SCONTENTS

STORES LaTeX CONTENTS

v2.4 2025-05-15[*]

©2019–2025 by Pablo González L[†]

CTAN: https://www.ctan.org/pkg/scontents
https://github.com/pablgonz/scontents

Abstract

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

Contents

1 Description of the package

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

2 Motivation and Acknowledgments

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

The general idea of this package is to try to imitate this implementation "*buffers*" that has ConTeXt which allows you to save content in memory, including *verbatim*, to be used later. The filecontentsdef[2] package solves this problem and since expl3[1] 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 to filecontentsdef package. 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)

Starting with version 2.3 the SCONTENTS package is fully compatible with *tagged* PDF and will have LaTeX release 2024-11-01 (TeX Live 2024) as a minimum requirement.

[*]This file describes a documentation for v2.4, last revised 2025-05-15.
[†]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 (lppl), version 1.3 or later (`https://www.latex-project.org/lppl.txt`). The software has the status "maintained".

The package SCONTENTS requires an updated version of LaTeX to work (minimum required to LaTeX release 2024-11-01). 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.

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

4.2 Loading and options

The package is loaded in the usual way:

For LaTeX users

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

For plain TeX users

```
\input scontents.tex
```

For ConTeXt users

```
\usemodule{scontents}
```

The package options are not available for plain TeX and ConTeXt and must be passed using `\setupsc` (see §4.4). ConTeXt users should use `-luatex`, the implementation does not support LuaMetaTeX.

4.3 The TAB character

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

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

In a standard LaTeX document, the character TAB '⇥' are treated as explicit spaces (in most contexts) and is the behavior when ⟨*stored content*⟩, but when ⟨*writing files*⟩ these are preserved.

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

As a general recommendation *"Do not use TAB character unless strictly necessary"*, for example within a *"verbatim environment"* that supports this character such as `Verbatim` of the packages fancyvrb[5], fvextra[7] or `lstlisting` of the package listings[6] or when you want to generate a `MakeFile` file.

[1]Check the answer given by Ulrich Diez in Keyboard TAB character in argument v (xparse).
[2]Check the answer given by Enrico Gregorio in How to output a tabulation into a file.

4.4 Configuration of the options

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

`\setupsc` `\setupsc{`⟨*key* = *val*⟩`}`

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 `\typestored`, `\mergesc` 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 *last 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 fancyvrb[5] or fvextra[7] packages 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 `\typestored`, `\mergesc` and `\meaningsc` commands.

4.5 Options Overview

Summary of available options:

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

5 User interface

The user interface consists in `scontents` environment, `\Scontents` and `\Scontents*` commands to ⟨*stored content*⟩, `\getstored` command to get the ⟨*stored content*⟩, `\typestored` and `\mergesc` commands to print *verbatim style* the ⟨*stored content*⟩ along with other utilities described in this documentation.

5.1 The environment scontents

scontents

```
\begin{scontents}[⟨key=val⟩]
    ⟨body env⟩
\end{scontents}
```

The `scontents` environment processes {⟨*body env*⟩} and *"stores"* it in a ⟨*sequence*⟩ or *"writes"* it to an ⟨*external file*⟩ if desired, 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 [⟨*key* = *val*⟩] 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, the ⟨*stored content*⟩ in the *sequence* {⟨*inner*⟩} is *only available* after ⟨*stored content*⟩ in the *sequence* {⟨*outer*⟩} one has been retrieved, either by using the key `print-env` or `\getstored` command.

It is advisable to ⟨*stored content*⟩ within *sequences* with different names, so as not to get lost in the order (position) in which they are stored.

Notes for plain TEX and ConTEXt users

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

\scontents
\endscontents

```
\scontents[⟨key=val⟩]
    ⟨body env⟩
\endscontents
```

ConTEXt users should use `\startscontents` and `\stopscontents`.

\startscontents
\stopscontents

```
\startscontents[⟨key=val⟩]
    ⟨body env⟩
\stopscontents
```

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 {⟨*body env*⟩} 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 ⟨*stored content*⟩ is extracted from the *sequence* {⟨*seq name*⟩} set by the key `store-env` at the time it is executed.

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

Sets the *name* of the ⟨*external file*⟩ in which the {⟨*body env*⟩} of the environment will be written. The ⟨*file.ext*⟩ will be created in the working directory and the {⟨*body env*⟩} will be *stored* in the *sequence* set by the key `store-env` at the time it is executed. If ⟨*file.ext*⟩ does not exist, it will be created or overwritten if the `overwrite` key is used.

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

Sets the *name* of the ⟨*external file*⟩ in which the {⟨*body env*⟩} of the environment will be written. The ⟨*file.ext*⟩ will be created in the working directory and the {⟨*body env*⟩} will NOT be *stored* in the *sequence* set by the key `store-env` at the time it is executed. If ⟨*file.ext*⟩ does not exist, it will be created or overwritten if the `overwrite` key is used.

In the keys `write-env` and `write-out` the character TAB will be written in ⟨*file.ext*⟩, relative or absolute paths are not supported. XƎLaTeX users using character TAB must add `-8bit` at the command line, otherwise you will get TeX character TAB '`^^I`' in ⟨*file.ext*⟩.

5.2 The command `\newenvsc`

`\newenvsc` `\newenvsc{`⟨*env name*⟩`}[`⟨*initial keys*⟩`]`

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

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

created the environment `myenvstore` that ⟨*stored content*⟩ in the *sequence* `{`⟨*myseq*⟩`}` and will NOT display the ⟨*stored content*⟩ when the environment it is executed.

5.3 The command `\Scontents`

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

The `\Scontents` command reads the `{`⟨*argument*⟩`}` in standard mode and *"stores"* it in the *sequence* set by the key `store-cmd` at the time it is executed. It is not possible to pass *verbatim things*, but it is possible to use the implementation of `\Verb` delimited by *braces* '`{ }`' provided by the fvextra[7] package for *verbatim one line* and `\lstinline` from listings[6] package, but it is preferable to use the *starred argument* '`*`'.

The `\Scontents*` command reads the `{`⟨*argument*⟩`}` under *verbatim catcode* regimen and *"stores"* it in the *sequence* set by the key `store-cmd` at the time it is executed. If its *"first"* delimiter is a *brace* '`{`', it will be assumed that the `{`⟨*argument*⟩`}` is nested into *braces*. Otherwise it will be assumed that `{`⟨*argument*⟩`}` is delimited by that first delimiter ⟨*del*⟩ like command `\verb`. Blank lines are preserved, escaped braces '`\{`' and '`\}`' must also be balanced if the `{`⟨*argument*⟩`}` is used with *braces* and character TAB typed from the keyboard are converted into spaces.

The *starred argument* '`*`' and `[`⟨*key = val*⟩`]` must NOT be separated by *spaces* '␣' between them and the command. Both versions can be used anywhere in the document and cannot be used as an `{`⟨*argument*⟩`}` for other command.

Options for command

The command options can be configured globally using option in package or the `\setupsc` command and locally using `[`⟨*key = val*⟩`]`.

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

Sets the *name* of the *sequence* in which the `{`⟨*argument*⟩`}` is stored. If the *sequence* does not exist, it will be created globally.

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

Sets if the ⟨*stored content*⟩ is displayed or not at the time of running the command. The ⟨*stored content*⟩ is extracted from the *sequence* `{`⟨*seq name*⟩`}` set by the key `store-cmd` at the time it is executed.

Options only for starred version

The `force-eol` and `overwrite` keys is available for this *starred version*.

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

Sets the *name* of the ⟨*external file*⟩ in which the `{`⟨*argument*⟩`}` will be written. The ⟨*file.ext*⟩ will be created in the working directory and the `{`⟨*argument*⟩`}` will be *stored* in the *sequence* set by the key `store-cmd` at the time it is executed. If ⟨*file.ext*⟩ does not exist, it will be created or overwritten if the `overwrite` key is used.

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

Sets the *name* of the ⟨*external file*⟩ in which the `{`⟨*argument*⟩`}` will be written. The ⟨*file.ext*⟩ will be created in the working directory and the `{`⟨*argument*⟩`}` will NOT be *stored* in any *sequence* set by the key `store-cmd` at the time it is executed. If ⟨*file.ext*⟩ does not exist, it will be created or overwritten if the `overwrite` key is used.

In the keys `write-cmd` and `write-out` the character TAB will be written in ⟨*file.ext*⟩, relative or absolute paths are not supported. XƎLaTeX users using character TAB must add `-8bit` at the command line, otherwise you will get TeX character TAB '`^^I`' in ⟨*file.ext*⟩.

5.4 The command `\getstored`

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

The `\getstored` command retrieves the ⟨*stored content*⟩ according to the *integer value* set in ⟨*index*⟩ which corresponds to the *position* of the ⟨*stored content*⟩ in the *sequence* {⟨*seq name*⟩}.

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"* ⟨*stored content*⟩ in the *sequence* {⟨*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 ⟨*stored content*⟩ in the *sequence* {⟨*seq name*⟩}. If the *optional argument* is not passed run `\getstored` on all ⟨*stored content*⟩ in the *sequence* {⟨*seq name*⟩}.

Options for command

`sep` = {⟨*code*⟩} default: *empty*

Establishes the *separation* between each ⟨*stored content*⟩ in the *sequence* {⟨*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 ⟨*stored content*⟩ in the *sequence* {⟨*seq name*⟩}. The value must be a ⟨*positive integer*⟩.

`start` = {⟨*integer*⟩} default: *1*

Set the *position* of the ⟨*stored content*⟩ within the *sequence* {⟨*seq name*⟩} from which to *start* executing. The value must be a ⟨*positive integer*⟩.

`stop` = {⟨*integer*⟩} default: *total*

Set the *position* of the ⟨*stored content*⟩ within the *sequence* {⟨*seq name*⟩} at which execution ends. The value must be a ⟨*positive integer*⟩.

`before` = {⟨*code*⟩} default: *empty*

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

`after` = {⟨*code*⟩} default: *empty*

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

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

Wraps the ⟨*stored content*⟩ within the *sequence* {⟨*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 `\typestored`

`\typestored` `\typestored[⟨index, start-stop, 1-end, keys⟩]{⟨seq name⟩}`

The command `\typestored` places the ⟨*stored content*⟩ in the *sequence* {⟨*seq name*⟩} into the internally `verbatimsc` environment (§5.9). The *integer value* set at ⟨*index*⟩ corresponds to the *position* of the ⟨*stored content*⟩ in the *sequence* {⟨*seq name*⟩} will be printed, if ⟨*1-end*⟩ is used *"all"* ⟨*stored content*⟩ in the *sequence* {⟨*seq name*⟩} will be printed.

The *integer values* set by ⟨*start-stop*⟩ define the *range* of the ⟨*stored content*⟩ in the *sequence* {⟨*seq name*⟩} that will be printed, the rest of the accepted ⟨*keys*⟩ are `print-cmd` with default value `true`, `write-out`, `width-tab` and `overwrite`.

If the *optional argument* is not passed, the *first* ⟨*stored content*⟩ in the *sequence* {⟨*seq name*⟩} will be printed.

In the key `write-out` the character TAB will be written in ⟨*file.ext*⟩, relative or absolute paths are not supported. XƎLaTeX users using character TAB must add `-8bit` at the command line, otherwise you will get TeX character TAB '`^^I`' in ⟨*file.ext*⟩.

5.7 The command `\meaningsc`

`\meaningsc` `\meaningsc[⟨index, start-stop, 1-end, keys⟩]{⟨seq name⟩}`

The command `\meaningsc` executes `\meaning` on the ⟨*stored content*⟩ in the *sequence* {⟨*seq name*⟩}. The ⟨*index*⟩, ⟨*start-stop*⟩, ⟨*1-end*⟩ and ⟨*keys*⟩ they have the same behavior as in the command `\typestored`. If the *optional argument* is not passed it defaults to the first ⟨*stored content*⟩ in the *sequence* {⟨*seq name*⟩}.

5.8 The command \mergesc

\mergesc

`\mergesc[⟨typestored | meaningsc, keys⟩]{⟨{⟨seq A⟩}[⟨index⟩], {⟨seq B⟩}[⟨start - stop⟩], {⟨seq C⟩}[⟨1-end⟩]⟩}`

The command `\mergesc` assembles the ⟨*stored content*⟩ in the *sequences* `{⟨seq A⟩}[1]`, `{⟨seq B⟩}[2-5]` and `{⟨seq C⟩}[1-end]` and then executes `\typestored` (§5.6) if the `typestored` key is active or `\meaningsc` (§5.7) if the `meaningsc` key is active.

The `{⟨argument⟩}` taken by this command is a *comma separated list* of the form `{⟨seq name⟩}` followed by either `[⟨index⟩]`, `[⟨start-stop⟩]` or `[⟨1-end⟩]`. The use of the keys `typestored` or `meaningsc` 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.

5.9 The environment verbatimsc

verbatimsc

The environment used by `\typestored` and `\mergesc` to display the ⟨*stored content*⟩ in *verbatim style*. The environment is compatible with *tagged* PDF and can be customized in the following ways after loading the SCONTENTS package:

Using the packages fvextra[7] or fancyvrb[5]:

```
\ExplSyntaxOn
\cs_undefine:N \verbatimsc
\cs_undefine:N \endverbatimsc
\ExplSyntaxOff
\usepackage{fancyvrb}
\DefineVerbatimEnvironment{verbatimsc}{Verbatim}{numbers=left}
```

Using the package minted[8]:

```
\ExplSyntaxOn
\cs_undefine:N \verbatimsc
\cs_undefine:N \endverbatimsc
\ExplSyntaxOff
\usepackage{minted}
\newminted{tex}{linenos}
\newenvironment{verbatimsc}{\VerbatimEnvironment\begin{texcode}}{\end{texcode}}
```

Using the package listings[6]:

```
\ExplSyntaxOn
\cs_undefine:N \verbatimsc
\cs_undefine:N \endverbatimsc
\ExplSyntaxOff
\usepackage{listings}
\lstnewenvironment{verbatimsc}
  {
   \lstset{
           basicstyle=\small\ttfamily,
           columns=fullflexible,
           language=[LaTeX]TeX,
           numbers=left,
           numberstyle=\tiny\color{gray},
           keywordstyle=\color{red}
          }
  }{}
```

At the moment, the fvextra[7] and fancyvrb[5] packages partially support *tagged* PDF.

6 Other commands provided

6.1 The command \countsc

\countsc

`\countsc{⟨seq name⟩}`

The command `\countsc` count a number of ⟨*stored content*⟩ in the *sequence* `{⟨seq name⟩}`.

6.2 The command `\cleanseqsc`

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

The command `\cleanseqsc` remove all ⟨*stored content*⟩ in the *sequence* {⟨*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} \emph{\enquote{buffers}} by
making use \mymeta{sequences}.
```

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

```
\begin{scontents}[store-env=abstract,print-env=true]
\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}
\end{scontents}
```

and

```
\begin{scontents}[store-env=description, print-env=true]
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} \emph{\enquote{buffers}} by
making use \mymeta{sequences}.
\end{scontents}
```

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

8 Examples

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

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

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

[3]The cool TeX automation tool: https://www.ctan.org/pkg/arara

8.1 From answers package

Example 1

Adaptation of example 1 of the package answers[17] 🗎.

```latex
% arara: pdflatex
% arara: clean: { extensions: [ aux, log] }
\documentclass{article}
\usepackage[store-cmd=solutions]{scontents}
\newtheorem{ex}{Exercise}
\setlength{\parindent}{0pt}
\pagestyle{empty}
\begin{document}
\section{Problems}
\begin{ex}
First exercise
\Scontents{First solution.}
\end{ex}

\begin{ex}
Second exercise
\Scontents{Second solution.}
\end{ex}

\section{Solutions}
\foreachsc[sep={\\[10pt]}]{solutions}
\end{document}
```

8.2 From filecontentsdef package

Example 2

Adaptation of example from package filecontentsdef[2] 🗎.

```latex
% arara: pdflatex
% arara: clean: { extensions: [ aux, log] }
\documentclass{article}
\usepackage[store-env=defexercise,store-cmd=defexercise]{scontents}
\setlength{\parindent}{0pt}
\pagestyle{empty}
\begin{document}
% not starred
\Scontents{
Prove that \[x^n+y^n=z^n\] is not solvable in positive integers if $n$ is at
most $-3$.\par
}
% starred
\Scontents*|Refute the existence of black holes in less than $140$ characters.|
% write environment to \jobname.txt
\begin{scontents}[write-env=\jobname.txt]
\def\NSA{NSA}%
Prove that factorization is easily done via probabilistic algorithms and
advance evidence from knowledge of the names of its employees in the
seventies that the \NSA\ has known that for 40 years.\par
\end{scontents}
% see all stored
\begin{itemize}
\foreachsc[before={\item }]{defexercise}
\end{itemize}
% \getstored are robust :)
\section{\getstored[2]{defexercise}}
\end{document}
```

8.3 From TeX-SX

Example 3

Adapted from LaTeX equivalent of ConTeXt buffers 🗎.

```latex
% arara: pdflatex
% arara: clean: { extensions: [ aux, log] }
```

```latex
\documentclass{article}
\usepackage[store-cmd=tikz]{scontents}
\usepackage{tikz}
\setlength{\parindent}{0pt}
\pagestyle{empty}
\Scontents{\matrix{ \node (a) {$a$} ; & \node (b) {$b$} ; \\ } ;}
\Scontents{\matrix[ampersand replacement=\&]
{ \node (a) {$a$} ; \& \node (b) {$b$} ; \\ } ;}
\Scontents{\matrix{\node (a) {$a$} ; & \node (b) {$b$} ; \\ } ; }
\begin{document}
\section{tikzpicture}
\begin{tikzpicture}
\getstored[1]{tikz}
\end{tikzpicture}

\begin{tikzpicture}
\getstored[2]{tikz}
\end{tikzpicture}

\begin{tikzpicture}
\getstored{tikz}
\end{tikzpicture}

\begin{scontents}[store-env=buffer]
Hello World!

This is a \verb*|fake poor man's buffer :)|.
\end{scontents}

\section{source tikz}
\typestored[1]{tikz}
\typestored[2]{tikz}
\typestored[3]{tikz}

\section{fake buffer}
\subsection{real content}
\getstored[1]{buffer}
\subsection{verbatim style}
\typestored[1]{buffer}
\subsection{meaning}
\meaningsc[1]{buffer}

\section{tikz again}
\foreachsc[before={\begin{tikzpicture}},after={\end{tikzpicture}},sep={\\[10pt]}]{tikz}
\end{document}
```

Example 4

Adapted from Collecting contents of environment and store them for later retrieval.

```latex
% arara: pdflatex
% arara: clean: { extensions: [ aux, log] }
\documentclass{article}
\usepackage{scontents}
\setlength{\parindent}{0pt}
\pagestyle{empty}
\begin{document}
\begin{scontents}[store-env=main]
Something for main A.
\end{scontents}

\begin{scontents}[store-env=main]
Something for \verb|main B|.
\end{scontents}

\begin{scontents}[store-env=other]
Something for \verb|other|.
\end{scontents}

\textbf{Let's print them}

This is first stored in main: \getstored[1]{main}\par
This is second stored in main: \getstored{main}\par
This is stored in other: \getstored{other}

\textbf{Print all of stored in main}\par
\foreachsc[sep={\\[10pt]}]{main}
\end{document}
```

Example 5

Adapted from Collect contents of an environment (that contains verbatim content) 🗎.

```
% arara: pdflatex
% arara: clean: { extensions: [ aux, log] }
\documentclass{article}
\usepackage{scontents}
\setlength{\parindent}{0pt}
\pagestyle{empty}
\begin{document}
\section{Problem stated the first time}
\begin{scontents}[print-env=true,store-env=problem]
This is normal text.
\verb|This is from the verb command.|
\verb*|This is from the verb* command.|
This is normal text.
\begin{verbatim}
This is from the verbatim environment:
&%{}~
\end{verbatim}
\end{scontents}
\section{Problem restated}
\getstored[1]{problem}
\section{Problem restated once more}
\getstored[1]{problem}
\end{document}
```

Example 6

Adapted from Environment hiding its content 🗎.

```
% arara: pdflatex
% arara: clean: { extensions: [ aux, log] }
\documentclass[10pt]{article}
\usepackage{scontents}
\newenvsc{forshort}[store-env=forshort,print-env=false]
\setlength{\parindent}{0pt}
\pagestyle{empty}
\begin{document}

Something in the whole course.

\begin{forshort}
    Just a summary...
\end{forshort}

\end{document}
```

8.4 Customization of verbatimsc

Example 7

Customization of verbatimsc using the fvextra[7] and tcolorbox[14] package 🗎.

```
% arara: pdflatex
% arara: clean: { extensions: [ aux, log] }
\documentclass{article}
\usepackage{scontents}
\ExplSyntaxOn
\cs_undefine:N \verbatimsc
\cs_undefine:N \endverbatimsc
\ExplSyntaxOff
\usepackage{fvextra}
\usepackage{xcolor}
```

```
\definecolor{mygray}{gray}{0.9}
\usepackage{tcolorbox}
\newenvironment{verbatimsc}%
{\VerbatimEnvironment
\begin{tcolorbox}[colback=mygray, boxsep=0pt, arc=0pt, boxrule=0pt]
\begin{Verbatim}[fontsize=\scriptsize, breaklines, breakafter=*, breaksymbolsep=0.5em,
breakaftersymbolpre={\,\tiny\ensuremath{\rfloor}}]}%
{\end{Verbatim}%
\end{tcolorbox}}
\setlength{\parindent}{0pt}
\pagestyle{empty}
\begin{document}
\section{Test \texttt{\textbackslash begin\{scontents\}} with \texttt{fancyvrb}}
Test \verb+{scontents}+ \par

\begin{scontents}
Using \verb+scontents+ env no \verb+[key=val]+, save in seq \verb+contents+
with index 1.

Prove new \Verb*{ fancyvrb with braces } and environment \verb+Verbatim*+
\begin{verbatim}
 verbatim  environment
\end{verbatim}
\end{scontents}

\section{Test \texttt{\textbackslash Scontents} with \texttt{fancyvrb}}
\Scontents{ We have coded this in \LaTeX: $E=mc^2$.}

\section{Test \texttt{\textbackslash getstored}}
\getstored[1]{contents}\par
\getstored{contents}

\section{Test \texttt{\textbackslash meaningsc}}
\meaningsc[1]{contents}\par
\meaningsc[2]{contents}

\section{Test \texttt{\textbackslash typestored}}
\typestored[1]{contents}
\typestored[2]{contents}
\end{document}
```

Example 8

Customization of verbatimsc using the listings[6] package.

```
% arara: pdflatex
% arara: clean: { extensions: [ aux, log] }
\documentclass{article}
\usepackage{scontents}
\ExplSyntaxOn
\cs_undefine:N \verbatimsc
\cs_undefine:N \endverbatimsc
\ExplSyntaxOff
\usepackage{xcolor}
\usepackage{listings}
\lstnewenvironment{verbatimsc}
  {
   \lstset{
           basicstyle=\small\ttfamily,
           breaklines=true,
           columns=fullflexible,
           language=[LaTeX]TeX,
           numbers=left,
           numbersep=1em,
           numberstyle=\tiny\color{gray},
           keywordstyle=\color{red}
          }
  }{}
\setlength{\parindent}{0pt}
\pagestyle{empty}
\begin{document}
\section{Test \texttt{\textbackslash begin\{scontents\}} with \texttt{listings}}
```

```latex
Test \verb+{scontents}+ \par

\begin{scontents}
Using \verb+scontents+ env no \verb+[key=val]+, save in seq \verb+contents+ with index 1.\par

Prove \lstinline[basicstyle=\ttfamily]| lstinline | and environment \verb+Verbatim*+
\begin{verbatim}
  verbatim  environment
\end{verbatim}
\end{scontents}

\section{Test \texttt{\textbackslash Scontents*} with \texttt{listings}}

\Scontents*+We have coded this in \lstinline[basicstyle=\ttfamily]|\LaTeX: $E=mc^2$|
and more.+

\section{Test \texttt{\textbackslash getstored}}
\getstored{contents}\par
\getstored[1]{contents}

\section{Test \texttt{\textbackslash typestored}}
\typestored[1]{contents}
\typestored[2]{contents}
\end{document}
```

Example 9

Customization of verbatimsc using the minted[8] package.

```latex
% arara: xelatex: {shell: true, options: [-8bit]}
% arara: xelatex: {shell: true, options: [-8bit]}
% arara: clean: { extensions: [ aux, log] }
\documentclass{article}
\usepackage{scontents}
\ExplSyntaxOn
\cs_undefine:N \verbatimsc
\cs_undefine:N \endverbatimsc
\ExplSyntaxOff
\usepackage{minted}
\newminted{tex}{linenos}
\newenvironment{verbatimsc}{\VerbatimEnvironment\begin{texcode}}{\end{texcode}}
\pagestyle{empty}
\setlength{\parindent}{0pt}
\begin{document}
\section{Test \texttt{\textbackslash begin\{scontents\}} with \texttt{minted}}
Test \verb+{scontents}+ \par

\begin{scontents}[overwrite,write-env=\jobname.tsc,force-eol=true]
Using \verb+scontents+ env no \verb+[key=val]+, save in seq \verb+contents+
with index 1.\par

Prove new \Verb*{ new fvextra with braces } and environment \verb+Verbatim*+
\begin{Verbatim}[obeytabs, showtabs, tab=\rightarrowfill, tabcolor=red]
No tab
        One real tab
                Two real Tab plus       one tab
\end{Verbatim}
\end{scontents}

\section{See \Verb{\jobname.tsc}}
Read \Verb{\jobname.tsc} (shows TABs as red arrows):
\VerbatimInput[obeytabs, showtabs, tab=\rightarrowfill, tabcolor=red]{\jobname.tsc}

\section{Test \texttt{\textbackslash Scontents} with \texttt{minted}}

\Scontents{ We have coded \par this in \LaTeX: $E=mc^2$.}

\section{Test \texttt{\textbackslash getstored}}
\getstored[1]{contents}\par
\getstored{contents}

\section{Test \texttt{\textbackslash typestored}}
```

```
\typestored[1]{contents}
\typestored[2]{contents}
\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.

```
% arara: pdflatex
% arara: clean: { extensions: [ aux, log] }
\documentclass{article}
\usepackage{scontents}
% Fix part of a MCE that should go before babel's loading
\begin{scontents}[store-env=mce]
\documentclass[french]{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage{lmodern}
\usepackage[a4paper]{geometry}
\end{scontents}
% Fix part of a MCE that should go after (>=) babel's loading
\begin{scontents}[store-env=mce]
\usepackage{babel}
\begin{document}
\end{scontents}
% Fix part of a MCE that should go after its body
\begin{scontents}[store-env=mce]
\end{document}
\end{scontents}
\begin{document}
\section{First annswer}
% Variable part of a MCE that should added to the fixed preamble, before babel's loading
\begin{scontents}[store-env=mce-1]
\usepackage{amsmath}
\end{scontents}
% Variable part of a MCE being the code snippet
\begin{scontents}[store-env=mce-1]
\begin{align}
  0 & \neq 1 \\
  1 & \neq 0
\end{align}
\end{scontents}
\begin{description}
\item[Preamble's addition]\leavevmode
  \typestored[1]{mce-1}
\item[Code snippet]\leavevmode
  \typestored[2]{mce-1}
\item[MCE]\leavevmode
  \mergesc[typestored, print-cmd=true]
    {
      {mce}[1], {mce-1}[1], {mce}[2], {mce-1}[2], {mce}[3]
    }
\end{description}
\section{Second annswer}
% Variable part of a MCE that should added to the fixed preamble, before babel's loading
\begin{scontents}[store-env=mce-2]
\usepackage{amsmath}
\end{scontents}
% Variable part of a MCE being the code snippet
\begin{scontents}[store-env=mce-2]
\begin{flalign}
  0 & \neq 1 \\
  1 & \neq 0
\end{flalign}
\end{scontents}

\begin{description}
\item[Preamble's addition]\leavevmode
  \typestored[1]{mce-2}
\item[Code snippet]\leavevmode
  \typestored[2]{mce-2}
```

```
\item[MCE]\leavevmode
  \mergesc[typestored, print-cmd=true, write-out=mce.txt, overwrite=true]
    {
      {mce}[1], {mce-2}[1], {mce}[2], {mce-2}[2], {mce}[3]
    }
\end{description}
\end{document}
```

8.6 The tagged PDF example

This example is just to show the compatibility of SCONTENTS with *tagged* `PDF` using `lualatex`. The attached files here are just for testing .

```
% arara: lualatex
% arara: clean: { extensions: [ aux, log] }
\DocumentMetadata{tagging=on, lang=en-US, pdfversion=2.0, pdfstandard=ua-2, testphase=latest}
\documentclass{article}
\usepackage{scontents,unicode-math,hyperref}
\hypersetup{pdftitle = {Test scontents package},}
\begin{document}
Some

\begin{scontents}[print-env=true]
    First code \verb|\foo|

    And more code \verb|\bar|
\end{scontents}

Text

\begin{scontents}[print-env=true]
    Second code \verb|\foo|

    And more code \verb|\bar|
\end{scontents}

Text

\Scontents*{code \verb|\baz|}

% \typestored
\typestored[1]{contents}

% \mergesc
\mergesc[typestored]{ {contents}[1-end] }

% \getstored
\getstored[2]{contents}
\end{document}
```

This example have been checked using `veraPDF` together with `ngpdf`.

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[2] package to the current version with only expl3[1].

v2.4 (ctan), 2025-05-15	– Optimization of expansion code from ‘x’ to ‘e’.
	– Restructuring code for documentation and implementation.
	– Add new keys for \typestored and \meaningsc.
	– Check the version of expl3 in plain TeX and ConTeXt.
v2.3 (ctan), 2025-04-23	– Adapting the verbatimsc environment for *tagged* PDF.
	– Update minimum required to LaTeX release 2024-11-01.
	– Safer code for replacement \obeyedline.
v2.2 (ctan), 2025-03-26	– Fix internal definition for some functions.
	– Replace \peek_charcode_ignore_spaces:NTF by \peek_charcode:NTF.
	– Set correct code for \obeyedline implement in LaTeX release 2024-06-01.
v2.1 (ctan), 2024-06-14	– Fix \cleanseqsc 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 scontents 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.

11 References

[1] The LaTeX Project. "The expl3 package". Available from CTAN, https://www.ctan.org/pkg/expl3, 2025.

[2] Burnol, Jean François. "The filecontentsdef package". Available from CTAN, https://ctan.org/pkg/filecontentsdef, 2019.

[3] The LaTeX Project. "The xparse package". Available from CTAN, https://www.ctan.org/pkg/xparse, 2024.

[4] Niederberger, Clemens. "xsim – eXercise Sheets IMproved". Available from CTAN, https://www.ctan.org/pkg/xsim, 2023.

[5] Van Zandt, Timothy. "The fancyvrb package - Fancy Verbatims in LaTeX". Available from CTAN, https://www.ctan.org/pkg/fancyvrb, 2024.

[6] Hoffmann, Jobst. "The listings package". Available from CTAN, https://www.ctan.org/pkg/listings, 2024.

[7] Poore, Geoffrey M. "The fvextra package - Highlighted source code in LaTeX". Available from CTAN, https://www.ctan.org/pkg/minted, 2025.

[8] Poore, Geoffrey M. "The minted package - Highlighted source code in LaTeX". Available from CTAN, https://www.ctan.org/pkg/minted, 2025.

[9] The LaTeX Project. "The LaTeX3 Interfaces". Available from CTAN, https://www.ctan.org/pkg/l3kernel, 2025.

[10] The LaTeX Project. “The LaTeX 2_ε sources”. Available from CTAN, https://ctan.org/tex-archive/macros/latex/base, 2025.

[11] The LaTeX Project. “LaTeX for authors current version”. Available from CTAN, https://ctan.org/pkg/latex-base, 2025.

[12] Fischer, Ulrike. “tagpdf – LaTeX kernel code for PDF tagging”. Available from CTAN, https://www.ctan.org/pkg/tagpdf, 2025.

[13] The LaTeX Project. “latex-lab – LaTeX laboratory”. Available from CTAN, https://www.ctan.org/pkg/latex-lab, 2025.

[14] F. Sturm, Thomas. “tcolorbox – Coloured boxes, for LaTeX examples and theorems, etc”. Available from CTAN, https://ctan.org/pkg/tcolorbox, 2025.

[15] The LaTeX Project. “verbatim – Reimplementation of and extensions to LaTeX verbatim”. Available from CTAN, https://www.ctan.org/pkg/verbatim, 2023.

[16] Wright, Joseph. “Programming key–value in expl3”. Available from TUGboat, https://www.tug.org/TUGboat/tb31-1/tb97wright-l3keys.pdf, 2010.

[17] Wright, Joseph. “answers – Setting questions (or exercises) and answers”. Available from CTAN, https://ctan.org/pkg/answers, 2014.

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.

All variables and functions defined in this package are private and are NOT intended to work or be used by another package or module.

12.1 Declaration of the package

First we set up the module name for DocStrip l3doc class:

```
⟨@@=scontents⟩
```

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

```
⟨loader⟩\def\ScontentsFileDate{2025-05-15}%
⟨core⟩\def\ScontentsCoreFileDate{2025-05-15}%
⟨*loader⟩
\def\ScontentsFileVersion{2.4}%
\def\ScontentsFileDescription{Stores LaTeX contents in memory or files}%
```

The LaTeX loader is quite simple, we just need to make sure of the minimum version for correct operation and then set interfaces up. The choice of LaTeX release 2024-11-01 is the latest available in TeX Live 2024 (frozen) and is necessary to be able to implement the package's full compatibility with *tagged* PDF.

```
⟨*latex⟩
\NeedsTeXFormat{LaTeX2e}[2024-11-01]
\ProvidesExplPackage
  {scontents} {\ScontentsFileDate} {\ScontentsFileVersion} {\ScontentsFileDescription}
⟨/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 the frozen xparse[3] later (§12.1.1).

```
⟨*!latex⟩
⟨context⟩\writestatus{loading}{User Module scontents v\ScontentsFileVersion}
⟨context⟩\unprotect
\input expl3-generic.tex
\ExplSyntaxOn
\tl_gset:ce { ver @ scontents . sty } { \ScontentsFileDate\space
  v\ScontentsFileVersion\space \ScontentsFileDescription }
\iow_log:e { Package: ~ scontents ~ \use:c { ver @ scontents . sty } }
```

For plain TeX and ConTeXt we must check the minimum requirement, in this case \int_step_tokens:nn which was added in release 2025-01-14 of expl3 included in TeX Live 2024 (frozen).

```
\cs_if_exist:NF \int_step_tokens:nn
  {
    \msg_new:nnn { scontents } { expl-too-old }
      {
        Please~install~an~up~to~date~TeX~distribution~or~update~using~
        your~TeX~package~manager~or~from~CTAN. \\
        See~documentation.~Loading~scontents~will~abort!
      }
    \msg_fatal:nn { scontents } { expl-too-old }
    \ExplSyntaxOff
    \file_input_stop:
  }
⟨/!latex⟩
```

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

```
⟨*plain⟩
\cs_if_exist:NT \__scontents_rescan_tokens:n
  {
    \msg_new:nnn { scontents } { already-loaded }
      {
        The~'scontents'~package~is~already~loaded.~Aborting~input~\msg_line_context:.
      }
    \msg_warning:nn { scontents } { already-loaded }
    \ExplSyntaxOff
    \file_input_stop:
  }
⟨/plain⟩
```

In ConTEXt we must take a precaution when running under LMTX since \tex_scantokens:D is a copy of the primitive ε-TEX \scantokens and \tl_gset_rescan:Nnn is a wrapper around it and are not available under LMTX.

This is an adaptation of the file t-lua-widow-control.mkxl part of Max Chernoff's lua-widow-control package, \contextlmtxmode is described at https://source.contextgarden.net/tex/context/base/mkxl/context.mkxl.

```
⟨*context⟩
\bool_if:NT \contextlmtxmode
  {
    \msg_new:nnn { scontents } { luametatex }
      {
        The~'scontents'~package~doesn't~work~under~LMTX.
      }
    \msg_error:nn { scontents } { luametatex }
  }
⟨/context⟩
⟨/loader⟩
```

12.1.1 Load xparse-generic in plain TEX and ConTEXt

\l__scontents_char_value_int

When loading the package outside of LATEX we can't usually use xparse[3] now ltcmd[10] part of the LATEX kernel. However since the old xparse[3] provide xparse-generic.tex is loadable in any format.

```
⟨*loader&!latex⟩
\int_new:N  \l__scontents_char_value_int
\int_set:Nn \l__scontents_char_value_int { \char_value_catcode:n { `\@ } }
\char_set_catcode_letter:N \@
\file_input:n { xparse-generic.tex }
\char_set_catcode:nn { `\@ } { \l__scontents_char_value_int }
⟨/loader&!latex⟩
```

The file TDS:/tex/latex/l3packages/xparse/xparse-generic.tex is always available and frozen since 2021.

(*End of definition for* \l__scontents_char_value_int.)

12.1.2 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
\c__scontents_end_env_tl
\l__scontents_env_name_tl

The global token list \g__scontents_end_verbatimsc_tl match when ending verbatimsc (§12.11).

```
⟨*loader⟩
\tl_new:N \g__scontents_end_verbatimsc_tl
\tl_gset_rescan:Nnn \g__scontents_end_verbatimsc_tl
  {
    \char_set_catcode_other:N \\
⟨*latex⟩
    \char_set_catcode_other:N \{
    \char_set_catcode_other:N \}
⟨/latex⟩
  }
⟨latex⟩  { \end{verbatimsc} }
⟨plain⟩  { \endverbatimsc }
⟨context⟩  { \stopverbatimsc }
```

The constant token list \c__scontents_end_env_tl match when ending environments defined by \newenvsc, the token list\l__scontents_env_name_tl storing the *name* of environments defined by \newenvsc (§12.9).

```
\tl_new:N    \l__scontents_env_name_tl
\tl_const:Ne \c__scontents_end_env_tl
  {
    \c_backslash_str
⟨latex|plain⟩     end
⟨context⟩     stop
⟨latex⟩     \c_left_brace_str
      \exp_not:N \l__scontents_env_name_tl
⟨latex⟩     \c_right_brace_str
```

```
  }
⟨/loader⟩
```

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

12.1.3 Loading the package core

Now we load the core SCONTENTS code:

```
⟨*loader⟩
\file_input:n { scontents-code.tex }
```

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

```
\cs_new:Npn \__scontents_format_case:nnn #1 #2 #3
⟨latex⟩   {#1} % LaTeX
⟨plain⟩   {#2} % Plain/Generic
⟨context⟩   {#3} % ConTeXt
⟨/loader⟩
```

(*End of definition for* `\__scontents_format_case:nnn`.)

12.1.4 Checking proper loader

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

```
⟨*core⟩
\begingroup
  \catcode32=10
  \endlinechar=32
  \def\next{\endgroup}%
  \expandafter\ifx\csname PackageError\endcsname\relax
    \begingroup
      \def\next{\endgroup\endgroup}%
      \def\PackageError#1#2#3%
        {%
          \endgroup
          \errhelp{#3}%
          \errmessage{#1 Error: #2!}%
        }%
  \fi
  \expandafter\ifx\csname ScontentsFileDate\endcsname\relax
    \def\next
      {%
        \PackageError{scontents}{No scontents loader detected}
          {%
            You have attempted to use the scontents code directly rather than using
            the correct loader. Loading of scontents will abort.
          }%
        \endgroup
        \endinput
      }%
  \else
    \ifx\ScontentsFileDate\ScontentsCoreFileDate
    \else
      \def\next
        {%
          \PackageError{scontents}{Mismatched scontents files detected}
            {%
              You have attempted to load scontents with mismatched files:
              probably you have one or more files 'locally installed' which
              are in conflict. Loading of scontents will abort.
            }%
          \endgroup
          \endinput
        }%
    \fi
  \fi
\next
```

12.2 Keys for the package

store-env
store-cmd
verb-font
print-env
print-cmd
force-eol
overwrite
width-tab
print-all
store-all

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.

```
\keys_define:nn { scontents }
  {
    store-env .tl_set:N          = \l__scontents_name_seq_env_tl,
    store-env .initial:n         = contents,
    store-env .value_required:n = true,
    store-cmd .tl_set:N          = \l__scontents_name_seq_cmd_tl,
    store-cmd .initial:n         = contents,
    store-cmd .value_required:n = true,
    verb-font .tl_set:N          = \l__scontents_verb_font_tl,
    verb-font .value_required:n = true,
    print-env .bool_set:N        = \l__scontents_print_env_bool,
    print-env .initial:n         = false,
    print-env .default:n         = true,
    print-cmd .bool_set:N        = \l__scontents_print_cmd_bool,
    print-cmd .initial:n         = false,
    print-cmd .default:n         = true,
    force-eol .bool_set:N        = \l__scontents_forced_eol_bool,
    force-eol .initial:n         = false,
    force-eol .default:n         = true,
    overwrite .bool_set:N        = \l__scontents_overwrite_bool,
    overwrite .initial:n         = false,
    overwrite .default:n         = true,
    width-tab .int_set:N         = \l__scontents_tab_width_int,
    width-tab .initial:n         = 1,
    width-tab .value_required:n = true,
    print-all .meta:n            = { print-env = #1, print-cmd = #1 },
    print-all .default:n         = true,
    store-all .meta:n            = { store-env = #1, store-cmd = #1 },
    store-all .value_required:n = true
  }
⟨/core⟩
```

Set default value for `verb-font` key.

```
⟨loader⟩\keys_define:nn { scontents }
⟨latex⟩   { verb-font .initial:n = \ttfamily }
⟨plain|context⟩  { verb-font .initial:n = \tt }
```

In LaTeX mode process the ⟨*keys*⟩ as options passed on to the package and will return an error when they are.

```
⟨*latex⟩
\ProcessKeyOptions [ scontents ]
⟨/latex⟩
```

(*End of definition for* `store-env` *and others.*)

12.3 Internal variables

\l__scontents_save_every_body_lines_tl
\l__scontents_processed_body_lines_tl
\l__scontents_environment_keys_tl
\l__scontents_nesting_env_int
\l__scontents_nesting_aux_int

The token list `\l__scontents_save_every_body_lines_tl` holds the {⟨*body env*⟩} of an environment, `scontents` by default, as it's being read, the token list `\l__scontents_processed_body_lines_tl` saves all sanitized lines saved in `\l__scontents_save_every_body_lines_tl`.

The token list `\l__scontents_environment_keys_tl` saves the ⟨*keys*⟩ passed to the *optional argument* after they are sanitized and the integer variables `\l__scontents_nesting_env_int` together with `\l__scontents_nesting_aux_int` are used to analyze the nesting of the environment.

All of these variables are used in the implementation of `\newenvsc` (§12.9) and the environments base functions (§12.8).

```
⟨*core⟩
\tl_new:N  \l__scontents_save_every_body_lines_tl
\tl_new:N  \l__scontents_processed_body_lines_tl
\tl_new:N  \l__scontents_environment_keys_tl
\int_new:N \l__scontents_nesting_env_int
\int_new:N \l__scontents_nesting_aux_int
```

(*End of definition for* `\l__scontents_save_every_body_lines_tl` *and others.*)

\l__scontents_cmd_name_tl

The token list \l__scontents_cmd_name_tl saves the *name* of the commands \Scontents, \foreachsc, \typestored, \meaningsc and \mergesc.

```
\tl_new:N \l__scontents_cmd_name_tl
```

(*End of definition for* \l__scontents_cmd_name_tl*.*)

\l__scontents_Scontents_arg_tl
\l__scontents_foreachsc_arg_tl
\l__scontents_typestored_arg_tl
\l__scontents_meaningsc_arg_tl
\l__scontents_mergesc_arg_tl

The token lists \l__scontents_Scontents_arg_tl, \l__scontents_foreachsc_arg_tl, \l__scontents_typestored_arg_tl and \l__scontents_meaningsc_arg_tl save the {⟨*argument*⟩} passed to the \Scontents (§12.12), \foreachsc (§12.14), \typestored (§12.15) and \meaningsc (§12.16) commands.

```
\tl_new:N \l__scontents_Scontents_arg_tl
\tl_new:N \l__scontents_foreachsc_arg_tl
\tl_new:N \l__scontents_typestored_arg_tl
\tl_new:N \l__scontents_meaningsc_arg_tl
```

(*End of definition for* \l__scontents_Scontents_arg_tl *and others.*)

\l__scontents_mergesc_arg_tl
\l__scontents_mergesc_keys_tl
\l__scontents_current_seq_name_str

The token list \l__scontents_mergesc_arg_tl save the {⟨*argument*⟩} and the token list \l__scontents_mergesc_keys_tl save the ⟨*keys*⟩ passed to the \mergesc (§12.17) command.

The string variable \l__scontents_current_seq_name_str stores the name of the *current sequence* passed as an {⟨*argument*⟩} to the \typestored and \meaningsc commands and is used by the function __scontents_parse_type_meaning_key:n.

```
\tl_new:N  \l__scontents_mergesc_arg_tl
\tl_new:N  \l__scontents_mergesc_keys_tl
\str_new:N \l__scontents_current_seq_name_str
```

(*End of definition for* \l__scontents_mergesc_arg_tl*,* \l__scontents_mergesc_keys_tl*, and* \l__scontents_current_seq_name_str*.*)

\l__scontents_file_name_tl
\l__scontents_file_write_iow
\l__scontents_writing_bool
l__scontents_storing_bool
\l__scontents_writable_bool

The token list \l__scontents_file_name_tl is used for store the name of the ⟨*output file*⟩, when there's one. Its value is set by the keys write-env, write-out and write-cmd (§12.5).

The variable \l__scontents_file_write_iow is an *output stream* for write the {⟨*body env*⟩} of an environment or {⟨*argument*⟩} for command to a ⟨*output file*⟩ when the keys write-env, write-out or write-cmd are active.

The boolean variables \l__scontents_writing_bool and \l__scontents_storing_bool (true by default) set by the write-out, write-env and write-cmd keys determine whether the content is stored and written or just written to a ⟨*output file*⟩.

The boolean variable \l__scontents_writable_bool keeps track of whether we should write to a file, it is in write-only or in mode overwrite when the key overwrite is used.

This variable is used by the function __scontents_file_if_writable:nTF (see 12.8.2).

```
\tl_new:N   \l__scontents_file_name_tl
\iow_new:N  \l__scontents_file_write_iow
\bool_new:N \l__scontents_writing_bool
\bool_new:N \l__scontents_storing_bool
\bool_set_true:N \l__scontents_storing_bool
\bool_new:N \l__scontents_writable_bool
```

(*End of definition for* \l__scontents_file_name_tl *and others.*)

\l__scontents_foreach_print_seq
\g__scontents_foreach_exec_tl
\l__scontents_foreach_before_tl
\l__scontents_foreach_before_bool
\l__scontents_foreach_after_tl
\l__scontents_foreach_after_bool
\l__scontents_foreach_stop_int
\l__scontents_foreach_wrapper_bool

Internal variables used by ⟨*keys*⟩ (§12.6.2) and implementation of \foreachsc command (§12.14).

```
\seq_new:N  \l__scontents_foreach_print_seq
\tl_new:N   \g__scontents_foreach_exec_tl
\tl_new:N   \l__scontents_foreach_before_tl
\bool_new:N \l__scontents_foreach_before_bool
\tl_new:N   \l__scontents_foreach_after_tl
\bool_new:N \l__scontents_foreach_after_bool
\int_new:N  \l__scontents_foreach_stop_int
\bool_new:N \l__scontents_foreach_stop_bool
\bool_new:N \l__scontents_foreach_wrapper_bool
```

(*End of definition for* \l__scontents_foreach_print_seq *and others.*)

\l__scontents_seq_item_seq
\g__scontents_name_sc!internal_seq

The sequence variable \l__scontents_seq_item_seq save the ⟨*indexes*⟩ in the *sequence* of the items requested to \typestored, \mergesc or \meaningsc and the sequence \g__scontents_name_sc!internal_seq assemble this.

```
\seq_new:N \l__scontents_seq_item_seq
\seq_new:c { g__scontents_name_sc!internal_seq }
```

(*End of definition for* `\l__scontents_seq_item_seq` *and* `\g__scontents_name_sc!internal_seq`.)

`\g__scontents_last_stored_tl`

The token list `\g__scontents_last_stored_tl` used by the function `\__scontents_lastfrom_seq:n` (§12.7) to execute the last ⟨*stored content*⟩ outside the group.

```
\tl_new:N \g__scontents_last_stored_tl
```

(*End of definition for* `\g__scontents_last_stored_tl`.)

`\c__scontents_hidden_space_str`

The variable `\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 *sequence* when the key `force-eol` is false.

```
\str_const:Ne \c__scontents_hidden_space_str
  { \c_percent_str \c_circumflex_str \c_circumflex_str A scheol \c_percent_str }
```

(*End of definition for* `\c__scontents_hidden_space_str`.)

`\l__scontents_save_sf_int`
`\l__scontents_save_skip`

Internal variables used by functions `\__scontents_bsphack:` and `\__scontents_esphack:` (§12.4.2).

```
\int_new:N  \l__scontents_save_sf_int
\skip_new:N \l__scontents_save_skip
```

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

`\q__scontents_stop`
`\q__scontents_mark`
`\s__scontents_stop`
`\s__scontents_mark`

Some quarks and scan's used along the code as macro delimiters.

```
\quark_new:N \q__scontents_stop
\quark_new:N \q__scontents_mark
\scan_new:N  \s__scontents_stop
\scan_new:N  \s__scontents_mark
⟨/core⟩
```

(*End of definition for* `\q__scontents_stop` *and others.*)

`\l__scontents_plain_bool`

The boolean variable `\l__scontents_plain_bool` used in the plain TeX implementation of the `verbatimsc` environment (§12.11).

```
⟨*plain⟩
\bool_new:N \l__scontents_plain_bool
⟨/plain⟩
```

(*End of definition for* `\l__scontents_plain_bool`.)

12.4 Utility functions

`\__scontents_rescan_tokens:n`
`\__scontents_rescan_tokens:V`
`\__scontents_rescan_tokens:e`

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 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?

```
⟨*core⟩
\cs_new_protected:Npn \__scontents_rescan_tokens:n #1 { \tex_scantokens:D {#1} }
\cs_generate_variant:Nn \__scontents_rescan_tokens:n { V, e }
```

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

`\tl_if_empty:fTF`

Some nonstandard kernel variant.

```
\prg_generate_conditional_variant:Nnn \tl_if_empty:n { f } { p, TF }
```

(*End of definition for* `\tl_if_empty:fTF`.)

`\__scontents_use_delimit_by_s_stop:nw`
`\__scontents_use_i_delimit_by_s_stop:nw`
`\__scontents_use_none_delimit_by_s_stop:w`
`\__scontents_use_none_delimit_by_q_stop:w`

Some functions used in the implementation of `\mergesc` (§12.17) and `scontents` (§12.10).

```
\cs_new:Npn \__scontents_use_delimit_by_s_stop:nw #1 \s__scontents_stop {#1}
\cs_new:Npn \__scontents_use_i_delimit_by_s_stop:nw #1 #2 \s__scontents_stop {#1}
\cs_new:Npn \__scontents_use_none_delimit_by_s_stop:w #1 \s__scontents_stop { }
\cs_new:Npn \__scontents_use_none_delimit_by_q_stop:w #1 \q__scontents_stop { }
```

(*End of definition for* `\__scontents_use_delimit_by_s_stop:nw` *and others.*)

__scontents_tl_if_head_is_q_mark:nTF

The conditional function __scontents_tl_if_head_is_q_mark:n tests if the head of the token list is \q__scontents_mark.

```
\prg_new_protected_conditional:Npnn \__scontents_tl_if_head_is_q_mark:n #1
  { T, F, TF }
  {
    \exp_after:wN \if_meaning:w
      \exp_after:wN
        \q__scontents_mark \__scontents_use_i_delimit_by_s_stop:nw #1 ? \s__scontents_stop
      \prg_return_true:
    \else:
      \prg_return_false:
    \fi:
  }
```

(End of definition for __scontents_tl_if_head_is_q_mark:nTF.*)*

__scontents_file_if_writable:n
__scontents_file_if_writable:nT
__scontents_file_if_writable:nF
__scontents_file_if_writable:nTF

The conditional function __scontents_file_if_writable:n used by the write-env, write-cmd, write-out and overwrite keys.

```
\prg_new_protected_conditional:Npnn \__scontents_file_if_writable:n #1 { T, F, TF }
  {
    \bool_if:NTF \l__scontents_writing_bool
      {
        \file_if_exist:nTF {#1}
          {
            \bool_if:NTF \l__scontents_overwrite_bool
              {
                \msg_warning:nne { scontents } { overwrite-file } {#1}
                \prg_return_true:
              }
              {
                \msg_warning:nne { scontents } { not-writing } {#1}
                \prg_return_false:
              }
          }
          {
            \msg_warning:nne { scontents } { writing-file } {#1}
            \prg_return_true:
          }
      }
      { \prg_return_false: }
  }
```

(End of definition for __scontents_file_if_writable:n *and others.)*

__scontents_file_write_cmd:nn
__scontents_file_write_cmd:VV

The function __scontents_file_write_cmd:nn used by the write-env, write-cmd, write-out and overwrite keys for commands.

```
\cs_new_protected:Npn \__scontents_file_write_cmd:nn #1#2
  {
    \__scontents_file_if_writable:nT {#1}
      {
        \iow_open:Nn \l__scontents_file_write_iow {#1}
        \iow_now:Nn  \l__scontents_file_write_iow {#2}
        \iow_close:N \l__scontents_file_write_iow
      }
  }
\cs_generate_variant:Nn \__scontents_file_write_cmd:nn { VV }
```

(End of definition for __scontents_file_write_cmd:nn.*)*

12.4.1 Functions for TAB and verbatimsc

__scontents_tab:
__scontents_par:

Control sequences to replace tab '^^I' and form feed '^^L' characters.

```
\cs_new:Ne \__scontents_tab: { \c_space_tl }
\cs_new:Nn \__scontents_par: { ^^J ^^J }
```

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

__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.

```
\cs_new:Nn \__scontents_tabs_to_spaces:
  { \prg_replicate:nn { \l__scontents_tab_width_int } { ~ } }
```

(*End of definition for* `\__scontents_tabs_to_spaces:`.)

`\__scontents_do_noligs:N`

The function `\__scontents_do_noligs:N` is an alternative definition for LaTeX 2ε's `\do@noligs` which makes sure to not consume following space tokens. The LaTeX 2ε version ends with `` \char`#1 ``, which leaves TeX still looking for an ⟨*optional space*⟩.

```
\cs_new_protected:Npn \__scontents_do_noligs:N #1
  {
    \char_set_catcode_active:N #1
    \cs_set:cpe { __scontents_active_char_ \token_to_str:N #1 : }
      {
        \mode_leave_vertical:
        \tex_kern:D \c_zero_dim
        \tex_char:D `\exp_not:N #1
      }
    \char_set_active_eq:Nc #1 { __scontents_active_char_ \token_to_str:N #1 : }
  }
```

(*End of definition for* `\__scontents_do_noligs:N`.)

`\__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.

```
\cs_new_protected:Npn \__scontents_set_active_eq:NN #1
  {
    \char_set_catcode_active:N #1
    \char_set_active_eq:NN #1
  }
⟨/core⟩
⟨*loader⟩
\group_begin:
⟨plain⟩  \char_set_catcode_active:n { `\* }
  \cs_new_protected:Nn \__scontents_plain_disable_outer_par:
⟨*plain⟩
    {
      \group_begin:
        \char_set_lccode:nn { `\* } { `\^^L }
        \tex_lowercase:D { \group_end:
        \tex_let:D * \scan_stop:
      }
    }
⟨/plain⟩
⟨latex|context⟩    { }
\group_end:
⟨/loader⟩
⟨*core⟩
\group_begin:
  \char_set_catcode_active:N \*
  \cs_new_protected:Nn \__scontents_make_control_chars_active:
    {
      \__scontents_plain_disable_outer_par:
      \__scontents_set_active_eq:NN \^^I \__scontents_tab:
      \__scontents_set_active_eq:NN \^^L \__scontents_par:
      \__scontents_set_active_eq:NN \^^M \__scontents_ret:w
    }
\group_end:
⟨/core⟩
```

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

12.4.2 Functions `\@bsphack` and `\@esphack`

`\__scontents_bsphack:`
`\__scontents_esphack:`

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

```
⟨*core⟩
\cs_new_protected:Nn \__scontents_bsphack:
  {
    \scan_stop:
    \mode_if_horizontal:T
      {
        \skip_set_eq:NN \l__scontents_save_skip \tex_lastskip:D
        \int_set_eq:NN \l__scontents_save_sf_int \tex_spacefactor:D
      }
  }
\cs_new_protected:Nn \__scontents_esphack:
  {
    \scan_stop:
    \mode_if_horizontal:T
      {
        \int_set_eq:NN \tex_spacefactor:D \l__scontents_save_sf_int
        \dim_compare:nNnT { \l__scontents_save_skip } > { \c_zero_skip }
          {
            \skip_if_eq:nnT { \tex_lastskip:D } { \c_zero_skip }
              {
                \nobreak
                \skip_horizontal:n { \c_zero_skip }
              }
            \tex_ignorespaces:D
          }
      }
  }
⟨/core⟩
⟨*latex⟩
\cs_gset_eq:NN \__scontents_bsphack: \@bsphack
\cs_gset_eq:NN \__scontents_esphack: \@esphack
⟨/latex⟩
```

(End of definition for `\__scontents_bsphack:` *and* `\__scontents_esphack:`*.)*

12.5 Keys for environment

write-env
write-out
print-env
store-env
force-eol
overwrite
unknown

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

```
⟨*core⟩
\keys_define:nn { scontents / scontents }
  {
    write-env .code:n           = {
                                    \bool_set_true:N \l__scontents_storing_bool
                                    \bool_set_true:N \l__scontents_writing_bool
                                    \tl_set:Nn \l__scontents_file_name_tl {#1}
                                  },
    write-out .code:n           = {
                                    \bool_set_false:N \l__scontents_storing_bool
                                    \bool_set_true:N  \l__scontents_writing_bool
                                    \tl_set:Nn \l__scontents_file_name_tl {#1}
                                  },
    write-env .value_required:n = true,
    write-out .value_required:n = true,
    print-env .meta:nn          = { scontents } { print-env = #1 },
    print-env .default:n        = true,
    store-env .meta:nn          = { scontents } { store-env = #1 },
    force-eol .meta:nn          = { scontents } { force-eol = #1 },
    force-eol .default:n        = true,
    overwrite .meta:nn          = { scontents } { overwrite = #1 },
    overwrite .default:n        = true,
    unknown   .code:n           = { \__scontents_unknown_keys_env:n {#1} },
  }
```

(End of definition for `write-env` *and others.)*

12.5.1 Handling unknown keys for environment scontents

The ⟨*keys*⟩ are save in the string variable `\l_keys_key_str` and the value (if any) is passed as an argument to each ⟨*function*⟩.

__scontents_unknown_keys_env:n
__scontents_unknown_keys_env:nn

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

```
\cs_new_protected:Npn \__scontents_unknown_keys_env:n #1
```

```
  { \exp_args:NV \__scontents_unknown_keys_env:nn \l_keys_key_str {#1} }
\cs_new_protected:Npn \__scontents_unknown_keys_env:nn #1#2
  {
    \tl_if_blank:nTF {#2}
      { \msg_error:nnn { scontents } { env-key-unknown } {#1} }
      { \msg_error:nnnn { scontents } { env-key-value-unknown } {#1} {#2} }
  }
```

(*End of definition for* `\__scontents_unknown_keys_env:n` *and* `\__scontents_unknown_keys_env:nn`*.*)

12.6 Keys for commands

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

12.6.1 Keys for command `\Scontents`

write-cmd
write-out
print-cmd
store-cmd
force-eol
overwrite
unknown

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

```
\keys_define:nn { scontents / Scontents }
  {
    write-cmd .code:n           = {
                                    \bool_set_true:N \l__scontents_storing_bool
                                    \bool_set_true:N \l__scontents_writing_bool
                                    \tl_set:Nn \l__scontents_file_name_tl {#1}
                                  },
    write-out .code:n           = {
                                    \bool_set_false:N \l__scontents_storing_bool
                                    \bool_set_true:N  \l__scontents_writing_bool
                                    \tl_set:Nn \l__scontents_file_name_tl {#1}
                                  },
    write-cmd .value_required:n = true,
    write-out .value_required:n = true,
    print-cmd .meta:nn          = { scontents } { print-cmd = #1 },
    print-cmd .default:n        = true,
    store-cmd .meta:nn          = { scontents } { store-cmd = #1 },
    force-eol .meta:nn          = { scontents } { force-eol = #1 },
    force-eol .default:n        = true,
    overwrite .meta:nn          = { scontents } { overwrite = #1 },
    overwrite .default:n        = true,
    unknown   .code:n           = { \__scontents_unknown_keys_cmd:n {#1} },
  }
```

(*End of definition for* `write-cmd` *and others.*)

12.6.2 Keys for command `\foreachsc`

before
after
start
stop
step
wrapper
sep
unknown

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

```
\keys_define:nn { scontents / foreachsc }
  {
    before  .code:n           = {
                                  \bool_set_true:N \l__scontents_foreach_before_bool
                                  \tl_set:Nn \l__scontents_foreach_before_tl {#1}
                                },
    before  .value_required:n = true,
    after   .code:n           = {
                                  \bool_set_true:N \l__scontents_foreach_after_bool
                                  \tl_set:Nn \l__scontents_foreach_after_tl {#1}
                                },
    after   .value_required:n = true,
    start   .int_set:N        = \l__scontents_foreach_start_int,
    start   .value_required:n = true,
    start   .initial:n        = 1,
    stop    .code:n           = {
                                  \bool_set_true:N \l__scontents_foreach_stop_bool
                                  \int_set:Nn \l__scontents_foreach_stop_int {#1}
                                },
    stop    .value_required:n = true,
    step    .int_set:N        = \l__scontents_foreach_step_int,
    step    .value_required:n = true,
    step    .initial:n        = 1,
    wrapper .code:n           = {
                                  \bool_set_true:N \l__scontents_foreach_wrapper_bool
```

```
                                       \cs_set_protected:Npn
                                       \__scontents_foreach_wrapper:n ##1 {#1}
                                     },
    wrapper .value_required:n = true,
    sep     .tl_set:N         = \l__scontents_foreach_sep_tl,
    sep     .initial:n        = { },
    sep     .value_required:n = true,
    unknown .code:n           = { \__scontents_unknown_keys_cmd:n {#1} },
  }
```

(*End of definition for* `before` *and others.*)

12.6.3 Handling unknown keys for \Scontents and \foreachsc

__scontents_unknown_keys_cmd:n
__scontents_unknown_keys_cmd:nn

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

```
\cs_new_protected:Npn \__scontents_unknown_keys_cmd:n #1
  { \exp_args:NV \__scontents_unknown_keys_cmd:nn \l_keys_key_str {#1} }
\cs_new_protected:Npn \__scontents_unknown_keys_cmd:nn #1#2
  {
    \tl_if_blank:nTF {#2}
      { \msg_error:nnn { scontents } { cmd-key-unknown } {#1} }
      { \msg_error:nnnn { scontents } { cmd-key-value-unknown } {#1} {#2} }
  }
```

(*End of definition for* `\__scontents_unknown_keys_cmd:n` *and* `\__scontents_unknown_keys_cmd:nn`*.*)

12.6.4 Keys for commands \typestored and \meaningsc

width-tab
write-out
overwrite
print-cmd
unknown

We define a ⟨*keys*⟩ for command \typestored and \meaningsc. Both commands accept the same type of *optional arguments*, just define a common ⟨*keys*⟩. Here we will implement the write-out, overwrite and print-cmd keys which are necessary in the implementation of the \mergesc command (§12.17).

```
\keys_define:nn { scontents / typemeaning }
  {
    width-tab .meta:nn    = { scontents } { width-tab = #1 },
    write-out .code:n     = {
                              \bool_set_false:N \l__scontents_storing_bool
                              \bool_set_true:N  \l__scontents_writing_bool
                              \tl_set:Nn \l__scontents_file_name_tl {#1}
                            },
    write-out .value_required:n = true,
    overwrite .meta:nn    = { scontents } { overwrite = #1 },
    overwrite .default:n  = true,
    print-cmd .bool_set:N = \l__scontents_print_verb_style_bool,
    print-cmd .initial:n  = true,
    print-cmd .default:n  = true,
    unknown   .code:n     = { \__scontents_parse_type_meaning_key:n {#1} },
  }
```

(*End of definition for* `width-tab` *and others.*)

12.6.5 Keys for command \mergesc

typestored
meaningsc
__scontents_mergesc_cmd:nn

We define two ⟨*keys*⟩ typestored and meaningsc as *mandatory*, returning an "error" through the function __scontents_mergesc_cmd:nn.

```
\keys_define:nn { scontents / mergesc }
  {
    typestored .code:n =
      {
        \cs_set_eq:NN \__scontents_mergesc_cmd:nn \__scontents_typestored:nn
      },
    typestored .value_forbidden:n = true,
    meaningsc  .code:n =
      {
        \cs_set_eq:NN \__scontents_mergesc_cmd:nn \__scontents_meaningsc:nn
      },
    meaningsc  .value_forbidden:n = true,
  }
\cs_new_protected:Npn \__scontents_mergesc_cmd:nn #1 #2
  {
    \msg_error:nn { scontents } { mergesc-missing-key }
  }
```

(*End of definition for* `typestored` *,* `meaningsc` *, and* `\__scontents_mergesc_cmd:nn`*.*)

12.6.6 Parsing keys for `\typestored`, `\meaningsc` and `\mergesc`

`\__scontents_parse_type_meaning_key:n`
`\__scontents_parse_type_meaning_key:nn`
`\__scontents_parse_type_meaning_range:w`
`\__scontents_range_parser:nnnn`
`\__scontents_range_parser:nnen`
`\__scontents_range_parser_aux:nnn`

The `\typestored`, `\meaningsc` and `\mergesc` commands (which internally uses the previous two) accept an *optional argument* containing the ⟨*index*⟩ position, ⟨*1-end*⟩ or the range of ⟨*start-stop*⟩ positions of the ⟨*stored content*⟩ in the *sequence* along with other ⟨*keys*⟩.

To avoid the awkward `\typestored[...][`⟨*keys*⟩`]{...}` syntax, we'll make the commands have a single *optional argument* which is processed by l3keys[9], and the *unknown* ⟨*keys*⟩ are brought here to `\__scontents_parse_typemeaning_key:n` to process.

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

```
\cs_new_protected:Npn \__scontents_parse_type_meaning_key:n #1
  { \exp_args:NV \__scontents_parse_type_meaning_key:nn \l_keys_key_str {#1} }
\cs_new_protected:Npn \__scontents_parse_type_meaning_key:nn #1#2
  {
    \tl_if_blank:nTF {#2}
      { \__scontents_parse_type_meaning_range:w #1 - \q__scontents_mark - \s__scontents_mark }
      { \msg_error:nnee { scontents } { cmd-key-value-unknown } {#1} {#2} }
  }
\cs_new_protected:Npn \__scontents_parse_type_meaning_range:w #1 - #2 - #3 \s__scontents_mark
  {
    \__scontents_range_parser:nnen {#1} {#2}
      { \seq_count:c { g__scontents_name_ \l__scontents_current_seq_name_str _seq } }
      { \msg_error:nnn { scontents } { cmd-key-unknown } {#1} }
  }
\cs_new_protected:Npn \__scontents_range_parser:nnnn #1 #2 #3 #4
  {
    \exp_args:Nee \__scontents_range_parser_aux:nnn
      { \str_if_eq:nnTF {#1} { end } {#3} { \exp_not:n {#1} } }
      { \str_if_eq:nnTF {#2} { end } {#3} { \exp_not:n {#2} } }
      { #4 }
  }
\cs_generate_variant:Nn \__scontents_range_parser:nnnn { nnen }
\cs_new_protected:Npn \__scontents_range_parser_aux:nnn #1 #2 #3
  {
    \__scontents_tl_if_head_is_q_mark:nTF {#2}
      {
        \tl_if_empty:fTF { \int_to_roman:n { -0 #1 } }
          { \seq_put_right:Ne \l__scontents_seq_item_seq { \int_eval:n {#1} } }
          { #3 {#1} }
      }
      {
        \bool_lazy_and:nnTF
          { \tl_if_empty_p:f { \int_to_roman:n { -0 #1 } } }
          { \tl_if_empty_p:f { \int_to_roman:n { -0 #2 } } }
          {
            \int_compare:nNnTF {#2} > {#1}
              { \int_step_inline:nnnn {#1} { 1 } {#2} }
              { \int_step_inline:nnnn {#1} { -1 } {#2} }
              { \seq_put_right:Nn \l__scontents_seq_item_seq {##1} }
          }
          { #3 { #1-#2 } }
      }
  }
⟨/core⟩
```

(*End of definition for* `\__scontents_parse_type_meaning_key:n` *and others.*)

12.7 Functions for sequences

The *storage system* of the package SCONTENTS is done using seq variables. Here we set up the macros that will manage the variables.

`\__scontents_append_contents:nn`
`\__scontents_append_contents:Ve`

The function `\__scontents_append_contents:nn` creates a *sequence* {⟨*seq name*⟩} pass to `#1` if one didn't exist and appends the {⟨*body env*⟩} for environments or {⟨*argument*⟩} for commands to the right of the *sequence* {⟨*seq name*⟩} pass to `#2`.

```
⟨*core⟩
\cs_new_protected:Npn \__scontents_append_contents:nn #1#2
  {
    \tl_if_blank:nT {#1}
      { \msg_error:nn { scontents } { empty-store-cmd } }
    \seq_if_exist:cF { g__scontents_name_#1_seq }
      { \seq_new:c { g__scontents_name_#1_seq } }
    \seq_gput_right:cn { g__scontents_name_#1_seq } {#2}
  }
\cs_generate_variant:Nn \__scontents_append_contents:nn { Ve }
```

(*End of definition for* `\__scontents_append_contents:nn`.)

`\__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`.

```
\cs_new_protected:Npn \__scontents_store_to_seq:NN #1#2
  {
    \tl_log:N #1
    \__scontents_append_contents:Ve #2 { \exp_not:V #1 }
  }
```

(*End of definition for* `\__scontents_store_to_seq:NN`.)

`\__scontents_finish_storing:NNN`

The `\__scontents_finish_storing:NNN` function will first check if we are in standard *storing mode*, that is, the `write-out` key is NOT active, then the state of the variable `\l__scontents_forced_eol_bool` set by the `force-eol` key is checked and if this is “false” (default value) we will add `\c__scontents_hidden_space_str` to the end of the token list passed in `{#1}` which contains {⟨*body env*⟩} for the generic environment `scontents` and the {⟨*argument*⟩} for the `\Scontents` command.

Then the function `\__scontents_store_to_seq:NN` is applied to “store” in the *sequence* passed in `{#2}` and finally the state of the boolean variable passed in `{#3}` established by the `print-env` and `print-cmd` keys is checked and if it is “true”, {⟨*body env*⟩} or {⟨*argument*⟩} will be printed from the *sequence* in which it was stored by means of the `\__scontents_lastfrom_seq:V` function.

```
\cs_new_protected:Npn \__scontents_finish_storing:NNN #1 #2 #3
  {
    \bool_if:NT \l__scontents_storing_bool
      {
        \bool_if:NF \l__scontents_forced_eol_bool
          { \tl_put_right:Ne #1 { \c__scontents_hidden_space_str } }
        \__scontents_store_to_seq:NN #1 #2
        \bool_if:NT #3 { \__scontents_lastfrom_seq:V #2 }
      }
  }
```

(*End of definition for* `\__scontents_finish_storing:NNN`.)

`\__scontents_getfrom_seq:Nn`
`\__scontents_getfrom_seq:nNn`
`\__scontents_getfrom_seq_aux:nnn`
`\__scontents_getfrom_seq:nn`
`\__scontents_getfrom_seq:nnn`

The function `\__scontents_getfrom_seq:Nn` retrieves the ⟨*stored content*⟩ pass to `{#1}` from the *sequence* {⟨*seq name*⟩} pass to `{#2}`.

```
\cs_new:Npn \__scontents_getfrom_seq:Nn #1#2
  {
    \seq_if_exist:cTF { g__scontents_name_#2_seq }
      {
        \exp_args:Nf \__scontents_getfrom_seq:nNn
          { \seq_count:c { g__scontents_name_#2_seq } } #1 {#2}
      }
      { \msg_expandable_error:nnn { scontents } { undefined-storage } {#2} }
  }
\cs_new:Npn \__scontents_getfrom_seq:nNn #1 #2 #3
  {
    \seq_map_tokens:Nn #2 { \__scontents_getfrom_seq_aux:nnn {#1} {#3} }
  }
\cs_new:Npn \__scontents_getfrom_seq_aux:nnn #1 #2 #3
  {
    \exp_args:Nnf \use:n { \__scontents_getfrom_seq:nnn {#1} } { \int_eval:n {#3} } {#2}
  }
\cs_new:Npn \__scontents_getfrom_seq:nn #1#2
  {
    \seq_if_exist:cTF { g__scontents_name_#2_seq }
      {
        \exp_args:Nf \__scontents_getfrom_seq:nnn
          { \seq_count:c { g__scontents_name_#2_seq } }
          {#1} {#2}
      }
      { \msg_expandable_error:nnn { scontents } { undefined-storage } {#2} }
  }
\cs_new:Npn \__scontents_getfrom_seq:nnn #1#2#3
  {
    \bool_lazy_or:nnTF
      { \int_compare_p:nNn {#2} = { 0 } }
      { \int_compare_p:nNn { \int_abs:n {#2} } > {#1} }
      { \msg_expandable_error:nnnnn { scontents } { index-out-of-range } {#2} {#3} {#1} }
      { \seq_item:cn { g__scontents_name_#3_seq } {#2} }
  }
```

(*End of definition for* `\__scontents_getfrom_seq:Nn` *and others.*)

`\__scontents_lastfrom_seq:n`
`\__scontents_lastfrom_seq:V`

The function `\__scontents_lastfrom_seq:n` save the last ⟨*stored content*⟩ from the *sequence* pass to `#1` in `\g__scontents_last_tl` then rescan with the function `\__scontents_rescan_tokens:V` when the keys `print-env` or `print-cmd` are active.

```
\cs_new_protected:Npn \__scontents_lastfrom_seq:n #1
  {
    \tl_gset:Ne \g__scontents_last_stored_tl
      {
        \seq_item:cn { g__scontents_name_#1_seq } { -1 }
      }
    \group_insert_after:N \__scontents_rescan_tokens:V
    \group_insert_after:N \g__scontents_last_stored_tl
    \group_insert_after:N \tl_gclear:N
    \group_insert_after:N \g__scontents_last_stored_tl
  }
\cs_generate_variant:Nn \__scontents_lastfrom_seq:n { V }
⟨/core⟩
```

(*End of definition for* `\__scontents_lastfrom_seq:n`.)

12.8 Base functions for environments

In version 1.8 (2019-11-18) the command `\newenvsc` (§12.9) was implemented, allowing you to create environments with the same behavior as the base environment `scontents`. Since that version, the base environment `scontents` (§12.10) is defined using `\newenvsc`.

The references to `\begin{scontents}` or `\end{scontents}` described in this section are for illustrative purposes only, but apply to any environment defined using `\newenvsc`.

12.8.1 Functions for keyval of environment

`\__scontents_grab_opt_arg:n`
`\__scontents_grab_opt_arg:w`

The function `\__scontents_grab_opt_arg: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 ltcmd[10] to exploit its delimited argument processor.

```
⟨*core⟩
\NewDocumentCommand \__scontents_grab_opt_arg:w { r[] }
  {
    \__scontents_grab_opt_arg:n {#1}
  }
```

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.

```
\cs_new_protected:Npn \__scontents_grab_opt_arg:n #1
  {
    \tl_if_novalue:nF {#1}
      {
        \tl_set:Nn \l__scontents_environment_keys_tl {#1}
        \__scontents_normalise_line_ends:N \l__scontents_environment_keys_tl
        \keys_set:nV { scontents / scontents } \l__scontents_environment_keys_tl
      }
    \__scontents_start_after_option:w
  }
⟨/core⟩
```

(*End of definition for* `\__scontents_grab_opt_arg:n` *and* `\__scontents_grab_opt_arg:w`.)

12.8.2 Functions for save the body of environment

__scontents_start_environment:w
__scontents_start_after_option:w
__scontents_check_line_process:en
__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_opt_arg:w is called to do the heavy lifting. __scontents_grab_opt_arg: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:en checks that everything past \begin{scontents} is empty and then process the environment.

The function __scontents_check_line_process:en 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 xsim[4] by Clemens Niederberger.

```
⟨*core⟩
\group_begin:
  \char_set_catcode_active:N \^^I
  \char_set_catcode_active:N \^^L
  \char_set_catcode_active:N \^^M
  \cs_new_protected:Npn \__scontents_normalise_line_ends:N #1
    { \tl_replace_all:Nnn #1 { ^^M } { ~ } }
  \cs_new_protected:Npn \__scontents_start_environment:w #1 ^^M
    {
      \tl_if_head_is_N_type:nTF {#1}
        {
          \str_if_eq:eeTF { \tl_head:n {#1} } { [ }
            { \__scontents_grab_opt_arg:w #1 ^^M }
            { \__scontents_check_line_process:en { } {#1} }
        }
        { \__scontents_check_line_process:en { } {#1} }
    }
  \cs_new_protected:Npn \__scontents_start_after_option:w #1 ^^M
    { \__scontents_check_line_process:en { [...] } {#1} }
  \cs_new_protected:Npn \__scontents_check_line_process:en #1 #2
    {
      \tl_if_blank:nF {#2}
        {
          \msg_error:nnee { scontents } { junk-after-begin }
            { after~\c_backslash_str begin { \l__scontents_env_name_tl } #1 } {#2}
        }
      \__scontents_make_control_chars_active:
      \__scontents_file_tl_write_start:V \l__scontents_file_name_tl
    }
  \cs_new_protected:Nn \__scontents_stop_environment:
    {
      \__scontents_file_write_stop:N \l__scontents_processed_body_lines_tl
      \bool_lazy_and:nnT
        { \l__scontents_storing_bool }
        { \tl_if_empty_p:N \l__scontents_processed_body_lines_tl }
        {
          \msg_warning:nne { scontents } { empty-environment } { \l__scontents_env_name_tl }
        }
    }
```

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

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

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

```
  \cs_new_protected:Npn \__scontents_file_tl_write_start:n #1
    {
      \group_begin:
```

```
      \__scontents_file_if_writable:nTF {#1}
        {
          \bool_set_true:N \l__scontents_writable_bool
          \iow_open:Nn \l__scontents_file_write_iow {#1}
        }
        { \bool_set_false:N \l__scontents_writable_bool }
      \tl_clear:N \l__scontents_save_every_body_lines_tl
      \seq_map_function:NN \l_char_special_seq \char_set_catcode_other:N
      \int_step_function:nnnN { 128 } { 1 } { 255 } \char_set_catcode_letter:n
      \cs_set_protected:Npe \__scontents_ret:w ##1 ^^M
        {
          \exp_not:N \__scontents_verb_processor_iterate:w
          ##1 \c__scontents_end_env_tl
              \c__scontents_end_env_tl
              \exp_not:N \q__scontents_stop
        }
      \__scontents_make_control_chars_active:
      \__scontents_ret:w
  }
\cs_new:Nn \__scontents_setup_verb_processor:
  {
    \use:e
      {
        \cs_set:Npn \exp_not:N \__scontents_verb_processor_iterate:w
          ##1 \c__scontents_end_env_tl
          ##2 \c__scontents_end_env_tl
          ##3 \exp_not:N \q__scontents_stop
      }   { \__scontents_verb_processor_iterate:nnn {##1} {##2} {##3} }
  }
\cs_new:Npn \__scontents_verb_processor_iterate:nnn #1 #2 #3
  {
    \tl_if_blank:nTF {#3}
      {
        \__scontents_analyse_nesting:n {#1}
        \__scontents_verb_processor_output:n {#1}
      }
      {
        \__scontents_if_nested:TF
          {
            \__scontents_nesting_decr:
            \__scontents_verb_processor_output:e
              { \exp_not:n {#1} \c__scontents_end_env_tl \exp_not:n {#2} }
          }
          {
            \tl_if_blank:nF {#1}
              { \__scontents_verb_processor_output:n {#1} }
            \cs_set_protected:Npe \__scontents_ret:w
              {
                \__scontents_env_end_function:
                \bool_lazy_or:nnF
                  { \tl_if_blank_p:n {#2} }
                  { \str_if_eq_p:ee {#2} { \c_percent_str } }
                  {
                    \str_if_eq:VnF \c__scontents_hidden_space_str {#2}
                      {
                        \msg_warning:nnnn { scontents } { rescanning-text }
                          {#2} { \tl_use:N \l__scontents_env_name_tl }
                      }
                    \__scontents_rescan_tokens:n {#2}
                  }
              }
            \char_set_active_eq:NN ^^M \__scontents_ret:w
          }
      }
    ^^M
  }
\cs_new:Nn \__scontents_env_end_function:
  {
    \__scontents_format_case:nnn
      { \exp_not:N \end { \if_false: } \fi: }
      { \exp_after:wN \exp_not:N \cs:w end }
```

```
          { \exp_after:wN \exp_not:N \cs:w stop }
        \tl_use:N \l__scontents_env_name_tl
        \__scontents_format_case:nnn
          { \if_false: { \fi: } }
          { \cs_end: }
          { \cs_end: }
      }
    \cs_new_protected:Npn \__scontents_file_write_stop:N #1
      {
        \bool_if:NT \l__scontents_writable_bool
          { \iow_close:N \l__scontents_file_write_iow }
        \use:e
          {
            \group_end:
            \bool_if:NT \l__scontents_storing_bool
              {
                \tl_set:Nn \exp_not:N #1
                  {
                    \exp_args:NV
                      \__scontents_remove_leading_nl:n \l__scontents_save_every_body_lines_tl
                  }
              }
          }
      }
    \cs_new:Npn \__scontents_remove_leading_nl:n #1
      {
        \tl_if_head_is_N_type:nTF {#1}
          {
            \exp_args:Nf
              \__scontents_remove_leading_nl:nn
                { \tl_head:n {#1} } {#1}
          }
          { \exp_not:n {#1} }
      }
    \cs_new:Npn \__scontents_remove_leading_nl:nn #1 #2
      {
        \token_if_eq_meaning:NNTF ^^J #1
          { \exp_not:o { \__scontents_remove_leading_nl:w #2 } }
          { \exp_not:n {#2} }
      }
    \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:e`

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`.

```
    \cs_new_protected:Npn \__scontents_verb_processor_output:n #1
      {
        \bool_if:NT \l__scontents_writable_bool
          { \iow_now:Nn \l__scontents_file_write_iow {#1} }
        \bool_if:NT \l__scontents_storing_bool
          { \tl_put_right:Nn \l__scontents_save_every_body_lines_tl { ^^J #1 } }
      }
  \group_end:
  \cs_generate_variant:Nn \__scontents_verb_processor_output:n { e }
  \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:w`
`\__scontents_nesting_decr:`
`\__scontents_use_none_delimit_by_q_stop:w`
`\__scontents_if_nested:TF`

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

```
  \cs_new_protected:Npn \__scontents_analyse_nesting:n #1
    {
      \int_zero:N \l__scontents_nesting_aux_int
      \__scontents_analyse_nesting_format:n {#1}
      \int_compare:nNnT { \l__scontents_nesting_aux_int } > { 1 }
      { \msg_warning:nn { scontents } { multiple-begin } }
  }
\cs_new_protected:Nn \__scontents_nesting_incr:
  {
    \int_incr:N \l__scontents_nesting_env_int
    \int_incr:N \l__scontents_nesting_aux_int
  }
\cs_new_protected:Nn \__scontents_nesting_decr:
  {
    \int_decr:N \l__scontents_nesting_env_int
  }
\prg_new_protected_conditional:Npnn \__scontents_if_nested: { TF }
  {
    \int_compare:nNnTF { \l__scontents_nesting_env_int } > { \c_zero_int }
      { \prg_return_true: }
      { \prg_return_false: }
  }
```

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

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.

```
\use:e
  {
    \cs_new_protected:Npn \exp_not:N \__scontents_analyse_nesting_latex:w #1
      \c_backslash_str begin \c_left_brace_str #2 \c_right_brace_str
  }   {
        \__scontents_tl_if_head_is_q_mark:nTF {#2}
          { \__scontents_use_none_delimit_by_q_stop:w }
          {
            \str_if_eq:VnT \l__scontents_env_name_tl {#2}
              { \__scontents_nesting_incr: }
            \__scontents_analyse_nesting_latex:w
          }
      }
\cs_new_protected:Npe \__scontents_analyse_nesting_latex:n #1
  {
    \__scontents_analyse_nesting_latex:w #1
      \c_backslash_str begin
        \c_left_brace_str \exp_not:N \q__scontents_mark \c_right_brace_str
    \exp_not:N \q__scontents_stop
  }
```

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).

```
\cs_new_protected:Npn \__scontents_analyse_nesting_generic_process:nn #1 #2
  {
    \tl_if_head_is_N_type:nTF {#2}
      {
        \__scontents_tl_if_head_is_q_mark:nF {#2}
          {
            \__scontents_nesting_incr:
            \__scontents_analyse_nesting_generic:w #2 \q__scontents_stop
          }
      }
      { \__scontents_analyse_nesting_generic:w #2 \q__scontents_stop }
  }
\cs_new_protected:Npn \__scontents_analyse_nesting_generic:nn #1 #2
  {
    \__scontents_define_generic_nesting_function:n {#1}
    \use:e
      {
        \exp_not:N \__scontents_analyse_nesting_generic:w #2
          \c_backslash_str #1 \tl_use:N \l__scontents_env_name_tl
            \exp_not:N \q__scontents_mark \exp_not:N \q__scontents_stop
      }
  }
\cs_new_protected:Npn \__scontents_define_generic_nesting_function:n #1
  {
    \use:e
      {
        \cs_set_protected:Npn \exp_not:N \__scontents_analyse_nesting_generic:w ##1
          \c_backslash_str #1 \tl_use:N \l__scontents_env_name_tl
            ##2 \exp_not:N \q__scontents_stop
      }   { \__scontents_analyse_nesting_generic_process:nn {##1} {##2} }
  }
⟨/core⟩
```

Now we just need to call the `\__scontents_analyse_nesting_format:n` function to analyze the nesting.

```
⟨*loader⟩
⟨latex⟩\cs_new_eq:NN \__scontents_analyse_nesting_format:n
⟨latex⟩  \__scontents_analyse_nesting_latex:n
⟨!latex⟩\cs_new_protected:Npn \__scontents_analyse_nesting_format:n
⟨plain⟩  { \__scontents_analyse_nesting_generic:nn { } }
⟨context⟩  { \__scontents_analyse_nesting_generic:nn { start } }
⟨/loader⟩
```

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

12.9 The command `\newenvsc`

In version 1.8 (2019-11-18) the command `\newenvsc` was implemented, allowing you to create environments with the same behavior as the base environment `scontents`. To achieve this, we will create new environments so that they wrap around the base functions (§12.8).

`\__scontents_generic_begin:`
`\__scontents_generic_end:`

The function `\__scontents_generic_begin:` leaves the '`^^M`' character active and calls the generic environment start function `\__scontents_start_environment:w`.

```
⟨*core⟩
\cs_new_protected:Npn \__scontents_generic_begin:
  {
    \char_set_catcode_active:N \^^M
    \__scontents_start_environment:w
  }
```

The function `\__scontents_generic_end:` calls the generic environment stop function `\__scontents_stop_environment:` and finally calls the function `\__scontents_finish_storing:NNN` which stores {⟨*body env*⟩} in the *sequence* {⟨*seq name*⟩} and prints it from the *sequence* if the `print-env` key is active.

```
\cs_new_protected:Npn \__scontents_generic_end:
  {
    \__scontents_stop_environment:
    \__scontents_finish_storing:NNN
      \l__scontents_processed_body_lines_tl
      \l__scontents_name_seq_env_tl
      \l__scontents_print_env_bool
  }
```

(*End of definition for* `\__scontents_generic_begin:` *and* `\__scontents_generic_end:`.)

`\__scontents_setting_env:nn`
`\__scontents_define_env:nnn`

The function `\__scontents_setting_env:nn` receives the environment *name* passed in `{#1}` and save it in the variable `\l__scontents_env_name_tl` along with the initial ⟨*keys*⟩ passed in `{#2}`.

Two functions will be created `\__scontents_#1_begin:` and `\__scontents_#1_end:` which will internally call the `\__scontents_generic_begin:` and `\__scontents_generic_end:` functions and expand the arguments of the function `\__scontents_define_env:nnn` function.

```
\cs_new_protected:Npn \__scontents_setting_env:nn #1 #2
  {
    \cs_new_protected:cpn { __scontents_#1_begin: }
      {
        \tl_set:Nn \l__scontents_env_name_tl {#1}
        \keys_set:nn { scontents } {#2}
        \__scontents_setup_verb_processor:
        \__scontents_generic_begin:
      }
    \cs_new_protected:cpn { __scontents_#1_end: }
      { \__scontents_generic_end: }
    \exp_args:Nooo \__scontents_define_env:nnn % http://nooooooooooooooo.com :) jeje
      { \tl_to_str:n {#1} }
      { \cs:w __scontents_#1_begin: \cs_end: }
      { \cs:w __scontents_#1_end: \cs_end: }
  }
⟨/core⟩
```

The function `\__scontents_define_env:nnn` will create the environments for LaTeX, plain TeX and ConTeXt.

```
⟨*loader⟩
\cs_new_protected:Npn \__scontents_define_env:nnn #1 #2 #3
  {
⟨latex|plain⟩    \NewDocumentEnvironment {#1} { }
⟨context⟩    \cs_new_protected:cpn { start #1 }
      {
⟨!latex⟩        \group_begin:
        #2
      }
⟨context⟩    \cs_new_protected:cpn { stop #1 }
      {
        #3
⟨!latex⟩        \group_end:
      }
  }
```

(*End of definition for* `\__scontents_setting_env:nn` *and* `\__scontents_define_env:nnn`*.*)

`\newenvsc` Now we just need to create the user command `\newenvsc` for LaTeX, plain TeX and ConTeXt.

```
\NewDocumentCommand \newenvsc { m O{} }
  {
⟨latex|plain⟩    \cs_if_exist:cTF { #1 }
⟨context⟩    \cs_if_exist:cTF { start #1 }
      { \msg_error:nnn { scontents } { env-already-defined } {#1} }
      { \__scontents_setting_env:nn {#1} {#2} }
  }
⟨/loader⟩
```

(*End of definition for* `\newenvsc`*. This function is documented on page 5.*)

12.10 The environment scontents

`scontents` `\scontents` `\endscontents` `\startscontents` `\stopscontents` Finally defining the `scontents` environment should be easy :)

```
⟨*loader⟩
\newenvsc{scontents}
⟨/loader⟩
```

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

12.11 The environment verbatimsc

The `verbatimsc` environment is, in a way, a customized version of the standard `verbatim` environment provided by LaTeX. For correct operation in plain TeX, LaTeX and ConTeXt, we must add a couple of additional functions.

`\dospecials` The `verbatim` environment in LaTeXrequires `\dospecials`. In case it doesn't exist (at the time scontents is loaded) we define `\dospecials` to use the `\l_char_special_seq`.

```
⟨*!latex⟩
\cs_if_exist:NF \dospecials
  {
    \cs_new:Npn \dospecials
      { \seq_map_function:NN \l_char_special_seq \do }
  }
⟨/!latex⟩
```

(*End of definition for* `\dospecials`*.*)

`\__scontents_xverb:w` `\end{verbatimsc}` `\endverbatimsc` `\stopverbatimsc` The environment `verbatimsc` needs to literally find the end of this `\end{verbatimsc}` in the case of LaTeX. Here we set this for plain TeX, LaTeX, and ConTeXt.

```
⟨*loader⟩
⟨*!context⟩
\use:e
  {
    \cs_new_protected:Npn \exp_not:N \__scontents_xverb:w
      #1 \g__scontents_end_verbatimsc_tl
⟨latex⟩      { #1 \exp_not:N \end{verbatimsc} }
⟨plain⟩      { #1 \exp_not:N \endverbatimsc }
⟨context⟩      { #1 \exp_not:N \stopverbatimsc }
  }
⟨/!context⟩
⟨/loader⟩
```

(*End of definition for* `\__scontents_xverb:w` *and others.*)

12.11.1 plain TeX version off `verbatimsc`

`\verbatimsc`
`\endverbatimsc`
`\__scontents_verbatimsc_aux:`
`\__scontents_vobeyspaces:`
`\__scontents_xverb:`
`\__scontents_nolig_list:`
`\__scontents_xobeysp:`

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

```
⟨*plain⟩
\cs_new_protected:Npn \verbatimsc
  {
    \group_begin:
      \__scontents_verbatimsc_aux: \frenchspacing \__scontents_vobeyspaces:
      \__scontents_xverb:
  }
\cs_new_protected:Npn \endverbatimsc
  { \group_end: }
\cs_new_protected:Nn \__scontents_verbatimsc_aux:
  {
    \skip_vertical:N \parskip
    \dim_zero:N \parindent
    \skip_set:Nn \parfillskip { 0pt plus 1fil }
    \skip_set:Nn \parskip { 0pt plus0pt minus0pt }
    \tex_par:D
    \bool_set_false:N \l__scontents_plain_bool
    \cs_set:Npn \par
      {
        \bool_if:NTF \l__scontents_plain_bool
          {
            \mode_leave_vertical:
            \null
            \tex_par:D
            \penalty \interlinepenalty
          }
          {
            \bool_set_true:N \l__scontents_plain_bool
            \mode_if_horizontal:T
              { \tex_par:D \penalty \interlinepenalty }
          }
      }
    \cs_set_eq:NN \do \char_set_catcode_other:N
    \dospecials \obeylines
    \tl_use:N \l__scontents_verb_font_tl
    \cs_set_eq:NN \do \__scontents_do_noligs:N
    \__scontents_nolig_list:
    \tex_everypar:D \exp_after:wN
      { \tex_the:D \tex_everypar:D \tex_unpenalty:D }
  }
\cs_new_protected:Nn \__scontents_nolig_list:
  { \do\`\do\<\do\>\do\,\do\'\do\- }
\cs_new_protected:Nn \__scontents_vobeyspaces:
  { \__scontents_set_active_eq:NN \  \__scontents_xobeysp: }
\cs_new_protected:Nn \__scontents_xobeysp:
  { \mode_leave_vertical: \nobreak \ }
⟨/plain⟩
```

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

12.11.2 ConTeXt version off `verbatimsc`

`\startverbatimsc`
`\stopverbatimsc`

In ConTeXt we use our own tool `\definetyping`.

```
⟨*loader⟩
⟨context⟩\definetyping[verbatimsc]
⟨/loader⟩
```

(*End of definition for* `\startverbatimsc` *and* `\stopverbatimsc`*.*)

12.11.3 LaTeX version off verbatimsc

__scontents_verbatimsc_instance:
verbatimsc

To be compatible with *tagged* PDF we must define the environment verbatimsc in terms of the xtemplate[10] module integrated into the LaTeX kernel and the new blocks-code from latex-lab[13]. This code is adapted directly from Mrs. Ulrike Fischer's answer to New verbatim environment with block code (tagged-pdf).

```
⟨*loader⟩
⟨*latex⟩
\cs_new_protected:Nn \__scontents_verbatimsc_instance:
  {
    \DeclareInstance{blockenv}{verbatimsc}{display}
      {
        env-name       = verbatimsc,
        tag-name       = verbatim,
        tag-class      = ,
        tagging-recipe = standard,
        inner-level-counter = ,
        level-increase = false,
        setup-code     = ,
        block-instance = displayblock,
        inner-instance = ,
        final-code     = \legacyverbatimsetup \tag_tool:n {paratag=codeline},
        para-flattened = true
      }
  }
\NewDocumentEnvironment { verbatimsc } { }
  {
    \IfDocumentMetadataTF
      {
        \__scontents_verbatimsc_instance:
        \UseInstance{blockenv}{verbatimsc}{}
        \@setupverbinvisiblespace\frenchspacing\@vobeyspaces
        \__scontents_xverb:
      }
      {
        \cs_set_eq:cN { @xverbatim } \__scontents_xverb:
        \verbatim
      }
  }
```

The \endverbatim in the second argument of the verbatimsc environment is only needed for compatibility with the verbatim[15] package.

```
  {
    \IfDocumentMetadataTF
      {
        \endblockenv
      }
      { \endverbatim }
  }
⟨/latex⟩
⟨/loader⟩
```

(*End of definition for* __scontents_verbatimsc_instance: *and* verbatimsc. *This function is documented on page 7.*)

12.12 The command \Scontents

\Scontents
__scontents_Scontents_code:nn
__scontents_Scontents_norm_arg:n
__scontents_Scontents_verb_arg:w

User command \Scontents to ⟨*stored content*⟩ in a *sequence*, adapted from code by Ulrich Diez in Stringify input - \string on token list and code by user @siracusa in Convert a macro from Latex2e to expl3.

```
⟨*core⟩
\NewDocumentCommand \Scontents { !s !O{} }
  {
    \tl_set:Nn \l__scontents_cmd_name_tl { Scontents }
    \__scontents_Scontents_code:nn {#1} {#2}
  }
```

The internal function __scontents_Scontents_code:nn first executes __scontents_bsphack:, opens a group and checks the ⟨*keys*⟩ passed in {#2}, leaves the TAB character active and then checks the *starred argument* '*' passed in {#1}. If the latter is present it will call the function __scontents_Scontents_verb_arg:w otherwise it will call the function __scontents_Scontents_norm_arg:n.

```
\cs_new_protected:Npn \__scontents_Scontents_code:nn #1 #2
```

```
  {
    \__scontents_bsphack:
    \group_begin:
      \tl_if_novalue:nF {#2}
        { \keys_set:nn { scontents / Scontents } {#2} }
      \char_set_catcode_active:n { 9 }
      \bool_if:NTF #1
        { \__scontents_Scontents_verb_arg:w }
        { \__scontents_Scontents_norm_arg:n }
  }
```

The function __scontents_Scontents_verb_arg:w saves the {⟨*argument*⟩} passed in {#1} under the *verbatim catcode* using v-type argument from ltcmd in \l__scontents_Scontents_arg_tl and replaces all '^^M' and \obeyedline added in LaTeX release 2024-06-01 by '^^J' and then calls the function __scontents_Scontents_finish:. Here we will apply \RenewDocumentCommand since \obeyedline can be modified by the user and if so the code would return a low-level error.

```
\NewDocumentCommand \__scontents_Scontents_verb_arg:w { +v }
  {
    \tl_set:Nn \l__scontents_Scontents_arg_tl {#1}
    \cs_if_exist:NT \obeyedline
      {
        \RenewDocumentCommand \obeyedline { } { \iow_char:N \^^M }
        \tl_replace_all:Nee \l__scontents_Scontents_arg_tl { \obeyedline } { \iow_char:N \^^M }
      }
    \tl_replace_all:Nee \l__scontents_Scontents_arg_tl { \iow_char:N \^^M } { \iow_char:N \^^J }
    \__scontents_Scontents_finish:
  }
```

The function __scontents_Scontents_norm_arg:n saves the {⟨*argument*⟩} passed in {#1} in the *standard catcode* regimen and then calls the function __scontents_Scontents_finish:.

```
\cs_new_protected:Npn \__scontents_Scontents_norm_arg:n #1
  {
    \tl_set:Nn \l__scontents_Scontents_arg_tl {#1}
    \__scontents_Scontents_finish:
  }
```

The function __scontents_Scontents_finish: will first call the function __scontents_file_write_cmd:VV used by the write-out and write-cmd keys, then call the function __scontents_finish_storing:NNN to *store* the {⟨*argument*⟩} passed to the \Scontents command saved in the \l__scontents_Scontents_arg_tl variable in the *sequence* \l__scontents_name_seq_cmd_tl and print it from this *sequence* according to the state of the \l__scontents_print_cmd_bool variable set by the print-cmd key. Finally it will close the *group* opened at the beginning of the command definition and run __scontents_esphack: if the print-cmd key is not active.

```
\cs_new_protected:Nn \__scontents_Scontents_finish:
  {
    \__scontents_file_write_cmd:VV \l__scontents_file_name_tl \l__scontents_Scontents_arg_tl
    \__scontents_finish_storing:NNN
      \l__scontents_Scontents_arg_tl \l__scontents_name_seq_cmd_tl \l__scontents_print_cmd_bool
    \use:e
      {
        \group_end:
        \bool_if:NF \l__scontents_print_cmd_bool { \__scontents_esphack: }
      }
  }
```

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

12.13 The command \getstored

\getstored
__scontents_getstored:nn

User command \getstored to extract ⟨*stored content*⟩ in *sequence* (robust).

```
\NewDocumentCommand \getstored { O{-1} m }
  {
    \__scontents_getstored:nn {#1} {#2}
  }
```

The internal function __scontents_getstored:nn will set the end of line then apply the function __scontents_rescan_tokens:e on the ⟨*stored content*⟩ at the ⟨*index*⟩ given by {#1} in the sequence {#2} via the function __scontents_getfrom_seq:nn.

```
\cs_new_protected:Npn \__scontents_getstored:nn #1 #2
```

```
  {
    \group_begin:
      \int_set:Nn \tex_newlinechar:D { `\^^J }
      \__scontents_rescan_tokens:e
        {
    \endgroup % This assumes \catcode`\\=0... Things might go off otherwise.
    \__scontents_getfrom_seq:nn {#1} {#2}
        }
  }
```

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

12.14 The command \foreachsc

\foreachsc
__scontents_foreachsc:nn
__scontents_foreach_add_body:n

User command \foreachsc to loop over ⟨*stored content*⟩ in *sequence.*

```
\NewDocumentCommand \foreachsc { o m }
  {
    \tl_set:Nn \l__scontents_cmd_name_tl { foreachsc }
    \__scontents_foreachsc:nn {#1} {#2}
  }
\cs_new_protected:Npn \__scontents_foreachsc:nn #1 #2
  {
    \group_begin:
      \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / foreachsc } {#1} }
      \tl_set:Nn \l__scontents_foreachsc_arg_tl {#2}
      \seq_clear:N \l__scontents_foreach_print_seq
      \bool_if:NF \l__scontents_foreach_stop_bool
        {
          \int_set:Nn \l__scontents_foreach_stop_int
            { \seq_count:c { g__scontents_name_#2_seq } }
        }
      \int_step_function:nnnN
        { \l__scontents_foreach_start_int }
        { \l__scontents_foreach_step_int }
        { \l__scontents_foreach_stop_int }
        \__scontents_foreach_add_body:n
      \tl_gset:Ne \g__scontents_foreach_exec_tl
        {
          \exp_args:NNV \seq_use:Nn
            \l__scontents_foreach_print_seq \l__scontents_foreach_sep_tl
        }
    \group_end:
    \exp_after:wN \tl_gclear:N
    \exp_after:wN \g__scontents_foreach_exec_tl
    \g__scontents_foreach_exec_tl
  }
\cs_new_protected:Npn \__scontents_foreach_add_body:n #1
  {
    \seq_put_right:Ne \l__scontents_foreach_print_seq
      {
        \bool_if:NT \l__scontents_foreach_before_bool
          { \exp_not:V \l__scontents_foreach_before_tl }
        \bool_if:NTF \l__scontents_foreach_wrapper_bool
          { \__scontents_foreach_wrapper:n }
          { \use:n }
            { \getstored [#1] { \tl_use:N \l__scontents_foreachsc_arg_tl } }
        \bool_if:NT \l__scontents_foreach_after_bool
          { \exp_not:V \l__scontents_foreach_after_tl }
      }
  }
```

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

12.15 The command \typestored

\typestored
__scontents_typestored:nn
__scontents_typestored:N
__scontents_xverb:w

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

```
\NewDocumentCommand \typestored { o m }
  {
    \tl_set:Nn \l__scontents_cmd_name_tl { typestored }
    \__scontents_typestored:nn {#1} {#2}
  }
\cs_new_protected:Npn \__scontents_typestored:nn #1 #2
  {
    \__scontents_bsphack:
    \group_begin:
      \seq_clear:N \l__scontents_seq_item_seq
      \str_set:Ne \l__scontents_current_seq_name_str {#2}
      \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / typemeaning } {#1} }
      \seq_if_empty:NT \l__scontents_seq_item_seq
        { \seq_set_from_clist:Nn \l__scontents_seq_item_seq { 1 } }
      \tl_set:Ne \l__scontents_typestored_arg_tl
        { \__scontents_getfrom_seq:Nn \l__scontents_seq_item_seq {#2} }
      \__scontents_remove_trailing_eol:N \l__scontents_typestored_arg_tl
      \tl_replace_all:Nen \l__scontents_typestored_arg_tl { \c__scontents_hidden_space_str } { ^^J }
      \tl_log:N \l__scontents_typestored_arg_tl
      \tl_if_empty:NF \l__scontents_typestored_arg_tl
        {
          \bool_if:NT \l__scontents_print_verb_style_bool
            {
              \__scontents_typestored:N \l__scontents_typestored_arg_tl
            }
        }
      \__scontents_file_write_cmd:VV \l__scontents_file_name_tl \l__scontents_typestored_arg_tl
      \use:e
        {
    \group_end:
    \bool_if:NF \l__scontents_print_verb_style_bool { \__scontents_esphack: }
        }
  }
```

The `\__scontents_typestored: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:e`.

```
\group_begin:
  \char_set_catcode_active:N \^^M
  \cs_new_protected:Npn \__scontents_typestored:N #1
    {
      \tl_if_blank:VT #1
        { \msg_error:nnn { scontents } { empty-variable } {#1} }
      \cs_set_eq:NN \__scontents_verb_print_EOL: ^^M
      \cs_set_eq:NN ^^M \scan_stop:
      \cs_set_eq:cN { do@noligs } \__scontents_do_noligs:N
      \int_set:Nn \tex_newlinechar:D { `\^^J }
      \__scontents_rescan_tokens:e
        {
          \__scontents_format_case:nnn
            { \exp_not:N \begin{verbatimsc} } % LaTeX
            { \verbatimsc } % Plain/Generic
            { \startverbatimsc } % ConTeXt
            ^^M
          \exp_not:V #1 ^^M
          \g__scontents_end_verbatimsc_tl
        }
      \cs_set_eq:NN ^^M \__scontents_verb_print_EOL:
    }
\group_end:
\cs_new_protected:Nn \__scontents_xverb:
  {
    \char_set_catcode_active:n { 9 }
    \char_set_active_eq:nN { 9 } \__scontents_tabs_to_spaces:
    \__scontents_xverb:w
  }
```

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

12.16 The command \meaningsc

`\meaningsc`
`\__scontents_meaningsc:nn`
`\__scontents_meaningsc:n`

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

```
\NewDocumentCommand \meaningsc { o m }
  {
    \tl_set:Nn \l__scontents_cmd_name_tl { meaningsc }
    \__scontents_meaningsc:nn {#1} {#2}
  }
\cs_new_protected:Npn \__scontents_meaningsc:nn #1 #2
  {
    \__scontents_bsphack:
    \group_begin:
      \seq_clear:N \l__scontents_seq_item_seq
      \str_set:Ne \l__scontents_current_seq_name_str {#2}
      \tl_if_novalue:nF {#1} { \keys_set:nn { scontents / typemeaning } {#1} }
      \seq_if_empty:NT \l__scontents_seq_item_seq
        { \seq_set_from_clist:Nn \l__scontents_seq_item_seq { 1 } }
      \__scontents_meaningsc:n {#2}
      \use:e
        {
    \group_end:
    \bool_if:NF \l__scontents_print_verb_style_bool { \__scontents_esphack: }
        }
  }
\group_begin:
  \char_set_catcode_active:N \^^I
  \cs_new_protected:Npn \__scontents_meaningsc:n #1
    {
      \tl_set:Ne \l__scontents_meaningsc_arg_tl
        { \__scontents_getfrom_seq:Nn \l__scontents_seq_item_seq {#1} }
      \tl_replace_all:Nen \l__scontents_meaningsc_arg_tl { \iow_char:N \^^J } { ~ }
      \tl_replace_all:Nen \l__scontents_meaningsc_arg_tl { \c__scontents_hidden_space_str } { ~ }
      \tl_log:N \l__scontents_meaningsc_arg_tl
      \tl_use:N \l__scontents_verb_font_tl
      \tl_replace_all:Nne \l__scontents_meaningsc_arg_tl { ^^I } { \__scontents_tabs_to_spaces: }
      \tl_if_empty:NF \l__scontents_meaningsc_arg_tl
        {
          \bool_if:NT \l__scontents_print_verb_style_bool
            {
              \cs_replacement_spec:N \l__scontents_meaningsc_arg_tl
            }
        }
      \__scontents_file_write_cmd:VV \l__scontents_file_name_tl \l__scontents_meaningsc_arg_tl
    }
\group_end:
```

(*End of definition for* `\meaningsc`*,* `\__scontents_meaningsc:nn`*, and* `\__scontents_meaningsc:n`*. This function is documented on page 6.*)

12.17 The command `\mergesc`

The `\mergesc` command parses a comma separated list given as {⟨*argument*⟩}, and just assembles it as a temporary *internal sequence*, then passes it to the `\typestored` or `\meaningsc` command.

`\mergesc`
`\__scontents_mergesc_code:nn`
`\__scontents_mergesc_parse_list:n`
`\__scontents_remove_trailing_eol:N`
`\__scontents_remove_trailing_eol:w`
`\__scontents_parse_mergesc:nw`
`\__scontents_parse_mergesc_aux:nw`
`\__scontents_parse_mergesc_range:nw`

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.

```
\NewDocumentCommand \mergesc { o m }
  {
    \tl_set:Nn \l__scontents_cmd_name_tl { mergesc }
    \__scontents_mergesc_code:nn {#1} {#2}
  }
\cs_new_protected:Npn \__scontents_mergesc_code:nn #1 #2
  {
    \group_begin:
      \tl_clear:N \l__scontents_mergesc_keys_tl
      \tl_if_novalue:nF {#1}
        {
          \keys_set_known:nnN { scontents / mergesc } {#1} \l__scontents_mergesc_keys_tl
        }
      \seq_gclear:c { g__scontents_name_sc!internal_seq }
      \__scontents_mergesc_parse_list:n {#2}
      \exp_args:Ne \__scontents_mergesc_cmd:nn
        { 1-end, \exp_not:V \l__scontents_mergesc_keys_tl } { sc!internal }
    \group_end:
  }
```

```
\cs_new_protected:Npn \__scontents_mergesc_parse_list:n #1
  {
    \clist_map_inline:nn {#1} { \__scontents_parse_mergesc:nw ##1 \s__scontents_stop }
    \seq_gpop_right:cN { g__scontents_name_sc!internal_seq } \l__scontents_mergesc_arg_tl
    \__scontents_remove_trailing_eol:N \l__scontents_mergesc_arg_tl
    \seq_gput_right:cV { g__scontents_name_sc!internal_seq } \l__scontents_mergesc_arg_tl
  }
\cs_new_protected:Npe \__scontents_remove_trailing_eol:N #1
  {
    \exp_not:N \exp_after:wN \exp_not:N \__scontents_remove_trailing_eol:w
      #1 \s__scontents_stop \c__scontents_hidden_space_str \s__scontents_stop \s__scontents_mark #1
  }
\use:e
  {
    \cs_new_protected:Npn \exp_not:N \__scontents_remove_trailing_eol:w #1
        \c__scontents_hidden_space_str \s__scontents_stop #2 \s__scontents_mark #3
  }   {
        \tl_set:Ne #3
          {
            \tl_if_empty:nTF {#2}
              { \exp_not:o { \__scontents_use_delimit_by_s_stop:nw #1 } }
              { \exp_not:n {#1} }
          }
      }
\cs_new_protected:Npn \__scontents_parse_mergesc:nw #1
  {
    \peek_remove_spaces:n
      {
        \peek_charcode:NTF [ % ]
          { \__scontents_parse_mergesc_aux:nw {#1} }
          { \__scontents_parse_mergesc_aux:nw {#1} [ 1-\seq_count:c { g__scontents_name_#1_seq } ] }
      }
  }
\cs_new_protected:Npn \__scontents_parse_mergesc_aux:nw #1 [#2]
  {
    \seq_clear:N \l__scontents_seq_item_seq
    \clist_map_inline:nn {#2}
      { \__scontents_parse_mergesc_range:nw {#1} ##1 - \q__scontents_mark - \s__scontents_mark }
    \seq_map_inline:Nn \l__scontents_seq_item_seq
      {
        \seq_gput_right:ce { g__scontents_name_sc!internal_seq }
          { \seq_item:cn { g__scontents_name_#1_seq } {##1} }
      }
    \__scontents_use_none_delimit_by_s_stop:w
  }
\cs_new_protected:Npn \__scontents_parse_mergesc_range:nw #1 #2 - #3 - #4 \s__scontents_mark
  {
    \cs_set_protected:Npn \__scontents_tmp:w ##1
      {
        \msg_error:nneee { scontents } { index-out-of-range }
          {##1} {#1} { \seq_count:c { g__scontents_name_#1_seq } }
      }
    \__scontents_range_parser:nnen {#2} {#3}
      { \seq_count:c { g__scontents_name_#1_seq } }
      { \__scontents_tmp:w }
  }
```

(*End of definition for* `\mergesc` *and others. This function is documented on page 7.*)

12.18 The command \setupsc

`\setupsc` User command `\setupsc` to setup module for `\keys_set:nn { scontents }`.

```
\NewDocumentCommand \setupsc { +m }
  {
    \keys_set:nn { scontents } {#1}
  }
```

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

12.19 The command \countsc

\countsc

User command \countsc to count number of ⟨*stored contents*⟩ in the *sequence*.

```
\NewExpandableDocumentCommand \countsc { m }
  {
    \seq_count:c { g__scontents_name_#1_seq }
  }
```

(*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) all ⟨*stored contents*⟩ in the *sequence*.

```
\NewDocumentCommand \cleanseqsc { m }
  {
    \seq_gclear_new:c { g__scontents_name_#1_seq }
  }
```

(*End of definition for* \cleanseqsc. *This function is documented on page 8.*)

12.21 Warning and error messages

Warning and error messages used throughout the package.

```
\msg_new:nnn { scontents } { junk-after-begin }
  {
    Junk~characters~#1~\msg_line_context: :
    \\ \\
    #2
  }
\msg_new:nnnn { scontents } { env-already-defined }
  { Environment~'#1'~already~defined! }
  {
    You~have~used~\newenvsc
    with~an~environment~that~already~has~a~definition. \\ \\
    The~existing~definition~of~'#1'~will~not~be~altered.
  }
\msg_new:nnn { scontents } { empty-stored-content }
  { Empty~value~for~key~'getstored'~\msg_line_context:. }
\msg_new:nnn { scontents } { empty-variable }
  { Variable~'#1'~empty~\msg_line_context:. }
\msg_new:nnn { scontents } { overwrite-file }
  { Overwriting~file~'#1'. }
\msg_new:nnn { scontents } { writing-file }
  { Writing~file~'#1'. }
\msg_new:nnn { scontents } { not-writing }
  { File~`#1'~already~exists.~Not~writing. }
\msg_new:nnn { scontents } { rescanning-text }
  { Rescanning~text~'#1'~after~\c_backslash_str end{#2}~\msg_line_context:.}
\msg_new:nnn { scontents } { multiple-begin }
  { Multiple~\c_backslash_str begin{ \l__scontents_env_name_tl }~\msg_line_context:.}
\msg_new:nnn { scontents } { undefined-storage }
  { Storage~named~'#1'~is~not~defined. }
\msg_new:nnnn { scontents } { mergesc-missing-key }
  {
    Need~mandatory~key~'typestored'~or~'meaningsc'~for~\\
    command~\c_backslash_str mergesc~\msg_line_context:.
  }
  {
    The~command~\c_backslash_str mergesc~need~a~mandatory~key~typestored~or~meaningsc.\\
    Check~that~you~have~spelled~the~key~name~correctly.
  }
\msg_new:nnn { scontents } { index-out-of-range }
  {
    \int_compare:nNnTF {#1} = { 0 }
      { Index~of~sequence~cannot~be~zero. }
      {
        Index~'#1'~out~of~range~for~'#2'.~
        \int_compare:nNnTF {#1} > { 0 }
          { Max = } { Min = -} #3.
      }
  }
\msg_new:nnnn { scontents } { env-key-unknown }
  {
    The~key~'#1'~is~unknown~by~environment~
    '\l__scontents_env_name_tl'~and~is~being~ignored.
  }
  {
    The~environment~'\l__scontents_env_name_tl'~does~not~have~a~key~called~'#1'.\\
    Check~that~you~have~spelled~the~key~name~correctly.
  }
\msg_new:nnnn { scontents } { env-key-value-unknown }
  {
    The~key~'#1=#2'~is~unknown~by~environment~
    '\l__scontents_env_name_tl'~and~is~being~ignored.
  }
  {
    The~environment~'\l__scontents_env_name_tl'~does~not~have~a~key~called~'#1'.\\
    Check~that~you~have~spelled~the~key~name~correctly.
  }
\msg_new:nnnn { scontents } { cmd-key-unknown }
  {
    The~key~'#1'~is~unknown~by~command~\c_backslash_str \l__scontents_cmd_name_tl \c_space_tl
    and~is~being~ignored~ \msg_line_context:.
  }
  {
    The~command~\c_backslash_str \l__scontents_cmd_name_tl \c_space_tl
    does~not~have~a~key~called~'#1'.\\
    Check~that~you~have~spelled~the~key~name~correctly.
  }
\msg_new:nnnn { scontents } { cmd-key-value-unknown }
  {
    The~key~'#1=#2'~is~unknown~by~command~\c_backslash_str \l__scontents_cmd_name_tl \c_space_tl
    and~is~being~ignored~ \msg_line_context:.
  }
  {
    The~command~\c_backslash_str \l__scontents_cmd_name_tl \c_space_tl
    does~not~have~a~key~called~'#1'.\\
    Check~that~you~have~spelled~the~key~name~correctly.
  }
\msg_new:nnn { scontents } { empty-environment }
  { environment~'#1'~empty~\msg_line_context:. }
\msg_new:nnnn { scontents } { verbatim-newline }
  { Verbatim~argument~of~#1~ended~by~end~of~line. }
  {
    The~verbatim~argument~of~the~#1~cannot~contain~more~than~one~line,~
    but~the~end~
    of~the~current~line~has~been~reached.~You~may~have~forgotten~the~
    closing~delimiter.
    \\ \\
    LaTeX~will~ignore~'#2'.
  }
\msg_new:nnnn { scontents } { verbatim-tokenized }
  { The~verbatim~#1~cannot~be~used~inside~an~argument. }
  {
    The~#1~takes~a~verbatim~argument.~
    It~may~not~appear~within~the~argument~of~another~function.~
    It~received~an~illegal~token \tl_if_empty:nF {#3} { ~'#3' } .
    \\ \\
    LaTeX~will~ignore~'#2'.
  }
⟨/core⟩
```

12.22 Finish package

Finish package implementation.

```
⟨plain|context⟩\ExplSyntaxOff
⟨plain|context⟩\endinput
⟨latex|core⟩\file_input_stop:
```

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.