\parindent}{0pt}
6 \pagestyle{empty}
7 \begin{document}
8 \section{Problem stated the first time}
9 \begin{scontents}[print-env=true,store-env=problem]
10 This is normal text.
11 \verb|This is from the verb command.|
12 \verb*|This is from the verb* command.|
13 This is normal text.
14 \begin{verbatim}

```

```

15 This is from the verbatim environment:
16 &{%{}}~
17 \end{verbatim}
18 \end{scontents}
19 \section{Problem restated}
20 \getstored[1]{problem}
21 \section{Problem restated once more}
22 \getstored[1]{problem}
23 \end{document}

```

Example 6

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


```

1 % arara: pdflatex
2 % arara: clean: { extensions: [ aux, log ] }
3 \documentclass[10pt]{article}
4 \usepackage{scontents}
5 \newenvsc{forshort}[store-env=forshort,print-env=false]
6 \setlength{\parindent}{0pt}
7 \pagestyle{empty}
8 \begin{document}
9
10 Something in the whole course.
11
12 \begin{forshort}
13     Just a summary...
14 \end{forshort}
15
16 \end{document}

```

8.4 Customization of verbatimsc

Example 7

Customization of `verbatimsc` using the `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{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 \lstineline[basicstyle=\ttfamily]| lstineline | 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 \lstineline[basicstyle=\ttfamily]|\LaTeX:  $E=mc^2$ |
42 and more.+
43
44 \section{Test \texttt{\textbackslash getstored}}
45 \getstored{contents}\par
46 \getstored[1]{contents}
47
48 \section{Test \texttt{\textbackslash typestored}}
49 \typestored[1]{contents}
50 \typestored[2]{contents}
51 \end{document}

```

Example 9

Customization of `verbatimsc` using the `minted` 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 \setlength{\parindent}{0pt}
14 \begin{document}
15 \section{Test \texttt{\textbackslash begin{scontents}} whit \texttt{minted}}
16 Test \verb+{scontents}+ \par
17
18 \begin{scontents}[overwrite,write-env=\jobname.tsc,force-eol=true]
19 Using \verb+scontents+ env no \verb+[key=val]+, save in seq \verb+contents+
20 with index 1.\par
21
22 Prove new \Verb*{ new fvextra whit braces } and environment \verb+Verbatim*+
23 \begin{Verbatim}[obeytabs, showtabs, tab=\rightarrowfill, tabcolor=red]
24 No tab
25         One real tab
26         Two real Tab plus         one tab
27 \end{Verbatim}
28 \end{scontents}
29
30 \section{See \Verb{\jobname.tsc}}
31 Read \Verb{\jobname.tsc} (shows TABs as red arrows):
32 \VerbatimInput[obeytabs, showtabs, tab=\rightarrowfill, tabcolor=red]{\jobname.tsc}
33
34 \section{Test \texttt{\textbackslash Scontents} whit \texttt{minted}}
35
36 \Scontents{ We have coded \par this in \LaTeX: $E=mc^2$}
37
38 \section{Test \texttt{\textbackslash getstored}}
39 \getstored[1]{contents}\par
40 \getstored{contents}
41
42 \section{Test \texttt{\textbackslash typestored}}
43 \typestored[1]{contents}
44 \typestored[2]{contents}
45 \end{document}

```

9 Change history

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

- v2.0 (ctan), 2022-04-04**
 - Adapting the `verbatimsc` environment (compatibility `verbatim` package).
 - Removed compatibility layer for older \TeX releases.
 - Fix loader in plain \TeX and `ConTeXt`.
 - Minor adjustments in the documentation.
- v1.9 (ctan), 2020-01-21**
 - Update and improvements in the internal code.
 - Updating the generic code for I/O verification.
 - Add `write-cmd` and `write-out` keys for `\Scontents*`.
 - Fix `sep` key in `\foreachsc`.
- v1.8 (ctan), 2019-11-18**
 - Add `\newenvsc` command.
 - Fix nested environment in plain \TeX and `ConTeXt`.
 - Modified default value in `\getstored`.
 - Add `overwrite` key to reduce I/O operations.
 - Deleted an unnecessary group in the code.
- v1.7 (ctan), 2019-10-29**
 - The `verbatimsc` environment was rewritten.
 - Minor adjustments in documentation.
- v1.6 (ctan), 2019-10-26**
 - The internal behavior of `\getstored` has been modified.
 - The internal behavior of `\foreachsc` has been modified.
 - Corrected file extension for `ConTeXt`.
 - Remove spurious warning.
- v1.5 (ctan), 2019-10-24**
 - Add support for plain \TeX and `ConTeXt`.
 - Split internal code for optimization.
 - Add support for vertical spaces in `key=val`.
 - Add `\foreachsc` command.
 - Check if `verbatim` package is loaded.
- v1.4 (ctan), 2019-10-03**
 - Add `store-all` key.
 - Messages and keys were separated.
 - Restructuring of documentation.
 - Now the version of `expl3` is checked instead of `xparse`.
 - The internal behavior of `force-eol` has been modified.
- v1.3 (ctan), 2019-09-24**
 - The environment can now nest.
 - Added `force-eol`, `verb-font` and `width-tab` keys.
 - The extra space has been removed when you run `\getstored`.
 - Internal code has been rewritten more efficiently.
 - Remove starred argument for `\tpestored`.
 - 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.

10 Index of Documentation

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

C	
Commands provide by SCONTENTS :	
<code>\Scontents*</code>	3, 5
<code>\Scontents</code>	3, 5
<code>\cleanseqsc</code>	7
<code>\countsc</code>	7
<code>\endscontents</code>	4
<code>\foreachsc</code>	6
<code>\getstored</code>	3, 4, 6
<code>\meaningsc</code>	3, 7
<code>\newenvsc</code>	3, 5
<code>\scontents</code>	4
<code>\setupsc</code>	3–5
<code>\startscontents</code>	4
<code>\stopscontents</code>	4
<code>\typestored</code>	3, 6
E	
Environment provide by SCONTENTS :	
<code>scontents</code>	3–5
<code>verbatimsc</code>	6, 11–13
Environments	
<code>Verbatim</code>	2
<code>filecontentsdefmacro</code>	1
<code>lstlisting</code>	2
K	
Keys	
<code>after</code>	6
<code>before</code>	6
<code>force-eol</code>	3–5
<code>overwrite</code>	3–5
<code>print-all</code>	3
<code>print-cmd</code>	3, 5
<code>print-env</code>	3–5
<code>sep</code>	6
<code>start</code>	6
<code>step</code>	6
<code>stop</code>	6
<code>store-all</code>	3
<code>store-cmd</code>	3, 5
<code>store-env</code>	3–5
<code>verb-font</code>	3
<code>width-tab</code>	3, 6, 7
<code>wrapper</code>	6
<code>write-cmd</code>	3, 5
<code>write-env</code>	3, 4
<code>write-out</code>	3–5
L	
<code>\lstinline</code>	5
M	
<code>\meaning</code>	7
P	
Packages	
<code>answers</code>	8
<code>expl3</code>	1, 2, 7, 14
<code>fancyvrb</code>	2, 3, 6, 11
<code>filecontentsdef</code>	1, 7, 9, 14
<code>fvextra</code>	5
<code>l3keys2e</code>	2
<code>l3seq</code>	1, 7
<code>listings</code>	2, 5, 7, 12
<code>minted</code>	7, 13
<code>scontents</code>	1, 2, 6–8, 14
<code>tcolorbox</code>	11
V	
<code>\Verb</code>	5
<code>\verb</code>	5

11 References

- [1] The L^AT_EX Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/expl3>, 2020.
- [2] The L^AT_EX Project. “The xparse package”. Available from CTAN, <https://www.ctan.org/pkg/xparse>, 2020.
- [3] The L^AT_EX Project. “The l3keys2e package”. Available from CTAN, <https://www.ctan.org/pkg/l3keys2e>, 2020.
- [4] WRIGHT, JOSEPH. “Programming key–value in expl3”. Available from TUGBOAT, <https://www.tug.org/TUGboat/tb31-1/tb97wright-l3keys.pdf>, 2010.

12 Implementation

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

12.1 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{2022-04-04}%
3 <core>\def\ScontentsCoreFileDate{2022-04-04}%
4 <*loader>
5 \def\ScontentsFileVersion{2.0}%
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.

```
7 <*latex>
8 \RequirePackage{l3keys2e}[2020/02/08]
9 \ProvidesExplPackage
10   {scontents} {\ScontentsFileDate} {\ScontentsFileVersion} {\ScontentsFileDescription}
11 </latex>
```

The plain \TeX 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) and add `\ExplSyntaxOn` to be able to load `xparse` later.

```
12 <!!latex>
13 <context>\writestatus{loading}{User Module scontents v\ScontentsFileVersion}
14 <context>\unprotect
15 \input expl3-generic.tex
16 \ExplSyntaxOn
17 \tl_gset:cx { ver @ scontents . sty } { \ScontentsFileDate\space
18   v\ScontentsFileVersion\space \ScontentsFileDescription }
19 \iow_log:x { Package: ~ scontents ~ \use:c { ver @ scontents . sty } }
20 </!!latex>
```

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

```
21 <*plain>
22 \msg_gset:nnn { scontents } { already-loaded }
23   { The~'scontents'~package~is~already~loaded.~Aborting~input~\msg_line_context:. }
24 \cs_if_exist:NT \__scontents_rescan_tokens:n
25   {
26     \msg_warning:nn { scontents } { already-loaded }
27     \ExplSyntaxOff
28     \file_input_stop:
29   }
30 </plain>
```

12.2 Definition of variables by format

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

`\g__scontents_end_verbatimsc_tl` A global token list `\g__scontents_end_verbatimsc_tl` match when ending `verbatimsc` environment.

```
31 \tl_new:N \g__scontents_end_verbatimsc_tl
32 \tl_gset_rescan:Nnn \g__scontents_end_verbatimsc_tl
33   {
34     \char_set_catcode_other:N \
35 <*latex>
36     \char_set_catcode_other:N {\
37     \char_set_catcode_other:N \}
38 </latex>
39   }
40 <latex> { \end{verbatimsc} }
```



```

41 <plain> { \endverbatim }
42 <context> { \stopverbatim }

```

(End definition for `\g__scontents_end_verbatimsc_tl`.)

`\c__scontents_end_env_tl` A token list `\c__scontents_end_env_tl` match when ending environments defined by `\newenvsc`,
`\l__scontents_env_name_tl` `\l__scontents_env_name_tl` storing the name of environments defined by `\newenvsc`.

```

43 \tl_new:N \l__scontents_env_name_tl
44 \tl_const:Nx \c__scontents_end_env_tl
45 {
46   \c_backslash_str
47 <latex|plain> end
48 <context> stop
49 <latex> \c_left_brace_str
50   \exp_not:N \l__scontents_env_name_tl
51 <latex> \c_right_brace_str
52 }

```

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

Now we load the core `SCONTENTS` code:

```

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

```

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

```

54 \cs_new:Npn \__scontents_format_case:nnn #1 #2 #3
55 <latex> {#1} % LaTeX
56 <plain> {#2} % Plain/Generic
57 <context> {#3} % ConTeXt

```

(End definition for `__scontents_format_case:nnn`.)

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

```

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

```

```

94         are in conflict. Loading of scontents will abort.
95     }%
96     \endgroup
97     \endinput
98 }%
99 \fi
100 \fi
101 \next

```

12.3 Definition of temporary variables

The token list `\l__scontents_macro_tmp_tl` is a temporary token list to hold the contents of the macro/environment. `\l__scontents_temp_tl`, `\g__scontents_temp_tl`, `\l__scontents_tmpa_int` and `\l__scontents_temp_bool` are generic temporary vars.

```

102 \tl_new:N \l__scontents_macro_tmp_tl
103 \tl_new:N \l__scontents_temp_tl
104 \tl_new:N \g__scontents_temp_tl
105 \int_new:N \l__scontents_tmpa_int
106 \bool_new:N \l__scontents_temp_bool

```

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

12.4 Compatibility layer with plain \TeX and \ConTeXt

When loading the package outside of \LaTeX we can't usually use `xparse`. However since `xparse` now `ltxcmds` is part of the \LaTeX kernel is loadable in any format.

```

107 </core>
108 <*loader&!latex>
109 \int_set:Nn \l__scontents_tmpa_int { \char_value_catcode:n { \@ } }
110 \char_set_catcode_letter:N \@
111 \file_input:n { xparse-generic.tex }
112 \char_set_catcode:nn { \@ } { \l__scontents_tmpa_int }
113 </loader&!latex>
114 <*core>

```

12.5 Definition of keys for the package

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

```

115 \keys_define:nn { scontents }
116 {
117     store-env .tl_set:N          = \l__scontents_name_seq_env_tl,
118     store-env .initial:n         = contents,
119     store-env .value_required:n = true,
120     store-cmd .tl_set:N          = \l__scontents_name_seq_cmd_tl,
121     store-cmd .initial:n         = contents,
122     store-cmd .value_required:n = true,
123     verb-font .tl_set:N          = \l__scontents_verb_font_tl,
124     verb-font .value_required:n = true,
125     print-env .bool_set:N        = \l__scontents_print_env_bool,
126     print-env .initial:n         = false,
127     print-env .default:n         = true,
128     print-cmd .bool_set:N        = \l__scontents_print_cmd_bool,
129     print-cmd .initial:n         = false,
130     print-cmd .default:n         = true,
131     force-eol .bool_set:N        = \l__scontents_forced_eol_bool,
132     force-eol .initial:n         = false,
133     force-eol .default:n         = true,
134     overwrite .bool_set:N        = \l__scontents_overwrite_bool,
135     overwrite .initial:n         = false,
136     overwrite .default:n         = true,
137     width-tab .int_set:N         = \l__scontents_tab_width_int,
138     width-tab .initial:n         = 1,
139     width-tab .value_required:n = true,
140     print-all .meta:n           = { print-env = #1 , print-cmd = #1 },
141     print-all .default:n        = true,
142     store-all .meta:n           = { store-env = #1 , store-cmd = #1 },
143     store-all .value_required:n = true
144 }

```

```

145 </core>
146 <loader>\keys_define:nn { scontents }
147 <latex> { verb-font .initial:n = \ttfamily }
148 <plain|context> { verb-font .initial:n = \tt }

```

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

```

149 <latex>\ProcessKeysOptions { scontents }
150 <*core>

```

12.6 Internal variables and utility functions

`\l__scontents_fname_out_tl` The token list `\l__scontents_fname_out_tl` is used for store the name of the $\langle output\ file \rangle$, when there's one. Its value is set by the keys `write-env`, `write-out` and `write-cmd`.

```

\l__scontents_every_line_env_tl
\l__scontents_file_iow

```

The token list `\l__scontents_every_line_env_tl` holds the contents of an environment, `scontents` by default, as it's being read. `\l__scontents_file_iow` is an output stream for saving the contents of an environment (or command) to a file.

This variables is used by the function `__scontents_file_tl_write_start:n` (see 12.10.5).

```

151 \tl_new:N \l__scontents_fname_out_tl
152 \tl_new:N \l__scontents_every_line_env_tl
153 \iow_new:N \l__scontents_file_iow

```

(End definition for `\l__scontents_fname_out_tl`, `\l__scontents_every_line_env_tl`, and `\l__scontents_file_iow`.)

`\l__scontents_foreach_name_seq_tl` `\l__scontents_foreach_name_seq_tl` is the name assigned to the sequence on which the loop will be made, `\l__scontents_foreach_before_tl` and `\l__scontents_foreach_after_tl` are token lists in which the assigned material will be placed before and after the execution of the `\foreachsc` loop.

```

\l__scontents_foreach_before_tl
\l__scontents_foreach_after_tl

```

```

154 \tl_new:N \l__scontents_foreach_name_seq_tl
155 \tl_new:N \l__scontents_foreach_before_tl
156 \tl_new:N \l__scontents_foreach_after_tl

```

(End definition for `\l__scontents_foreach_name_seq_tl`, `\l__scontents_foreach_before_tl`, and `\l__scontents_foreach_after_tl`.)

`\l__scontents_seq_item_int` `\l__scontents_seq_item_int` stores the index in the sequence of the item requested to `\tpestored` 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.

```

\l__scontents_env_nesting_int
\l__scontents_foreach_stop_int

```

```

157 \int_new:N \l__scontents_foreach_stop_int
158 \int_new:N \l__scontents_seq_item_int
159 \int_new:N \l__scontents_env_nesting_int

```

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

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

```

\l__scontents_storing_bool
\l__scontents_writable_bool

```

```

160 \bool_new:N \l__scontents_writing_bool
161 \bool_set_false:N \l__scontents_writing_bool
162 \bool_new:N \l__scontents_storing_bool
163 \bool_set_true:N \l__scontents_storing_bool
164 \bool_new:N \l__scontents_writable_bool

```

(End definition for `\l__scontents_writing_bool`, `\l__scontents_storing_bool`, and `\l__scontents_writable_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

```

```

165 \bool_new:N \l__scontents_foreach_before_bool
166 \bool_set_false:N \l__scontents_foreach_before_bool
167 \bool_new:N \l__scontents_foreach_after_bool
168 \bool_set_false:N \l__scontents_foreach_after_bool
169 \bool_new:N \l__scontents_foreach_stop_bool
170 \bool_set_false:N \l__scontents_foreach_stop_bool
171 \bool_new:N \l__scontents_foreach_wrapper_bool
172 \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`.

```
173 \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.

```
174 \str_const:Nx \c__scontents_hidden_space_str
175 { \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.

`\q__scontents_mark`

```
176 \quark_new:N \q__scontents_stop
```

```
177 \quark_new:N \q__scontents_mark
```

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

`\l__scontents_save_sf_int` Internal variables used by functions `__scontents_bsphack:` and `__scontents_esphack:`.

`\l__scontents_save_skip`

```
178 \int_new:N \l__scontents_save_sf_int
```

```
179 \skip_new:N \l__scontents_save_skip
```

(End definition for `\l__scontents_save_sf_int` and `\l__scontents_save_skip`.)

`__scontents_rescan_tokens:n` The function `\tl_rescan:nn` provided by `expl3` 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. See the answer by Ulrich Diez in [How do use {<setup>} in \tl_set_rescan:Nnn to replace \scantokens?](#)

`__scontents_rescan_tokens:x`

`__scontents_rescan_tokens:v`

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

```
181 \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:`

```
182 \cs_new:Npx \__scontents_tab: { \c_space_tl }
```

```
183 \cs_new:Npn \__scontents_par: { ^^J ^^J }
```

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

`\tl_remove_once:NV` Some nonstandard kernel variants.

`\tl_replace_all:Nxx`

`\tl_replace_all:Nxn`

`\tl_replace_all:Nnx`

`\tl_if_empty:fTF`

```
184 \cs_generate_variant:Nn \tl_remove_once:Nn { NV }
```

```
185 \cs_generate_variant:Nn \tl_replace_all:Nnn { Nx, Nxx, Nnx }
```

```
186 \cs_generate_variant:Nn \msg_error:nnnn { nnx }
```

```
187 \prg_generate_conditional_variant:Nnn \tl_if_empty:n { f } { TF }
```

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

12.7 Defining keys for the environment and commands

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

12.7.1 Keys for environment scontents

We define a set of *(keys)* for environment `scontents`.

```

188 \keys_define:nn { scontents / scontents }
189 {
190   write-env .code:n          = {
191     \bool_set_true:N \__scontents_writing_bool
192     \tl_set:Nn \__scontents_fname_out_tl {#1}
193   },
194   write-out .code:n         = {
195     \bool_set_false:N \__scontents_storing_bool
196     \bool_set_true:N  \__scontents_writing_bool
197     \tl_set:Nn \__scontents_fname_out_tl {#1}
198   },
199   write-env .value_required:n = true,
200   write-out .value_required:n = true,
201   print-env .meta:nn         = { scontents } { print-env = #1 },
202   print-env .default:n       = true,
203   store-env .meta:nn         = { scontents } { store-env = #1 },
204   force-eol .meta:nn         = { scontents } { force-eol = #1 },
205   force-eol .default:n       = true,
206   overwrite .meta:nn         = { scontents } { overwrite = #1 },
207   overwrite .default:n       = true,
208   unknown .code:n           = { \__scontents_parse_environment_keys:n {#1} }
209 }

```

12.7.2 Keys for command \Scontents

We define a set of *(keys)* for commands `\Scontents` and `\Scontents*`.

```

210 \keys_define:nn { scontents / Scontents }
211 {
212   write-cmd .code:n          = {
213     \bool_set_true:N \__scontents_writing_bool
214     \tl_set:Nn \__scontents_fname_out_tl {#1}
215   },
216   write-out .code:n         = {
217     \bool_set_false:N \__scontents_storing_bool
218     \bool_set_true:N  \__scontents_writing_bool
219     \tl_set:Nn \__scontents_fname_out_tl {#1}
220   },
221   write-cmd .value_required:n = true,
222   write-out .value_required:n = true,
223   print-cmd .meta:nn         = { scontents } { print-cmd = #1 },
224   print-cmd .default:n       = true,
225   store-cmd .meta:nn         = { scontents } { store-cmd = #1 },
226   force-eol .meta:nn         = { scontents } { force-eol = #1 },
227   force-eol .default:n       = true,
228   overwrite .meta:nn         = { scontents } { overwrite = #1 },
229   overwrite .default:n       = true,
230   unknown .code:n           = { \__scontents_parse_command_keys:n {#1} }
231 }

```

12.7.3 Keys for command \foreachsc

We define a set of *(keys)* for command `\foreachsc`.

```

232 \keys_define:nn { scontents / foreachsc }
233 {
234   before .code:n           = {
235     \bool_set_true:N \__scontents_foreach_before_bool
236     \tl_set:Nn \__scontents_foreach_before_tl {#1}
237   },
238   before .value_required:n = true,
239   after .code:n            = {
240     \bool_set_true:N \__scontents_foreach_after_bool
241     \tl_set:Nn \__scontents_foreach_after_tl {#1}
242   },
243   after .value_required:n = true,
244   start .int_set:N         = \__scontents_foreach_start_int,
245   start .value_required:n = true,
246   start .initial:n         = 1,

```

```

247   stop   .code:n          = {
248           \bool_set_true:N \__scontents_foreach_stop_bool
249           \int_set:Nn \__scontents_foreach_stop_int {#1}
250         },
251   stop   .value_required:n = true,
252   step   .int_set:N         = \__scontents_foreach_step_int,
253   step   .value_required:n = true,
254   step   .initial:n        = 1,
255   wrapper .code:n          = {
256           \bool_set_true:N \__scontents_foreach_wrapper_bool
257           \cs_set_protected:Npn
258           \__scontents_foreach_wrapper:n ##1 {#1}
259         },
260   wrapper .value_required:n = true,
261   sep     .tl_set:N         = \__scontents_foreach_sep_tl,
262   sep     .initial:n        = {},
263   sep     .value_required:n = true,
264   unknown .code:n          = { \__scontents_parse_foreach_keys:n {#1} }
265 }

```

12.7.4 Key for commands \tystored and \meaningsc

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

```

266 \keys_define:nn { scontents / typemeaning }
267 {
268   width-tab .meta:nn = { scontents } { width-tab = #1 },
269   unknown   .code:n = { \__scontents_parse_type_meaning_key:n {#1} }
270 }

```

12.8 Handling undefined keys

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

12.8.1 Undefined keys for environment scontents

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

```

271 \cs_new_protected:Npn \__scontents_parse_environment_keys:n #1
272 { \exp_args:NV \__scontents_parse_environment_keys:nn \l_keys_key_str {#1} }
273 \cs_new_protected:Npn \__scontents_parse_environment_keys:nn #1#2
274 {
275   \tl_if_blank:nTF {#2}
276   { \msg_error:nnn { scontents } { env-key-unknown } {#1} }
277   { \msg_error:nnnn { scontents } { env-key-value-unknown } {#1} {#2} }
278 }

```

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

12.8.2 Undefined keys for \Scontents and \Scontents*

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

```

279 \cs_new_protected:Npn \__scontents_parse_command_keys:n #1
280 { \exp_args:NV \__scontents_parse_command_keys:nn \l_keys_key_str {#1} }
281 \cs_new_protected:Npn \__scontents_parse_command_keys:nn #1#2
282 {
283   \tl_if_blank:nTF {#2}
284   { \msg_error:nnn { scontents } { cmd-key-unknown } {#1} }
285   { \msg_error:nnnn { scontents } { cmd-key-value-unknown } {#1} {#2} }
286 }

```

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

12.8.3 Undefined keys for \foreachsc

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.

```

\__scontents_parse_foreach_keys:n
\__scontents_parse_foreach_keys:n

```

```

287 \cs_new_protected:Npn \__scontents_parse_foreach_keys:nn #1#2
288   {
289     \tl_if_blank:nTF {#2}
290       { \msg_error:nnn { scontents } { for-key-unknown } {#1} }
291       { \msg_error:nnnn { scontents } { for-key-value-unknown } {#1} {#2} }
292   }
293 \cs_new_protected:Npn \__scontents_parse_foreach_keys:n #1
294   { \exp_args:NV \__scontents_parse_foreach_keys:nn \l_keys_key_str {#1} }

```

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

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

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

```

295 \cs_new_protected:Npn \__scontents_parse_type_meaning_key:n #1
296   { \exp_args:NV \__scontents_parse_type_meaning_key:nn \l_keys_key_str {#1} }
297 \cs_new_protected:Npn \__scontents_parse_type_meaning_key:nn #1#2
298   {
299     \tl_if_empty:ftF { \int_to_roman:n { -0 #1 } }
300     {
301       \tl_if_blank:nTF {#2}
302         { \int_set:Nn \l__scontents_seq_item_int {#1} }
303         { \msg_error:nnnn { scontents } { type-key-value-unknown } {#1} {#2} }
304     }
305     {
306       \tl_if_blank:nTF {#2}
307         { \msg_error:nnn { scontents } { type-key-unknown } {#1} }
308         { \msg_error:nnnn { scontents } { type-key-value-unknown } {#1} {#2} }
309     }
310   }

```

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

12.9 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:Vx

```

The function `__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.

```

311 \cs_new_protected:Npn \__scontents_append_contents:nn #1#2
312   {
313     \tl_if_blank:nT {#1}
314       { \msg_error:nn { scontents } { empty-store-cmd } }
315     \seq_if_exist:cF { g__scontents_name_#1_seq }
316       { \seq_new:c { g__scontents_name_#1_seq } }
317     \seq_gput_right:cn { g__scontents_name_#1_seq } {#2}
318   }
319 \cs_generate_variant:Nn \__scontents_append_contents:nn { Vx }

```

(End definition for `__scontents_append_contents:nn`)

```

\__scontents_getfrom_seq:nn
\__scontents_getfrom_seq:nnn

```

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

```

320 \cs_new:Npn \__scontents_getfrom_seq:nn #1#2
321   {
322     \seq_if_exist:cTF { g__scontents_name_#2_seq }
323     {
324       \exp_args:Nf \__scontents_getfrom_seq:nnn
325         { \seq_count:c { g__scontents_name_#2_seq } }
326         {#1} {#2}
327     }

```

```

328     { \msg_expandable_error:nnn { scontents } { undefined-storage } {#2} }
329   }
330 \cs_new:Npn \__scontents_getfrom_seq:nnn #1#2#3
331   {
332     \bool_lazy_or:nnTF
333     { \int_compare_p:nNn {#2} = { 0 } }
334     { \int_compare_p:nNn { \int_abs:n {#2} } > {#1} }
335     { \msg_expandable_error:nnnn { scontents } { index-out-of-range } {#2} {#3} {#1} }
336     { \seq_item:cn { g__scontents_name_#3_seq } {#2} }
337   }

```

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

`__scontents_lastfrom_seq:n` The function `__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.

`__scontents_lastfrom_seq:V`

```

338 \cs_new_protected:Npn \__scontents_lastfrom_seq:n #1
339   {
340     \tl_gset:Nx \g__scontents_temp_tl { \seq_item:cn { g__scontents_name_#1_seq } {-1} }
341     \group_insert_after:N \__scontents_rescan_tokens:V
342     \group_insert_after:N \g__scontents_temp_tl
343     \group_insert_after:N \tl_gclear:N
344     \group_insert_after:N \g__scontents_temp_tl
345   }
346 \cs_generate_variant:Nn \__scontents_lastfrom_seq:n { V }

```

(End definition for `__scontents_lastfrom_seq:n`.)

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

`__scontents_store_to_seq:NN`

```

347 \cs_new_protected:Npn \__scontents_store_to_seq:NN #1#2
348   {
349     \tl_log:N #1
350     \__scontents_append_contents:Vx #2 { \exp_not:V #1 }
351   }

```

(End definition for `__scontents_store_to_seq:NN`.)

12.10 The command `\newenvsc` and 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.

12.10.1 The command `\newenvsc`

`\newenvsc`
`__scontents_env_setting:nn`
`__scontents_env_define:nnn`

The internal function `__scontents_env_setting:nn` defines two functions `__scontents_#1_ env_begin:` and `__scontents_#1_ env_end:`, which set the current environment's name in `#1` and `\l__scontents_env_name_tl` and default properties in `#2` then call `__scontents_setup_verb_processor:`, the generic `__scontents_env_generic_begin:` and `__scontents_env_generic_end:`. Finally the function `__scontents_env_define:nnn` will create the environments.

```

352 \cs_new_protected:Npn \__scontents_env_setting:nn #1 #2
353   {
354     \cs_new_protected:cpn { __scontents_#1_env_begin: }
355     {
356       \tl_set:Nn \l__scontents_env_name_tl {#1}
357       \keys_set:nn { scontents } {#2}
358       \__scontents_setup_verb_processor:
359       \__scontents_env_generic_begin:
360     }
361     \cs_new_protected:cpn { __scontents_#1_env_end: }
362     { \__scontents_env_generic_end: }
363     \exp_args:Nooo % http://nooooooooooooooooo.com :) jeje
364     \__scontents_env_define:nnn { \tl_to_str:n {#1} }
365     { \cs:w __scontents_#1_env_begin: \cs_end: }
366     { \cs:w __scontents_#1_env_end: \cs_end: }
367   }
368 </core>
369 <*loader>
370 \NewDocumentCommand \newenvsc { m O{} }

```



```

371 {
372 <latex | plain> \cs_if_exist:cTF { #1 }
373 <context> \cs_if_exist:cTF { start #1 }
374 { \msg_error:nnn { scontents } { env-already-defined } {#1} }
375 { \__scontents_env_setting:nn {#1} {#2} }
376 }
377 \cs_new_protected:Npn \__scontents_env_define:nnn #1 #2 #3
378 {
379 <latex | plain> \NewDocumentEnvironment {#1} { }
380 <context> \cs_new_protected:cpn { start #1 }
381 {
382 <!!latex> \group_begin:
383 #2
384 }
385 <context> \cs_new_protected:cpn { stop #1 }
386 {
387 #3
388 <!!latex> \group_end:
389 }
390 }
391 </loader>
392 <*core>

```

(End definition for `\newenvsc`, `__scontents_env_setting:nn`, and `__scontents_env_define:nnn`. This function is documented on page 5.)

12.10.2 Generic definition of the environment

`__scontents_env_generic_begin:` Now we define the generic environment functions `__scontents_env_generic_begin:` and `__scontents_env_generic_end:`

```

393 \cs_new_protected:Npn \__scontents_env_generic_begin:
394 {
395 \char_set_catcode_active:N ^^M
396 \__scontents_start_environment:w
397 }
398 \cs_new_protected:Npn \__scontents_env_generic_end:
399 {
400 \__scontents_stop_environment:
401 \__scontents_finish_storing:NNN \l__scontents_macro_tmp_tl
402 \l__scontents_name_seq_env_tl \l__scontents_print_env_bool
403 }

```

(End definition for `__scontents_env_generic_begin:` and `__scontents_env_generic_end:`.)

12.10.3 Definition of the environment scontents

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

```

scontents
\contents
\endscontents
\startcontents
\stopscontents

```

```

404 </core>
405 <loader>\newenvsc{scontents}
406 <*core>

```

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

12.10.4 key val for environment

`__scontents_grab_optional:n` 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.

```

407 </core>
408 <*loader>
409 \NewDocumentCommand \__scontents_grab_optional:w { r[] }
410 { \__scontents_grab_optional:n {#1} }
411 </loader>
412 <*core>
413 \cs_new_protected:Npn \__scontents_grab_optional:n #1
414 {
415 \tl_if_novalue:nF {#1}
416 {

```

```

417     \tl_set:Nn \__scontents_temp_tl {#1}
418     \__scontents_normalise_line_ends:N \__scontents_temp_tl
419     \keys_set:nV { scontents / scontents } \__scontents_temp_tl
420   }
421   \__scontents_start_after_option:w
422 }

```

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

12.10.5 The environment itself

```

\__scontents_start_environment:w
\__scontents_start_after_option:w
\__scontents_check_line_process:xn
\__scontents_stop_environment:

```

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

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

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

In all cases, the function `__scontents_check_line_process:xn` checks that everything past `\begin{scontents}` is empty and then process the environment.

The function `__scontents_check_line_process:xn` calls the function `__scontents_file_tl_write_start:V` which will then read the contents of the environment and optionally store them in a token list or write to an external file.

When that’s done, the function `__scontents_file_write_stop:N` does the cleanup. This part of the code is inspired and adapted from the code of the package `xsimverb` by Clemens Niederberger.

```

423 \group_begin:
424   \char_set_catcode_active:N \^^I
425   \char_set_catcode_active:N \^^L
426   \char_set_catcode_active:N \^^M
427   \cs_new_protected:Npn \__scontents_normalise_line_ends:N #1
428     { \tl_replace_all:Nnn #1 { ^^M } { ~ } }
429   \cs_new_protected:Npn \__scontents_start_environment:w #1 ^^M
430     {
431       \tl_if_head_is_N_type:nTF {#1}
432         {
433           \str_if_eq:eeTF { \tl_head:n {#1} } { [ ]
434             { \__scontents_grab_optional:w #1 ^^M }
435             { \__scontents_check_line_process:xn { } {#1} }
436           }
437         { \__scontents_check_line_process:xn { } {#1} }
438       }
439   \cs_new_protected:Npn \__scontents_start_after_option:w #1 ^^M
440     { \__scontents_check_line_process:xn { [...] } {#1} }
441   \cs_new_protected:Npn \__scontents_check_line_process:xn #1 #2
442     {
443       \tl_if_blank:nF {#2}
444       {
445         \msg_error:nnxn { scontents } { junk-after-begin }
446         { after~\c_backslash_str begin { \l__scontents_env_name_tl } #1 } {#2}
447       }
448       \__scontents_make_control_chars_active:
449       \__scontents_file_tl_write_start:V \l__scontents_fname_out_tl
450     }
451   \cs_new_protected:Npn \__scontents_stop_environment:
452     {
453       \__scontents_file_write_stop:N \l__scontents_macro_tmp_tl
454       \bool_lazy_and:nnT
455         { \l__scontents_storing_bool }
456         { \tl_if_empty_p:N \l__scontents_macro_tmp_tl }
457         {
458           \msg_warning:nnx { scontents } { empty-environment } { \l__scontents_env_name_tl }
459         }
460     }

```

(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:

```

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_n\l:n` (which expects an active `^^J` token at the head of the token list; a low level TeX error is raised otherwise).

```

461 \cs_new_protected:Npn \__scontents_file_tl_write_start:n #1
462 {
463   \group_begin:
464     \__scontents_file_if_writable:nTF {#1}
465     {
466       \bool_set_true:N \__scontents_writable_bool
467       \iow_open:Nn \__scontents_file_iow {#1}
468     }
469     { \bool_set_false:N \__scontents_writable_bool }
470   \tl_clear:N \__scontents_every_line_env_tl
471   \seq_map_function:NN \l_char_special_seq \char_set_catcode_other:N
472   \int_step_function:nnnN { 128 } { 1 } { 255 } \char_set_catcode_letter:n
473   \cs_set_protected:Npx \__scontents_ret:w ##1 ^^M
474   {
475     \exp_not:N \__scontents_verb_processor_iterate:w
476     ###1 \c__scontents_end_env_tl
477     \c__scontents_end_env_tl
478     \exp_not:N \q__scontents_stop
479   }
480   \__scontents_make_control_chars_active:
481   \__scontents_ret:w
482 }
483 \cs_new:Npn \__scontents_setup_verb_processor:
484 {
485   \use:x
486   {
487     \cs_set:Npn \exp_not:N \__scontents_verb_processor_iterate:w
488     ###1 \c__scontents_end_env_tl
489     ###2 \c__scontents_end_env_tl
490     ###3 \exp_not:N \q__scontents_stop
491   } { \__scontents_verb_processor_iterate:nnn {##1} {##2} {##3} }
492 }
493 \cs_new:Npn \__scontents_verb_processor_iterate:nnn #1 #2 #3
494 {
495   \tl_if_blank:nTF {#3}
496   {
497     \__scontents_analyse_nesting:n {#1}
498     \__scontents_verb_processor_output:n {#1}
499   }
500   {
501     \__scontents_if_nested:TF
502     {
503       \__scontents_nesting_decr:
504       \__scontents_verb_processor_output:x
505       { \exp_not:n {#1} \c__scontents_end_env_tl \exp_not:n {#2} }
506     }
507     {
508       \tl_if_blank:nF {#1}
509       { \__scontents_verb_processor_output:n {#1} }
510       \cs_set_protected:Npx \__scontents_ret:w
511       {
512         \__scontents_env_end_function:
513         \bool_lazy_or:nnF
514         { \tl_if_blank_p:n {#2} }
515         { \str_if_eq_p:ee {#2} { \c_percent_str } }
516         {
517           \str_if_eq:VnF \c__scontents_hidden_space_str {#2}
518           {
519             \msg_warning:nnnn { scontents } { rescanning-text }
520             {#2} { \tl_use:N \l__scontents_env_name_tl }
521           }
522           \__scontents_rescan_tokens:n {#2}
523         }
524       }
525       \char_set_active_eq:NN ^^M \__scontents_ret:w

```

```

526     }
527   }
528   ^^M
529 }
530 \cs_new:Npn \__scontents_env_end_function:
531 {
532   \__scontents_format_case:nnn
533   { \exp_not:N \end { \if_false: } \fi: }
534   { \exp_after:wN \exp_not:N \cs:w end }
535   { \exp_after:wN \exp_not:N \cs:w stop }
536   \tl_use:N \__scontents_env_name_tl
537   \__scontents_format_case:nnn
538   { \if_false: { \fi: } }
539   { \cs_end: }
540   { \cs_end: }
541 }
542 \cs_new_protected:Npn \__scontents_file_write_stop:N #1
543 {
544   \bool_if:NT \l__scontents_writable_bool
545   { \iow_close:N \l__scontents_file_iow }
546   \use:x
547   {
548     \group_end:
549     \bool_if:NT \l__scontents_storing_bool
550     {
551       \tl_set:Nn \exp_not:N #1
552       { \exp_args:NV \__scontents_remove_leading_nl:n \l__scontents_every_line_env_tl }
553     }
554   }
555 }
556 \cs_new:Npn \__scontents_remove_leading_nl:n #1
557 {
558   \tl_if_head_is_N_type:nTF {#1}
559   {
560     \exp_args:Nf
561     \__scontents_remove_leading_nl:nn
562     { \tl_head:n {#1} } {#1}
563   }
564   { \exp_not:n {#1} }
565 }
566 \cs_new:Npn \__scontents_remove_leading_nl:nn #1 #2
567 {
568   \token_if_eq_meaning:NNTF ^^J #1
569   { \exp_not:o { \__scontents_remove_leading_nl:w #2 } }
570   { \exp_not:n {#2} }
571 }
572 \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:x`

The function `__scontents_verb_processor_output:n` does the output of each line read, to a token list and to a file, depending on the booleans `\l__scontents_writing_bool` and `\l__scontents_storing_bool`.

```

573 \cs_new_protected:Npn \__scontents_verb_processor_output:n #1
574 {
575   \bool_if:NT \l__scontents_writable_bool
576   { \iow_now:Nn \l__scontents_file_iow {#1} }
577   \bool_if:NT \l__scontents_storing_bool
578   { \tl_put_right:Nn \l__scontents_every_line_env_tl { ^^J #1 } }
579 }
580 \group_end:
581 \cs_generate_variant:Nn \__scontents_verb_processor_output:n { x }
582 \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:w`
`__scontents_nesting_decr:`
`__scontents_use_none_delimit_by_q_stop:w`
`__scontents_if_nested:TF`

`__scontents_analyse_nesting:n` scans nested `\begin{scontents}` and steps a `\l__scontents_env_nesting_int` counter. The `__scontents_if_nested:` conditional tests if we're in a nested environment, and `__scontents_nesting_decr:` reduces the nesting level, if an `\end{scontents}` is found.

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

```

583 \cs_new_protected:Npn \__scontents_analyse_nesting:n #1
584 {
585   \int_zero:N \l__scontents_tmpa_int
586   \__scontents_analyse_nesting_format:n {#1}
587   \int_compare:nNnT { \l__scontents_tmpa_int } > { 1 }
588     { \msg_warning:nn { scontents } { multiple-begin } }
589 }
590 \cs_new_protected:Npn \__scontents_nesting_incr:
591 {
592   \int_incr:N \l__scontents_env_nesting_int
593   \int_incr:N \l__scontents_tmpa_int
594 }
595 \cs_new_protected:Npn \__scontents_nesting_decr:
596 { \int_decr:N \l__scontents_env_nesting_int }
597 \prg_new_protected_conditional:Npnn \__scontents_if_nested: { TF }
598 {
599   \int_compare:nNnTF { \l__scontents_env_nesting_int } > { \c_zero_int }
600     { \prg_return_true: }
601     { \prg_return_false: }
602 }
603 \cs_new:Npn \__scontents_use_none_delimit_by_q_stop:w #1 \q__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.

```

604 \use:x
605 {
606   \cs_new_protected:Npn \exp_not:N \__scontents_analyse_nesting_latex:w ##1
607     \c_backslash_str begin \c_left_brace_str ##2 \c_right_brace_str
608   } {
609     \__scontents_tl_if_head_is_q_mark:nTF {#2}
610       { \__scontents_use_none_delimit_by_q_stop:w }
611       {
612         \str_if_eq:VnT \l__scontents_env_name_tl {#2}
613           { \__scontents_nesting_incr: }
614         \__scontents_analyse_nesting_latex:w
615       }
616   }
617 \cs_new_protected:Npx \__scontents_analyse_nesting_latex:n #1
618 {
619   \__scontents_analyse_nesting_latex:w #1
620   \c_backslash_str begin
621     \c_left_brace_str \exp_not:N \q__scontents_mark \c_right_brace_str
622   \exp_not:N \q__scontents_stop
623 }

```

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 `\startcontents`).

```

624 \cs_new_protected:Npn \__scontents_analyse_nesting_generic_process:nn #1 #2
625 {
626   \tl_if_head_is_N_type:nTF {#2}
627     {
628       \__scontents_tl_if_head_is_q_mark:nF {#2}
629       {
630         \__scontents_nesting_incr:
631         \__scontents_analyse_nesting_generic:w #2 \q__scontents_stop
632       }
633     }
634     { \__scontents_analyse_nesting_generic:w #2 \q__scontents_stop }
635 }
636 \cs_new_protected:Npn \__scontents_analyse_nesting_generic:nn #1 #2
637 {
638   \__scontents_define_generic_nesting_function:n {#1}
639   \use:x
640   {
641     \exp_not:N \__scontents_analyse_nesting_generic:w #2
642     \c_backslash_str #1 \tl_use:N \l__scontents_env_name_tl
643     \exp_not:N \q__scontents_mark \exp_not:N \q__scontents_stop
644   }

```

```

645 }
646 \cs_new_protected:Npn \__scontents_define_generic_nesting_function:n #1
647 {
648   \use:x
649   {
650     \cs_set_protected:Npn \exp_not:N \__scontents_analyse_nesting_generic:w ####1
651       \c_backslash_str #1 \tl_use:N \__scontents_env_name_tl
652       ####2 \exp_not:N \q__scontents_stop
653     } { \__scontents_analyse_nesting_generic_process:nn {##1} {##2} }
654   }
655 </core>
656 <*loader>
657 <latex>\cs_new_eq:NN \__scontents_analyse_nesting_format:n
658 <latex> \__scontents_analyse_nesting_latex:n
659 <!latex>\cs_new_protected:Npn \__scontents_analyse_nesting_format:n
660 <plain> { \__scontents_analyse_nesting_generic:nn { } }
661 <context> { \__scontents_analyse_nesting_generic:nn { start } }
662 </loader>
663 <*core>

```

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

12.10.6 Recording of the content in the sequence

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

```

664 \cs_new_protected:Npn \__scontents_finish_storing:NNN #1 #2 #3
665 {
666   \bool_if:NT \l__scontents_storing_bool
667   {
668     \bool_if:NF \l__scontents_forced_eol_bool
669     { \tl_put_right:Nx #1 { \c__scontents_hidden_space_str } }
670     \__scontents_store_to_seq:NN #1 #2
671     \bool_if:NT #3 { \__scontents_lastfrom_seq:V #2 }
672   }
673 }
674 </core>

```

(End definition for `__scontents_finish_storing:NNN`.)

12.11 The environment `verbatimsc`

In plain \TeX we emulate \LaTeX 's `verbatim` environment.

```

\verbatimsc
\endverbatimsc
\__scontents_verbatimsc_aux:
  \__scontents_vobeyspaces:
    \__scontents_xverb:
  \__scontents_nolig_list:
  \__scontents_xobeysp:
675 <*plain>
676 \cs_new_protected:Npn \verbatimsc
677 {
678   \group_begin:
679   \__scontents_verbatimsc_aux: \frenchspacing \__scontents_vobeyspaces:
680   \__scontents_xverb:
681 }
682 \cs_new_protected:Npn \endverbatimsc
683 { \group_end: }
684 \cs_new_protected:Npn \__scontents_verbatimsc_aux:
685 {
686   \skip_vertical:N \parskip
687   \dim_zero:N \parindent
688   \skip_set:Nn \parfillskip { 0pt plus 1fil }
689   \skip_set:Nn \parskip { 0pt plus 0pt minus 0pt }
690   \tex_par:D
691   \bool_set_false:N \l__scontents_temp_bool
692   \cs_set:Npn \par
693   {
694     \bool_if:NTF \l__scontents_temp_bool
695     {
696       \mode_leave_vertical:
697       \null
698       \tex_par:D
699       \penalty \interlinepenalty
700     }
701     {

```

```

702         \bool_set_true:N \__scontents_temp_bool
703         \mode_if_horizontal:T
704         { \tex_par:D \penalty \interlinepenalty }
705     }
706 }
707 \cs_set_eq:NN \do \char_set_catcode_other:N
708 \dospecials \obeylines
709 \tl_use:N \__scontents_verb_font_tl
710 \cs_set_eq:NN \do \__scontents_do_noligs:N
711 \__scontents_nolig_list:
712 \tex_everypar:D \exp_after:wN
713   { \tex_the:D \tex_everypar:D \tex_unpenalty:D }
714 }
715 \cs_new_protected:Npn \__scontents_nolig_list:
716   { \do\` \do\< \do\> \do\, \do\' \do\ - }
717 \cs_new_protected:Npn \__scontents_vobeyspaces:
718   { \__scontents_set_active_eq:NN \ \__scontents_xobeysp: }
719 \cs_new_protected:Npn \__scontents_xobeysp:
720   { \mode_leave_vertical: \nobreak \ }
721 </plain>

```

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

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

```

722 <!*latex>
723 \cs_if_exist:NF \dospecials
724   {
725     \cs_new:Npn \dospecials
726       { \seq_map_function:NN \l_char_special_seq \do }
727   }
728 </!*latex>

```

(End definition for `\dospecials`.)

12.12 The command `\Scontents`

User command to *stored content*, adapted from code by Ulrich Diez in `Stringify input - \string` on token list and code by user `siracusa` in `Convert a macro from Latexze to expl3`

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

`__scontents_esphack:`

```

729 <*core>
730 \cs_new_protected:Npn \__scontents_bsphack:
731   {
732     \scan_stop:
733     \mode_if_horizontal:T
734     {
735       \skip_set_eq:NN \__scontents_save_skip \tex_lastskip:D
736       \int_set_eq:NN \__scontents_save_sf_int \tex_spacefactor:D
737     }
738   }
739 \cs_new_protected:Npn \__scontents_esphack:
740   {
741     \scan_stop:
742     \mode_if_horizontal:T
743     {
744       \int_set_eq:NN \tex_spacefactor:D \__scontents_save_sf_int
745       \dim_compare:nNnT { \__scontents_save_skip } > { \c_zero_skip }
746       {
747         \skip_if_eq:nnT { \tex_lastskip:D } { \c_zero_skip }
748         {
749           \nobreak
750           \skip_horizontal:n { \c_zero_skip }
751         }
752         \tex_ignorespaces:D
753       }
754     }
755   }
756 </core>

```

```

757 <*/latex>
758 \cs_gset_eq:NN \__scontents_bsphack: \@bsphack
759 \cs_gset_eq:NN \__scontents_esphack: \@esphack
760 </latex>

```

(End definition for `__scontents_bsphack:` and `__scontents_esphack:`.)

`\Scontents` The `\Scontents` command starts by parsing an optional argument to the function `__scontents_`
`__scontents_Scontents_internal:nn` `Scontents_internal:nn` then delegates to `__scontents_verb_arg:w` or `__scontents_norm_arg:n`
`__scontents_norm_arg:n` depending whether a star (*) argument is present.
`__scontents_verb_arg:w`

```

761 <*/loader>
762 \NewDocumentCommand \Scontents { !s !O{} }
763 { \__scontents_Scontents_internal:nn {#1} {#2} }
764 </loader>
765 <*/core>
766 \cs_new_protected:Npn \__scontents_Scontents_internal:nn #1 #2
767 {
768   \__scontents_bsphack:
769   \group_begin:
770     \tl_if_novalue:nF {#2}
771     { \keys_set:nn { scontents / Scontents } {#2} }
772     \char_set_catcode_active:n { 9 }
773     \bool_if:NTF #1
774     { \__scontents_verb_arg:w }
775     { \__scontents_norm_arg:n }
776 }

```

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

```

777 \cs_new_protected:Npn \__scontents_norm_arg:n #1
778 {
779   \tl_set:Nn \__scontents_temp_tl {#1}
780   \__scontents_Scontents_finish:
781 }

```

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

```

782 </core>
783 <*/loader>
784 \NewDocumentCommand \__scontents_verb_arg:w { +v }
785 { \__scontents_verb_arg_internal:n {#1} }
786 </loader>
787 <*/core>

```

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

`__scontents_verb_arg_internal:n` The function `__scontents_verb_arg_internal:n` replace all `\^^M` by `\^^J` then adds it to the seq
`__scontents_Scontents_finish:` variable.
`__scontents_file_write_cmd:nn`
`__scontents_file_write_cmd:VV`

```

788 \cs_new_protected:Npn \__scontents_verb_arg_internal:n #1
789 {
790   \tl_set:Nn \__scontents_temp_tl {#1}
791   \tl_replace_all:Nxx \__scontents_temp_tl { \iow_char:N \^^M } { \iow_char:N \^^J }
792   \__scontents_Scontents_finish:
793 }
794 \cs_new_protected:Npn \__scontents_Scontents_finish:
795 {
796   \__scontents_file_write_cmd:VV \__scontents_fname_out_tl \__scontents_temp_tl
797   \__scontents_finish_storing:NNN
798   \__scontents_temp_tl
799   \__scontents_name_seq_cmd_tl
800   \__scontents_print_cmd_bool
801   \use:x
802   {
803     \group_end:
804     \bool_if:NF \__scontents_print_cmd_bool { \__scontents_esphack: }
805   }
806 }
807 \cs_new_protected:Npn \__scontents_file_write_cmd:nn #1#2
808 {

```



```

809   \__scontents_file_if_writable:nT {#1}
810   {
811     \iow_open:Nn \__scontents_file_iow {#1}
812     \iow_now:Nn \__scontents_file_iow {#2}
813     \iow_close:N \__scontents_file_iow
814   }
815 }
816 \cs_generate_variant:Nn \__scontents_file_write_cmd:nn { VV }
817 \prg_new_protected_conditional:Npnn \__scontents_file_if_writable:n #1 { T, F, TF }
818 {
819   \bool_if:NTF \__scontents_writing_bool
820   {
821     \file_if_exist:nTF {#1}
822     {
823       \bool_if:NTF \__scontents_overwrite_bool
824       {
825         \msg_warning:nnx { scontents } { overwrite-file } {#1}
826         \prg_return_true:
827       }
828       {
829         \msg_warning:nnx { scontents } { not-writing } {#1}
830         \prg_return_false:
831       }
832     }
833     {
834       \msg_warning:nnx { scontents } { writing-file } {#1}
835       \prg_return_true:
836     }
837   }
838   { \prg_return_false: }
839 }

```

(End definition for `__scontents_verb_arg_internal:n`, `__scontents_Scontents_finish:`, and `__scontents_file_write_cmd:nn`.)

12.13 The command `\getstored`

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

```

\__scontents_getstored_internal:nn
840 </core>
841 <*loader>
842 \NewDocumentCommand \getstored { 0{-1} m }
843 { \__scontents_getstored_internal:nn {#1} {#2} }
844 </loader>
845 <*core>
846 \cs_new_protected:Npn \__scontents_getstored_internal:nn #1 #2
847 {
848   \group_begin:
849   \int_set:Nn \tex_newlinechar:D { \^^J }
850   \__scontents_rescan_tokens:x
851   {
852     \endgroup % This assumes \catcode\=0... Things might go off otherwise.
853     \__scontents_getfrom_seq:nn {#1} {#2}
854   }
855 }

```

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

12.14 The command `\foreachsc`

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

```

\__scontents_foreachsc_internal:nn
\__scontents_foreach_add_body:n
856 </core>
857 <*loader>
858 \NewDocumentCommand \foreachsc { o m }
859 { \__scontents_foreachsc_internal:nn {#1} {#2} }
860 </loader>
861 <*core>
862 \cs_new_protected:Npn \__scontents_foreachsc_internal:nn #1 #2
863 {
864   \group_begin:
865   \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / foreachsc } {#1} }

```

```

866     \tl_set:Nn \l__scontents_foreach_name_seq_tl {#2}
867     \seq_clear:N \l__scontents_foreach_print_seq
868     \bool_if:NF \l__scontents_foreach_stop_bool
869     {
870         \int_set:Nn \l__scontents_foreach_stop_int
871         { \seq_count:c { g__scontents_name_#2_seq } }
872     }
873     \int_step_function:nnnN
874     { \l__scontents_foreach_start_int }
875     { \l__scontents_foreach_step_int }
876     { \l__scontents_foreach_stop_int }
877     \__scontents_foreach_add_body:n
878     \tl_gset:Nx \g__scontents_temp_tl
879     {
880         \exp_args:NNV \seq_use:Nn
881         \l__scontents_foreach_print_seq \l__scontents_foreach_sep_tl
882     }
883     \group_end:
884     \exp_after:wN \tl_gclear:N
885     \exp_after:wN \g__scontents_temp_tl
886     \g__scontents_temp_tl
887 }
888 \cs_new_protected:Npn \l__scontents_foreach_add_body:n #1
889 {
890     \seq_put_right:Nx \l__scontents_foreach_print_seq
891     {
892         \bool_if:NT \l__scontents_foreach_before_bool
893         { \exp_not:V \l__scontents_foreach_before_tl }
894         \bool_if:NTF \l__scontents_foreach_wrapper_bool
895         { \__scontents_foreach_wrapper:n }
896         { \use:n }
897         { \getstored [#1] { \tl_use:N \l__scontents_foreach_name_seq_tl } }
898         \bool_if:NT \l__scontents_foreach_after_bool
899         { \exp_not:V \l__scontents_foreach_after_tl }
900     }
901 }

```

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

12.15 The command `\typestored`

```

\typestored
\__scontents_tpestored_internal:nn
\__scontents_verb_print:N
\__scontents_xverb:w

```

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

```

902 </core>
903 <*loader>
904 \NewDocumentCommand \typestored { o m }
905 { \__scontents_tpestored_internal:nn {#1} {#2} }
906 </loader>
907 <*core>
908 \cs_new_protected:Npn \__scontents_tpestored_internal:nn #1 #2
909 {
910     \group_begin:
911     \int_set:Nn \l__scontents_seq_item_int { 1 }
912     \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / typemeaning } {#1} }
913     \tl_set:Nx \l__scontents_temp_tl
914     { \exp_args:NV \__scontents_getfrom_seq:nn \l__scontents_seq_item_int {#2} }
915     \tl_remove_once:NV \l__scontents_temp_tl \c__scontents_hidden_space_str
916     \tl_log:N \l__scontents_temp_tl
917     \tl_if_empty:NF \l__scontents_temp_tl
918     { \__scontents_verb_print:N \l__scontents_temp_tl }
919     \group_end:
920 }

```

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`.

```

921 \group_begin:
922 \char_set_catcode_active:N ^^M
923 \cs_new_protected:Npn \__scontents_verb_print:N #1

```

```

924 {
925   \tl_if_blank:VT #1
926   { \msg_error:nnn { scontents } { empty-variable } {#1} }
927   \cs_set_eq:NN \__scontents_verb_print_EOL: ^^M
928   \cs_set_eq:NN ^^M \scan_stop:
929   \cs_set_eq:cN { do@noligs } \__scontents_do_noligs:N
930   \int_set:Nn \tex_newlinechar:D { \^^J }
931   \__scontents_rescan_tokens:x
932   {
933     \__scontents_format_case:nnn
934     { \exp_not:N \begin{verbatim} } % LaTeX
935     { \verbatim } % Plain/Generic
936     { \startverbatim } % ConTeXt
937     ^^M
938     \exp_not:V #1 ^^M
939     \g__scontents_end_verbatim_sc_tl
940   }
941   \cs_set_eq:NN ^^M \__scontents_verb_print_EOL:
942 }
943 \group_end:
944 \cs_new_protected:Npn \__scontents_xverb:
945 {
946   \char_set_catcode_active:n { 9 }
947   \char_set_active_eq:nN { 9 } \__scontents_tabs_to_spaces:
948   \__scontents_xverb:w
949 }
950 \</core>

```

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

`verbatimsc` Finally the \LaTeX and \ConTeXt version of `verbatimsc` environment is defined.

`\startverbatimsc` The macro `\endverbatim` in the second argument of the `verbatimsc` environment is only needed for

`\stopverbatimsc` compatibility with the `verbatim` package.

```

951 \<loader>
952 \<!context>
953 \use:x
954 {
955   \cs_new_protected:Npn \exp_not:N \__scontents_xverb:w
956   ##1 \g__scontents_end_verbatim_sc_tl
957   \<latex> { ##1 \exp_not:N \end{verbatim} }
958   \<plain> { ##1 \exp_not:N \endverbatim }
959   \<context> { ##1 \exp_not:N \stopverbatim }
960 }
961 \</!context>
962 \<!latex>
963 \NewDocumentEnvironment { verbatimsc } { }
964 {
965   \cs_set_eq:cN { @xverbatim } \__scontents_xverb:
966   \verbatim
967 }
968 { \endverbatim }
969 \</latex>
970 \<context>\definetytyping[verbatimsc]
971 \</loader>
972 \<*core>

```

(End definition for `verbatimsc`, `\startverbatimsc`, and `\stopverbatimsc`. These functions are documented on page 6.)

12.15.1 Some auxiliaries functions

`__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.

```

973 \cs_new:Npn \__scontents_tabs_to_spaces:
974 { \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 $\LaTeX 2_{\epsilon}$'s `\do@noligs` which makes sure to not consume following space tokens. The $\LaTeX 2_{\epsilon}$ version ends with `\char`#1`, which leaves \TeX still looking for an *(optional space)*. This version uses `\char_generate:nn` to ensure that doesn't happen.

```

975 \cs_new:Npn \__scontents_do_noligs:N #1
976 {
977   \char_set_catcode_active:N #1
978   \char_set_active_eq:Nc #1 { __scontents_active_char_ \token_to_str:N #1 : }
979   \cs_set:cpx { __scontents_active_char_ \token_to_str:N #1 : }
980     {
981       \mode_leave_vertical:
982       \tex_kern:D \c_zero_dim
983       \char_generate:nn { `#1 } { 12 }
984     }
985 }

```

(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`.

```

986 \prg_new_protected_conditional:Npnn \__scontents_tl_if_head_is_q_mark:n #1
987 { T, F, TF }
988 {
989   \if_meaning:w \q__scontents_mark #1 \scan_stop:
990     \prg_return_true:
991   \else:
992     \prg_return_false:
993   \fi:
994 }

```

(End definition for `__scontents_tl_if_head_is_q_mark:nTF`.)

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

```

995 \cs_new_protected:Npn \__scontents_set_active_eq:NN #1
996 {
997   \char_set_catcode_active:N #1
998   \char_set_active_eq:NN #1
999 }
1000 </core>
1001 <*loader>
1002 \group_begin:
1003 <plain> \char_set_catcode_active:n { \* }
1004 \cs_new_protected:Npn \__scontents_plain_disable_outer_par:
1005 <*plain>
1006 {
1007   \group_begin:
1008   \char_set_lccode:nn { \* } { \^^L }
1009   \tex_lowercase:D { \group_end:
1010     \tex_let:D * \scan_stop:
1011   }
1012 }
1013 </plain>
1014 <latex | context> { }
1015 \group_end:
1016 </loader>
1017 <*core>
1018 \group_begin:
1019 \char_set_catcode_active:N \*
1020 \cs_new_protected:Npn \__scontents_make_control_chars_active:
1021 {
1022   \__scontents_plain_disable_outer_par:
1023   \__scontents_set_active_eq:NN \^^I \__scontents_tab:
1024   \__scontents_set_active_eq:NN \^^L \__scontents_par:
1025   \__scontents_set_active_eq:NN \^^M \__scontents_ret:w
1026 }
1027 \group_end:

```

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

12.16 The command `\setupsc`

User command `\setupsc` to setup module.

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

```

1028 </core>
1029 <*loader>
1030 \NewDocumentCommand \setupsc { +m }
1031 { \keys_set:nn { scontents } {#1} }
1032 </loader>
1033 <*core>

```

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

12.17 The command `\meaningsc`

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

`__scontents_meaningsc_internal:nn`
`__scontents_meaningsc:n`

```

1034 </core>
1035 <*loader>
1036 \NewDocumentCommand \meaningsc { o m }
1037 { __scontents_meaningsc_internal:nn {#1} {#2} }
1038 </loader>
1039 <*core>
1040 \cs_new_protected:Npn __scontents_meaningsc_internal:nn #1 #2
1041 {
1042   \group_begin:
1043     \int_set:Nn \l__scontents_seq_item_int { 1 }
1044     \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / typemeaning } {#1} }
1045     __scontents_meaningsc:n {#2}
1046   \group_end:
1047 }
1048 \group_begin:
1049 \char_set_catcode_active:N ^^I
1050 \cs_new_protected:Npn __scontents_meaningsc:n #1
1051 {
1052   \tl_set:Nx \l__scontents_temp_tl
1053   { \exp_args:NV __scontents_getfrom_seq:nn \l__scontents_seq_item_int {#1} }
1054   \tl_replace_all:Nxn \l__scontents_temp_tl { \iow_char:N ^^J } { ~ }
1055   \tl_remove_once:NV \l__scontents_temp_tl \c__scontents_hidden_space_str
1056   \tl_log:N \l__scontents_temp_tl
1057   \tl_use:N \l__scontents_verb_font_tl
1058   \tl_replace_all:Nnx \l__scontents_temp_tl { ^^I } { __scontents_tabs_to_spaces: }
1059   \cs_replacement_spec:N \l__scontents_temp_tl
1060 }
1061 \group_end:

```

(End definition for `\meaningsc`, `__scontents_meaningsc_internal:nn`, and `__scontents_meaningsc:n`. This function is documented on page 7.)

12.18 The command `\countsc`

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

```

1062 </core>
1063 <*loader>
1064 \NewExpandableDocumentCommand \countsc { m }
1065 { \seq_count:c { g__scontents_name_#1_seq } }
1066 </loader>
1067 <*core>

```

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

12.19 The command `\cleanseqsc`

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

```

1068 </core>
1069 <*loader>
1070 \NewDocumentCommand \cleanseqsc { m }
1071 { \seq_clear_new:c { g__scontents_name_#1_seq } }
1072 </loader>
1073 <*core>

```

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

12.20 Warning and error messages

Warning and error messages used throughout the package.

```

1074 \msg_new:nnn { scontent } { junk-after-begin }
1075 {
1076   Junk~characters~#1~\msg_line_context: :
1077   \\ \\
1078   #2
1079 }
1080 \msg_new:nnnn { scontent } { env-already-defined }
1081 { Environment~'#1'~already~defined! }
1082 {
1083   You~have~used~\newenvsc
1084   with~an~environment~that~already~has~a~definition. \\ \\
1085   The~existing~definition~of~'#1'~will~not~be~altered.
1086 }
1087 \msg_new:nnn { scontent } { empty-stored-content }
1088 { Empty~value~for~key~'getstored'~\msg_line_context:. }
1089 \msg_new:nnn { scontent } { empty-variable }
1090 { Variable~'#1'~empty~\msg_line_context:. }
1091 \msg_new:nnn { scontent } { overwrite-file }
1092 { Overwriting~file~'#1'. }
1093 \msg_new:nnn { scontent } { writing-file }
1094 { Writing~file~'#1'. }
1095 \msg_new:nnn { scontent } { not-writing }
1096 { File~'#1'~already~exists.~Not~writing. }
1097 \msg_new:nnn { scontent } { rescanning-text }
1098 { Rescanning~text~'#1'~after~\c_backslash_str end{#2}~\msg_line_context:. }
1099 \msg_new:nnn { scontent } { multiple-begin }
1100 { Multiple~\c_backslash_str begin{ \l__scontent_env_name_tl }~\msg_line_context:. }
1101 \msg_new:nnn { scontent } { undefined-storage }
1102 { Storage~named~'#1'~is~not~defined. }
1103 \msg_new:nnn { scontent } { index-out-of-range }
1104 {
1105   \int_compare:nNnTF {#1} = { 0 }
1106   { Index~of~sequence~cannot~be~zero. }
1107   {
1108     Index~'#1'~out~of~range~for~'#2'.~
1109     \int_compare:nNnTF {#1} > { 0 }
1110     { Max = } { Min = - } #3.
1111   }
1112 }
1113 \msg_new:nnnn { scontent } { env-key-unknown }
1114 {
1115   The~key~'#1'~is~unknown~by~environment~
1116   '\l__scontent_env_name_tl'~and~is~being~ignored.
1117 }
1118 {
1119   The~environment~'\l__scontent_env_name_tl'~does~not~have~a~key~called~'#1'.~\l
1120   Check~that~you~have~spelled~the~key~name~correctly.
1121 }
1122 \msg_new:nnnn { scontent } { env-key-value-unknown }
1123 {
1124   The~key~'#1=#2'~is~unknown~by~environment~
1125   '\l__scontent_env_name_tl'~and~is~being~ignored.
1126 }
1127 {
1128   The~environment~'\l__scontent_env_name_tl'~does~not~have~a~key~called~'#1'.~\l
1129   Check~that~you~have~spelled~the~key~name~correctly.
1130 }
1131 \msg_new:nnnn { scontent } { cmd-key-unknown }
1132 { The~key~'#1'~is~unknown~by~'\c_backslash_str Scontent'~and~is~being~ignored. }
1133 {
1134   The~command~'\c_backslash_str Scontent'~does~not~have~a~key~called~'#1'.~\l
1135   Check~that~you~have~spelled~the~key~name~correctly.
1136 }
1137 \msg_new:nnnn { scontent } { cmd-key-value-unknown }
1138 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str Scontent'~and~is~being~ignored. }
1139 {

```

```

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

```

12.21 Finish package

Finish package implementation.

```

1188 </core>
1189 <plain | context>\ExplSyntaxOff

```

13 Index of Implementation

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

Symbols	
\'	716
*	1003, 1008, 1019
\,	716
\-	716
\<	716
\>	716
\\	34, 852, 1077, 1084, 1119, 1128, 1134, 1140, 1146, 1152, 1158, 1164, 1176, 1185
\`	716
B	
\beginngroup	60, 65
bool commands:	
\bool_if:NTF	544, 549, 575, 577, 666, 668, 671, 694, 773, 804, 819, 823, 868, 892, 894, 898
\bool_lazy_and:nnTF	454
\bool_lazy_or:nnTF	332, 513
\bool_new:N	106, 160, 162, 164, 165, 167, 169, 171
\bool_set_false:N	161, 166, 168, 170, 172, 195, 217, 469, 691
\bool_set_true:N	163, 191, 196, 213, 218, 235, 240, 248, 256, 466, 702
C	
\catcode	61, 852
char commands:	
\char_generate:nn	36, 983
\char_set_active_eq:NN	525, 978, 998
\char_set_active_eq:nN	947
\char_set_catcode:nn	112
\char_set_catcode_active:N	395, 424, 425, 426, 922, 977, 997, 1019, 1049
\char_set_catcode_active:n	772, 946, 1003
\char_set_catcode_letter:N	110
\char_set_catcode_letter:n	472
\char_set_catcode_other:N	34, 36, 37, 471, 707
\char_set_lccode:nn	1008
\l_char_special_seq	31, 471, 726
\char_value_catcode:n	109
\cleanseqsc	7, 37, <u>1068</u>
\countsc	7, 37, <u>1062</u>
cs commands:	
\cs:w	365, 366, 534, 535
\cs_end:	365, 366, 539, 540
\cs_generate_variant:Nn	181, 184, 185, 186, 319, 346, 581, 582, 816
\cs_gset_eq:NN	758, 759
\cs_if_exist:NTF	24, 372, 373, 723
\cs_new:Npn	54, 183, 320, 330, 483, 493, 530, 556, 566, 572, 603, 725, 973, 975
\cs_new:Npx	182
\cs_new_eq:NN	657
\cs_new_protected:Npn	180, 271, 273, 279, 281, 287, 293, 295, 297, 311, 338, 347, 352, 354, 361, 377, 380, 385, 393, 398, 413, 427, 429, 439, 441, 451, 461, 542, 573, 583, 590, 595, 606, 624, 636, 646, 659, 664, 676, 682, 684, 715, 717, 719, 730, 739, 766, 777, 788, 794, 807, 846, 862, 888, 908, 923, 944, 955, 995, 1004, 1020, 1040, 1050
\cs_new_protected:Npx	617
\cs_replacement_spec:N	1059
\cs_set:Npn	487, 692
\cs_set:Npx	979
\cs_set_eq:NN	707, 710, 927, 928, 929, 941, 965
\cs_set_protected:Npn	257, 650
\cs_set_protected:Npx	473, 510
\csname	64, 74
D	
\def	2, 3, 5, 6, 63, 66, 67, 75, 88
\definetying	970
dim commands:	
\dim_compare:nNnTF	745
\dim_zero:N	687
\c_zero_dim	982
\do	707, 710, 716, 726
\dospecials	708, <u>722</u>
E	
\else	85, 87
else commands:	
\else:	991
\end	40, 533, 957
\endcsname	64, 74
\endgroup	63, 66, 69, 82, 96, 852
\endinput	83, 97
\endlinechar	62
\endscontents	4, <u>404</u>
\endverbatim	968
\endverbatimsc	41, <u>675</u> , 958
Environments	
scontents	24, 25
\errhelp	70
\errmessage	71
exp commands:	
\exp_after:wN	534, 535, 712, 884, 885
\exp_args:Nf	324, 560
\exp_args:NNV	880
\exp_args:Nooo	363
\exp_args:NV	272, 280, 294, 296, 552, 914, 1053
\exp_not:N	50, 475, 478, 487, 490, 533, 534, 535, 551, 606, 621, 622, 641, 643, 650, 652, 934, 955, 957, 958, 959
\exp_not:n	350, 505, 564, 569, 570, 893, 899, 938
\expandafter	64, 74
\ExplSyntaxOff	27, 1189
\ExplSyntaxOn	16
F	
\fi	73, 99, 100
fi commands:	
\fi:	533, 538, 993
file commands:	
\file_if_exist:NTF	821
\file_input:n	53, 111
\file_input_stop:	28
\foreachsc	6, 21, 22, 33, <u>856</u>
\frenchspacing	679
G	
\getstored	6, 33, <u>840</u> , 897

group commands:

`\group_begin:` . 382, 423, 463, 678, 769, 848, 864, 910, 921, 1002, 1007, 1018, 1042, 1048
`\group_end:` 388, 548, 580, 683, 803, 883, 919, 943, 1009, 1015, 1027, 1046, 1061
`\group_insert_after:N` 341, 342, 343, 344

I

if commands:

`\if_false:` 533, 538
`\if_meaning:w` 989
`\ifx` 64, 74, 86
`\input` 15

int commands:

`\int_abs:n` 334
`\int_compare:nNnTF` 587, 599, 1105, 1109
`\int_compare_p:nNn` 333, 334
`\int_decr:N` 596
`\int_incr:N` 592, 593
`\int_new:N` 105, 157, 158, 159, 178
`\int_set:Nn` 109, 249, 302, 849, 870, 911, 930, 1043
`\int_set_eq:NN` 736, 744
`\int_step_function:nnnN` 472, 873
`\int_to_roman:n` 23, 299
`\int_zero:N` 585
`\c_zero_int` 599
`\interlinepenalty` 699, 704

iow commands:

`\iow_char:N` 791, 1054
`\iow_close:N` 545, 813
`\iow_log:n` 19
`\iow_new:N` 153
`\iow_now:Nn` 576, 812
`\iow_open:Nn` 467, 811

K

keys commands:

`\keys_define:nn` 115, 146, 188, 210, 232, 266
`\l_keys_key_str` 22, 272, 280, 294, 296
`\keys_set:nn` 37, 357, 419, 771, 865, 912, 1031, 1044

M

`\meaningsc` 7, 22, 23, 37, 1034

mode commands:

`\mode_if_horizontal:TF` 703, 733, 742
`\mode_leave_vertical:` 696, 720, 981

msg commands:

`\msg_error:nn` 314
`\msg_error:nnn` 276, 284, 290, 307, 374, 926
`\msg_error:nnnn` 186, 277, 285, 291, 303, 308, 445
`\msg_expandable_error:nnn` 328
`\msg_expandable_error:nnnn` 335
`\msg_gset:nnn` 22
`\msg_line_context:` 23, 1076, 1088, 1090, 1098, 1100, 1168
`\msg_new:nnn` 1074, 1087, 1089, 1091, 1093, 1095, 1097, 1099, 1101, 1103, 1167
`\msg_new:nnnn` 1080, 1113, 1122, 1131, 1137, 1143, 1149, 1155, 1161, 1169, 1179
`\msg_warning:nn` 26, 588
`\msg_warning:nnn` 458, 825, 829, 834
`\msg_warning:nnnn` 519

N

`\NewDocumentCommand` 370, 409, 762, 784, 842, 858, 904, 1030, 1036, 1070

`\NewDocumentEnvironment` 379, 963
`\newenvsc` 5, 17, 24, 352, 405, 1083
`\NewExpandableDocumentCommand` 1064
`\next` 63, 66, 75, 88, 101
`\nobreak` 720, 749
`\null` 697

O

`\obeylines` 708

P

`\PackageError` 67, 77, 90
 Packages

`l3keys2e` 19
`scontents` 16, 17, 20
`xparse` 25

`\par` 692
`\parfillskip` 688
`\parindent` 687
`\parskip` 686, 689
`\penalty` 699, 704

prg commands:

`\prg_generate_conditional_variant:Nnn` 187
`\prg_new_protected_conditional:Npnn` 597, 817, 986
`\prg_replicate:nn` 974
`\prg_return_false:` 601, 830, 838, 992
`\prg_return_true:` 600, 826, 835, 990

`\ProcessKeysOptions` 149
`\ProvidesExplPackage` 9

Q

quark commands:

`\quark_new:N` 176, 177

quark internal commands:

`\q_scontents_mark` 36, 176, 621, 643, 989
`\q_scontents_stop` 176, 478, 490, 603, 622, 631, 634, 643, 652

R

`\relax` 64, 74
`\RequirePackage` 8

S

scan commands:

`\scan_stop:` 732, 741, 928, 989, 1010
`\Scontents` 5, 21, 22, 31, 761
`\scontents` 4, 404
`scontents` 4, 404

scontents internal commands:

`__scontents_analyse_nesting:n` 28, 497, 583
`__scontents_analyse_nesting:w` 583
`__scontents_analyse_nesting_format:n` 586, 657, 659
`__scontents_analyse_nesting_generic:nn` 636, 660, 661
`__scontents_analyse_nesting_generic:w` 631, 634, 641, 650
`__scontents_analyse_nesting_generic_process:nn` 624, 653
`__scontents_analyse_nesting_latex:n` 617, 658
`__scontents_analyse_nesting_latex:w` 606, 614, 619
`__scontents_append_contents:nn` 23, 311, 350
`__scontents_bsphack:` 20, 729, 768
`__scontents_check_line_process:nn` 26, 423

_scontents_define_generic_nesting_
 function:n 638, 646
 _scontents_do_noligs:N 36, 710, 929, 975
 \c_scontents_end_env_tl 17, 43, 476, 477, 488, 489,
 505
 \g_scontents_end_verbatimsc_tl 16, 31, 939, 956
 _scontents_env_define:nnn 24, 352
 _scontents_env_end_function: 512, 530
 _scontents_env_generic_begin: 24, 25, 359, 393
 _scontents_env_generic_end: .. 24, 25, 362, 393
 \l_scontents_env_name_tl 17, 24, 43, 356, 446, 458,
 520, 536, 612, 642, 651, 1100, 1116, 1119, 1125, 1128
 \l_scontents_env_nesting_int .. 19, 28, 157, 592,
 596, 599
 _scontents_env_setting:nn 24, 352
 _scontents_esphack: 20, 729, 804
 \l_scontents_every_line_env_tl 19, 151, 470, 552,
 578
 _scontents_file_if_writable:n 817
 _scontents_file_if_writable:nTF ... 464, 809
 \l_scontents_file_iow 19, 151, 467, 545, 576, 811,
 812, 813
 _scontents_file_tl_write_start:n . 19, 26, 449,
 461, 582
 _scontents_file_write_cmd:nn 788
 _scontents_file_write_stop:N ... 26, 453, 461
 _scontents_finish_storing:NNN . 401, 664, 797
 \l_scontents_fname_out_tl . 19, 151, 192, 197, 214,
 219, 449, 796
 \l_scontents_forced_eol_bool 131, 668
 _scontents_foreach_add_body:n 856
 \l_scontents_foreach_after_bool . 165, 240, 898
 \l_scontents_foreach_after_tl 19, 154, 241, 899
 \l_scontents_foreach_before_bool 165, 235, 892
 \l_scontents_foreach_before_tl 19, 154, 236, 893
 \l_scontents_foreach_name_seq_tl . 19, 154, 866,
 897
 \l_scontents_foreach_print_seq 20, 173, 867, 881,
 890
 \l_scontents_foreach_sep_tl 261, 881
 \l_scontents_foreach_start_int 244, 874
 \l_scontents_foreach_step_int 252, 875
 \l_scontents_foreach_stop_bool . 165, 248, 868
 \l_scontents_foreach_stop_int 19, 157, 249, 870,
 876
 _scontents_foreach_wrapper:n 258, 895
 \l_scontents_foreach_wrapper_bool 165, 256, 894
 _scontents_foreachsc_internal:nn 856
 _scontents_format_case:nnn .. 54, 532, 537, 933
 _scontents_getfrom_seq:nn 23, 320, 853, 914, 1053
 _scontents_getfrom_seq:nnn 320
 _scontents_getstored_internal:nn 840
 _scontents_grab_optional:n 407
 _scontents_grab_optional:w ... 25, 26, 407, 434
 \c_scontents_hidden_space_str 20, 174, 517, 669,
 915, 1055
 _scontents_if_nested: 28
 _scontents_if_nested:TF 501, 583
 _scontents_lastfrom_seq:n 24, 338, 671
 \l_scontents_macro_tmp_tl . 18, 102, 401, 453, 456
 _scontents_make_control_chars_active: . 448,
 480, 995
 _scontents_meaningsc:n 1034
 _scontents_meaningsc_internal:nn 1034
 \l_scontents_name_seq_cmd_tl 120, 799
 \l_scontents_name_seq_env_tl 117, 402
 _scontents_nesting_decr: 28, 503, 583
 _scontents_nesting_incr: 590, 613, 630
 _scontents_nolig_list: 675
 _scontents_norm_arg:n 32, 761
 _scontents_normalise_line_ends:N 25, 418, 427
 \l_scontents_overwrite_bool 134, 823
 _scontents_par: 182, 1024
 _scontents_parse_command_keys:n . 22, 230, 279
 _scontents_parse_command_keys:nn 279
 _scontents_parse_environment_keys:n 22, 208,
 271
 _scontents_parse_environment_keys:nn .. 271
 _scontents_parse_foreach_keys:n . 22, 264, 287
 _scontents_parse_foreach_keys:nn 287
 _scontents_parse_type_meaning_key:n 269, 295
 _scontents_parse_type_meaning_key:nn .. 295
 _scontents_parse_typemeaning_key:n 23
 _scontents_plain_disable_outer_par: .. 995
 \l_scontents_print_cmd_bool .. 24, 128, 800, 804
 \l_scontents_print_env_bool 24, 125, 402
 _scontents_remove_leading_nl:n 27, 461
 _scontents_remove_leading_nl:nn ... 561, 566
 _scontents_remove_leading_nl:w 461
 _scontents_rescan_tokens:n 34, 24, 180, 341, 522,
 850, 931
 _scontents_ret:w ... 26, 473, 481, 510, 525, 1025
 \l_scontents_save_sf_int 178, 736, 744
 \l_scontents_save_skip 178, 735, 745
 _scontents_Scontents_finish: 780, 788
 _scontents_Scontents_internal:nn ... 32, 761
 \l_scontents_seq_item_int . 19, 157, 302, 911, 914,
 1043, 1053
 _scontents_set_active_eq:NN 718, 995
 _scontents_setup_verb_processor: 24, 358, 461
 _scontents_start_after_option:w . 26, 421, 423
 _scontents_start_environment:w 396, 423
 _scontents_stop_environment: 400, 423
 _scontents_store_to_seq: 30
 _scontents_store_to_seq:NN 24, 347, 670
 \l_scontents_storing_bool . 19, 28, 160, 195, 217,
 455, 549, 577, 666
 _scontents_tab: 182, 1023
 \l_scontents_tab_width_int 137, 974
 _scontents_tabs_to_spaces: . 35, 947, 973, 1058
 \l_scontents_temp_bool ... 18, 102, 691, 694, 702
 \g_scontents_temp_tl . 18, 102, 340, 342, 344, 878,
 885, 886
 \l_scontents_temp_tl . 18, 102, 417, 418, 419, 779,
 790, 791, 796, 798, 913, 915, 916, 917, 918, 1052, 1054,
 1055, 1056, 1058, 1059
 _scontents_tl_if_head_is_q_mark:nTF 609, 628,
 986
 \l_scontents_tmpa_int 18, 102, 109, 112, 585, 587,
 593
 _scontents_tpestored_internal:nn 902
 _scontents_use_none_delimit_by_q_stop:w 583
 _scontents_verb_arg:w 32, 761
 _scontents_verb_arg_internal:n .. 32, 785, 788
 \l_scontents_verb_font_tl 123, 709, 1057
 _scontents_verb_print:N 34, 902

<code>_scontents_verb_print_EOL:</code>	927, 941
<code>_scontents_verb_processor_iterate:nnn</code>	461
<code>_scontents_verb_processor_iterate:w</code>	461
<code>_scontents_verb_processor_output:n</code>	28, 498, 504, 509, 573
<code>_scontents_verbatimmsc_aux:</code>	675
<code>_scontents_vobeyspaces:</code>	675
<code>\l_scontents_writable_bool</code>	160, 466, 469, 544, 575
<code>\l_scontents_writing_bool</code>	19, 28, 160, 191, 196, 213, 218, 819
<code>_scontents_xobeysp:</code>	675
<code>_scontents_xverb:</code>	675, 944, 965
<code>_scontents_xverb:w</code>	902, 955
<code>\Scontents*</code>	22
<code>\ScontentsCoreFileDate</code>	3, 86
<code>\ScontentsFileDate</code>	2, 10, 17, 86
<code>\ScontentsFileDescription</code>	6, 10, 18
<code>\ScontentsFileVersion</code>	5, 10, 13, 18
seq commands:	
<code>\seq_clear:N</code>	867
<code>\seq_clear_new:N</code>	1071
<code>\seq_count:N</code>	325, 871, 1065
<code>\seq_gput_right:Nn</code>	317
<code>\seq_if_exist:NTF</code>	315, 322
<code>\seq_item:Nn</code>	336, 340
<code>\seq_map_function:NN</code>	471, 726
<code>\seq_new:N</code>	173, 316
<code>\seq_put_right:Nn</code>	890
<code>\seq_use:Nn</code>	880
<code>\setupsc</code>	3, 37, 1028
skip commands:	
<code>\skip_horizontal:n</code>	750
<code>\skip_if_eq:nnTF</code>	747
<code>\skip_new:N</code>	179
<code>\skip_set:Nn</code>	688, 689
<code>\skip_set_eq:NN</code>	735
<code>\skip_vertical:N</code>	686
<code>\c_zero_skip</code>	745, 747, 750
<code>\space</code>	17, 18
<code>\startscontents</code>	4, 404
<code>\startverbatimmsc</code>	936, 951
<code>\stopscontents</code>	4, 404
<code>\stopverbatimmsc</code>	42, 951
str commands:	
<code>\c_backslash_str</code>	46, 446, 607, 620, 642, 651, 1098, 1100, 1132, 1134, 1138, 1140, 1144, 1146, 1150, 1152
<code>\c_circumflex_str</code>	175
<code>\c_left_brace_str</code>	49, 607, 621
<code>\c_percent_str</code>	175, 515
<code>\c_right_brace_str</code>	51, 607, 621
<code>\str_const:Nn</code>	174
<code>\str_if_eq:nnTF</code>	433, 517, 612
<code>\str_if_eq_p:nn</code>	515
T	
T _E X and L ^A T _E X 2 _ε commands:	
<code>\@</code>	109, 110, 112
<code>\@bsphack</code>	758
<code>\@esphack</code>	759
tex commands:	
<code>\tex_everypar:D</code>	712, 713
<code>\tex_ignorespaces:D</code>	752
<code>\tex_kern:D</code>	982
<code>\tex_lastskip:D</code>	735, 747
<code>\tex_let:D</code>	1010
<code>\tex_lowercase:D</code>	1009
<code>\tex_newlinechar:D</code>	849, 930
<code>\tex_par:D</code>	690, 698, 704
<code>\tex_scantokens:D</code>	20, 180
<code>\tex_spacefactor:D</code>	736, 744
<code>\tex_the:D</code>	713
<code>\tex_unpenalty:D</code>	713
tl commands:	
<code>\c_space_tl</code>	182
<code>\tl_clear:N</code>	470
<code>\tl_const:Nn</code>	44
<code>\tl_gclear:N</code>	343, 884
<code>\tl_gset:Nn</code>	17, 340, 878
<code>\tl_gset_rescan:Nnn</code>	32
<code>\tl_head:n</code>	433, 562
<code>\tl_if_blank:nTF</code>	275, 283, 289, 301, 306, 313, 443, 495, 508, 925
<code>\tl_if_blank_p:n</code>	514
<code>\tl_if_empty:n</code>	187
<code>\tl_if_empty:NTF</code>	917
<code>\tl_if_empty:nTF</code>	184, 299, 1184
<code>\tl_if_empty_p:N</code>	456
<code>\tl_if_head_is_N_type:nTF</code>	431, 558, 626
<code>\tl_if_noalue:nTF</code>	415, 770, 865, 912, 1044
<code>\tl_log:N</code>	349, 916, 1056
<code>\tl_new:N</code>	31, 43, 102, 103, 104, 151, 152, 154, 155, 156
<code>\tl_put_right:Nn</code>	578, 669
<code>\tl_remove_once:Nn</code>	184, 184, 915, 1055
<code>\tl_replace_all:Nnn</code>	184, 185, 428, 791, 1054, 1058
<code>\tl_rescan:nn</code>	20
<code>\tl_set:Nn</code>	192, 197, 214, 219, 236, 241, 356, 417, 551, 779, 790, 866, 913, 1052
<code>\tl_to_str:n</code>	364
<code>\tl_use:N</code>	520, 536, 642, 651, 709, 897, 1057
token commands:	
<code>\token_if_eq_meaning:NNTF</code>	568
<code>\token_to_str:N</code>	978, 979
<code>\tt</code>	148
<code>\ttfamily</code>	147
<code>\typetostored</code>	6, 22, 23, 34, 902
U	
<code>\unprotect</code>	14
use commands:	
<code>\use:N</code>	19
<code>\use:n</code>	485, 546, 604, 639, 648, 801, 896, 953
V	
<code>\verbatim</code>	966
<code>\verbatimmsc</code>	675, 935
<code>verbatimmsc</code>	6, 951
W	
<code>\writestatus</code>	13