SCONTENTS

Stores LaTeX contents

v1.4 — 2019/10/03*

©2019 by Pablo González †

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

Abstract

The scontents package stores valid LaTeX code in ⟨*sequences*⟩ using the l3seq module of expl3. The ⟨*stored content*⟩, including *verbatim*, can be used as many times as desired in the document, additionally can be written to ⟨*external files*⟩.

Contents

1 Motivation and Acknowledgments

In LaTeX there is no direct way to record content for later use, although you can do this using \macros, recording ⟨*verbatim content*⟩ is a problem, usually you can avoid this by creating external files or boxes. The general idea of this package is to try to imitate this implementation *buffers* that has ConTeXt which allows you to save content in memory, including *verbatim*, to be used later. The package filecontentsdef solves this problem and since expl3 has an excellent way to manage data, ideas were combined giving rise to this package.

This package would not be possible without the great work of Jean François Burnol who was kind enough to take my requirements into account and add the filecontentsdefmacro environment. Also a special thanks to Phelype Oleinik who has collaborated and adapted a large part of the code and all LaTeX3 team for their great work and to the different members of the TeX-SX community who have provided great answers and ideas. Here a note of the main ones:

1. Stack datastructure using LaTeX
2. LaTeX equivalent of ConTeXt buffers
3. Storing an array of strings in a command
4. Collecting contents of environment and store them for later retrieval
5. Collect contents of an environment (that contains verbatim content)

2 License and Requirements

The scontents package loads expl3, xparse and l3keys2e. This package can be used with xelatex, lualatex , pdflatex and the classical workflow latex»dvips»ps2pdf.

*This file describes a documentation for v1.4, last revised 2019/10/03.

†E-mail: «pablgonz@educarchile.cl»

3 The scontents package

3.1 Description of the package and load

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

The package uses a [⟨*key = val*⟩] system for the options and is loaded in the usual way:

```
\usepackage{scontents}
```

or

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

3.2 The TAB character

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

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

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

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

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

3.3 Configuration of the options

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

\setupsc `\setupsc{⟨keyval list⟩}`

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

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

Sets the ⟨*font family*⟩ used to display the ⟨*stored content*⟩ for the commands \typestored and \meaningsc. 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.

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

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

`force-eol = {⟨true | false⟩}` (default: *false*)

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

`width-tab = {⟨integer⟩}` (default: *1*)

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

[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.

3.4 Options Overview

Summary of available options.

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

4 User interface

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

4.1 The environment scontents

scontents

\begin{scontents}[⟨*keyval list*⟩]
⟨*env contents*⟩
\end{scontents}

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

For the correct operation \begin{scontents} and \end{scontents} must be in different lines, all ⟨*keys*⟩ must be passed in a ⟨*single line*⟩ separated by commas "without vertical spaces" and "without separation" of the start of the environment.

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

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

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

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

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

Options for environment

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

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

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

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

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

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

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

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

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

4.2 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. It is not possible to pass environments such as *verbatim*, but it is possible to use the implementation of `\Verb` provided by the fvextra package for contents on one line and `\lstinline` from listings package, but it is preferable to use the starred version.

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

Blank lines are preserved, escaped braces "`\{`" and "`\}`" must also be balanced if the argument is used with braces and `TAB`s characters typed from the keyboard are converted into spaces.

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

Options for command

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

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

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

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

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

4.3 The command `\getstored`

`\getstored`

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

The command `\getstored` gets the content stored in ⟨*seq name*⟩ according to the ⟨*index*⟩ in which it was stored. The command is robust and can be used as an ⟨*argument*⟩ for another command. If the optional argument is not passed it defaults to the first element stored in the ⟨*seq name*⟩.

4.4 The command `\typestored`

`\typestored`

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

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

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

The verbatim package is not compatible with the implementation of the `verbatimsc` environment used by this command, if you are a user of that package consider customizing the `verbatimsc` environment.

4.5 The environment verbatimsc

verbatimsc

Internal environment used by \typestored to display ⟨*verbatim style*⟩ contents.

One consideration to keep in mind is that this is a *representation* of the ⟨*stored content*⟩ in a *verbatim* environment and not a real *verbatim* environment.

The verbatimsc environment can be customized in the following ways:

Using the package fancyvrb:

```
\makeatletter
\let\verbatimsc\@undefined
\let\endverbatimsc\@undefined
\makeatother
\DefineVerbatimEnvironment{verbatimsc}{Verbatim}{numbers=left}
```

Using the package minted:

```
\makeatletter
\let\verbatimsc\@undefined
\let\endverbatimsc\@undefined
\makeatother
\usepackage{minted}
\newminted{tex}{linenos}
\newenvironment{verbatimsc}{\VerbatimEnvironment\begin{texcode}}{\end{texcode}}
```

Using the package listings:

```
\makeatletter
\let\verbatimsc\@undefined
\let\endverbatimsc\@undefined
\makeatother
\usepackage{listings}
\lstnewenvironment{verbatimsc}
  {
   \lstset{
           basicstyle=\small\ttfamily,
           columns=fullflexible,
           language=[LaTeX]TeX,
           numbers=left,
           numberstyle=\tiny\color{gray},
           keywordstyle=\color{red}
          }
  }{}
```

5 Other commands provided

5.1 The command \meaningsc

\meaningsc

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

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

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

5.2 The command \countsc

\countsc

\countsc{⟨*seq name*⟩}

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

5.3 The command \cleanseqsc

\cleanseqsc

\cleanseqsc{⟨*seq name*⟩}

The command \cleanseqsc remove all contents stored in ⟨*seq name*⟩.

6 The scontents package in action

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

Abstract

The scontents package stores valid (La)TeX code in ⟨*sequences*⟩ using the l3seq module of expl3. The ⟨*stored content*⟩, including *verbatim*, can be used as many times as desired in the document, additionally can be written to ⟨*external files*⟩.

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}
The \mypkg*{scontents} package stores valid \hologo{(La)TeX} code in
\mymeta{sequences} using the \mypkg{l3seq} module of \mypkg{expl3}.
The \mymeta{stored content}, including \emph{verbatim}, can be used
as many times as desired in the document, additionally can be written
to \mymeta{external files}.
\end{abstract}
```

and

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

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

```
\begin{scontents}[store-env=abstract,print-env=true]
\begin{abstract}
The \mypkg*{scontents} package stores valid \hologo{(La)TeX} code in
\mymeta{sequences} using the \mypkg{l3seq} module of \mypkg{expl3}.
The \mymeta{stored content}, including \emph{verbatim}, can be used
as many times as desired in the document, additionally can be written
to \mymeta{external files}.
\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} \enquote{\emph{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.

7 Examples

These are some (adapted) examples that have served as inspiration for the creation of this package.

7.1 From answers package

Example 1

Adaptation of example 1 (ansexam1) of the package answers .

```
\documentclass[12pt,a4paper]{article}
\usepackage[store-cmd=solutions]{scontents}
\usepackage{pgffor}
\newtheorem{ex}{Exercise}
\begin{document}
\section{Problems}
\begin{ex}
First exercise
\Scontents{
   First solution.
}
\end{ex}

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

\section{Solutions}
\foreach \i in {1,...,\countsc{solutions}} {
\noindent\textbf{\i} \getstored[\i]{solutions}\par
}
\end{document}
```

7.2 From filecontentsdef package

Example 2

Adaptation of example from package filecontentsdef .

```
\documentclass{article}
\usepackage[store-env=defexercise,store-cmd=defexercise]{scontents}
\usepackage{pgffor}
\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
\foreach \i in {1,...,3} {
\begin{itemize}
\item \getstored[\i]{defexercise}
\end{itemize}}
% \getstored are robust :)
\section{\getstored[2]{defexercise}}
\end{document}
```

7.3 From TeX-SX

Example 3

Adapted from LaTeX equivalent of ConTeXt buffers.

```
\documentclass{article}
\usepackage[store-cmd=tikz]{scontents}
\usepackage{tikz}
\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[3]{tikz}
\end{tikzpicture}

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

Example 4

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

```
\documentclass{article}
\usepackage{scontents}
\usepackage{pgffor}
\pagestyle{empty}
\begin{document}
\begin{scontents}[store-env=a]
Something for a
\end{scontents}

\begin{scontents}[store-env=a]
Something for b
\end{scontents}

\begin{scontents}[store-env=a]
Something with no label
\end{scontents}

\textbf{Let's print them}

This is a: \getstored[1]{a}\par
This is b: \getstored[2]{a}

\textbf{Print all of them}

\foreach \i in {1,...,\countsc{a}} {\getstored[\i]{a}\par}
\end{document}
```

Example 5

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

```
\documentclass{article}
\usepackage{scontents}
```

```
\pagestyle{empty}
\setlength{\parindent}{0pt}
\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}
```

7.4 Customization of verbatimsc

Example 6

Customization of verbatimsc using the fancyvrb and tcolorbox package .

```
\documentclass{article}
\usepackage{scontents}
\makeatletter
\let\verbatimsc\@undefined
\let\endverbatimsc\@undefined
\makeatother
\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\}} whit \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 whit braces } and environment \verb+Verbatim*+
\begin{verbatim}
 verbatim  environment
\end{verbatim}
\end{scontents}

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

\section{Test \texttt{\textbackslash getstored}}
\getstored[1]{contents}\par
\getstored[2]{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 7

Customization of verbatimsc using the listings package .

```
\documentclass{article}
\usepackage{scontents}
\makeatletter
\let\verbatimsc\@undefined
\let\endverbatimsc\@undefined
\makeatother
\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\}} whit \texttt{listings}}
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*} whit \texttt{listings}}

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

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

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

Example 8

Customization of verbatimsc using the minted package .

```
% need ---shell-escape
\documentclass{article}
\usepackage{scontents}
\makeatletter
\let\verbatimsc\@undefined
\let\endverbatimsc\@undefined
\makeatother
\usepackage{minted}
```

```
\newminted{tex}{linenos}
\newenvironment{verbatimsc}{\VerbatimEnvironment\begin{texcode}}{\end{texcode}}
\pagestyle{empty}
\begin{document}
\section{Test \texttt{\textbackslash begin\{scontents\}} whit \texttt{minted}}
Test \verb+{scontents}+ \par

\begin{scontents}[write-env=usingtab.tex,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 whit braces } and environment \verb+Verbatim*+
% Real TABs here :)
\begin{Verbatim}[obeytabs, showtabs, tab=\rightarrowfill, tabcolor=red,showspaces, spacecolor=blue]
No tab
        One real tab
                Two real Tab plus       one tab
\end{Verbatim}
\end{scontents}

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

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

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

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

8 Change history

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

v1.4 (ctan), 2019/10/03
- Add `store-all` key.
- Messages and keys were separated.
- Restructuring of documentation.
- Now the version of `expl3` is checked instead of `xparse`.
- The internal behavior of `force-eol` has been modified.

v1.3 (ctan), 2019/09/24
- The environment can now nest.
- Added `force-eol`, `verb-font` and `width-tab` keys.
- The extra space has been removed when you run `\getstored`.
- Internal code has been rewritten more efficiently.
- Remove `\typestored*`.
- Remove `filecontentsdef` dependency.
- Changing `\regex_replace_all:` for `\tl_replace_all:`.

v1.2 (ctan), 2019/08/28
- Restructuring of documentation.
- Added copy of `\tex_scantokens:D`.

v1.1 (ctan), 2019/08/12
- Extension of documentation.
- Replace `\tex_endinput:D` by `\file_input_stop:`.

v1.0 (ctan), 2019/07/30
- First public release,

9 Index of Documentation

10 Implementation

10.1 Declaration of the package

First we set up the module name for l3doc:

```
⟨@@=scontents⟩
```

Then, we can give the traditional declaration of a package written with expl3 and the necessary packages for its operation.

```
\RequirePackage{l3keys2e}
\RequirePackage{xparse}[2019/05/28]
\ProvidesExplPackage{scontents}{2019/10/03}{1.4}
  {Stores LaTeX contents in memory or files}
```

A check to make sure that expl3 is not too old

```
\@ifpackagelater { expl3 } { 2019/09/19 }
  { }
  {
    \PackageError { scontents } { Support~package~expl3~too~old }
      {
        You~need~to~update~your~installation~of~the~bundles~
        'l3kernel'~and~'l3packages'.\MessageBreak
        Loading~scontents~will~abort!
      }
    \file_input_stop:
  }
```

10.2 Definition of common keys

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

```
\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 .initial:n        = \ttfamily,
    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,
    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
  }
```

We process the ⟨*keys*⟩ as options passed on to the package, the package l3keys2e will verify the ⟨*keys*⟩ and will return an error when they are *unknown*.

```
\ProcessKeysOptions { scontents }
```

10.3 Internal variables

Now we declare the internal variables we will use.

\l__scontents_macro_tmp_tl
\l__scontents_fname_out_tl
\l__scontents_temp_tl
\l__scontents_file_tl
\g__scontents_temp_tl

`\l__scontents_macro_tmp_tl` is a temporary token list to hold the contents of the macro/environment, `\l__scontents_fname_out_tl` is used as the name of the output file, when there's one, `\l__scontents_file_tl` holds the contents of an environment as it's being read, and `\l__scontents_temp_tl` and `\g__scontents_temp_tl` are generic temporary token lists.

```
\tl_new:N \l__scontents_macro_tmp_tl
\tl_new:N \l__scontents_fname_out_tl
\tl_new:N \l__scontents_temp_tl
\tl_new:N \l__scontents_file_tl
\tl_new:N \g__scontents_temp_tl
```

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

\l__scontents_seq_item_int
\l__scontents_env_nesting_int
\l__scontents_tmpa_int

`\l__scontents_seq_item_int` stores the index in the sequence of the item requested to `\typestored` or `\meaningsc`. `\l__scontents_env_nesting_int` stores the current nesting level of the `scontents` environment.

```
\int_new:N \l__scontents_seq_item_int
\int_new:N \l__scontents_env_nesting_int
\int_new:N \l__scontents_tmpa_int
```

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

\l__scontents_writing_bool
\l__scontents_storing_bool

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

```
\bool_new:N \l__scontents_writing_bool
\bool_set_false:N \l__scontents_writing_bool
\bool_new:N \l__scontents_storing_bool
\bool_set_true:N  \l__scontents_storing_bool
```

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

\g__scontents_end_verbatimsc_tl
\c__scontents_end_env_tl

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

```
\tl_new:N \g__scontents_end_verbatimsc_tl
\tl_gset_rescan:Nnn
  \g__scontents_end_verbatimsc_tl
  {
    \char_set_catcode_other:N \\
    \char_set_catcode_other:N \{
    \char_set_catcode_other:N \}
  }
  { \end{verbatimsc} }
\tl_const:Nx \c__scontents_end_env_tl
  { \c_backslash_str end \c_left_brace_str scontents \c_right_brace_str }
```

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

\c__scontents_hidden_space_str

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

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

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

\q__scontents_stop
\q__scontents_mark

Some quarks used along the code as macro delimiters.

```
\quark_new:N \q__scontents_stop
\quark_new:N \q__scontents_mark
```

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

\l__scontents_file_iow

An output stream for saving the contents of an environment to a file.

```
\iow_new:N \l__scontents_file_iow
```

(*End definition for* `\l__scontents_file_iow`.)

`\__scontents_rescan_tokens:n` `\tl_rescan:nn` doesn't fit the needs of this package because it does not allow catcode changes inside the argument, so verbatim commands used inside one of scontents's commands/environments will not work. Here we create a private copy of `\tex_scantokens:D` which will serve our purposes.

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

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

`\__scontents_tab:` `\__scontents_par:` Control sequences to replace tab (`^^I`) and form feed (`^^L`) characters.

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

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

`\tl_remove_once:NV` `\tl_replace_all:Nxx` `\tl_replace_all:Nxn` `\tl_replace_all:Nnx` `\tl_if_empty:fTF` Some nonstandard variants.

```
\cs_generate_variant:Nn \tl_remove_once:Nn { NV }
\cs_generate_variant:Nn \tl_replace_all:Nnn { Nx, Nxx, Nnx }
\prg_generate_conditional_variant:Nnn \tl_if_empty:n { f } { TF }
```

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

10.4 Defining keys for the environment and commands

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

10.4.1 Keys for environment scontents

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

```
\keys_define:nn { scontents / scontents }
  {
    write-env .code:n           = {
                                    \bool_set_true:N \l__scontents_writing_bool
                                    \tl_set:Nn \l__scontents_fname_out_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_fname_out_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,
    unknown   .code:n           = { \__scontents_parse_environment_keys:n {#1} }
  }
```

10.4.2 Keys for command \Scontents

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

```
\keys_define:nn { scontents / Scontents }
  {
    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,
    unknown   .code:n    = { \__scontents_parse_command_keys:n {#1} }
  }
```

10.4.3 Key for commands \typestored and \meaningsc

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

```
\keys_define:nn { scontents / typemeaning }
  {
    width-tab .meta:nn = { scontents } { width-tab = #1 },
    unknown   .code:n  = { \__scontents_parse_type_meaning_key:n {#1} }
  }
```

10.5 Handling undefined keys

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

10.5.1 Undefined keys for environment scontents

__scontents_parse_environment_keys:n
__scontents_parse_environment_keys: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_parse_environment_keys:n #1
  { \exp_args:NV \__scontents_parse_environment_keys:nn \l_keys_key_tl {#1} }
\cs_new_protected:Npn \__scontents_parse_environment_keys: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 definition for* __scontents_parse_environment_keys:n *and* __scontents_parse_environment_keys:nn.)

10.5.2 Undefined keys for \Scontents and \Scontents*

__scontents_parse_command_keys:n
__scontents_parse_command_keys:nn

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

```
\cs_new_protected:Npn \__scontents_parse_command_keys:n #1
  { \exp_args:NV \__scontents_parse_command_keys:nn \l_keys_key_tl {#1} }
\cs_new_protected:Npn \__scontents_parse_command_keys: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 definition for* __scontents_parse_command_keys:n *and* __scontents_parse_command_keys:nn.)

10.5.3 Undefined keys for \typestored and \meaningsc

__scontents_parse_type_meaning_key:n
__scontents_parse_type_meaning_key:nn

The commands \typestored and \meaningsc accept an optional argument for setting the width-tab to print the stored contents. However their optional argument also contains the number of the item to retrieve from the stored sequence. To avoid the awkward \typestored[][⟨*options*⟩]{...} syntax, we'll make the commands have a single optional argument which is processed by l3keys, and the unknown keys are brought here to __scontents_parse_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_tl {#1} }
\cs_new_protected:Npn \__scontents_parse_type_meaning_key:nn #1#2
  {
    \tl_if_empty:fTF { \int_to_roman:n { -0 #1 } }
      {
        \tl_if_blank:nTF {#2}
          { \int_set:Nn \l__scontents_seq_item_int {#1} }
          { \msg_error:nnnn { scontents } { type-key-value-unknown } {#1} {#2} }
      }
      {
        \tl_if_blank:nTF {#2}
```

```
143           { \msg_error:nnn { scontents } { type-key-unknown } {#1} }
144           { \msg_error:nnnn { scontents } { type-key-value-unknown } {#1} {#2} }
145       }
146   }
```

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

10.6 Programming of the sequences

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

`\__scontents_append_contents:nn`

`\__scontents_append_contents:nn` creates a seq variable if one didn't exist and appends the contents in the argument to the right of the sequence.

```
147 \cs_new_protected:Npn \__scontents_append_contents:nn #1#2
148   {
149     \tl_if_blank:nT {#1}
150       { \msg_error:nn { scontents } { empty-store-cmd } }
151     \seq_if_exist:cF { g__scontents_name_#1_seq }
152       { \seq_new:c { g__scontents_name_#1_seq } }
153     \seq_gput_right:cn { g__scontents_name_#1_seq } {#2}
154   }
155 \cs_generate_variant:Nn \__scontents_append_contents:nn { Vx }
```

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

`\__scontents_getfrom_seq:nn`
`\__scontents_getfrom_seq:nnn`

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

```
156 \cs_new:Npn \__scontents_getfrom_seq:nn #1#2
157   {
158     \seq_if_exist:cTF { g__scontents_name_#2_seq }
159       {
160         \exp_args:Nf \__scontents_getfrom_seq:nnn
161           { \seq_count:c { g__scontents_name_#2_seq } }
162           {#1} {#2}
163       }
164       { \msg_expandable_error:nnn { scontents } { undefined-storage } {#2} }
165   }
166 \cs_new:Npn \__scontents_getfrom_seq:nnn #1#2#3
167   {
168     \bool_lazy_or:nnTF
169       { \int_compare_p:nNn {#2} = { 0 } }
170       { \int_compare_p:nNn { \int_abs:n {#2} } > {#1} }
171       { \msg_expandable_error:nnnnn { scontents } { index-out-of-range } {#2} {#3} {#1} }
172       { \seq_item:cn { g__scontents_name_#3_seq } {#2} }
173   }
```

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

`\__scontents_lastfrom_seq:n`

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

```
174 \cs_new_protected:Npn \__scontents_lastfrom_seq:n #1
175   {
176     \tl_gset:Nx \g__scontents_temp_tl { \seq_item:cn { g__scontents_name_#1_seq } {-1} }
177     \group_insert_after:N \__scontents_rescan_tokens:V
178     \group_insert_after:N \g__scontents_temp_tl
179     \group_insert_after:N \tl_gclear:N
180     \group_insert_after:N \g__scontents_temp_tl
181   }
```

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

`\__scontents_store_to_seq:NN`

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

```
182 \cs_new_protected:Npn \__scontents_store_to_seq:NN #1#2
183   {
184     \tl_log:N #1
185     \__scontents_append_contents:Vx #2 { \exp_not:V #1 }
186   }
```

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

10.7 Construction of environment scontents

We define the environment scontents, next to the system [⟨*key* = *val*⟩]. The environment is divided into three parts. This implementation is taken from answer by Enrico Gregorio in https://tex.stackexchange.com/a/487746/7832.

scontents This is the main environment used in the document.

```
\NewDocumentEnvironment { scontents } { }
  {
    \char_set_catcode_active:N \^^M
    \__scontents_start_environment:w
  }
  {
    \__scontents_stop_environment:
    \__scontents_atend_environment:
  }
```

(*End definition for* scontents. *This function is documented on page* 3.)

10.7.1 key val for environment

Define a [⟨*key* = *val*⟩] for environment scontents

__scontents_environment_inline:w
__scontents_environment_keys:w
__scontents_environment_junk:nw
__scontents_environment_junk:xw

The macro __scontents_environment_inline:w is called from the scontents environment with the tokens following the \begin{scontents}. If the immediate next token (ignoring spaces) is a [, then we look for an optional argument delimited by a]. All the remaining tokens are treated as junk and an error is raised if they are non-blank.

```
\cs_new_protected:Npn \__scontents_environment_inline:w
  {
    \peek_charcode_ignore_spaces:NTF [ % ]
      { \__scontents_environment_keys:w }
      {
        \__scontents_environment_junk:xw
          { after~\c_backslash_str begin{scontents} }
      }
  }
\cs_new_protected:Npn \__scontents_environment_keys:w [ #1 ]
  {
    \keys_set:nn { scontents / scontents } {#1}
    \__scontents_environment_junk:xw
      { after~optional~argument~to~\c_backslash_str begin{scontents} }
  }
\cs_new_protected:Npn \__scontents_environment_junk:nw #1 #2 \q__scontents_mark
  {
    \tl_if_blank:nF {#2}
      { \msg_error:nnnn { scontents } { junk-after-begin } {#1} {#2} }
  }
\cs_generate_variant:Nn \__scontents_environment_junk:nw { x }
```

(*End definition for* __scontents_environment_inline:w, __scontents_environment_keys:w, *and* __scontents_environment_junk:nw.)

10.7.2 The environment itself

__scontents_start_environment:w
__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 line directly after \begin{scontents} contains an optional argument enclosed in [...] or other tokens. The trailing tokens are treated as junk and an error is raised. The __scontents_environment_inline:w macro checks for those cases.

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

When that's done, __scontents_file_write_stop:N does the cleanup and the read token list is smuggled out of the verbatim group. This part of the code is inspired and adapted from the code of the package xsimverb by Clemens Niederberger.

```
\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_start_environment:w #1 ^^M
  {
    \__scontents_environment_inline:w #1 \q__scontents_mark
    \__scontents_make_control_chars_active:
    \group_begin:
      \__scontents_file_tl_write_start:V \l__scontents_fname_out_tl
  }
\cs_new_protected:Npn \__scontents_stop_environment:
  {
      \__scontents_file_write_stop:N \l__scontents_macro_tmp_tl
      \exp_args:NNNV
    \group_end:
    \tl_set:Nn \l__scontents_macro_tmp_tl \l__scontents_macro_tmp_tl
  }
```

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

`\__scontents_file_tl_write_start:n`
`\__scontents_file_tl_write_start:V`
`\__scontents_verb_processor_iterate:w`
`\__scontents_file_write_stop:N`
`\__scontents_remove_leading_nl:n`
`\__scontents_remove_leading_nl:w`

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

```
\cs_new_protected:Npn \__scontents_file_tl_write_start:n #1
  {
    \group_begin:
      \bool_if:NT \l__scontents_writing_bool
        {
          \file_if_exist:nTF {#1}
            { \msg_warning:nnx { scontents } { rewriting-file } {#1} }
            { \msg_warning:nnx { scontents } { writing-file } {#1} }
          \iow_open:Nn \l__scontents_file_iow {#1}
        }
      \tl_clear:N \l__scontents_file_tl
      \seq_map_function:NN \l_char_special_seq \char_set_catcode_other:N
      \int_step_function:nnN { 128 } { 255 } \char_set_catcode_letter:n
      \cs_set_protected:Npx \__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
  }
\use:x
  {
    \cs_new: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
  } {
      \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:x
                { \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:Npx \__scontents_ret:w
                {
                  \exp_not:N \end{scontents}
                  \bool_lazy_or:nnF
                    { \tl_if_blank_p:n {#2} }
                    { \str_if_eq_p:ee {#2} { \c_percent_str } }
                    {
                      \msg_warning:nnn { scontents } { rescanning-text } {#2}
                      \__scontents_rescan_tokens:n {#2}
                    }
                }
              \char_set_active_eq:NN ^^M \__scontents_ret:w
            }
        }
      ^^M
    }
  \cs_new_protected:Npn \__scontents_file_write_stop:N #1
    {
      \bool_if:NT \l__scontents_writing_bool
        { \iow_close:N \l__scontents_file_iow }
      \use:x
        {
          \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_file_tl }
            }
        }
    }
  \cs_new:Npn \__scontents_remove_leading_nl:n #1
    { \exp_not:o { \__scontents_remove_leading_nl:w #1 } }
  \cs_new:Npn \__scontents_remove_leading_nl:w ^^J { }
```

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

`\__scontents_verb_processor_output:n`
`\__scontents_verb_processor_output:x`
`\__scontents_analyse_nesting:n`
`\__scontents_analyse_nesting:w`
`\__scontents_nesting_decr:`
`\__scontents_use_none_delimit_by_q_stop:w`
`\__scontents_if_nested:`*TF*

`\__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`.

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

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

```
  \cs_new_protected:Npn \__scontents_verb_processor_output:n #1
    {
      \bool_if:NT \l__scontents_writing_bool
        { \iow_now:Nn \l__scontents_file_iow {#1} }
      \bool_if:NT \l__scontents_storing_bool
        { \tl_put_right:Nn \l__scontents_file_tl { ^^J #1 } }
    }
  \cs_generate_variant:Nn \__scontents_verb_processor_output:n { x }
  \cs_new_protected:Npx \__scontents_analyse_nesting:n #1
    {
      \int_zero:N \l__scontents_tmpa_int
      \exp_not:N \__scontents_analyse_nesting: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
      \int_compare:nNnT { \l__scontents_tmpa_int } > { 1 }
        { \msg_warning:nn { scontents } { multiple-begin } }
    }
  \use:x
    {
      \cs_new_protected:Npn \exp_not:N \__scontents_analyse_nesting:w ##1
        \c_backslash_str begin \c_left_brace_str ##2 \c_right_brace_str
```

```
    }   {
          \if_meaning:w \q__scontents_mark #2
            \exp_after:wN \use_i:nn
          \else:
            \exp_after:wN \use_ii:nn
          \fi:
            { \__scontents_use_none_delimit_by_q_stop:w }
            {
              \str_if_eq:eeT {#2} {scontents}
                {
                  \int_incr:N \l__scontents_env_nesting_int
                  \int_incr:N \l__scontents_tmpa_int
                  \__scontents_analyse_nesting:w
                }
              \__scontents_analyse_nesting:w
            }
        }
  \cs_new_protected:Npn \__scontents_nesting_decr:
    { \int_decr:N \l__scontents_env_nesting_int }
  \prg_new_protected_conditional:Npnn \__scontents_if_nested: { TF }
    {
      \int_compare:nNnTF { \l__scontents_env_nesting_int } > { \c_zero_int }
        { \prg_return_true: }
        { \prg_return_false: }
    }
  \cs_new:Npn \__scontents_use_none_delimit_by_q_stop:w #1 \q__scontents_stop { }
\group_end:
\cs_generate_variant:Nn \__scontents_file_tl_write_start:n { V }
```

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

10.7.3 Recording of the content in the sequence

`\__scontents_atend_environment:`

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

```
\cs_new_protected:Npn \__scontents_atend_environment:
  {
    \bool_if:NT \l__scontents_storing_bool
      {
        \bool_if:NF \l__scontents_forced_eol_bool
          { \tl_put_right:Nx \l__scontents_macro_tmp_tl { \c__scontents_hidden_space_str } }
        \__scontents_store_to_seq:NN \l__scontents_macro_tmp_tl \l__scontents_name_seq_env_tl
        \bool_if:NT \l__scontents_print_env_bool
          { \__scontents_lastfrom_seq:n \l__scontents_name_seq_env_tl }
      }
  }
```

(*End definition for* `\__scontents_atend_environment:`.)

10.8 The `\Scontents` command

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

`\Scontents`
`\__scontents_norm_arg:n`
`\__scontents_verb_arg:w`

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

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

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

```
\NewDocumentCommand { \Scontents }{ !s !O{} }
  {
    \group_begin:
      \IfNoValueF {#2}
        { \keys_set:nn { scontents / Scontents } {#2} }
      \char_set_catcode_active:n { 9 }
      \IfBooleanTF {#1}
        { \__scontents_verb_arg:w }
        { \__scontents_norm_arg:n }
  }
```

```
\cs_new_protected:Npn \__scontents_norm_arg:n #1
  {
      \tl_set:Nx \l__scontents_temp_tl { \exp_not:n {#1} }
      \tl_put_right:Nx \l__scontents_temp_tl { \c__scontents_hidden_space_str }
      \__scontents_store_to_seq:NN \l__scontents_temp_tl \l__scontents_name_seq_cmd_tl
      \bool_if:NT \l__scontents_print_cmd_bool
        { \__scontents_lastfrom_seq:n \l__scontents_name_seq_cmd_tl }
    \group_end:
  }
\NewDocumentCommand { \__scontents_verb_arg:w } { +v }
  {
      \tl_set:Nx \l__scontents_temp_tl { \exp_not:n {#1} }
      \tl_replace_all:Nxx \l__scontents_temp_tl { \iow_char:N \^^M } { \iow_char:N \^^J }
      \bool_if:NF \l__scontents_forced_eol_bool
        { \tl_put_right:Nx \l__scontents_temp_tl { \c__scontents_hidden_space_str } }
      \__scontents_store_to_seq:NN \l__scontents_temp_tl \l__scontents_name_seq_cmd_tl
      \bool_if:NT \l__scontents_print_cmd_bool
        { \__scontents_lastfrom_seq:n \l__scontents_name_seq_cmd_tl }
    \group_end:
  }
```

(*End definition for* `\Scontents`, `\__scontents_norm_arg:n`, *and* `\__scontents_verb_arg:w`. *This function is documented on page* 4.)

10.9 The command `\getstored`

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

```
\NewDocumentCommand { \getstored } { O{1} m }
  {
    \group_begin:
      \__scontents_rescan_tokens:x
        { \__scontents_getfrom_seq:nn {#1} {#2} }
    \group_end:
  }
```

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

10.10 The `\typestored` command

This implementation is an adaptation taken from answer by Phelype Oleinik in (https://tex.stackexchange.com/a/497651/7832).

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

`\__scontents_verb_print:N`

`\__scontents_xverb:w`

`verbatimsc`

```
\NewDocumentCommand { \typestored } { o m }
  {
    \group_begin:
      \int_set:Nn \l__scontents_seq_item_int { 1 }
      \IfValueT {#1} { \keys_set:nn { scontents / typemeaning } {#1} }
      \tl_set:Nx \l__scontents_temp_tl
        {
          \exp_args:NV \__scontents_getfrom_seq:nn \l__scontents_seq_item_int {#2}
        }
      \tl_remove_once:NV \l__scontents_temp_tl \c__scontents_hidden_space_str
      \tl_log:N \l__scontents_temp_tl
      \tl_if_empty:NF \l__scontents_temp_tl
        { \__scontents_verb_print:N \l__scontents_temp_tl }
    \group_end:
  }
```

The `\__scontents_verb_print:N` macro is defined with active carriage return (ASCII 13) characters to mimick an actual verbatim environment "on the loose". The contents of the environment are placed in a `verbatimsc` environment and rescanned using `\__scontents_rescan_tokens:x`.

```
\group_begin:
  \char_set_catcode_active:N \^^M
  \cs_new_protected:Npn \__scontents_verb_print: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
      \__scontents_rescan_tokens:x
        {
          \exp_not:N \begin{verbatimsc} ^^M
          \exp_not:V #1 ^^M
          \g__scontents_end_verbatimsc_tl
        }
      \cs_set_eq:NN ^^M \__scontents_verb_print_EOL:
    }
\group_end:
```

Finally, the `verbatimsc` environment is defined.

```
\cs_new_protected:Npn \__scontents_xverb:
  {
    \char_set_catcode_active:n { 9 }
    \char_set_active_eq:nN { 9 } \__scontents_tabs_to_spaces:
    \__scontents_xverb:w
  }
\use:x
  {
    \cs_new_protected:Npn \exp_not:N \__scontents_xverb:w
      ##1 \g__scontents_end_verbatimsc_tl
  }
      { #1 \end{verbatimsc} }
\NewDocumentEnvironment { verbatimsc } { }
  {
    \cs_set_eq:cN { @xverbatim } \__scontents_xverb:
    \verbatim
  }
  { }
```

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

10.11 Some auxiliaries

`\__scontents_tabs_to_spaces:`

In a verbatim context the TAB character is made active and set equal to `\__scontents_tabs_to_spaces:`, to produce as many spaces as the `width-tab` key was set to.

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

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

`\__scontents_do_noligs:N`

`\__scontents_do_noligs:N` is an alternative definition for LaTeX 2_ε's `\do@noligs` which makes sure to not consume following space tokens. The LaTeX 2_ε version ends with `\char`#1`, which leaves TeX still looking for an ⟨*optional space*⟩. This version uses `\char_generate:nn` to ensure that doesn't happen.

```
\cs_new:Npn \__scontents_do_noligs:N #1
  {
    \char_set_catcode_active:N #1
    \char_set_active_eq:Nc #1 { __scontents_active_char_ \token_to_str:N #1 : }
    \cs_set:cpx { __scontents_active_char_ \token_to_str:N #1 : }
      {
        \mode_leave_vertical:
        \tex_kern:D \c_zero_dim
        \char_generate:nn { `#1 } { 12 }
      }
  }
```

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

`\__scontents_set_active_eq:NN`
`\__scontents_make_control_chars_active:`

Shortcut definitions for common catcode changes.

```
\cs_new_protected:Npn \__scontents_set_active_eq:NN #1
  {
    \char_set_catcode_active:N #1
```

```
    \char_set_active_eq:NN #1
  }
\cs_new_protected:Npn \__scontents_make_control_chars_active:
  {
    \__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
  }
```

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

10.12 The command \setupsc

User command \setupsc to setup module.

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

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

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

10.13 The command meaningsc

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

```
\NewDocumentCommand { \meaningsc } { o m }
  {
    \group_begin:
      \int_set:Nn \l__scontents_seq_item_int { 1 }
      \IfValueT {#1} { \keys_set:nn {  scontents / typemeaning } {#1} }
      \__scontents_meaningsc:n {#2}
    \group_end:
  }
\group_begin:
  \char_set_catcode_active:N \^^I
  \cs_new_protected:Npn \__scontents_meaningsc:n #1
    {
      \tl_set:Nx \l__scontents_temp_tl
        {
          \exp_args:NV \__scontents_getfrom_seq:nn \l__scontents_seq_item_int {#1}
        }
      \tl_replace_all:Nxn \l__scontents_temp_tl { \iow_char:N \^^J } { ~ }
      \tl_remove_once:NV  \l__scontents_temp_tl \c__scontents_hidden_space_str
      \tl_log:N \l__scontents_temp_tl
      \tl_use:N \l__scontents_verb_font_tl
      \tl_replace_all:Nnx \l__scontents_temp_tl { ^^I } { \__scontents_tabs_to_spaces: }
      \cs_replacement_spec:N \l__scontents_temp_tl
    }
\group_end:
```

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

10.14 The command \countsc

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

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

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

10.15 The command \cleanseqsc

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

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

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

10.16 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: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 } { rewriting-file }
  { Overwriting ~ file ~ '#1' }
\msg_new:nnn { scontents } { writing-file }
  { Writing ~ file ~ '#1' }
\msg_new:nnn { scontents } { rescanning-text }
  { Rescanning~text~'#1'~after~\c_backslash_str end{scontents}~\msg_line_context:.}
\msg_new:nnn { scontents } { multiple-begin }
  { Multiple~\c_backslash_str begin{scontents}~\msg_line_context:.}
\msg_new:nnn { scontents } { undefined-storage }
  { Storage~named~'#1'~is~not~defined. }
\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~'scontents'~and~is~being~ignored.}
  {
    The~environment~'scontents'~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~'scontents'~and~is~being~ignored. }
  {
    The~environment~'scontents'~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~'\c_backslash_str Scontents'~and~is~being~ignored.}
  {
    The~command~'\c_backslash_str Scontents'~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~'\c_backslash_str Scontents'~and~is~being~ignored. }
  {
    The~command~'\c_backslash_str Scontents'~does~not~have~a~key~called~'#1'.\\
    Check~that~you~have~spelled~the~key~name~correctly.
  }
\msg_new:nnnn { scontents } { type-key-unknown }
  { The~key~'#1'~is~unknown~and~is~being~ignored. }
  {
    This~command~does~not~have~a~key~called~'#1'.\\
    This~command~only~accepts~the~key~'width-tab'.
  }
\msg_new:nnnn { scontents } { type-key-value-unknown }
  { The~key~'#1'~to~which~you~passed~'#2'~is~unknown~and~is~being~ignored. }
  {
    This~command~does~not~have~a~key~called~'#1'.\\
    This~command~only~accepts~the~key~'width-tab'.
  }
```

10.17 Finish package

Finish package implementation.

```
\file_input_stop:
```

11 Index of Implementation

T

U

V