SCONTENTS

Stores LaTeX contents

v1.2 — 2019/08/28[*]

©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 memory (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 if desired.

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, I decided to combine the best of both.

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 xparse, filecontentsdef and l3keys2e. This package can be used with `xelatex`, `lualatex`, `pdflatex` and the classical workflow `latex-dvips-ps2pdf`.

[*] This file describes a documentation for v1.2, last revised 2019/08/28.

[†] E-mail: «pablgonz@educarchile.cl»

3 The scontents package

3.1 Description

The scontents package encapsulates the filecontentsdef package of Jean François Burnol which allows you to save the content in a \macro and save it in external files, adding a user interface style [⟨*key* = *val*⟩] along with the ability to save content in sequences for later use in different parts of the document.

3.2 Loading package

The package is loaded in the usual way:

```
\usepackage{scontents}
```

or

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

3.3 Configuration of the options

Most of the options can be passed directly to the package or can be configured by means of the command \setupsc.

\setupsc

```
\setupsc {⟨key=val⟩}
```

The command \setupsc configures the options in a global way, for \Scontents, \Scontents* and environment scontents. it can be used both in the preamble and in the body of the document as many times as desired.

3.4 Options Overview

Summary table of available options.

key	package	\setupsc	scontents	\Scontents	\Scontents*
store-env	✓	✓	✓	✗	✗
store-cmd	✓	✓	✗	✓	✓
print-env	✓	✓	✓	✗	✗
print-cmd	✓	✓	✗	✓	✓
print-all	✓	✓	✓	✗	✗
write-env	✗	✗	✓	✗	✗
write-out	✗	✗	✓	✗	✗

4 User interface

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

4.1 The environment scontents

scontents

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

The scontents environment encapsulates the filecontentsdef* and filecontentsdefmacro environments provided by the filecontentsdef package. This allows you to record content including verbatim for later reuse.

Some considerations to keep in mind:

1. The environment cannot be nested.
2. Both \begin and \end must be on different lines.
3. The [⟨*key=val*⟩] options must be passed on one line right after starting the environment.
4. The content of the environment is treated in the same way as filecontents* environment.
5. If you don't want the extra space added by TEX, you should use \relax or % at the end of environment.

For more technical information about the environment it is better to read the documentation of the filecontentsdef package.

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.

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

The name of the sequence in which the content recorded by the environment was stored.

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

It will show the current content of the environment.

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

In addition to storing the content of the environment will write this in an external file.

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

It will write the contents of the environment in an external file, but, it will not store the contents of this one. It is analogous to the `filecontents*` environment.

4.2 The command `\Scontents`

The command to store content directly in memory, the star version allows to save ⟨*verbatim*⟩ contents.

`\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.

It can be used anywhere in the document and cannot be used as an ⟨*argument*⟩ for another command.

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-like the argument of `\verb`. Some considerations to keep in mind:

1. Blank lines are preserved.
2. The command cannot be used as an argument for another command.
3. If you don't want the extra space added by TEX, you should use `\relax` or `%` at the end.

Options for command

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

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

The name of the sequence in which the content recorded by `\Scontents` was stored.

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

It will show the current content of `\Scontents`.

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 last element saved in the ⟨*seq name*⟩.

4.4 The command `\typestored`

`\typestored`

```
\typestored [⟨index⟩]{⟨seq name⟩}
\typestored*[⟨index⟩]{⟨seq name⟩}
```

The command `\typestored` shows the content stored in ⟨*seq name*⟩ in *verbatim* mode. Internally places the content into the `verbatimsc` environment.

The command \typestored* must be used for content stored by \Scontents* command. Internally places the content into the verbatimsc environment.

If the optional argument is not passed it defaults to the last element saved in the ⟨*seq name*⟩.

4.5 The environment verbatimsc

verbatimsc

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

One consideration to keep in mind is that this is a *representation* of the content in a ⟨*verbatim*⟩ environment and not a real ⟨*verbatim*⟩ environment, the line ends are not respected. The verbatimsc environment can be 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*⟩]{⟨*seq name*⟩}

The command \meaningsc executes \meaning on the content stored in ⟨*seq name*⟩. If the optional argument is not passed it defaults to the last element saved in the ⟨*seq name*⟩.

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 \cleansc

\cleanseqsc

\cleansc {⟨*seq name*⟩}

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

6 Examples

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

6.1 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}
```

6.2 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}
\Scontents{
Prove that \[x^n+y^n=z^n\] is not solvable in positive integers if $n$ is at
most $-3$.\par
}
\Scontents*{Refute the existence of black holes in less than $140$ characters.\relax}
\begin{scontents}[write-env=\jobname-3.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}

\foreach \i in {1,...,3} {
\begin{itemize}
\item \getstored[\i]{defexercise}
\end{itemize}}

\section{\getstored[2]{defexercise}} % \getstored are robust :)
\end{document}
```

6.3 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}
\foreach \i in {1,...,\countsc{tikz}}{
\typestored*[\i]{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}

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+. This is normal text.
\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}
```

6.4 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+\begin{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+\begin{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 🗎.

```
\documentclass{article} % need shell-escape
\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+\begin{scontents}+ \par

\begin{scontents}
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*+
\begin{verbatim}
     verbatim environment
\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[2]{contents}\par
\getstored[1]{contents}

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

7 Change history

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,

8 Index of Documentation

9 Implementation

9.1 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{filecontentsdef}[2019/04/20]
\RequirePackage{l3keys2e}
\RequirePackage{xparse}[2019/05/28]
\ProvidesExplPackage{scontents}{2019/08/28}{1.2}
  {Stores LaTeX contents in memory or files}
```

A check to make sure that xparse is not too old

```
\@ifpackagelater { xparse } { 2019/05/03 }
  { }
  {
    \PackageError { scontents } { Support~package~xparse~too~old }
      {
        You~need~to~update~your~installation~of~the~bundles~
        'l3kernel'~and~'l3packages'.\MessageBreak
        Loading~scontents~will~abort!
      }
    \file_input_stop:
  }
```

9.2 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,
    print-env .bool_set:N       = \l__scontents_print_env_bool,
    print-env .initial:n        = false,
    store-cmd .tl_set:N         = \l__scontents_name_seq_cmd_tl,
    store-cmd .initial:n        = contents,
    print-cmd .bool_set:N       = \l__scontents_print_cmd_bool,
    print-cmd .initial:n        = false,
    print-all .meta:n           = { print-env = true , print-cmd = true },
    store-env .value_required:n = true,
    store-cmd .value_required:n = true,
    print-env .value_required:n = true,
    print-cmd .value_required:n = true,
    print-all .value_required:n = true
  }
```

We process the keys as options passed on to the package.

```
\ProcessKeysOptions { scontents }
```

9.3 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_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, and \l__scontents_temp_tl is a generic temporary token list.

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

(*End definition for* \l__scontents_macro_tmp_tl *,* \l__scontents_fname_out_tl *, and* \l__scontents_temp_tl*.*)

\l__scontents_typeverb_env_bool
\l__scontents_writing_bool
\l__scontents_storing_bool

The boolean \l__scontents_typeverb_env_bool keeps track whether the starred variant of the \typestored macro was used, \l__scontents_writing_bool if 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_typeverb_env_bool
\bool_set_true:N  \l__scontents_typevrb_env_bool
\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_typeverb_env_bool *,* \l__scontents_writing_bool *, and* \l__scontents_storing_bool*.*)

\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*.*)

__scontents_rescan_tokens:w

\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_eq:NN \__scontents_rescan_tokens:w \tex_scantokens:D
```

(*End definition for* __scontents_rescan_tokens:w*.*)

\g__scontents_end_verbatimsc_tl

A token list to match when ending verbatim 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} }
```

(*End definition for* \g__scontents_end_verbatimsc_tl*.*)

9.4 Add keys for environment

We define a set of keys for environment scontents.

```
\keys_define:nn { 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
  }
```

9.5 Define keys for command

A sub/keys for command \Scontents and \Scontents*

```
\keys_define:nn { scontents / Scontents }
  {
    print-cmd .meta:nn = { scontents } { print-cmd = #1 },
    store-cmd .meta:nn = { scontents } { store-cmd = #1 }
  }
```

9.6 Programming of the sequences

__scontents_append_contents:nn
__scontents_getfrom_seq:nn

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 creates a seq variable if one didn't exist and appends the contents in the argument to the right of the sequence. __scontents_getfrom_seq:nn retrieves the saved item from the sequence.

```
\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_seq_name_#1_seq }
      { \seq_new:c { g__scontents_seq_name_#1_seq } }
    \seq_gput_right:cn { g__scontents_seq_name_#1_seq } {#2}
  }
\cs_generate_variant:Nn \__scontents_append_contents:nn { Vx }
\cs_new:Npn \__scontents_getfrom_seq:nn #1#2
  { \seq_item:cn { g__scontents_seq_name_#2_seq } {#1} }
```

(*End definition for* __scontents_append_contents:nn *and* __scontents_getfrom_seq:nn*.*)

9.7 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 2.*)

9.7.1 The environment itself

The environment itself

__scontents_start_environment:w
__scontents_stop_environment:

Here we make ^^M an active character so that the end of line can be "seen" to be used as a delimiter. 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.

```
\group_begin:
  \char_set_catcode_active:N \^^M
  \cs_new_protected:Npn \__scontents_start_environment:w #1 ^^M
    {
      \__scontents_environment_inline:w #1 \q__scontents_mark
      \group_begin:
        \bool_if:NTF \l__scontents_writing_bool
          {
            \use:c { filecontentsdef* } { \l__scontents_fname_out_tl }
                                        { \l__scontents_macro_tmp_tl } ^^M
          }
          { \filecontentsdefmacro { \l__scontents_macro_tmp_tl } ^^M }
    }
  \cs_new_protected:Npn \__scontents_stop_environment:
    {
        \bool_if:NTF \l__scontents_writing_bool
          { \endfilecontentsdef }
          { \endfilecontentsdefmacro }
      \group_end:
    }
\group_end:
```

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

9.7.2 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_known:nn { 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`.)

9.7.3 Recording of the content in the sequence

`\__scontents_atend_environment:`
`\__scontents_stored_to_seq:`

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

The `\__scontents_stored_to_seq:` function replaces a carriage return (ASCII 13) by a new line character (ASCII 10) for optionally logging the contents of the current `scontents` environment.

```
\cs_new_protected:Npn \__scontents_atend_environment:
  {
    \bool_if:NT \l__scontents_storing_bool
      {
        \__scontents_stored_to_seq:
        \bool_if:NT \l__scontents_print_env_bool
          { \__scontents_getfrom_seq:nn { -1 } { \l__scontents_name_seq_env_tl } }
      }
    \tl_clear:N \l__scontents_macro_tmp_tl
  }
\cs_new_protected:Npn \__scontents_stored_to_seq:
  {
    \regex_replace_all:nnN { \^^M } { \^^J } \l__scontents_macro_tmp_tl
    \tl_log:N \l__scontents_macro_tmp_tl
    \__scontents_append_contents:Vx \l__scontents_name_seq_env_tl
    { \exp_not:N \__scontents_rescan_tokens:w { \tl_use:N \l__scontents_macro_tmp_tl } }
  }
```

(End definition for `\__scontents_atend_environment:` and `\__scontents_stored_to_seq:`.)

9.8 The `\Scontents` command

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

`\Scontents`
`\__scontents_norm:n`
`\__scontents_verb:w`

The `\Scontents` macro starts by parsing an optional argument and then delegates to `\__scontents_verb:w` or `\__scontents_norm:n` depending whether a star argument is present.

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

__scontents_verb:w grabs a verbatim argument using xparse's v argument parser.

```
\NewDocumentCommand { \Scontents }{ !s !O{} }
  {
    \group_begin:
      \IfNoValueF {#2}
        { \keys_set_known:nn { scontents / Scontents } {#2} }
      \IfBooleanTF{#1}
        { \__scontents_verb:w }
        { \__scontents_norm:n }
  }
\cs_new_protected:Npn \__scontents_norm:n #1
  {
     \exp_args:NV \__scontents_append_contents:nn \l__scontents_name_seq_cmd_tl {#1}
     \bool_if:NT \l__scontents_print_cmd_bool
       { \__scontents_getfrom_seq:nn { -1 } { \l__scontents_name_seq_cmd_tl } }
   \group_end:
  }
\NewDocumentCommand { \__scontents_verb:w } { +v }
  {
      \tl_set:Nn \l__scontents_temp_tl {#1}
      \regex_replace_all:nnN { \^^M } { \^^J } \l__scontents_temp_tl
      \tl_log:N \l__scontents_temp_tl
      \exp_args:NVx \__scontents_append_contents:nn \l__scontents_name_seq_cmd_tl
       { \exp_not:N \__scontents_rescan_tokens:w { \tl_use:N \l__scontents_temp_tl } }
      \bool_if:NT \l__scontents_print_cmd_bool
        { \__scontents_getfrom_seq:nn { -1 } { \l__scontents_name_seq_cmd_tl } }
    \group_end:
  }
```

(End definition for \Scontents*,* __scontents_norm:n*, and* __scontents_verb:w*. This function is documented on page 3.)*

9.9 The command \getstored

\getstored

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

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

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

9.10 The \typestored command

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

\typestored
__scontents_fcdef_print:N
__scontents_xverb:w
verbatimsc

The \typestored commands fetches a buffer from memory, prints it to the log file, and then calls filecontentsdef's \filecontentsdef@get macro to read the contents of the token list and pass them to __scontents_fcdef_print:N.

```
\NewDocumentCommand { \typestored } { s O{1} m }
  {
    \group_begin:
      \tl_set:Nx \l__scontents_temp_tl { \__scontents_getfrom_seq:nn {#2} {#3} }
      \tl_log:N \l__scontents_temp_tl
      \IfBooleanTF {#1}
        { \bool_set_false:N \l__scontents_typeverb_env_bool }
        { \bool_set_true:N  \l__scontents_typeverb_env_bool }
      \use:c { filecontentsdef@get } \__scontents_fcdef_print:N \l__scontents_temp_tl
    \group_end:
  }
```

The __scontents_fcdef_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 \scantokens.

```
\group_begin:
 \char_set_catcode_active:N \^^M
 \cs_new_protected:Npn \__scontents_fcdef_print:N #1
    {
      \tl_if_blank:VT #1
        { \msg_error:nnn { scontents } { empty-variable } {#1} }
      \cs_set_eq:NN \__scontents_fcdef_saved_EOL: ^^M
      \cs_set_eq:NN ^^M \scan_stop:
      \use:x
        {
          \exp_not:N \__scontents_rescan_tokens:w
            {
              \exp_not:N \begin{verbatimsc} ^^M
              \__scontents_strip_scantokens:N #1
              \bool_if:NF \l__scontents_typeverb_env_bool { ^^M }
              \g__scontents_end_verbatimsc_tl
            }
        }
      \cs_set_eq:NN ^^M \__scontents_fcdef_saved_EOL:
    }
\group_end:
```

Finally, the `verbatimsc` environment is defined.

```
\use:x
 { \cs_gset_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:w
    \verbatim
  }
  { }
```

(*End definition for* `\typestored` *and others. These functions are documented on page 3.*)

__scontents_strip_scantokens:N
__scontents_strip_scantokens:n
__scontents_if_scantokens:Nw

The `\__scontents_strip_scantokens:n` (`\__scontents_strip_scantokens:N`) macro takes a token list (variable) as argument and examines it. If the argument token list is *exactly* `\scantokens{`⟨*stuff*⟩`}`, then the function returns ⟨*stuff*⟩, otherwise it returns the input token list without change. The token list is wrapped in `\exp_not:n` to avoid further expansion.

```
\cs_new:Npn \__scontents_strip_scantokens:N #1
  { \exp_args:NV \__scontents_strip_scantokens:n #1 }
\cs_new:Npn \__scontents_strip_scantokens:n #1
  {
    \tl_if_head_is_N_type:nTF {#1}
      {
        \__scontents_if_scantokens:NwTF #1 \q__scontents_mark
          {
            \exp_args:No \tl_if_single_token:nTF { \use_none:nn #1 ? }
              { \exp_not:o { \use_ii:nn #1 } }
              { \exp_not:n {#1} }
          }
          { \exp_not:n {#1} }
      }
      { \exp_not:n {#1} }
  }
\prg_new_conditional:Npnn \__scontents_if_scantokens:Nw #1#2 \q__scontents_mark { TF }
  {
    \token_if_eq_meaning:NNTF \__scontents_rescan_tokens:w #1
      { \prg_return_true: }
      { \prg_return_false: }
  }
```

(*End definition for* `\__scontents_strip_scantokens:N`, `\__scontents_strip_scantokens:n`, *and* `\__scontents_if_scantokens:Nw`.)

9.11 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.)

9.12 The command meaningsc

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

```
\NewDocumentCommand { \meaningsc } { O{1} m }
  {
    \group_begin:
      \tl_set:Nx \l__scontents_temp_tl { \__scontents_getfrom_seq:nn {#1} {#2} }
      \tl_log:N \l__scontents_temp_tl
    \tl_set:Nx \l__scontents_temp_tl { \__scontents_strip_scantokens:N \l__scontents_temp_tl }
      \regex_replace_all:nnN { \v{1,} } {   } \l__scontents_temp_tl
      \ttfamily
      \cs_replacement_spec:N \l__scontents_temp_tl
    \group_end:
  }
```

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

9.13 The command `\countsc`

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

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

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

9.14 The command `\cleanseqsc`

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

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

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

9.15 Messages

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:. }
```

9.16 Finish package

Finish package

```
\file_input_stop:
```

10 Index of Implementation