

SCONTENTS

Stores \LaTeX CONTENTS

V2.1 2024-06-14*

©2019–2024 by Pablo González†

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

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

Abstract

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

Contents

| | | | | | |
|-----|--|---|-----|---|----|
| 1 | Description of the package | 1 | 5.8 | The environment <code>verbatimsc</code> | 7 |
| 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 | 8 |
| 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 | 8.5 | The command <code>\mergesc</code> in action | 13 |
| 5.3 | The command <code>\Scontents</code> | 5 | 9 | Change history | 15 |
| 5.4 | The command <code>\getstored</code> | 6 | 10 | Index of Documentation | 16 |
| 5.5 | The command <code>\foreachsc</code> | 6 | 11 | References | 16 |
| 5.6 | The command <code>\tpestored</code> | 6 | 12 | Implementation | 17 |
| 5.7 | The command <code>\mergesc</code> | 6 | 13 | Index of Implementation | 44 |

 **The next update removes compatibility with versions prior to 2024.**

1 Description of the package

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

2 Motivation and Acknowledgments

In \LaTeX there is no direct way to record content for later use, although you can do this using `\macros`, recording \langle *verbatim content* \rangle is a problem, usually you can avoid this by creating external files or boxes.

The general idea of this package is to try to imitate this implementation “*buffers*” that has ConTeXt which allows you to save content in memory, including *verbatim*, to be used later. The package `filecontentsdef` solves this problem and since `expl3` has an excellent way to manage data, ideas were combined giving rise to this package.

This package would not be possible without the great work of JEAN FRANÇOIS BURNOL who was kind enough to take my requirements into account and add the `filecontentsdefmacro` environment. Also a special thanks to Phelype Oleinik who has collaborated and adapted a large part of the code and all \LaTeX 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 v2.1, last revised 2024-06-14.

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

The The `SCONTENTS` package is written (mostly) using `expl3`, it requires an updated version of \LaTeX to work (minimum version 2022-06-01). 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 \LaTeX 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}
```

 \ConTeXt users should use `-luatex`, the implementation does not support `LuaMetaTeX`.

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*” (`\u` or `\uuuu`) 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 \langle *reading* \rangle ¹ and \langle *writing* \rangle ² are different and may have adverse consequences.

In a standard \LaTeX document, the character TAB “” are treated as explicit spaces (in most contexts) and is the behavior when \langle *stored contents* \rangle , but when \langle *writing files* \rangle 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 \langle *output file* \rangle .

As a general recommendation “Do not use TAB character unless strictly necessary”, for example within

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](#).

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.

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

```
\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 `\startcontents` and `\stopcontents`.

```
\startcontents \startcontents[(keyval list)]
\stopcontents (env contents)
\stopcontents
```

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{ $\langle env name \rangle$ }[ $\langle initial keys \rangle$ ]
```

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

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 = {  $\langle seq name \rangle$  } 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 = {  $\langle true \mid false \rangle$  } 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 = {  $\langle file.ext \rangle$  } 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 = {  $\langle file.ext \rangle$  } 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 <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 <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 <tpestored>[<index, 1-end, 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 `<1-end>` is used “all” content stored in `{<seq name>}` will be printed.

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 command `\mergesc`

```
\mergesc <mergesc>[<tpestored | meanings, keys>]{<{<seq A>}<[<index>]>, <{<seq B>}<[<start - stop>]>, <{<seq C>}<[<1-end>]>}
```

The command `\mergesc` internally assembles the content stored in the `{<seq A>}<[1]>`, `{<seq B>}<[2-5]>` and `{<seq C>}<[1-end]>` into a temporary internal `<seq temp>`.

The use of the keys `tpestored` or `menaingsc` are “mandatory” and disjoint from each other, the rest of the accepted `<keys>` are `print-cmd`, `write-out`, `width-tab` and `overwrite`.

The use of the `write-out` key with this command follows the same rules already described, the main advantage is that it allows to join stored content *without rewriting* the file over and over again, by design \TeX does not have an append mode for writing files, this effectively allows you to write chunks of code and then merge them into a single file.

5.8 The environment `verbatimsc`

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

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

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 ConTeXt “*buffers*” by making use `(sequences)`.

I’ve only written:

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

and

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

```
$ pdftdetach -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` .

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



```

1 % arara: pdflatex
2 % arara: clean: { extensions: [ aux, log ] }
3 \documentclass{article}
4 \usepackage[store-cmd=solutions]{scontents}
5 \newtheorem{ex}{Exercise}
6 \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
20 \section{Solutions}
21 \foreachsc[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 \foreachsc[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 \tpestored[1]{tikz}
34 \tpestored[2]{tikz}
35 \tpestored[3]{tikz}
36
37 \section{fake buffer}
38 \subsection{real content}
39 \getstored[1]{buffer}
40 \subsection{verbatim style}
41 \tpestored[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 % 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{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\}} with \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 with braces } and environment \verb+Verbatim*+
31 \begin{verbatim}
32 verbatim environment
33 \end{verbatim}
34 \end{scontents}
35
36 \section{Test \texttt{\textbackslash Scontents} with \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 `verbatim` 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\}} with \texttt{listings}}
28 Test \verb+{scontents}+ \par
29
30 \begin{scontents}
31 Using \verb+scontents+ env no \verb+[key=val]+, save in seq \verb+contents+ with index 1.\par
32
33 Prove \lstinline[basicstyle=\ttfamily] | \lstinline | and environment \verb+Verbatim*+
34 \begin{verbatim}

```

```

35   verbatim environment
36 \end{verbatim}
37 \end{scontents}
38
39 \section[Test \texttt{\textbackslash Scontents*} with \texttt{listings}]
40
41 \Scontents*+ We have coded this in \lstinlne[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 tpestored}]
49 \tpestored[1]{contents}
50 \tpestored[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}} with \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 with 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} with \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 tpestored}]
43 \tpestored[1]{contents}
44 \tpestored[2]{contents}
45 \end{document}

```

8.5 The command `\mergesc` in action

The command `\mergesc` in action, adapted from Denis Bitouzé request at <https://github.com/pablgonz/scontents/issues/2> .

```

1 % arara: pdflatex
2 % arara: clean: { extensions: [ aux, log ] }
3 \documentclass{article}
4 \usepackage{scontents}
5 % Fix part of a MCE that should go before babel's loading
6 \begin{scontents}[store-env=mce]
7 \documentclass[french]{article}
8 \usepackage[T1]{fontenc}
9 \usepackage[utf8]{inputenc}
10 \usepackage{lmodern}
11 \usepackage[a4paper]{geometry}
12 \end{scontents}
13 % Fix part of a MCE that should go after (>=) babel's loading
14 \begin{scontents}[store-env=mce]
15 \usepackage{babel}
16 \begin{document}
17 \end{scontents}
18 % Fix part of a MCE that should go after its body
19 \begin{scontents}[store-env=mce]
20 \end{document}
21 \end{scontents}
22 \begin{document}
23 \section{First annswer}
24 % Variable part of a MCE that should added to the fixed preamble, before babel's loading
25 \begin{scontents}[store-env=mce-1]
26 \usepackage{amsmath}
27 \end{scontents}
28 % Variable part of a MCE being the code snippet
29 \begin{scontents}[store-env=mce-1]
30 \begin{align}
31   0 & \neq 1 \\
32   1 & \neq 0
33 \end{align}
34 \end{scontents}
35 \begin{description}
36 \item[Preamble's addition]\leavevmode
37   \typestored[1]{mce-1}
38 \item[Code snippet]\leavevmode
39   \typestored[2]{mce-1}
40 \item[MCE]\leavevmode
41   \mergesc[typestored, print-cmd=true]
42     {
43       {mce}[1], {mce-1}[1], {mce}[2], {mce-1}[2], {mce}[3]
44     }
45 \end{description}
46 \section{Second annswer}
47 % Variable part of a MCE that should added to the fixed preamble, before babel's loading
48 \begin{scontents}[store-env=mce-2]
49 \usepackage{amsmath}
50 \end{scontents}
51 % Variable part of a MCE being the code snippet
52 \begin{scontents}[store-env=mce-2]
53 \begin{flalign}
54   0 & \neq 1 \\
55   1 & \neq 0
56 \end{flalign}
57 \end{scontents}
58
59 \begin{description}
60 \item[Preamble's addition]\leavevmode
61   \typestored[1]{mce-2}
62 \item[Code snippet]\leavevmode
63   \typestored[2]{mce-2}
64 \item[MCE]\leavevmode
65   \mergesc[typestored, print-cmd=true, write-out=mce.txt, overwrite=true]
66     {
67       {mce}[1], {mce-2}[1], {mce}[2], {mce-2}[2], {mce}[3]
68     }
69 \end{description}
70 \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`.

- v2.1 (ctan), 2024-06-14**
 - Fix `\cleansc` command.
 - Add `\mergesc` command.
 - Fix internal definition for `seq` var.
 - Fix internal code for `\typestored`.
 - Replace `\cs_argument_spec:N` by `\cs_parameter_spec:N`.
 - Detect `l3keys2e` package (obsolete in june 2022 \LaTeX release).
 - Minor adjustments in the documentation.
- v2.0 (ctan), 2022-04-04**
 - Adapting the `verbatimsc` environment (compatibility `verbatim` package).
 - Removed compatibility layer for older \LaTeX 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 `\typestored`.
 - Remove `filecontentsdef` dependency.
 - Changing `\regex_replace_all:` for `\tl_replace_all:`.
- v1.2 (ctan), 2019-08-28**
 - Restructuring of documentation.
 - Added copy of `\tex_scantokens:`.
- v1.1 (ctan), 2019-08-12**
 - Extension of documentation.
 - Replace `\tex_endinput:D` by `\file_input_stop:`.
- v1.0 (ctan), 2019-07-30**
 - First public release.

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>\mergesc</code> | 6 |
| <code>\newenvsc</code> | 3, 5 |
| <code>\scontents</code> | 4 |
| <code>\setupsc</code> | 3–5 |
| <code>\startscontents</code> | 4 |
| <code>\stopscontents</code> | 4 |
| <code>\tpestored</code> | 3, 6, 7 |
| <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>typestored</code> | 6 |
| <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> | 3–6 |
| E | |
| Environments provide by SCONTENTS : | |
| <code>scontents</code> | 3–5 |
| <code>verbatimsc</code> | 6, 7, 11–13 |
| Environments: | |
| <code>Verbatim</code> | 3 |
| <code>filecontentsdefmacro</code> | 1 |
| <code>lstlisting</code> | 3 |
| K | |
| Keys provide by SCONTENTS : | |
| <code>after</code> | 6 |
| <code>before</code> | 6 |
| <code>force-eol</code> | 3–5 |
| <code>menaingsc</code> | 6 |
| <code>overwrite</code> | 3–6 |
| <code>print-all</code> | 3 |
| <code>print-cmd</code> | 3, 5, 6 |
| <code>print-env</code> | 3–5 |
| L | |
| <code>\lstinline</code> | 5 |
| M | |
| <code>\meaning</code> | 7 |
| P | |
| Packages: | |
| <code>answers</code> | 8 |
| <code>expl3</code> | 1, 2, 8, 15 |
| <code>fancyvrb</code> | 3, 7, 11 |
| <code>filecontentsdef</code> | 1, 8, 9, 15 |
| <code>fvextra</code> | 5 |
| <code>l3seq</code> | 1, 8 |
| <code>listings</code> | 3, 5, 7, 12 |
| <code>minted</code> | 7, 13 |
| <code>scontents</code> | 1, 2, 7, 8, 15 |
| <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>, 2023.
- [2] The L^AT_EX Project. “The xparse package”. Available from CTAN, <https://www.ctan.org/pkg/xparse>, 2023.
- [3] The L^AT_EX Project. “The l3keys2e package”. Available from CTAN, <https://www.ctan.org/pkg/l3keys2e>, 2022.
- [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{2024-06-14}%
3 <core>\def\ScontentsCoreFileDate{2024-06-14}%
4 <*loader>
5 \def\ScontentsFileVersion{2.1}%
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 \IfFormatAtLeastTF { 2022-06-01 }
9 { }
10 {
11   \RequirePackage{l3keys2e}[2020/02/08]
12   \PackageWarning { scontents }
13     {
14       The next update removes compatibility with versions prior to 2024.
15     }
16 }
17 \ProvidesExplPackage
18   {scontents} {\ScontentsFileDate} {\ScontentsFileVersion} {\ScontentsFileDescription}
19 </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.

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

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

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

```
39 \tl_new:N \g_scontents_end_verbatimsc_tl
40 \tl_gset_rescan:Nnn \g_scontents_end_verbatimsc_tl
```

```

41 {
42   \char_set_catcode_other:N \
43   ⟨*latex⟩
44   \char_set_catcode_other:N \{
45   \char_set_catcode_other:N \}
46   ⟨/latex⟩
47 }
48 ⟨latex⟩ { \end{verbatim} }
49 ⟨plain⟩ { \endverbatim }
50 ⟨context⟩ { \stopverbatim }

```

(End of definition for `\g__scontents_end_verbatim_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`.

```

51 \tl_new:N \l__scontents_env_name_tl
52 \tl_const:Nx \c__scontents_end_env_tl
53 {
54   \c_backslash_str
55   ⟨latex|plain⟩ end
56   ⟨context⟩ stop
57   ⟨latex⟩ \c_left_brace_str
58           \exp_not:N \l__scontents_env_name_tl
59   ⟨latex⟩ \c_right_brace_str
60 }

```

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

Now we load the core `SCONTENTS` code:

```

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

```

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

```

62 \cs_new:Npn \__scontents_format_case:nnn #1 #2 #3
63   ⟨latex⟩ {#1} % LaTeX
64   ⟨plain⟩ {#2} % Plain/Generic
65   ⟨context⟩ {#3} % ConTeXt

```

(End of 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`.

```

66 ⟨/loader⟩
67 ⟨*core⟩
68 \begingroup
69   \catcode32=10
70   \endlinechar=32
71   \def\next{\endgroup}%
72   \expandafter\ifx\csname PackageError\endcsname\relax
73     \begingroup
74       \def\next{\endgroup\endgroup}%
75       \def\PackageError#1#2#3%
76         {%
77           \endgroup
78           \errhelp{#3}%
79           \errmessage{#1 Error: #2!}%
80         }%
81   \fi
82   \expandafter\ifx\csname ScontentsFileDate\endcsname\relax
83     \def\next
84       {%
85         \PackageError{scontents}{No scontents loader detected}
86         {%
87           You have attempted to use the scontents code directly rather than using
88           the correct loader. Loading of scontents will abort.
89         }%
90       \endgroup
91     \endinput
92   }%
93   \else

```

```

94 \ifx\ScontentsFileDate\ScontentsCoreFileDate
95 \else
96 \def\next
97 {%
98 \PackageError{scontents}{Mismatched scontents files detected}
99 {%
100 You have attempted to load scontents with mismatched files:
101 probably you have one or more files 'locally installed' which
102 are in conflict. Loading of scontents will abort.
103 }%
104 \endgroup
105 \endinput
106 }%
107 \fi
108 \fi
109 \next

```

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.

```

110 \tl_new:N \l__scontents_macro_tmp_tl
111 \tl_new:N \l__scontents_temp_tl
112 \tl_new:N \g__scontents_temp_tl
113 \int_new:N \l__scontents_tmpa_int
114 \bool_new:N \l__scontents_temp_bool

```

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

`\l__scontents_keys_tl` Stores unused keys to be forwarded to other commands.

```
115 \tl_new:N \l__scontents_keys_tl
```

(End of definition for `\l__scontents_keys_tl`.)

12.4 Compatibility layer with plain T_EX and ConT_EXt

When loading the package outside of L^AT_EX we can't usually use `xparse`. However since `xparse` now `ltxcmd` is part of the L^AT_EX kernel is loadable in any format.

```

116 </core>
117 <*loader&!latex>
118 \int_set:Nn \l__scontents_tmpa_int { \char_value_catcode:n { \@ } }
119 \char_set_catcode_letter:N \@
120 \file_input:n { xparse-generic.tex }
121 \char_set_catcode:n { \@ } { \l__scontents_tmpa_int }
122 </loader&!latex>
123 <*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.

```

124 \keys_define:nn { scontents }
125 {
126 store-env .tl_set:N = \l__scontents_name_seq_env_tl,
127 store-env .initial:n = contents,
128 store-env .value_required:n = true,
129 store-cmd .tl_set:N = \l__scontents_name_seq_cmd_tl,
130 store-cmd .initial:n = contents,
131 store-cmd .value_required:n = true,
132 verb-font .tl_set:N = \l__scontents_verb_font_tl,
133 verb-font .value_required:n = true,
134 print-env .bool_set:N = \l__scontents_print_env_bool,
135 print-env .initial:n = false,
136 print-env .default:n = true,
137 print-cmd .bool_set:N = \l__scontents_print_cmd_bool,
138 print-cmd .initial:n = false,
139 print-cmd .default:n = true,

```

```

140 force-eol .bool_set:N      = \l__scontents_forced_eol_bool,
141 force-eol .initial:n      = false,
142 force-eol .default:n      = true,
143 overwrite .bool_set:N     = \l__scontents_overwrite_bool,
144 overwrite .initial:n      = false,
145 overwrite .default:n      = true,
146 width-tab .int_set:N      = \l__scontents_tab_width_int,
147 width-tab .initial:n      = 1,
148 width-tab .value_required:n = true,
149 print-all .meta:n        = { print-env = #1 , print-cmd = #1 },
150 print-all .default:n      = true,
151 store-all .meta:n        = { store-env = #1 , store-cmd = #1 },
152 store-all .value_required:n = true
153 }
154 </core>
155 <loader>\keys_define:nn { scontents }
156 <latex> { verb-font .initial:n = \ttfamily }
157 <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

```

158 <*latex>
159 \IfFormatAtLeastTF { 2022-06-01 }
160 {
161   \ProcessKeyOptions [ scontents ]
162 }
163 { \ProcessKeysOptions { scontents } }
164 </latex>
165 <*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` 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).

```

166 \tl_new:N \l__scontents_fname_out_tl
167 \tl_new:N \l__scontents_every_line_env_tl
168 \iow_new:N \l__scontents_file_iow

```

(End of 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.

```

169 \tl_new:N \l__scontents_foreach_name_seq_tl
170 \tl_new:N \l__scontents_foreach_before_tl
171 \tl_new:N \l__scontents_foreach_after_tl

```

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

`\l__scontents_env_nesting_int` `\l__scontents_seq_item_seq` stores the indexes in the sequence of the items 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.

```

172 \int_new:N \l__scontents_foreach_stop_int
173 \seq_new:N \l__scontents_seq_item_seq
174 \int_new:N \l__scontents_env_nesting_int

```

(End of definition for `\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.

```

175 \bool_new:N \l__scontents_writing_bool
176 \bool_set_false:N \l__scontents_writing_bool
177 \bool_new:N \l__scontents_storing_bool
178 \bool_set_true:N \l__scontents_storing_bool
179 \bool_new:N \l__scontents_writable_bool

```

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

```

\l__scontents_foreach_before_bool
\l__scontents_foreach_after_bool
\l__scontents_foreach_stop_bool
\l__scontents_foreach_wrapper_bool

```

Boolean variables used by the `\foreachsc` loop.

```

180 \bool_new:N \l__scontents_foreach_before_bool
181 \bool_set_false:N \l__scontents_foreach_before_bool
182 \bool_new:N \l__scontents_foreach_after_bool
183 \bool_set_false:N \l__scontents_foreach_after_bool
184 \bool_new:N \l__scontents_foreach_stop_bool
185 \bool_set_false:N \l__scontents_foreach_stop_bool
186 \bool_new:N \l__scontents_foreach_wrapper_bool
187 \bool_set_false:N \l__scontents_foreach_wrapper_bool

```

(End of 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`.

```

188 \seq_new:N \l__scontents_foreach_print_seq
189 \seq_new:c { g__scontents_name_sc!internal_seq }

```

(End of 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.

```

190 \str_const:Nx \c__scontents_hidden_space_str
191 { \c_percent_str \c_circumflex_str \c_circumflex_str A scheol \c_percent_str }

```

(End of definition for `\c__scontents_hidden_space_str`.)

```

\q__scontents_stop
\q__scontents_mark

```

Some quarks used along the code as macro delimiters.

```

192 \quark_new:N \q__scontents_stop
193 \quark_new:N \q__scontents_mark

```

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

```

\s__scontents_stop
\s__scontents_mark

```

```

194 \scan_new:N \s__scontents_stop
195 \scan_new:N \s__scontents_mark

```

(End of definition for `\s__scontents_stop` and `\s__scontents_mark`.)

```
\l__scontents_cur_seq_name_str
```

```
196 \str_new:N \l__scontents_cur_seq_name_str
```

(End of definition for `\l__scontents_cur_seq_name_str`.)

```

\__scontents_use_i_delimit_by_s_stop:nw
\__scontents_use_none_delimit_by_s_stop:w

```

```

197 \cs_new:Npn \__scontents_use_delimit_by_s_stop:nw #1 \s__scontents_stop {#1}
198 \cs_new:Npn \__scontents_use_i_delimit_by_s_stop:nw #1 #2 \s__scontents_stop {#1}
199 \cs_new:Npn \__scontents_use_none_delimit_by_s_stop:w #1 \s__scontents_stop { }

```

(End of definition for `__scontents_use_i_delimit_by_s_stop:nw` and `__scontents_use_none_delimit_by_s_stop:w`.)

```

\l__scontents_save_sf_int
\l__scontents_save_skip

```

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

```

200 \int_new:N \l__scontents_save_sf_int
201 \skip_new:N \l__scontents_save_skip

```

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

```
\__scontents_rescan_tokens:n
\__scontents_rescan_tokens:x
\__scontents_rescan_tokens:v
```

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?](#)

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

(End of definition for `__scontents_rescan_tokens:n`.)

```
\__scontents_tab: Control sequences to replace tab ( $\text{\char"0009}$ ) and form feed ( $\text{\char"000A}$ ) characters.
\__scontents_par:
```

```
204 \cs_new:Npx \__scontents_tab: { \c_space_tl }
205 \cs_new:Npn \__scontents_par: { ^^J ^^J }
```

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

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

Some nonstandard kernel variants.

```
206 \cs_generate_variant:Nn \tl_remove_once:Nn { NV }
207 \cs_generate_variant:Nn \tl_replace_all:Nnn { Nx, Nxx, Nnx }
208 \cs_generate_variant:Nn \msg_error:nnnn { nnx }
209 \prg_generate_conditional_variant:Nnn \tl_if_empty:n { f } { p, TF }
```

(End of 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`.

```
210 \keys_define:nn { scontents / scontents }
211 {
212   write-env .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-env .value_required:n = true,
222   write-out .value_required:n = true,
223   print-env .meta:nn        = { scontents } { print-env = #1 },
224   print-env .default:n      = true,
225   store-env .meta:nn        = { scontents } { store-env = #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_environment_keys:n {#1} }
231 }
```

12.7.2 Keys for command \Scontents

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

```
232 \keys_define:nn { scontents / Scontents }
233 {
234   write-cmd .code:n          = {
235     \bool_set_true:N \__scontents_writing_bool
236     \tl_set:Nn \__scontents_fname_out_tl {#1}
237   },
238   write-out .code:n         = {
239     \bool_set_false:N \__scontents_storing_bool
```

```

240         \bool_set_true:N \l__scontents_writing_bool
241         \tl_set:Nn \l__scontents_fname_out_tl {#1}
242     },
243     write-cmd .value_required:n = true,
244     write-out .value_required:n = true,
245     print-cmd .meta:nn = { scontents } { print-cmd = #1 },
246     print-cmd .default:n = true,
247     store-cmd .meta:nn = { scontents } { store-cmd = #1 },
248     force-eol .meta:nn = { scontents } { force-eol = #1 },
249     force-eol .default:n = true,
250     overwrite .meta:nn = { scontents } { overwrite = #1 },
251     overwrite .default:n = true,
252     unknown .code:n = { \__scontents_parse_command_keys:n {#1} }
253 }

```

12.7.3 Keys for command \foreachsc

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

```

254 \keys_define:nn { scontents / foreachsc }
255 {
256     before .code:n = {
257         \bool_set_true:N \l__scontents_foreach_before_bool
258         \tl_set:Nn \l__scontents_foreach_before_tl {#1}
259     },
260     before .value_required:n = true,
261     after .code:n = {
262         \bool_set_true:N \l__scontents_foreach_after_bool
263         \tl_set:Nn \l__scontents_foreach_after_tl {#1}
264     },
265     after .value_required:n = true,
266     start .int_set:N = \l__scontents_foreach_start_int,
267     start .value_required:n = true,
268     start .initial:n = 1,
269     stop .code:n = {
270         \bool_set_true:N \l__scontents_foreach_stop_bool
271         \int_set:Nn \l__scontents_foreach_stop_int {#1}
272     },
273     stop .value_required:n = true,
274     step .int_set:N = \l__scontents_foreach_step_int,
275     step .value_required:n = true,
276     step .initial:n = 1,
277     wrapper .code:n = {
278         \bool_set_true:N \l__scontents_foreach_wrapper_bool
279         \cs_set_protected:Npn
280         \__scontents_foreach_wrapper:n ##1 {#1}
281     },
282     wrapper .value_required:n = true,
283     sep .tl_set:N = \l__scontents_foreach_sep_tl,
284     sep .initial:n = {},
285     sep .value_required:n = true,
286     unknown .code:n = { \__scontents_parse_foreach_keys:n {#1} }
287 }

```

12.7.4 Key for commands \typestored and \meaningsc

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

```

288 \bool_new:N \l__scontents_print_aux_bool
289 \bool_set_true:N \l__scontents_print_aux_bool
290 \keys_define:nn { scontents / typemeaning }
291 {
292     width-tab .meta:nn = { scontents } { width-tab = #1 },
293     write-out .code:n = {
294         \bool_set_false:N \l__scontents_storing_bool
295         \bool_set_true:N \l__scontents_writing_bool
296         \tl_set:Nn \l__scontents_fname_out_tl {#1}
297     },
298     overwrite .meta:nn = { scontents } { overwrite = #1 },
299     overwrite .default:n = true,
300     unknown .code:n = { \__scontents_parse_type_meaning_key:n {#1} }

```

```
301 }
```

12.8 Handling undefined keys

The $\langle keys \rangle$ are stored in the string variable `\l_keys_key_str`, and the value (if any) is passed as an argument to each $\langle function \rangle$.

12.8.1 Undefined keys for environment scontents

```
\__scontents_parse_environment_keys:n
\__scontents_parse_environment_keys:nn
```

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

```
302 \cs_new_protected:Npn \__scontents_parse_environment_keys:n #1
303 { \exp_args:NV \__scontents_parse_environment_keys:nn \l_keys_key_str {#1} }
304 \cs_new_protected:Npn \__scontents_parse_environment_keys:nn #1#2
305 {
306   \tl_if_blank:nTF {#2}
307     { \msg_error:nnn { scontents } { env-key-unknown } {#1} }
308     { \msg_error:nnnn { scontents } { env-key-value-unknown } {#1} {#2} }
309 }
```

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

12.8.2 Undefined keys for \Scontents and \Scontents*

```
\__scontents_parse_command_keys:n
\__scontents_parse_command_keys:nn
```

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

```
310 \cs_new_protected:Npn \__scontents_parse_command_keys:n #1
311 { \exp_args:NV \__scontents_parse_command_keys:nn \l_keys_key_str {#1} }
312 \cs_new_protected:Npn \__scontents_parse_command_keys:nn #1#2
313 {
314   \tl_if_blank:nTF {#2}
315     { \msg_error:nnn { scontents } { cmd-key-unknown } {#1} }
316     { \msg_error:nnnn { scontents } { cmd-key-value-unknown } {#1} {#2} }
317 }
```

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

12.8.3 Undefined keys for \foreachsc

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

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

```
318 \cs_new_protected:Npn \__scontents_parse_foreach_keys:nn #1#2
319 {
320   \tl_if_blank:nTF {#2}
321     { \msg_error:nnn { scontents } { for-key-unknown } {#1} }
322     { \msg_error:nnnn { scontents } { for-key-value-unknown } {#1} {#2} }
323 }
324 \cs_new_protected:Npn \__scontents_parse_foreach_keys:n #1
325 { \exp_args:NV \__scontents_parse_foreach_keys:nn \l_keys_key_str {#1} }
```

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

12.8.4 Undefined keys for \tpestored and \meaningsc

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

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

First we check if the $\langle key \rangle$ 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 $\langle index \rangle$. Otherwise raise an error about an *unknown* option.

```
326 \cs_new_protected:Npn \__scontents_parse_type_meaning_key:n #1
327 { \exp_args:NV \__scontents_parse_type_meaning_key:nn \l_keys_key_str {#1} }
328 \cs_new_protected:Npn \__scontents_parse_type_meaning_key:nn #1#2
329 {
330   \tl_if_blank:nTF {#2}
331     { \__scontents_parse_type_meaning_range:w #1 - \q__scontents_mark - \s__scontents_mark }
```



```

332     { \msg_error:nnnn { scontents } { type-key-value-unknown } {#1} {#2} }
333   }
334 \cs_new_protected:Npn \__scontents_parse_type_meaning_range:w #1 - #2 - #3 \s__scontents_mark
335   {
336     \__scontents_range_parser:nxxn {#1} {#2}
337     { \seq_count:c { g__scontents_name_\l__scontents_cur_seq_name_str _seq } }
338     { \msg_error:nnn { scontents } { type-key-unknown } }
339   }
340 \cs_generate_variant:Nn \__scontents_range_parser:nnnn { nxx }
341 \cs_new_protected:Npn \__scontents_range_parser:nnnn #1 #2 #3
342   {
343     \exp_args:Nxx \__scontents_range_parser_aux:nnn
344     { \str_if_eq:nnTF {#1} { end } {#3} { \exp_not:n {#1} } }
345     { \str_if_eq:nnTF {#2} { end } {#3} { \exp_not:n {#2} } }
346   }
347 \cs_new_protected:Npn \__scontents_range_parser_aux:nnn #1 #2 #3
348   {
349     \__scontents_tl_if_head_is_q_mark:nTF {#2}
350     {
351       \tl_if_empty:ftf { \int_to_roman:n { -0 #1 } }
352       { \seq_put_right:Nx \l__scontents_seq_item_seq { \int_eval:n {#1} } }
353         { #3 {#1} }
354     }
355     {
356       \bool_lazy_and:nnTF
357         { \tl_if_empty_p:f { \int_to_roman:n { -0 #1 } } }
358         { \tl_if_empty_p:f { \int_to_roman:n { -0 #2 } } }
359         {
360           \int_compare:nNnTF {#2} > {#1}
361           { \int_step_inline:nnnn {#1} { 1 } {#2} }
362           { \int_step_inline:nnnn {#1} { -1 } {#2} }
363             { \seq_put_right:Nn \l__scontents_seq_item_seq {##1} }
364         }
365         { #3 { #1-#2 } }
366     }
367   }

```

(End of 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.

```

368 \cs_new_protected:Npn \__scontents_append_contents:nn #1#2
369   {
370     \tl_if_blank:nT {#1}
371     { \msg_error:nn { scontents } { empty-store-cmd } }
372     \seq_if_exist:cF { g__scontents_name_#1_seq }
373     { \seq_new:c { g__scontents_name_#1_seq } }
374     \seq_gput_right:cn { g__scontents_name_#1_seq } {#2}
375   }
376 \cs_generate_variant:Nn \__scontents_append_contents:nn { Vx }

```

(End of definition for `__scontents_append_contents:nn`)

```

\__scontents_getfrom_seq:nn
\__scontents_getfrom_seq:Nn
\__scontents_getfrom_seq:nnn

```

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

```

377 \cs_new:Npn \__scontents_getfrom_seq:Nn #1#2
378   {
379     \seq_if_exist:cTF { g__scontents_name_#2_seq }
380     {
381       \exp_args:Nf \__scontents_getfrom_seq:nNn
382       { \seq_count:c { g__scontents_name_#2_seq } } #1 {#2}
383     }
384     { \msg_expandable_error:nnn { scontents } { undefined-storage } {#2} }
385   }
386 \cs_new:Npn \__scontents_getfrom_seq:nNn #1 #2 #3

```

```

387 { \seq_map_tokens:Nn #2 { \__scontents_getfrom_seq_aux:nnn {#1} {#3} } }
388 \cs_new:Npn \__scontents_getfrom_seq_aux:nnn #1 #2 #3
389 { \exp_args:Nnf \use:n { \__scontents_getfrom_seq:nnn {#1} } { \int_eval:n {#3} } {#2} }
390 \cs_new:Npn \__scontents_getfrom_seq:nn #1#2
391 {
392   \seq_if_exist:cTF { g__scontents_name_#2_seq }
393   {
394     \exp_args:Nf \__scontents_getfrom_seq:nnn
395     { \seq_count:c { g__scontents_name_#2_seq } }
396     {#1} {#2}
397   }
398   { \msg_expandable_error:nnn { scontents } { undefined-storage } {#2} }
399 }
400 \cs_new:Npn \__scontents_getfrom_seq:nnn #1#2#3
401 {
402   \bool_lazy_or:nnTF
403   { \int_compare_p:nNn {#2} = { 0 } }
404   { \int_compare_p:nNn { \int_abs:n {#2} } > {#1} }
405   { \msg_expandable_error:nnnn { scontents } { index-out-of-range } {#2} {#3} {#1} }
406   { \seq_item:cn { g__scontents_name_#3_seq } {#2} }
407 }

```

(End of definition for `__scontents_getfrom_seq:nn`, `__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`

```

408 \cs_new_protected:Npn \__scontents_lastfrom_seq:n #1
409 {
410   \tl_gset:Nx \g__scontents_temp_tl { \seq_item:cn { g__scontents_name_#1_seq } {-1} }
411   \group_insert_after:N \__scontents_rescan_tokens:V
412   \group_insert_after:N \g__scontents_temp_tl
413   \group_insert_after:N \tl_gclear:N
414   \group_insert_after:N \g__scontents_temp_tl
415 }
416 \cs_generate_variant:Nn \__scontents_lastfrom_seq:n { V }

```

(End of 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`

```

417 \cs_new_protected:Npn \__scontents_store_to_seq:NN #1#2
418 {
419   \tl_log:N #1
420   \__scontents_append_contents:Vx #2 { \exp_not:V #1 }
421 }

```

(End of 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` The internal function `__scontents_env_setting:nn` defines two functions `__scontents_#1_`
`__scontents_env_setting:nn` `env_begin:` and `__scontents_#1_env_end:`, which set the current environment's name in `#1` and
`__scontents_env_define:nnn` `__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.

```

422 \cs_new_protected:Npn \__scontents_env_setting:nn #1 #2
423 {
424   \cs_new_protected:cpn { __scontents_#1_env_begin: }
425   {
426     \tl_set:Nn \__scontents_env_name_tl {#1}
427     \keys_set:nn { scontents } {#2}
428     \__scontents_setup_verb_processor:
429     \__scontents_env_generic_begin:

```

```

430     }
431     \cs_new_protected:cpn { __scontents_#1_env_end: }
432     { \__scontents_env_generic_end: }
433     \exp_args:Nooo % http://nooooooooooooooooo.com :) jeje
434     \__scontents_env_define:nnn { \tl_to_str:n {#1} }
435     { \cs:w __scontents_#1_env_begin: \cs_end: }
436     { \cs:w __scontents_#1_env_end: \cs_end: }
437   }
438 </core>
439 <*loader>
440 \NewDocumentCommand \newenvsc { m O{} }
441 {
442 <latex|plain>   \cs_if_exist:cTF { #1 }
443 <context>       \cs_if_exist:cTF { start #1 }
444               { \msg_error:nnn { scontents } { env-already-defined } {#1} }
445               { \__scontents_env_setting:nn {#1} {#2} }
446   }
447 \cs_new_protected:Npn \__scontents_env_define:nnn #1 #2 #3
448 {
449 <latex|plain>   \NewDocumentEnvironment {#1} { }
450 <context>       \cs_new_protected:cpn { start #1 }
451               {
452 <!!latex>       \group_begin:
453                 #2
454               }
455 <context>       \cs_new_protected:cpn { stop #1 }
456               {
457                 #3
458 <!!latex>       \group_end:
459               }
460   }
461 </loader>
462 <*core>

```

(End of 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:`.

```

463 \cs_new_protected:Npn \__scontents_env_generic_begin:
464 {
465   \char_set_catcode_active:N ^^M
466   \__scontents_start_environment:w
467 }
468 \cs_new_protected:Npn \__scontents_env_generic_end:
469 {
470   \__scontents_stop_environment:
471   \__scontents_finish_storing:NNN \l__scontents_macro_tmp_tl
472   \l__scontents_name_seq_env_tl \l__scontents_print_env_bool
473 }

```

(End of 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
\scntents
\endscontents
\startscntents
\stopscntents

```

```

474 </core>
475 <loader>\newenvsc{scontents}
476 <*core>

```

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

12.10.4 key val for environment

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.

```

477 </core>
478 <*loader>
479 \NewDocumentCommand \__scontents_grab_optional:w { r[] }
480 { \__scontents_grab_optional:n {#1} }
481 </loader>
482 <*core>
483 \cs_new_protected:Npn \__scontents_grab_optional:n #1
484 {
485   \tl_if_novalue:nF {#1}
486   {
487     \tl_set:Nn \__scontents_temp_tl {#1}
488     \__scontents_normalise_line_ends:N \__scontents_temp_tl
489     \keys_set:nV { scontents / scontents } \__scontents_temp_tl
490   }
491   \__scontents_start_after_option:w
492 }

```

(End of 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.

```

493 \group_begin:
494   \char_set_catcode_active:N ^^I
495   \char_set_catcode_active:N ^^L
496   \char_set_catcode_active:N ^^M
497   \cs_new_protected:Npn \__scontents_normalise_line_ends:N #1
498   { \tl_replace_all:Nnn #1 { ^^M } { ~ } }
499   \cs_new_protected:Npn \__scontents_start_environment:w #1 ^^M
500   {
501     \tl_if_head_is_N_type:nTF {#1}
502     {
503       \str_if_eq:eeTF { \tl_head:n {#1} } { [ ]
504       { \__scontents_grab_optional:w #1 ^^M }
505       { \__scontents_check_line_process:xn { } {#1} }
506     }
507     { \__scontents_check_line_process:xn { } {#1} }
508   }
509   \cs_new_protected:Npn \__scontents_start_after_option:w #1 ^^M
510   { \__scontents_check_line_process:xn { [...] } {#1} }
511   \cs_new_protected:Npn \__scontents_check_line_process:xn #1 #2
512   {
513     \tl_if_blank:nF {#2}
514     {
515       \msg_error:nnxn { scontents } { junk-after-begin }
516       { after~\c_backslash_str begin { \__scontents_env_name_tl } #1 } {#2}
517     }
518     \__scontents_make_control_chars_active:
519     \__scontents_file_tl_write_start:V \__scontents_fname_out_tl
520   }
521   \cs_new_protected:Npn \__scontents_stop_environment:
522   {
523     \__scontents_file_write_stop:N \__scontents_macro_tmp_tl
524     \bool_lazy_and:nnT
525     { \__scontents_storing_bool }
526     { \tl_if_empty_p:N \__scontents_macro_tmp_tl }

```

```

527     {
528     \msg_warning:nxx { scontents } { empty-environment } { \l__scontents_env_name_tl }
529     }
530 }

```

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

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

```

531 \cs_new_protected:Npn \__scontents_file_tl_write_start:n #1
532 {
533   \group_begin:
534   \__scontents_file_if_writable:nTF {#1}
535   {
536     \bool_set_true:N \l__scontents_writable_bool
537     \iow_open:Nn \l__scontents_file_iow {#1}
538   }
539   { \bool_set_false:N \l__scontents_writable_bool }
540   \tl_clear:N \l__scontents_every_line_env_tl
541   \seq_map_function:NN \l_char_special_seq \char_set_catcode_other:N
542   \int_step_function:nnnN { 128 } { 1 } { 255 } \char_set_catcode_letter:n
543   \cs_set_protected:Npx \__scontents_ret:w ##1 ^^M
544   {
545     \exp_not:N \__scontents_verb_processor_iterate:w
546     ##1 \c__scontents_end_env_tl
547     \c__scontents_end_env_tl
548     \exp_not:N \q__scontents_stop
549   }
550   \__scontents_make_control_chars_active:
551   \__scontents_ret:w
552 }
553 \cs_new:Npn \__scontents_setup_verb_processor:
554 {
555   \use:x
556   {
557     \cs_set:Npn \exp_not:N \__scontents_verb_processor_iterate:w
558     ###1 \c__scontents_end_env_tl
559     ###2 \c__scontents_end_env_tl
560     ###3 \exp_not:N \q__scontents_stop
561   } { \__scontents_verb_processor_iterate:nnn {##1} {##2} {##3} }
562 }
563 \cs_new:Npn \__scontents_verb_processor_iterate:nnn #1 #2 #3
564 {
565   \tl_if_blank:nTF {#3}
566   {
567     \__scontents_analyse_nesting:n {#1}
568     \__scontents_verb_processor_output:n {#1}
569   }
570   {
571     \__scontents_if_nested:TF
572     {
573       \__scontents_nesting_decr:
574       \__scontents_verb_processor_output:x
575       { \exp_not:n {#1} \c__scontents_end_env_tl \exp_not:n {#2} }
576     }
577     {
578       \tl_if_blank:nF {#1}
579       { \__scontents_verb_processor_output:n {#1} }
580       \cs_set_protected:Npx \__scontents_ret:w
581       {
582         \__scontents_env_end_function:
583         \bool_lazy_or:nnF
584         { \tl_if_blank_p:n {#2} }
585         { \str_if_eq_p:ee {#2} { \c_percent_str } }

```

```

586         {
587         \str_if_eq:VnF \c__scontents_hidden_space_str {#2}
588         {
589         \msg_warning:nnnn { scontents } { rescanning-text }
590         {#2} { \tl_use:N \l__scontents_env_name_tl }
591         }
592         \__scontents_rescan_tokens:n {#2}
593         }
594     }
595     \char_set_active_eq:NN ^^M \__scontents_ret:w
596 }
597 }
598 ^^M
599 }
600 \cs_new:Npn \__scontents_env_end_function:
601 {
602     \__scontents_format_case:nnn
603     { \exp_not:N \end { \if_false: } \fi: }
604     { \exp_after:wN \exp_not:N \cs:w end }
605     { \exp_after:wN \exp_not:N \cs:w stop }
606     \tl_use:N \l__scontents_env_name_tl
607     \__scontents_format_case:nnn
608     { \if_false: { \fi: } }
609     { \cs_end: }
610     { \cs_end: }
611 }
612 \cs_new_protected:Npn \__scontents_file_write_stop:N #1
613 {
614     \bool_if:NT \l__scontents_writable_bool
615     { \iow_close:N \l__scontents_file_iow }
616     \use:x
617     {
618         \group_end:
619         \bool_if:NT \l__scontents_storing_bool
620         {
621             \tl_set:Nn \exp_not:N #1
622             { \exp_args:NV \__scontents_remove_leading_nl:n \l__scontents_every_line_env_tl }
623         }
624     }
625 }
626 \cs_new:Npn \__scontents_remove_leading_nl:n #1
627 {
628     \tl_if_head_is_N_type:nTF {#1}
629     {
630         \exp_args:Nf
631         \__scontents_remove_leading_nl:nn
632         { \tl_head:n {#1} } {#1}
633     }
634     { \exp_not:n {#1} }
635 }
636 \cs_new:Npn \__scontents_remove_leading_nl:nn #1 #2
637 {
638     \token_if_eq_meaning:NNTF ^^J #1
639     { \exp_not:o { \__scontents_remove_leading_nl:w #2 } }
640     { \exp_not:n {#2} }
641 }
642 \cs_new:Npn \__scontents_remove_leading_nl:w ^^J { }

```

(End of 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`.

```

643 \cs_new_protected:Npn \__scontents_verb_processor_output:n #1
644 {
645     \bool_if:NT \l__scontents_writable_bool
646     { \iow_now:Nn \l__scontents_file_iow {#1} }
647     \bool_if:NT \l__scontents_storing_bool
648     { \tl_put_right:Nn \l__scontents_every_line_env_tl { ^^J #1 } }
649 }

```

```

650 \group_end:
651 \cs_generate_variant:Nn \__scontents_verb_processor_output:n { x }
652 \cs_generate_variant:Nn \__scontents_file_tl_write_start:n { V }

```

(End of definition for `__scontents_verb_processor_output:n`.)

```

\__scontents_analyse_nesting:n \__scontents_analyse_nesting:n scans nested \begin{scontents} and steps a \l__scontents_
\__scontents_analyse_nesting:w env_nesting_int counter. The \__scontents_if_nested: conditional tests if we're in a nested en-
\__scontents_nesting_decr: vironment, and \__scontents_nesting_decr: reduces the nesting level, if an \end{scontents} is
\__scontents_use_none_delimit_by_q_stop:w found.
\__scontents_if_nested:TF

```

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

```

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

```

674 \use:x
675 {
676   \cs_new_protected:Npn \exp_not:N \__scontents_analyse_nesting_latex:w ##1
677     \c_backslash_str begin \c_left_brace_str ##2 \c_right_brace_str
678   } {
679     \__scontents_tl_if_head_is_q_mark:nTF {#2}
680     { \__scontents_use_none_delimit_by_q_stop:w }
681     {
682       \str_if_eq:VnT \l__scontents_env_name_tl {#2}
683       { \__scontents_nesting_incr: }
684       \__scontents_analyse_nesting_latex:w
685     }
686   }
687 \cs_new_protected:Npx \__scontents_analyse_nesting_latex:n #1
688 {
689   \__scontents_analyse_nesting_latex:w #1
690   \c_backslash_str begin
691   \c_left_brace_str \exp_not:N \q__scontents_mark \c_right_brace_str
692   \exp_not:N \q__scontents_stop
693 }

```

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

```

694 \cs_new_protected:Npn \__scontents_analyse_nesting_generic_process:nn #1 #2
695 {
696   \tl_if_head_is_N_type:nTF {#2}
697   {
698     \__scontents_tl_if_head_is_q_mark:nF {#2}
699     {
700       \__scontents_nesting_incr:
701       \__scontents_analyse_nesting_generic:w #2 \q__scontents_stop
702     }
703   }

```

```

704     { \_scontents_analyse_nesting_generic:w #2 \q\_scontents_stop }
705   }
706 \cs_new_protected:Npn \_scontents_analyse_nesting_generic:nn #1 #2
707   {
708     \_scontents_define_generic_nesting_function:n {#1}
709     \use:x
710     {
711       \exp_not:N \_scontents_analyse_nesting_generic:w #2
712       \c_backslash_str #1 \tl_use:N \_scontents_env_name_tl
713       \exp_not:N \q\_scontents_mark \exp_not:N \q\_scontents_stop
714     }
715   }
716 \cs_new_protected:Npn \_scontents_define_generic_nesting_function:n #1
717   {
718     \use:x
719     {
720       \cs_set_protected:Npn \exp_not:N \_scontents_analyse_nesting_generic:w ####1
721       \c_backslash_str #1 \tl_use:N \_scontents_env_name_tl
722       ####2 \exp_not:N \q\_scontents_stop
723     } { \_scontents_analyse_nesting_generic_process:nn {##1} {##2} }
724   }
725 </core>
726 <*loader>
727 <latex>\cs_new_eq:NN \_scontents_analyse_nesting_format:n
728 <latex> \_scontents_analyse_nesting_latex:n
729 <!latex>\cs_new_protected:Npn \_scontents_analyse_nesting_format:n
730 <plain> { \_scontents_analyse_nesting_generic:nn { } }
731 <context> { \_scontents_analyse_nesting_generic:nn { start } }
732 </loader>
733 <*core>

```

(End of 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.

```

734 \cs_new_protected:Npn \_scontents_finish_storing:NNN #1 #2 #3
735   {
736     \bool_if:NT \l\_scontents_storing_bool
737     {
738       \bool_if:NF \l\_scontents_forced_eol_bool
739       { \tl_put_right:Nx #1 { \c\_scontents_hidden_space_str } }
740       \_scontents_store_to_seq:NN #1 #2
741       \bool_if:NT #3 { \_scontents_lastfrom_seq:V #2 }
742     }
743   }
744 </core>

```

(End of 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:
745 <*plain>
746 \cs_new_protected:Npn \verbatimsc
747   {
748     \group_begin:
749     \_scontents_verbatimsc_aux: \frenchspacing \_scontents_vobeyspaces:
750     \_scontents_xverb:
751   }
752 \cs_new_protected:Npn \endverbatimsc
753   { \group_end: }
754 \cs_new_protected:Npn \_scontents_verbatimsc_aux:
755   {
756     \skip_vertical:N \parskip
757     \dim_zero:N \parindent
758     \skip_set:Nn \parfillskip { 0pt plus 1fil }
759     \skip_set:Nn \parskip { 0pt plus 0pt minus 0pt }
760     \tex_par:D

```



```

761 \bool_set_false:N \l__scontents_temp_bool
762 \cs_set:Npn \par
763 {
764   \bool_if:NTF \l__scontents_temp_bool
765   {
766     \mode_leave_vertical:
767     \null
768     \tex_par:D
769     \penalty \interlinepenalty
770   }
771   {
772     \bool_set_true:N \l__scontents_temp_bool
773     \mode_if_horizontal:T
774     { \tex_par:D \penalty \interlinepenalty }
775   }
776 }
777 \cs_set_eq:NN \do \char_set_catcode_other:N
778 \dospecials \obeylines
779 \tl_use:N \l__scontents_verb_font_tl
780 \cs_set_eq:NN \do \__scontents_do_noligs:N
781 \__scontents_nolig_list:
782 \tex_everypar:D \exp_after:wN
783 { \tex_the:D \tex_everypar:D \tex_unpenalty:D }
784 }
785 \cs_new_protected:Npn \__scontents_nolig_list:
786 { \do\` \do\< \do\> \do\| \do\|' \do\| - }
787 \cs_new_protected:Npn \__scontents_vobeyspaces:
788 { \__scontents_set_active_eq:NN \ \__scontents_xobeysp: }
789 \cs_new_protected:Npn \__scontents_xobeysp:
790 { \mode_leave_vertical: \nobreak \ }
791 </plain>

```

(End of 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`.

```

792 <!*latex>
793 \cs_if_exist:NF \dospecials
794 {
795   \cs_new:Npn \dospecials
796   { \seq_map_function:NN \l_char_special_seq \do }
797 }
798 </!*latex>

```

(End of 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:`

```

799 <*core>
800 \cs_new_protected:Npn \__scontents_bsphack:
801 {
802   \scan_stop:
803   \mode_if_horizontal:T
804   {
805     \skip_set_eq:NN \l__scontents_save_skip \tex_lastskip:D
806     \int_set_eq:NN \l__scontents_save_sf_int \tex_spacefactor:D
807   }
808 }
809 \cs_new_protected:Npn \__scontents_esphack:
810 {
811   \scan_stop:
812   \mode_if_horizontal:T
813   {
814     \int_set_eq:NN \tex_spacefactor:D \l__scontents_save_sf_int
815     \dim_compare:nNnT { \l__scontents_save_skip } > { \c_zero_skip }

```

```

816     {
817         \skip_if_eq:nnT { \tex_lastskip:D } { \c_zero_skip }
818         {
819             \nobreak
820             \skip_horizontal:n { \c_zero_skip }
821         }
822         \tex_ignorespaces:D
823     }
824 }
825 }
826 </core>
827 <*latex>
828 \cs_gset_eq:NN \__scontents_bspack: \@bspack
829 \cs_gset_eq:NN \__scontents_espack: \@espack
830 </latex>

```

(End of definition for `__scontents_bspack:` and `__scontents_espack:`.)

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

```

831 <*loader>
832 \NewDocumentCommand \Scontents { !s !O{} }
833 { \__scontents_Scontents_internal:nn {#1} {#2} }
834 </loader>
835 <*core>
836 \cs_new_protected:Npn \__scontents_Scontents_internal:nn #1 #2
837 {
838     \__scontents_bspack:
839     \group_begin:
840     \tl_if_novalue:nF {#2}
841     { \keys_set:nn { scontents / Scontents } {#2} }
842     \char_set_catcode_active:n { 9 }
843     \bool_if:NTF #1
844     { \__scontents_verb_arg:w }
845     { \__scontents_norm_arg:n }
846 }

```

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

```

847 \cs_new_protected:Npn \__scontents_norm_arg:n #1
848 {
849     \tl_set:Nn \__scontents_temp_tl {#1}
850     \__scontents_Scontents_finish:
851 }

```

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

```

852 </core>
853 <*loader>
854 \NewDocumentCommand \__scontents_verb_arg:w { +v }
855 { \__scontents_verb_arg_internal:n {#1} }
856 </loader>
857 <*core>

```

(End of 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` variable.

```

858 \cs_new_protected:Npn \__scontents_verb_arg_internal:n #1
859 {
860     \tl_set:Nn \__scontents_temp_tl {#1}
861     \tl_replace_all:Nxx \__scontents_temp_tl { \iow_char:N ^^M } { \iow_char:N ^^J }
862     \__scontents_Scontents_finish:
863 }
864 \cs_new_protected:Npn \__scontents_Scontents_finish:
865 {
866     \__scontents_file_write_cmd:VV \__scontents_fname_out_tl \__scontents_temp_tl
867     \__scontents_finish_storing:NNN

```

```

868     \l__scontents_temp_tl
869     \l__scontents_name_seq_cmd_tl
870     \l__scontents_print_cmd_bool
871     \use:x
872     {
873     \group_end:
874     \bool_if:NF \l__scontents_print_cmd_bool { \l__scontents_esphack: }
875     }
876 }
877 \cs_new_protected:Npn \l__scontents_file_write_cmd:nn #1#2
878 {
879     \l__scontents_file_if_writable:nT {#1}
880     {
881         \iow_open:Nn \l__scontents_file_iow {#1}
882         \iow_now:Nn \l__scontents_file_iow {#2}
883         \iow_close:N \l__scontents_file_iow
884     }
885 }
886 \cs_generate_variant:Nn \l__scontents_file_write_cmd:nn { VV }
887 \prg_new_protected_conditional:Npnn \l__scontents_file_if_writable:n #1 { T, F, TF }
888 {
889     \bool_if:NTF \l__scontents_writing_bool
890     {
891         \file_if_exist:nTF {#1}
892         {
893             \bool_if:NTF \l__scontents_overwrite_bool
894             {
895                 \msg_warning:nxx { scontents } { overwrite-file } {#1}
896                 \prg_return_true:
897             }
898             {
899                 \msg_warning:nxx { scontents } { not-writing } {#1}
900                 \prg_return_false:
901             }
902         }
903         {
904             \msg_warning:nxx { scontents } { writing-file } {#1}
905             \prg_return_true:
906         }
907     }
908     { \prg_return_false: }
909 }

```

(End of definition for `\l__scontents_verb_arg_internal:n`, `\l__scontents_Scontents_finish:`, and `\l__scontents_file_write_cmd:nn`.)

12.13 The command `\getstored`

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

```

\l__scontents_getstored_internal:nn
910 </core>
911 <*loader>
912 \NewDocumentCommand \getstored { 0{-1} m }
913 { \l__scontents_getstored_internal:nn {#1} {#2} }
914 </loader>
915 <*core>
916 \cs_new_protected:Npn \l__scontents_getstored_internal:nn #1 #2
917 {
918     \group_begin:
919     \int_set:Nn \tex_newlinechar:D { ``^^J }
920     \l__scontents_rescan_tokens:x
921     {
922         \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
923         \l__scontents_getfrom_seq:nn {#1} {#2}
924     }
925 }

```

(End of definition for `\getstored` and `\l__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
926  ⟨/core⟩
927  ⟨*loader⟩
928  \NewDocumentCommand \foreachsc { o m }
929    { \__scontents_foreachsc_internal:nn {#1} {#2} }
930  ⟨/loader⟩
931  ⟨*core⟩
932  \cs_new_protected:Npn \__scontents_foreachsc_internal:nn #1 #2
933    {
934      \group_begin:
935        \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / foreachsc } {#1} }
936        \tl_set:Nn \l__scontents_foreach_name_seq_tl {#2}
937        \seq_clear:N \l__scontents_foreach_print_seq
938        \bool_if:NF \l__scontents_foreach_stop_bool
939          {
940            \int_set:Nn \l__scontents_foreach_stop_int
941              { \seq_count:c { g__scontents_name_#2_seq } }
942          }
943        \int_step_function:nnnN
944          { \l__scontents_foreach_start_int }
945          { \l__scontents_foreach_step_int }
946          { \l__scontents_foreach_stop_int }
947          \__scontents_foreach_add_body:n
948        \tl_gset:Nx \g__scontents_temp_tl
949          {
950            \exp_args:NNV \seq_use:Nn
951              \l__scontents_foreach_print_seq \l__scontents_foreach_sep_tl
952          }
953        \group_end:
954        \exp_after:wN \tl_gclear:N
955        \exp_after:wN \g__scontents_temp_tl
956        \g__scontents_temp_tl
957    }
958  \cs_new_protected:Npn \__scontents_foreach_add_body:n #1
959    {
960      \seq_put_right:Nx \l__scontents_foreach_print_seq
961        {
962          \bool_if:NT \l__scontents_foreach_before_bool
963            { \exp_not:V \l__scontents_foreach_before_tl }
964          \bool_if:NTF \l__scontents_foreach_wrapper_bool
965            { \__scontents_foreach_wrapper:n }
966            { \use:n }
967            { \getstored [#1] { \tl_use:N \l__scontents_foreach_name_seq_tl } }
968          \bool_if:NT \l__scontents_foreach_after_bool
969            { \exp_not:V \l__scontents_foreach_after_tl }
970        }
971    }

```

(End of 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` The `\typestored` commands fetches a buffer from memory, prints it to the log file, and then calls

```

\__scontents_typedstored_internal:nn
\__scontents_verb_print:N
\__scontents_xverb:w
972  ⟨/core⟩
973  ⟨*loader⟩
974  \NewDocumentCommand \typestored { o m }
975    { \__scontents_typedstored_internal:nn {#1} {#2} }
976  ⟨/loader⟩
977  ⟨*core⟩
978  \cs_new_protected:Npn \__scontents_typedstored_internal:nn #1 #2
979    {
980      \__scontents_bsphack:
981      \group_begin:
982        \seq_clear:N \l__scontents_seq_item_seq
983        \str_set:Nx \l__scontents_cur_seq_name_str {#2}
984        \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / typemeaning } {#1} }

```

```

985 \seq_if_empty:NT \l__scontents_seq_item_seq
986 { \seq_set_from_clist:Nn \l__scontents_seq_item_seq { 1 } }
987 \tl_set:Nx \l__scontents_temp_tl
988 { \__scontents_getfrom_seq:Nn \l__scontents_seq_item_seq {#2} }
989 \__scontents_remove_trailing_eol:N \l__scontents_temp_tl
990 \tl_replace_all:Nxn \l__scontents_temp_tl \c__scontents_hidden_space_str { ^^J }
991 \tl_log:N \l__scontents_temp_tl
992 \tl_if_empty:NF \l__scontents_temp_tl
993 {
994   \bool_if:NT \l__scontents_print_aux_bool
995   {
996     \__scontents_verb_print:N \l__scontents_temp_tl
997   }
998 }
999 \__scontents_file_write_cmd:VV \l__scontents_fname_out_tl \l__scontents_temp_tl
1000 \use:x
1001 {
1002   \group_end:
1003   \bool_if:NF \l__scontents_print_aux_bool { \__scontents_espack: }
1004 }
1005 }

```

The `__scontents_verb_print:N` macro is defined with active carriage return (ASCII 13) characters to mimic 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`.

```

1006 \group_begin:
1007   \char_set_catcode_active:N ^^M
1008   \cs_new_protected:Npn \__scontents_verb_print:N #1
1009   {
1010     \tl_if_blank:VT #1
1011     { \msg_error:nnn { scontents } { empty-variable } {#1} }
1012     \cs_set_eq:NN \__scontents_verb_print_EOL: ^^M
1013     \cs_set_eq:NN ^^M \scan_stop:
1014     \cs_set_eq:cN { do@noligs } \__scontents_do_noligs:N
1015     \int_set:Nn \tex_newlinechar:D { `^^J }
1016     \__scontents_rescan_tokens:x
1017     {
1018       \__scontents_format_case:nnn
1019       { \exp_not:N \begin{verbatimsc} } % LaTeX
1020       { \verbatimsc } % Plain/Generic
1021       { \startverbatimsc } % ConTeXt
1022       ^^M
1023       \exp_not:V #1 ^^M
1024       \g__scontents_end_verbatimsc_tl
1025     }
1026     \cs_set_eq:NN ^^M \__scontents_verb_print_EOL:
1027   }
1028 \group_end:
1029 \cs_new_protected:Npn \__scontents_xverb:
1030 {
1031   \char_set_catcode_active:n { 9 }
1032   \char_set_active_eq:nN { 9 } \__scontents_tabs_to_spaces:
1033   \__scontents_xverb:w
1034 }
1035 </core>

```

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

12.16 The command `\mergesc`

`\mergesc` The `\mergesc` command parses a list given as argument, and just assembles it as a temporary internal sequence, then passes it to the requested command.

```

1036 <*loader>
1037 \NewDocumentCommand \mergesc { o m }
1038 { \__scontents_mergesc_internal:nn {#1} {#2} }
1039 </loader>
1040 <*core>
1041 \keys_define:nn { scontents / mergesc }
1042 {
1043   , typestored .code:n =

```

```

1044     { \cs_set_eq:NN \__scontents_mergesc_output_cmd:nn \__scontents_tpestored_internal:nn }
1045     , meaningsc .code:n =
1046     { \cs_set_eq:NN \__scontents_mergesc_output_cmd:nn \__scontents_meaningsc_internal:nn }
1047   }
1048 \cs_new_protected:Npn \__scontents_mergesc_output_cmd:nn #1 #2
1049 { \msg_error:nn { scontents } { mergesc-missing-cmd } }
1050 \msg_new:nnn { scontents } { mergesc-missing-cmd }
1051 { Missing-output~command~for~\iow_char:N\mergesc~\msg_line_context:. }
1052 \cs_new_protected:Npn \__scontents_mergesc_internal:nn #1 #2
1053 {
1054   \group_begin:
1055   \tl_clear:N \l__scontents_keys_tl
1056   \tl_if_novalue:nF {#1}
1057   {
1058     % Add print-cmd here :D
1059     \keys_define:nn { scontents / typemeaning }
1060     {
1061       print-cmd .bool_set:N = \l__scontents_print_aux_bool,
1062       print-cmd .initial:n = false,
1063       print-cmd .default:n = true,
1064     }
1065     \keys_set_known:nnN { scontents / mergesc } {#1} \l__scontents_keys_tl
1066   }
1067   \seq_gclear:c { g__scontents_name_sc!internal_seq }
1068   \__scontents_mergesc_parse_list:n {#2}
1069   \exp_args:Nx \__scontents_mergesc_output_cmd:nn
1070     { 1-end, \exp_not:V \l__scontents_keys_tl } { sc!internal }
1071   \group_end:
1072 }

1073 \cs_new_protected:Npn \__scontents_mergesc_parse_list:n #1
1074 {
1075   \clist_map_inline:nn {#1} { \__scontents_parse_mergesc:nw ##1 \s__scontents_stop }
1076   \seq_gpop_right:cN { g__scontents_name_sc!internal_seq } \l__scontents_temp_tl
1077   \__scontents_remove_trailing_eol:N \l__scontents_temp_tl
1078   \seq_gput_right:cV { g__scontents_name_sc!internal_seq } \l__scontents_temp_tl
1079 }
1080 \cs_new_protected:Npx \__scontents_remove_trailing_eol:N #1
1081 {
1082   \exp_not:N \exp_after:wN \exp_not:N \__scontents_remove_trailing_eol:w
1083   #1 \s__scontents_stop \c__scontents_hidden_space_str \s__scontents_stop \s__scontents_mark #1
1084 }
1085 \use:e
1086 {
1087   \cs_new_protected:Npn \exp_not:N \__scontents_remove_trailing_eol:w #1
1088     \c__scontents_hidden_space_str \s__scontents_stop #2 \s__scontents_mark #3
1089 } {
1090   \tl_set:Nx #3
1091   {
1092     \tl_if_empty:nTF {#2}
1093     { \exp_not:o { \__scontents_use_delimit_by_s_stop:nw #1 } }
1094     { \exp_not:n {#1} }
1095   }
1096 }
1097 \cs_new_protected:Npn \__scontents_parse_mergesc:nw #1
1098 {
1099   \peek_charcode_ignore_spaces:NTF [ % ]
1100   { \__scontents_parse_mergesc_aux:nw {#1} }
1101   { \__scontents_parse_mergesc_aux:nw {#1} [ 1-\seq_count:c { g__scontents_name_#1_seq } ] }
1102 }
1103 \cs_new_protected:Npn \__scontents_parse_mergesc_aux:nw #1 [#2]
1104 {
1105   \seq_clear:N \l__scontents_seq_item_seq
1106   \clist_map_inline:nn {#2}
1107   { \__scontents_parse_mergesc_range:nw {#1} ##1 - \q__scontents_mark - \s__scontents_mark }
1108   \seq_map_inline:Nn \l__scontents_seq_item_seq
1109   {
1110     \seq_gput_right:cx { g__scontents_name_sc!internal_seq }
1111     { \seq_item:cn { g__scontents_name_#1_seq } {##1} }
1112   }
1113   \__scontents_use_none_delimit_by_s_stop:w

```

```

1114 }
1115 \cs_new_protected:Npn \__scontents_parse_mergesc_range:nw #1 #2 - #3 - #4 \s__scontents_mark
1116 {
1117   \cs_set_protected:Npn \__scontents_tmp:w ##1
1118   {
1119     \msg_error:nnxxx { scontents } { index-out-of-range }
1120     {##1} {#1} { \seq_count:c { g__scontents_name_#1_seq } }
1121   }
1122   \__scontents_range_parser:nnxn {#2} {#3}
1123   { \seq_count:c { g__scontents_name_#1_seq } }
1124   { \__scontents_tmp:w }
1125 }
1126 </core>

```

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

`verbatimsc` Finally the \LaTeX and Con \TeX t version of `verbatimsc` environment is defined.

`\startverbatimsc` The macro `\endverbatim` in the second argument of the `verbatimsc` environment is only needed for compatibility with the `verbatim` package.

`\stopverbatimsc`

```

1127 <*loader>
1128 <!context>
1129 \use:x
1130 {
1131   \cs_new_protected:Npn \exp_not:N \__scontents_xverb:w
1132   ##1 \g__scontents_end_verbatimsc_tl
1133   <latex> { ##1 \exp_not:N \end{verbatimsc} }
1134   <plain> { ##1 \exp_not:N \endverbatimsc }
1135   <context> { ##1 \exp_not:N \stopverbatimsc }
1136 }
1137 </!context>
1138 <*latex>
1139 \NewDocumentEnvironment { verbatimsc } { }
1140 {
1141   \cs_set_eq:cN { @xverbatim } \__scontents_xverb:
1142   \verbatim
1143 }
1144 { \endverbatim }
1145 </latex>
1146 <context>\definetyping[verbatimsc]
1147 </loader>
1148 <*core>

```

(End of definition for `verbatimsc`, `\startverbatimsc`, and `\stopverbatimsc`. These functions are documented on page 7.)

12.16.1 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.

```

1149 \cs_new:Npn \__scontents_tabs_to_spaces:
1150 { \prg_replicate:nn { \l__scontents_tab_width_int } { ~ } }

```

(End of definition for `__scontents_tabs_to_spaces:`.)

`__scontents_do_noligs:` `__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*.

```

1151 \cs_new_protected:Npn \__scontents_do_noligs:N #1
1152 {
1153   \char_set_catcode_active:N #1
1154   \cs_set:cpx { __scontents_active_char_ \token_to_str:N #1 : }
1155   {
1156     \mode_leave_vertical:
1157     \tex_kern:D \c_zero_dim
1158     \tex_char:D \exp_not:N #1
1159   }
1160   \char_set_active_eq:Nc #1 { __scontents_active_char_ \token_to_str:N #1 : }
1161 }

```

(End of 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`.

```

1162 \\prg_new_protected_conditional:Npnn \\_scontents_tl_if_head_is_q_mark:n #1
1163 { T, F, TF }
1164 {
1165   \\exp_after:wN \\if_meaning:w
1166     \\exp_after:wN \\q_\\_scontents_mark \\_scontents_use_i_delimit_by_s_stop:nw #1 ? \\s_\\_scontents_stop
1167     \\prg_return_true:
1168   \\else:
1169     \\prg_return_false:
1170   \\fi:
1171 }

```

(End of 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.

```

1172 \\cs_new_protected:Npn \\_scontents_set_active_eq:NN #1
1173 {
1174   \\char_set_catcode_active:N #1
1175   \\char_set_active_eq:NN #1
1176 }
1177 </core>
1178 <*loader>
1179 \\group_begin:
1180 <plain> \\char_set_catcode_active:n { \\* }
1181 \\cs_new_protected:Npn \\_scontents_plain_disable_outer_par:
1182 <*plain>
1183 {
1184   \\group_begin:
1185     \\char_set_lccode:nn { \\* } { \\^^L }
1186     \\tex_lowercase:D { \\group_end:
1187       \\tex_let:D * \\scan_stop:
1188     }
1189   }
1190 </plain>
1191 <latex|context> { }
1192 \\group_end:
1193 </loader>
1194 <*core>
1195 \\group_begin:
1196   \\char_set_catcode_active:N \\*
1197   \\cs_new_protected:Npn \\_scontents_make_control_chars_active:
1198   {
1199     \\_scontents_plain_disable_outer_par:
1200     \\_scontents_set_active_eq:NN \\^^I \\_scontents_tab:
1201     \\_scontents_set_active_eq:NN \\^^L \\_scontents_par:
1202     \\_scontents_set_active_eq:NN \\^^M \\_scontents_ret:w
1203   }
1204 \\group_end:

```

(End of definition for `_scontents_set_active_eq:NN`, `_scontents_make_control_chars_active:`, and `_scontents_plain_disable_outer_par:`.)

12.17 The command `\\setupsc`

User command `\\setupsc` to setup module.

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

```

1205 </core>
1206 <*loader>
1207 \\NewDocumentCommand \\setupsc { +m }
1208 { \\keys_set:nn { scontents } {#1} }
1209 </loader>
1210 <*core>

```

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

12.18 The command `\meaningc`

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

```

1211 </core>
1212 <*loader>
1213 \NewDocumentCommand \meaningc { o m }
1214 { \__scontents_meaningc_internal:nn {#1} {#2} }
1215 </loader>
1216 <*core>
1217 \cs_new_protected:Npn \__scontents_meaningc_internal:nn #1 #2
1218 {
1219   \group_begin:
1220     \seq_clear:N \l__scontents_seq_item_seq
1221     \str_set:Nx \l__scontents_cur_seq_name_str {#2}
1222     \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / typemeaning } {#1} }
1223     \seq_if_empty:NT \l__scontents_seq_item_seq
1224       { \seq_set_from_clist:Nn \l__scontents_seq_item_seq { 1 } }
1225     \__scontents_meaningc:n {#2}
1226   \group_end:
1227 }
1228 \group_begin:
1229 \char_set_catcode_active:N ^^I
1230 \cs_new_protected:Npn \__scontents_meaningc:n #1
1231 {
1232   \tl_set:Nx \l__scontents_temp_tl
1233     { \__scontents_getfrom_seq:Nn \l__scontents_seq_item_seq {#1} }
1234   \tl_replace_all:Nxn \l__scontents_temp_tl { \iow_char:N ^^J } { ~ }
1235   \tl_replace_all:Nxn \l__scontents_temp_tl \c__scontents_hidden_space_str { ~ }
1236   \tl_log:N \l__scontents_temp_tl
1237   \tl_use:N \l__scontents_verb_font_tl
1238   \tl_replace_all:Nnx \l__scontents_temp_tl { ^^I } { \__scontents_tabs_to_spaces: }
1239   \cs_replacement_spec:N \l__scontents_temp_tl
1240 }
1241 \group_end:

```

(End of definition for `\meaningc`, `__scontents_meaningc_internal:nn`, and `__scontents_meaningc:n`. This function is documented on page 7.)

12.19 The command `\countsc`

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

```

1242 </core>
1243 <*loader>
1244 \NewExpandableDocumentCommand \countsc { m }
1245 { \seq_count:c { g__scontents_name_#1_seq } }
1246 </loader>
1247 <*core>

```

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

12.20 The command `\cleanseqsc`

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

```

1248 </core>
1249 <*loader>
1250 \NewDocumentCommand \cleanseqsc { m }
1251 { \seq_gclear_new:c { g__scontents_name_#1_seq } }
1252 </loader>
1253 <*core>

```

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

12.21 Warning and error messages

Warning and error messages used throughout the package.

```

1254 \msg_new:nnn { scontents } { junk-after-begin }
1255 {
1256   Junk~characters~#1~\msg_line_context: :
1257   \\ \\

```

```

1258     #2
1259   }
1260   \msg_new:nnnn { scontents } { env-already-defined }
1261   { Environment~'#1'~already~defined! }
1262   {
1263     You~have~used~\newenvsc
1264     with~an~environment~that~already~has~a~definition. \ \ \
1265     The~existing~definition~of~'#1'~will~not~be~altered.
1266   }
1267   \msg_new:nnn { scontents } { empty-stored-content }
1268   { Empty~value~for~key~'getstored'~\msg_line_context:. }
1269   \msg_new:nnn { scontents } { empty-variable }
1270   { Variable~'#1'~empty~\msg_line_context:. }
1271   \msg_new:nnn { scontents } { overwrite-file }
1272   { Overwriting~file~'#1'. }
1273   \msg_new:nnn { scontents } { writing-file }
1274   { Writing~file~'#1'. }
1275   \msg_new:nnn { scontents } { not-writing }
1276   { File~'#1'~already~exists.~Not~writing. }
1277   \msg_new:nnn { scontents } { rescanning-text }
1278   { Rescanning~text~'#1'~after~\c_backslash_str end{#2}~\msg_line_context:. }
1279   \msg_new:nnn { scontents } { multiple-begin }
1280   { Multiple~\c_backslash_str begin{ \l__scontents_env_name_tl }~\msg_line_context:. }
1281   \msg_new:nnn { scontents } { undefined-storage }
1282   { Storage~named~'#1'~is~not~defined. }
1283   \msg_new:nnn { scontents } { index-out-of-range }
1284   {
1285     \int_compare:nNnTF {#1} = { 0 }
1286     { Index~of~sequence~cannot~be~zero. }
1287     {
1288       Index~'#1'~out~of~range~for~'#2'.~
1289       \int_compare:nNnTF {#1} > { 0 }
1290       { Max = } { Min = - } #3.
1291     }
1292   }
1293   \msg_new:nnnn { scontents } { env-key-unknown }
1294   {
1295     The~key~'#1'~is~unknown~by~environment~
1296     '\l__scontents_env_name_tl'~and~is~being~ignored.
1297   }
1298   {
1299     The~environment~'\l__scontents_env_name_tl'~does~not~have~a~key~called~'#1'. \ \
1300     Check~that~you~have~spelled~the~key~name~correctly.
1301   }
1302   \msg_new:nnnn { scontents } { env-key-value-unknown }
1303   {
1304     The~key~'#1=#2'~is~unknown~by~environment~
1305     '\l__scontents_env_name_tl'~and~is~being~ignored.
1306   }
1307   {
1308     The~environment~'\l__scontents_env_name_tl'~does~not~have~a~key~called~'#1'. \ \
1309     Check~that~you~have~spelled~the~key~name~correctly.
1310   }
1311   \msg_new:nnnn { scontents } { cmd-key-unknown }
1312   { The~key~'#1'~is~unknown~by~'\c_backslash_str Scontents'~and~is~being~ignored. }
1313   {
1314     The~command~'\c_backslash_str Scontents'~does~not~have~a~key~called~'#1'. \ \
1315     Check~that~you~have~spelled~the~key~name~correctly.
1316   }
1317   \msg_new:nnnn { scontents } { cmd-key-value-unknown }
1318   { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str Scontents'~and~is~being~ignored. }
1319   {
1320     The~command~'\c_backslash_str Scontents'~does~not~have~a~key~called~'#1'. \ \
1321     Check~that~you~have~spelled~the~key~name~correctly.
1322   }
1323   \msg_new:nnnn { scontents } { for-key-unknown }
1324   { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachsc'~and~is~being~ignored. }
1325   {
1326     The~command~'\c_backslash_str foreachsc'~does~not~have~a~key~called~'#1'. \ \
1327     Check~that~you~have~spelled~the~key~name~correctly.
1328   }

```

```

1329 \msg_new:nnnn { scontents } { for-key-value-unknown }
1330 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachsc'~and~is~being~ignored. }
1331 {
1332   The~command~'\c_backslash_str foreachsc'~does~not~have~a~key~called~'#1'.\\
1333   Check~that~you~have~spelled~the~key~name~correctly.
1334 }
1335 \msg_new:nnnn { scontents } { type-key-unknown }
1336 { The~key~'#1'~is~unknown~and~is~being~ignored. }
1337 {
1338   This~command~does~not~have~a~key~called~'#1'.\\
1339   This~command~only~accepts~the~key~'width-tab'.
1340 }
1341 \msg_new:nnnn { scontents } { type-key-value-unknown }
1342 { The~key~'#1'~to~which~you~passed~'#2'~is~unknown~and~is~being~ignored. }
1343 {
1344   This~command~does~not~have~a~key~called~'#1'.\\
1345   This~command~only~accepts~the~key~'width-tab'.
1346 }
1347 \msg_new:nnn { scontents } { empty-environment }
1348 { environment~'#1'~empty~\msg_line_context:. }
1349 \msg_new:nnnn { scontents } { verbatim-newline }
1350 { Verbatim~argument~of~#1~ended~by~end~of~line. }
1351 {
1352   The~verbatim~argument~of~the~#1~cannot~contain~more~than~one~line,~
1353   but~the~end~
1354   of~the~current~line~has~been~reached.~You~may~have~forgotten~the~
1355   closing~delimiter.
1356   \\ \\
1357   LaTeX~will~ignore~'#2'.
1358 }
1359 \msg_new:nnnn { scontents } { verbatim-tokenized }
1360 { The~verbatim~#1~cannot~be~used~inside~an~argument. }
1361 {
1362   The~#1~takes~a~verbatim~argument.~
1363   It~may~not~appear~within~the~argument~of~another~function.~
1364   It~received~an~illegal~token \tl_if_empty:nF {#3} { ~'#3' } .
1365   \\ \\
1366   LaTeX~will~ignore~'#2'.
1367 }

```

12.22 Finish package

Finish package implementation.

```

1368 </core>
1369 <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 | |
| ' | 786 |
| * | 1180, 1185, 1196 |
| , | 786 |
| - | 786 |
| < | 786 |
| > | 786 |
| \\ | 42, 922, 1051, 1257, 1264, 1299, 1308, 1314, 1320, 1326, 1332, 1338, 1344, 1356, 1365 |
| ` | 786 |
| B | |
| \beginngroup | 68, 73 |
| bool commands: | |
| \bool_if:NTF | 614, 619, 645, 647, 736, 738, 741, 764, 843, 874, 889, 893, 938, 962, 964, 968, 994, 1003 |
| \bool_lazy_and:nnTF | 356, 524 |
| \bool_lazy_or:nnTF | 402, 583 |
| \bool_new:N | 114, 175, 177, 179, 180, 182, 184, 186, 288 |
| \bool_set_false:N | 176, 181, 183, 185, 187, 217, 239, 294, 539, 761 |
| \bool_set_true:N | 178, 213, 218, 235, 240, 257, 262, 270, 278, 289, 295, 536, 772 |
| C | |
| \catcode | 69, 922 |
| char commands: | |
| \char_set_active_eq:NN | 595, 1160, 1175 |
| \char_set_active_eq:nN | 1032 |
| \char_set_catcode:nn | 121 |
| \char_set_catcode_active:N | 465, 494, 495, 496, 1007, 1153, 1174, 1196, 1229 |
| \char_set_catcode_active:n | 842, 1031, 1180 |
| \char_set_catcode_letter:N | 119 |
| \char_set_catcode_letter:n | 542 |
| \char_set_catcode_other:N | 42, 44, 45, 541, 777 |
| \char_set_lccode:nn | 1185 |
| \l_char_special_seq | 33, 541, 796 |
| \char_value_catcode:n | 118 |
| \cleanseqsc | 7, 41, <u>1248</u> |
| clist commands: | |
| \clist_map_inline:nn | 1075, 1106 |
| \countsc | 7, 41, <u>1242</u> |
| cs commands: | |
| \cs:w | 435, 436, 604, 605 |
| \cs_end: | 435, 436, 609, 610 |
| \cs_generate_variant:Nn | 203, 206, 207, 208, 340, 376, 416, 651, 652, 886 |
| \cs_gset_eq:NN | 828, 829 |
| \cs_if_exist:NTF | 32, 442, 443, 793 |
| \cs_new:Npn | 62, 197, 198, 199, 205, 377, 386, 388, 390, 400, 553, 563, 600, 626, 636, 642, 673, 795, 1149 |
| \cs_new:Npx | 204 |
| \cs_new_eq:NN | 727 |
| \cs_new_protected:Npn | 202, 302, 304, 310, 312, 318, 324, 326, 328, 334, 341, 347, 368, 408, 417, 422, 424, 431, 447, 450, 455, 463, 468, 483, 497, 499, 509, 511, 521, 531, 612, 643, 653, 660, 665, 676, 694, 706, 716, 729, 734, 746, 752, 754, 785, 787, 789, 800, 809, 836, 847, 858, 864, 877, 916, 932, 958, 978, 1008, 1029, 1048, 1052, 1073, 1087, 1097, 1103, 1115, 1131, 1151, 1172, 1181, 1197, 1217, 1230 |
| \cs_new_protected:Npx | 687, 1080 |
| \cs_replacement_spec:N | 1239 |
| \cs_set:Npn | 557, 762 |
| \cs_set:Npx | 1154 |
| \cs_set_eq:NN | 777, 780, 1012, 1013, 1014, 1026, 1044, 1046, 1141 |
| \cs_set_protected:Npn | 279, 720, 1117 |
| \cs_set_protected:Npx | 543, 580 |
| \csname | 72, 82 |
| D | |
| \def | 2, 3, 5, 6, 71, 74, 75, 83, 96 |
| \definetyping | 1146 |
| dim commands: | |
| \dim_compare:nNnTF | 815 |
| \dim_zero:N | 757 |
| \c_zero_dim | 1157 |
| \do | 777, 780, 786, 796 |
| \dospecials | 778, <u>792</u> |
| E | |
| \else | 93, 95 |
| else commands: | |
| \else: | 1168 |
| \end | 48, 603, 1133 |
| \endcsname | 72, 82 |
| \endgroup | 71, 74, 77, 90, 104, 922 |
| \endinput | 91, 105 |
| \endlinechar | 70 |
| \endscontents | 4, <u>474</u> |
| \endverbatim | 1144 |
| \endverbatimsc | 49, <u>745</u> , 1134 |
| Environments: | |
| scontents | 26, 27 |
| \errhelp | 78 |
| \errmessage | 79 |
| exp commands: | |
| \exp_after:wN | 604, 605, 782, 954, 955, 1082, 1165, 1166 |
| \exp_args:Nf | 381, 394, 630 |
| \exp_args:Nnf | 389 |
| \exp_args:NNV | 950 |
| \exp_args:Nooo | 433 |
| \exp_args:NV | 303, 311, 325, 327, 622 |
| \exp_args:Nx | 1069 |
| \exp_args:Nxx | 343 |
| \exp_not:N | 58, 545, 548, 557, 560, 603, 604, 605, 621, 676, 691, 692, 711, 713, 720, 722, 1019, 1082, 1087, 1131, 1133, 1134, 1135, 1158 |
| \exp_not:n | 344, 345, 420, 575, 634, 639, 640, 963, 969, 1023, 1070, 1093, 1094 |
| \expandafter | 72, 82 |
| \ExplSyntaxOff | 35, 1369 |
| \ExplSyntaxOn | 24 |
| F | |
| \fi | 81, 107, 108 |
| fi commands: | |
| \fi: | 603, 608, 1170 |

file commands:

\file_if_exist:nTF 891
 \file_input:n 61, 120
 \file_input_stop: 36
 \foreachsc 6, 23, 24, 36, 926
 \frenchspacing 749

G

\getstored 6, 35, 910, 967
 group commands:

\group_begin: . 452, 493, 533, 748, 839, 918, 934, 981,
 1006, 1054, 1179, 1184, 1195, 1219, 1228
 \group_end: . 458, 618, 650, 753, 873, 953, 1002, 1028,
 1071, 1186, 1192, 1204, 1226, 1241
 \group_insert_after:N 411, 412, 413, 414

I

if commands:

\if_false: 603, 608
 \if_meaning:w 1165
 \IfFormatAtLeastTF 8, 159
 \ifx 72, 82, 94
 \input 23
 int commands:

\int_abs:n 404
 \int_compare:nNnTF 360, 657, 669, 1285, 1289
 \int_compare_p:nNn 403, 404
 \int_decr:N 666
 \int_eval:n 352, 389
 \int_incr:N 662, 663
 \int_new:N 113, 172, 174, 200
 \int_set:Nn 118, 271, 919, 940, 1015
 \int_set_eq:NN 806, 814
 \int_step_function:nnnN 542, 943
 \int_step_inline:nnnn 361, 362
 \int_to_roman:n 24, 351, 357, 358
 \int_zero:N 655
 \c_zero_int 669
 \interlinepenalty 769, 774
 iow commands:

\iow_char:N 861, 1051, 1234
 \iow_close:N 615, 883
 \iow_log:n 27
 \iow_new:N 168
 \iow_now:Nn 646, 882
 \iow_open:Nn 537, 881

K

keys commands:

\keys_define:nn .. 124, 155, 210, 232, 254, 290, 1041,
 1059
 \l_keys_key_str 24, 303, 311, 325, 327
 \keys_set:nn . 40, 427, 489, 841, 935, 984, 1208, 1222
 \keys_set_known:nnN 1065

M

\meaningsc 7, 23, 24, 41, 1211
 \mergesc 6, 37, 1036
 mode commands:

\mode_if_horizontal:TF 773, 803, 812
 \mode_leave_vertical: 766, 790, 1156

msg commands:

\msg_error:nn 371, 1049
 \msg_error:nnn 307, 315, 321, 338, 444, 1011
 \msg_error:nnnn 208, 308, 316, 322, 332, 515
 \msg_error:nnnnn 1119

\msg_expandable_error:nnn 384, 398
 \msg_expandable_error:nnnnn 405
 \msg_gset:nnn 30
 \msg_line_context: 31, 1051, 1256, 1268, 1270, 1278,
 1280, 1348
 \msg_new:nnn 1050, 1254, 1267, 1269, 1271, 1273, 1275,
 1277, 1279, 1281, 1283, 1347
 \msg_new:nnnn 1260, 1293, 1302, 1311, 1317, 1323, 1329,
 1335, 1341, 1349, 1359
 \msg_warning:nn 34, 658
 \msg_warning:nnn 528, 895, 899, 904
 \msg_warning:nnnn 589

N

\NewDocumentCommand 440, 479, 832, 854, 912, 928, 974, 1037,
 1207, 1213, 1250
 \NewDocumentEnvironment 449, 1139
 \newenvsc 5, 18, 26, 422, 475, 1263
 \NewExpandableDocumentCommand 1244
 \next 71, 74, 83, 96, 109
 \nobreak 790, 819
 \null 767

O

\obeylines 778

P

\PackageError 75, 85, 98
 Packages:
 l3keys2e 20
 scontents 17, 18, 22
 xparse 27
 \PackageWarning 12
 \par 762
 \parfillskip 758
 \parindent 757
 \parskip 756, 759
 peek commands:
 \peek_charcode_ignore_spaces:NnTF 1099
 \penalty 769, 774
 prg commands:
 \prg_generate_conditional_variant:Nnn .. 209
 \prg_new_protected_conditional:Npnn . 667, 887,
 1162
 \prg_replicate:nn 1150
 \prg_return_false: 671, 900, 908, 1169
 \prg_return_true: 670, 896, 905, 1167
 \ProcessKeyOptions 161
 \ProcessKeysOptions 163
 \ProvidesExplPackage 17

Q

quark commands:

\quark_new:N 192, 193

quark internal commands:

\q_scontents_mark 40, 192, 331, 691, 713, 1107, 1166
 \q_scontents_stop 192, 548, 560, 673, 692, 701, 704,
 713, 722

R

\relax 72, 82
 \RequirePackage 11

S

scan commands:

\scan_new:N 194, 195
 \scan_stop: 802, 811, 1013, 1187

scan internal commands:

- \s__scontents_mark 194, 331, 334, 1083, 1088, 1107,
 1115
- \s__scontents_stop . 194, 197, 198, 199, 1075, 1083,
 1088, 1166
- \Scontents 5, 22, 24, 33, 831
- \scontents 4, 474
- scontents 4, 474
- scontents internal commands:
- __scontents_analyse_nesting:n 31, 567, 653, 653
- __scontents_analyse_nesting:w 653
- __scontents_analyse_nesting_format:n 656, 727,
 729
- __scontents_analyse_nesting_generic:nn . 706,
 730, 731
- __scontents_analyse_nesting_generic:w . . 701,
 704, 711, 720
- __scontents_analyse_nesting_generic_
 process:nn 694, 723
- __scontents_analyse_nesting_latex:n 687, 728
- __scontents_analyse_nesting_latex:w 676, 684,
 689
- __scontents_append_contents:nn 25, 368, 368, 376,
 420
- __scontents_bsphack: . . 21, 799, 800, 828, 838, 980
- __scontents_check_line_process:nn 28, 493, 505,
 507, 510, 511
- \l__scontents_cur_seq_name_str . . 196, 337, 983,
 1221
- __scontents_define_generic_nesting_
 function:n 708, 716
- __scontents_do_noligs:N 39, 780, 1014, 1151, 1151
- \c__scontents_end_env_tl 18, 51, 546, 547, 558, 559,
 575
- \g__scontents_end_verbatimsc_tl . . 17, 39, 1024,
 1132
- __scontents_env_define:nnn . . . 26, 422, 434, 447
- __scontents_env_end_function: 582, 600
- __scontents_env_generic_begin: 26, 27, 429, 463,
 463
- __scontents_env_generic_end: 26, 27, 432, 463, 468
- \l__scontents_env_name_tl 18, 26, 51, 426, 516, 528,
 590, 606, 682, 712, 721, 1280, 1296, 1299, 1305, 1308
- \l__scontents_env_nesting_int . . 20, 31, 172, 662,
 666, 669
- __scontents_env_setting:nn . . . 26, 422, 422, 445
- __scontents_esphack: . 21, 799, 809, 829, 874, 1003
- \l__scontents_every_line_env_tl 20, 166, 540, 622,
 648
- __scontents_file_if_writable:n 887
- __scontents_file_if_writable:nTF . . . 534, 879
- \l__scontents_file_iow 20, 166, 537, 615, 646, 881,
 882, 883
- __scontents_file_tl_write_start:n . 20, 28, 519,
 531, 531, 652
- __scontents_file_write_cmd:nn 858, 866, 877, 886,
 999
- __scontents_file_write_stop:N 28, 523, 531, 612
- __scontents_finish_storing:NNN . 471, 734, 734,
 867
- \l__scontents_fname_out_tl . 20, 166, 214, 219, 236,
 241, 296, 519, 866, 999
- \l__scontents_forced_eol_bool 140, 738
- __scontents_foreach_add_body:n . 926, 947, 958
- \l__scontents_foreach_after_bool . 180, 262, 968
- \l__scontents_foreach_after_tl 20, 169, 263, 969
- \l__scontents_foreach_before_bool 180, 257, 962
- \l__scontents_foreach_before_tl 20, 169, 258, 963
- \l__scontents_foreach_name_seq_tl . 20, 169, 936,
 967
- \l__scontents_foreach_print_seq 21, 188, 937, 951,
 960
- \l__scontents_foreach_sep_tl 283, 951
- \l__scontents_foreach_start_int 266, 944
- \l__scontents_foreach_step_int 274, 945
- \l__scontents_foreach_stop_bool . 180, 270, 938
- \l__scontents_foreach_stop_int 20, 172, 271, 940,
 946
- __scontents_foreach_wrapper:n 280, 965
- \l__scontents_foreach_wrapper_bool 180, 278, 964
- __scontents_foreachsc_internal:nn 926, 929, 932
- __scontents_format_case:nnn 62, 62, 602, 607, 1018
- __scontents_getfrom_seq:Nn . 377, 377, 988, 1233
- __scontents_getfrom_seq:nn . . . 25, 377, 390, 923
- __scontents_getfrom_seq:nNn 381, 386
- __scontents_getfrom_seq:nnn . 377, 389, 394, 400
- __scontents_getfrom_seq_aux:nnn 387, 388
- __scontents_getstored_internal:nn 910, 913, 916
- __scontents_grab_optional:n 477, 480, 483
- __scontents_grab_optional:w 27, 28, 477, 479, 504
- \c__scontents_hidden_space_str 21, 190, 587, 739,
 990, 1083, 1088, 1235
- __scontents_if_nested: 31, 667
- __scontents_if_nested:TF 571, 653
- \l__scontents_keys_tl 115, 1055, 1065, 1070
- __scontents_lastfrom_seq:n 26, 408, 408, 416, 741
- \l__scontents_macro_tmp_tl . 19, 110, 471, 523, 526
- __scontents_make_control_chars_active: . 518,
 550, 1172, 1197
- __scontents_meaningsc:n 1211, 1225, 1230
- __scontents_meaningsc_internal:nn . 1046, 1211,
 1214, 1217
- __scontents_mergesc_internal:nn . . . 1038, 1052
- __scontents_mergesc_output_cmd:nn . 1044, 1046,
 1048, 1069
- __scontents_mergesc_parse_list:n . . 1068, 1073
- \l__scontents_name_seq_cmd_tl 129, 869
- \l__scontents_name_seq_env_tl 126, 472
- __scontents_nesting_decr: 31, 573, 653, 665
- __scontents_nesting_incr: 660, 683, 700
- __scontents_nolig_list: 745, 781, 785
- __scontents_norm_arg:n 34, 831, 845, 847
- __scontents_normalise_line_ends:N 27, 488, 497
- \l__scontents_overwrite_bool 143, 893
- __scontents_par: 204, 205, 1201
- __scontents_parse_command_keys:n . 24, 252, 310,
 310
- __scontents_parse_command_keys:nn 310, 311, 312
- __scontents_parse_environment_keys:n 24, 230,
 302, 302
- __scontents_parse_environment_keys:nn . . 302,
 303, 304
- __scontents_parse_foreach_keys:n . 24, 286, 318,
 324
- __scontents_parse_foreach_keys:nn 318, 318, 325
- __scontents_parse_mergesc:nw 1075, 1097
- __scontents_parse_mergesc_aux:nw . . 1100, 1101,
 1103

__scontents_parse_mergesc_range:nw 1107, 1115
 __scontents_parse_type_meaning_key:n 300, 326, 326
 __scontents_parse_type_meaning_key:nn .. 326, 327, 328
 __scontents_parse_type_meaning_range:w . 331, 334
 __scontents_parse_typemeaning_key:n 24
 __scontents_plain_disable_outer_par: . 1172, 1181, 1199
 \l__scontents_print_aux_bool 288, 289, 994, 1003, 1061
 \l__scontents_print_cmd_bool .. 26, 137, 870, 874
 \l__scontents_print_env_bool 26, 134, 472
 __scontents_range_parser:nnnn .. 336, 340, 341, 1122
 __scontents_range_parser_aux:nnn ... 343, 347
 __scontents_remove_leading_nl:n .. 29, 531, 622, 626
 __scontents_remove_leading_nl:nn ... 631, 636
 __scontents_remove_leading_nl:w . 531, 639, 642
 __scontents_remove_trailing_eol:N . 989, 1077, 1080
 __scontents_remove_trailing_eol:w . 1082, 1087
 __scontents_rescan_tokens:n 37, 32, 202, 202, 203, 411, 592, 920, 1016
 __scontents_ret:w ... 29, 543, 551, 580, 595, 1202
 \l__scontents_save_sf_int 200, 806, 814
 \l__scontents_save_skip 200, 805, 815
 __scontents_Scontents_finish: 850, 858, 862, 864
 __scontents_Scontents_internal:nn 34, 831, 833, 836
 \l__scontents_seq_item_seq . 20, 173, 352, 363, 982, 985, 986, 988, 1105, 1108, 1220, 1223, 1224, 1233
 __scontents_set_active_eq:NN . 788, 1172, 1172, 1200, 1201, 1202
 __scontents_setup_verb_processor: 26, 428, 531, 553
 __scontents_start_after_option:w . 28, 491, 493, 509
 __scontents_start_environment:w . 466, 493, 499
 __scontents_stop_environment: .. 470, 493, 521
 __scontents_store_to_seq: 32
 __scontents_store_to_seq:NN .. 26, 417, 417, 740
 \l__scontents_storing_bool . 20, 30, 175, 217, 239, 294, 525, 619, 647, 736
 __scontents_tab: 204, 204, 1200
 \l__scontents_tab_width_int 146, 1150
 __scontents_tabs_to_spaces: 39, 1032, 1149, 1149, 1238
 \l__scontents_temp_bool ... 19, 110, 761, 764, 772
 \g__scontents_temp_tl . 19, 110, 410, 412, 414, 948, 955, 956
 \l__scontents_temp_tl . 19, 110, 487, 488, 489, 849, 860, 861, 866, 868, 987, 989, 990, 991, 992, 996, 999, 1076, 1077, 1078, 1232, 1234, 1235, 1236, 1238, 1239
 __scontents_tl_if_head_is_q_mark:n 1162
 __scontents_tl_if_head_is_q_mark:nTF 349, 679, 698, 1162
 __scontents_tmp:w 1117, 1124
 \l__scontents_tmpa_int 19, 110, 118, 121, 655, 657, 663
 __scontents_tpestored_internal:nn . 972, 975, 978, 1044
 __scontents_use_delimit_by_s_stop:nw 197, 1093
 __scontents_use_i_delimit_by_s_stop:nw . 197, 198, 1166
 __scontents_use_none_delimit_by_q_stop:w 653, 673, 680
 __scontents_use_none_delimit_by_s_stop:w 197, 199, 1113
 __scontents_verb_arg:w 34, 831, 844, 854
 __scontents_verb_arg_internal:n .. 34, 855, 858, 858
 \l__scontents_verb_font_tl 132, 779, 1237
 __scontents_verb_print:N .. 36, 37, 972, 996, 1008
 __scontents_verb_print_EOL: 1012, 1026
 __scontents_verb_processor_iterate:nnn . 531, 561, 563
 __scontents_verb_processor_iterate:w 531, 545, 557
 __scontents_verb_processor_output:n . 30, 568, 574, 579, 643, 643, 651
 __scontents_verbatimsc_aux: 745, 749, 754
 __scontents_vobeyspaces: 745, 749, 787
 \l__scontents_writable_bool 175, 536, 539, 614, 645
 \l__scontents_writing_bool . 20, 30, 175, 213, 218, 235, 240, 295, 889
 __scontents_xobeysp: 745, 788, 789
 __scontents_xverb: 745, 750, 1029, 1141
 __scontents_xverb:w 972, 1033, 1131
 \Scontents* 24
 \ScontentsCoreFileDate 3, 94
 \ScontentsFileDate 2, 18, 25, 94
 \ScontentsFileDescription 6, 18, 26
 \ScontentsFileVersion 5, 18, 21, 26
 seq commands:
 \seq_clear:N 937, 982, 1105, 1220
 \seq_count:N 337, 382, 395, 941, 1101, 1120, 1123, 1245
 \seq_gc_clear:N 1067
 \seq_gc_clear_new:N 1251
 \seq_gpop_right:NN 1076
 \seq_gput_right:Nn 374, 1078, 1110
 \seq_if_empty:NTF 985, 1223
 \seq_if_exist:NTF 372, 379, 392
 \seq_item:Nn 406, 410, 1111
 \seq_map_function:NN 541, 796
 \seq_map_inline:Nn 1108
 \seq_map_tokens:Nn 387
 \seq_new:N 173, 188, 189, 373
 \seq_put_right:Nn 352, 363, 960
 \seq_set_from_clist:Nn 986, 1224
 \seq_use:Nn 950
 \setupsc 3, 40, 1205
 skip commands:
 \skip_horizontal:n 820
 \skip_if_eq:nnTF 817
 \skip_new:N 201
 \skip_set:Nn 758, 759
 \skip_set_eq:NN 805
 \skip_vertical:N 756
 \c_zero_skip 815, 817, 820
 \space 25, 26
 \startscontents 4, 474
 \startverbatim 1021, 1127
 \stopscontents 4, 474
 \stopverbatim 50, 1127
 str commands:
 \c_backslash_str .. 54, 516, 677, 690, 712, 721, 1278,

| | |
|---|--|
| 1280, 1312, 1314, 1318, 1320, 1324, 1326, 1330, 1332 | |
| <code>\c_circumflex_str</code> | 191 |
| <code>\c_left_brace_str</code> | 57, 677, 691 |
| <code>\c_percent_str</code> | 191, 585 |
| <code>\c_right_brace_str</code> | 59, 677, 691 |
| <code>\str_const:Nn</code> | 190 |
| <code>\str_if_eq:nnTF</code> | 344, 345, 503, 587, 682 |
| <code>\str_if_eq_p:nn</code> | 585 |
| <code>\str_new:N</code> | 196 |
| <code>\str_set:Nn</code> | 983, 1221 |
| T | |
| TeX and \LaTeX z_{ϵ} commands: | |
| <code>\@</code> | 118, 119, 121 |
| <code>\@bsphack</code> | 828 |
| <code>\@esphack</code> | 829 |
| tex commands: | |
| <code>\tex_char:D</code> | 1158 |
| <code>\tex_everypar:D</code> | 782, 783 |
| <code>\tex_ignorespaces:D</code> | 822 |
| <code>\tex_kern:D</code> | 1157 |
| <code>\tex_lastskip:D</code> | 805, 817 |
| <code>\tex_let:D</code> | 1187 |
| <code>\tex_lowercase:D</code> | 1186 |
| <code>\tex_newlinechar:D</code> | 919, 1015 |
| <code>\tex_par:D</code> | 760, 768, 774 |
| <code>\tex_scantokens:D</code> | 22, 202 |
| <code>\tex_spacefactor:D</code> | 806, 814 |
| <code>\tex_the:D</code> | 783 |
| <code>\tex_unpenalty:D</code> | 783 |
| tl commands: | |
| <code>\c_space_tl</code> | 204 |
| <code>\tl_clear:N</code> | 540, 1055 |
| <code>\tl_const:Nn</code> | 52 |
| <code>\tl_gclear:N</code> | 413, 954 |
| <code>\tl_gset:Nn</code> | 25, 410, 948 |
| <code>\tl_gset_rescan:Nnn</code> | 40 |
| <code>\tl_head:n</code> | 503, 632 |
| <code>\tl_if_blank:nTF</code> 306, 314, 320, 330, 370, 513, 565, 578, 1010 | |
| <code>\tl_if_blank_p:n</code> | 584 |
| <code>\tl_if_empty:n</code> | 209 |
| <code>\tl_if_empty:NTF</code> | 992 |
| <code>\tl_if_empty:nTF</code> | 206, 351, 1092, 1364 |
| <code>\tl_if_empty_p:N</code> | 526 |
| <code>\tl_if_empty_p:n</code> | 357, 358 |
| <code>\tl_if_head_is_N_type:nTF</code> | 501, 628, 696 |
| <code>\tl_if_novalue:nTF</code> . | 485, 840, 935, 984, 1056, 1222 |
| <code>\tl_log:N</code> | 419, 991, 1236 |
| <code>\tl_new:N</code> 39, 51, 110, 111, 112, 115, 166, 167, 169, 170, 171 | |
| <code>\tl_put_right:Nn</code> | 648, 739 |
| <code>\tl_remove_once:Nn</code> | 206, 206 |
| <code>\tl_replace_all:Nnn</code> .. | 206, 207, 498, 861, 990, 1234, 1235, 1238 |
| <code>\tl_rescan:nn</code> | 22 |
| <code>\tl_set:Nn</code> 214, 219, 236, 241, 258, 263, 296, 426, 487, 621, 849, 860, 936, 987, 1090, 1232 | |
| <code>\tl_to_str:n</code> | 434 |
| <code>\tl_use:N</code> | 590, 606, 712, 721, 779, 967, 1237 |
| token commands: | |
| <code>\token_if_eq_meaning:NNTF</code> | 638 |
| <code>\token_to_str:N</code> | 1154, 1160 |
| <code>\tt</code> | 157 |
| <code>\ttfamily</code> | 156 |
| <code>\typestored</code> | 6, 23, 24, 36, 972 |
| U | |
| <code>\unprotect</code> | 22 |
| use commands: | |
| <code>\use:N</code> | 27 |
| <code>\use:n</code> 389, 555, 616, 674, 709, 718, 871, 966, 1000, 1085, 1129 | |
| V | |
| <code>\verbatim</code> | 1142 |
| <code>\verbatimsc</code> | 745, 1020 |
| <code>verbatimsc</code> | 7, 1127 |
| W | |
| <code>\writestatus</code> | 21 |