Saving Write Handles with the Experimental Package scrwfile

Write-Handles sparen mit dem experimentellen Paket scrwfile

Markus Kohm

Version v0.1.99 2023-03-31

The TEX engines TEX, pdfTEX, and XETEX (but not LuaTEX) do provide only 16 handles for simultaneously write open files. Some of them are already used by LATEX. Moreover, for every file using `\@starttoc` LATEX keeps a write file open from usage of `\@starttoc`, e.g., inside `\tableofcontents`, `\listoffigures` etc., until the end of the document. Additional write files are used for the index, glossary etc. So sometimes it happens, that there are not enough handles to open another one. scrwfile was made to change the LATEX internal handling of all the helper files bases on `\@starttoc` to not keep them open simultaneously, but use only one file handle for all of them. So you should almost never run out of write file handles.

Die TEX-Engines TEX, pdfTEXund XETEX (nicht jedoch LuaTEX) bieten lediglich 16 Handles, die gleichzeitig zum Schreiben geöffnet sind. Einige davon werden bereits von LATEX selbst benötigt. Darüber hinaus hält LATEX jede Datei, die per `\@starttoc` geöffnet wird, vom verwendeten `\@starttoc` bis zum Ende des Dokuments zum Schreiben offen. Das betrifft beispielsweise `\tableofcontents`, `\listoffigure` etc. Weitere Dateien werden gegebenenfalls für Index, Glossar und ähnliche Verzeichnisse benötigt. Daher kann es manchmal geschehen, dass keine weitere Datei mehr zum Schreiben geöffnet werden kann. scrwfile wurde entwickelt, um LATEX-Interna bei der Verwendung von `\@starttoc` so abzuändern, dass nicht mehr alle Dateien gleichzeitig offen bleiben müssen. Stattdessen wird nur noch ein Handle für all diese Dateien benötigt. Damit sollte der Fall, dass keine weiteren Dateien zum Schreiben geöffnet werden können, kaum noch auftreten.

Contents / Inhalt

Part I.
English User Manual

This is an experimental package. Usage of this package is on your own risk. There is no support, if you are using this package, not only for this package but also for the class you are using or other packages, that use `\@starttoc` or `\@writefile` like caption, float, tocbasic, listings and many more.

1. Background

One of the problems not solved by the introduction of ε-TeX is the fact that TeX can support only 18 open write handles. This number seems quite large at first, but many of these handles are already reserved. TeX itself uses handle 0 for the log file. LaTeX needs handle 1 for `\@mainaux`, handle 2 for `\@partaux`, one handle for `\tableofcontents`, one handle for `\listoffigures`, one handle for `\listoftables`, and one handle for `\makeindex`. Every other such list generates another handle, and packages like hyperref or minitoc require write handles too.

The bottom line is that eventually you may get the following error message:

```
! No room for a new \write .
\ch@ck ...\else \errmessage {No room for a new #3}
                                                \fi
```

For some time, the simplest solution to this problem has been to use LuaLaTeX instead of pdfLaTeX or XeLaTeX. This eliminates the restriction, and the maximum number of files you can have open for writing is then limited only by the operating system. In reality, you usually so not need to worry about it any more.

The fact that LaTeX always opens a new file for writing every table of contents, list of figures, list of tables, etc. has another disadvantage. Such lists are not only output by their respective commands, they also could not be output a second time because the associated auxiliary file[1] is empty after the respective command until the end of the document.

The scrwfile package makes a fundamental change to the LaTeX kernel, which can solve both problems not only for LuaLaTeX but also for pdfLaTeX or XeLaTeX.

2. Fundamental Changes to the LaTeX Kernel

LaTeX classes use the LaTeX kernel command `\@starttoc` to allocate a new file handle, such as for `\tableofcontents` or `\listoffigures`. This command not only loads the associated auxiliary file but also reopens it for writing. If entries to these lists are added using `\addcontentsline`, however, the system does not write directly to these auxiliary files. Instead, LaTeX writes `\@writefile` commands to the `aux` file. Only while reading the `aux` file at the end of the document do those `\@writefile` commands become actual write operations in the auxiliary files. Additionally, LaTeX does not close the auxiliary files explicitly. Instead it relies on TeX to close all open files at the end.

This procedure ensures that the auxiliary files are only written to within `\end{document}`, but they remain open throughout the entire LaTeX run. scrwfile takes is starting point here, by redefining `\@starttoc` and `\@writefile`.

Of course changes to the LaTeX kernel always have the potential to cause incompatibilities with other packages. Those primarily affected may be those which also redefine `\@starttoc` or `\@writefile`. In some cases, it may help to change the order the packages are loaded. If you encounter such a problem, please contact the KOMA-Script author.

3. The Single-File Method

As soon as the package is loaded with

```
\usepackage{scrwfile}
```

scrwfile redefines `\@starttoc` so that it no longer allocates a write handle or opens a file for writing. `\@writefile` is redefined so that immediately before closing the `aux` file in `\end{document}`, it writes not to the usual auxiliary files but to a single new file with extension `wrt`. After reading the `aux` file, this `wrt` file will be processed once for each of the auxiliary files that `\@writefile` writes information to. However these auxiliary files do not all have to be open at the same time. Instead, only one is open at a time and is explicitly

[1] The term *auxiliary file* here refers not to the main `aux` file but to the other internal files used indirectly via the `aux` file, e. g. the `toc` file, the `lof` file, or the `lot` file.

closed afterwards. Since LaTeX reuses an internal write file, scrwfile doesn't need its own write handle for this type of table of contents or list of floating environments.

Because of this behaviour, even if you have only one table of contents, once you load scrwfile you will have access to a write file handle for bibliographies, indexes, glossaries, and similar lists that do not use `\@starttoc`. Additionally, you can create any number of tables of contents and other lists that use `\@starttoc`.

4. The File Cloning Method

Since `\@writefile` has already been modified for the single-file method described in the previous system so that it no longer writes directly to the corresponding auxiliary file, a further possibility suggests itself. When copying the `\@writefile` statements into the `wrt` file, you can also copy them to other destinations. It should be explicitly noted that this only works in the document preamble.

`\TOCclone[`⟨*list heading*⟩`]{`⟨*source extension*⟩`}{`⟨*destination extension*⟩`}`

This cloning of file entries copies entire tables of contents or other lists. For this, you only need to specify the extension of the auxiliary file whose entries should be copied and the extension of a destination file. The entries are then copied there. Of course, you can also write additional entries to this cloned file.

You can manage the ⟨*destination extention*⟩ using tocbasic. If such a file is already under the control of tocbasic, a warning will be issued. Otherwise, a new list for this extension will be created using tocbasic. You can set the heading for this list with the optional argument ⟨*list heading*⟩.

`\listof`⟨*destination extension*⟩

You can then output this new content list, for example, with the command `\listof`⟨*destination extension*⟩. The content-list attributes `leveldown`, `numbered`, `onecolumn`, and `totoc` (see `\setuptoc` in the tocbasic chapter of the KOMA-Script manual) are automatically copied to the destination list if they were already set in the source list. The `nobabel` attribute is always set for cloned content lists, because the language-selection commands in the source content list are already copied anyway.

Example: Suppose you want a short table of contents with only the chapter level in addition to the normal the table of contents:

```
\usepackage{scrwfile}
\TOCclone[Summary Contents]{toc}{stoc}
```

This creates a new table of contents with the heading "Summary Contents". The new table of contents uses an auxiliary file with the extension `stoc`. All entries to the auxiliary file with extension `toc` will also be copied to this new auxiliary file.

In order to have the new short table of contents display only the chapter entries, we use:

```
\addtocontents{stoc}{\protect\value{tocdepth}=0}
```

Although normally you cannot write to an auxiliary file before `\begin{document}`, the code above works in the preamble after loading scrwfile. Owing to the unconventional way of changing the `tocdepth` counter within the TOC file, this change only applies to this content list.

Later in the document, we then output the content list with the file extension `stoc` with:

```
\listofstoc
```

and this shows only the parts and chapters of the document.

Things become a bit more difficult if the summary contents are to be listed in the table of contents. This would seem to be possible with

```
\addtocontents{toc}{%
  \protect\addxcontentsline
    {stoc}{chapter}{\protect\contentsname}%
}
```

However, since all entries in `toc` are also copied to `stoc`, this entry would also be copied from the summary contents. So we cannot generate the entry from the content list. Because we use the tocbasic package, we can use the following:

```
\BeforeStartingTOC[toc]{%
  \addcontentslinedefault{stoc}{chapter}
                  {\protect\contentsname}%
}
```

Of course, this assumes that the `toc` file is under the control of the tocbasic package, which is indeed the case for all KOMA-Script classes. See the tocbasic chapter of the KOMA-Script manual for more information about `\BeforeStartingTOC`.

Incidentally, the `\addxcontentsline` command used in the examples is also documented in the tocbasic chapter of the KOMA-Script manual.

5. Note on the State of Development

Although this package has already been tested by many users and is often in production use, its development is still ongoing. Therefore, it is theoretically possible that there might be changes, especially to the internal functionality. It is likely that the package will be extended in the future. Some code for such extensions is already in the package. However, as there are no user commands that make use of these features, they are currently undocumented.

6. Known Package Incompatibilities

As mentioned in section 2 scrwfile redefines some commands of the LaTeX kernel. This happens not only while loading the package, but indeed at various times while the document is processed, for example just before reading the `aux` file. This results in incompatibility with packages that also redefine these commands at run time.

The titletocpackage is an example for such an incompatibility. That package redefines `\@writefile` under some conditions at run time. If you use both scrwfile and titletoc, there is no warranty for the correct behaviour of either one. This is neither an error of titletoc nor of scrwfile.

Teil II.
Deutsche Benutzeranleitung

Dies ist ein experimentelles Paket. Die Verwendung dieses Pakets erfolgt auf eigene Gefahr. Es gibt keine Unterstützung, wenn Sie dieses Paket verwenden. Das gilt nicht nur für dieses Paket, sondern auch für die Klasse, die Sie verwenden, oder andere Pakete, die `\@starttoc` oder `\@writefile` verwenden, beispielsweise caption, float, tocbasic, listings und viele weitere.

7. Hintergrund

Eines der Probleme, die auch durch die Einführung von ε-TEX nicht gelöst wurden, ist die Tatsache, dass TEX nur 18 Dateien gleichzeitig zum Schreiben geöffnet haben kann. Diese Zahl erscheint zunächst recht groß. Allerdings ist zu berücksichtigen, dass bereits LATEX selbst einige dieser Dateien belegt. Inhaltsverzeichnis, Tabellenverzeichnis, Abbildungsverzeichnis, Index, Glossar und jedes weitere Verzeichnis, das von LATEX aus erzeugt wird, belegt in der Regel eine weitere Datei. Dazu kommen Hilfsdateien von Paketen wie hyperref oder minitoc.

Im Endeffekt kann es daher geschehen, dass irgendwann die Meldung

```
! No room for a new \write .
\ch@ck ...\else \errmessage {No room for a new #3}
                                                \fi
```

erscheint. Seit einiger Zeit ist die einfachste Lösung dieses Problems die Verwendung von LuaLATEX anstelle von PDFLATEX oder XeLATEX. Damit entfällt die Beschränkung und die maximale Anzahl der gleichzeitig zum Schreiben geöffneten Dateien wird nur noch durch das Betriebssystem bestimmt. In der Realität braucht man sich darüber dann normalerweise keine Gedanken mehr zu machen.

Dass LATEX bei Verzeichnissen wie dem Inhaltsverzeichnis, dem Tabellenverzeichnis und dem Abbildungsverzeichnis immer sofort eine neue Datei zum Schreiben öffnet, hat aber auch noch einen weiteren Nachteil. Solche Verzeichnisse werden durch deren Befehle nicht nur direkt gesetzt, sie können auch kein weiteres Mal gesetzt werden, da die zugehörige Hilfsdatei nach dem jeweiligen Befehl bis zum Ende des Dokuments leer ist.

Das Paket scrwfile bietet hier eine grundsätzliche Änderung im LATEX-Kern, durch die beide Probleme nicht nur für LuaLATEX sondern auch bei Verwendung von PDFLATEX oder XeLATEX gelöst werden können.

8. Grundsätzliche Änderungen am LATEX-Kern

LATEX-Klassen verwenden zum Öffnen eines Verzeichnisses, beispielsweise mit `\tableofcontents` oder `\listoffigures`, die LATEX-Kern-Anweisung `\@starttoc`. LATEX selbst lädt bei dieser Anweisung nicht nur die zugehörige Hilfsdatei, sondern öffnet diese Hilfsdatei auch neu zum Schreiben. Werden anschließend mit `\addtocontents` oder `\addcontentsline` Einträge in dieses Verzeichnis vorgenommen, so wird jedoch nicht direkt in die geöffnete Hilfsdatei

geschrieben. Stattdessen schreibt LaTeX `\@writefile`-Anweisungen in die `aux`-Datei. Erst beim Einlesen der `aux`-Dateien am Ende des Dokuments wird dann über diese `\@writefile`-Anweisungen in die tatsächlichen Hilfsdateien geschrieben. Die Hilfsdateien werden von LaTeX auch nicht explizit geschlossen. Stattdessen verlässt sich LaTeX hier darauf, dass TeX die Dateien am Ende ohnehin schließt.

Dieses Vorgehen sorgt dafür, dass die Hilfsdateien zwar erst innerhalb von `\end{document}` tatsächlich beschrieben werden, aber trotzdem während des gesamten LaTeX-Laufs gleichzeitig offen sind. scrwfile hat nun genau hier einen Ansatzpunkt: die Umdefinierung von `\@starttoc` und `\@writefile`.

Natürlich besitzen Änderungen am LaTeX-Kern immer das Potential, dass es zu Unverträglichkeiten mit anderen Paketen kommen kann. Betroffen können in erster Linie Pakete sein, die ebenfalls `\@starttoc` oder `\@writefile` umdefinieren. In einigen Fällen kann es helfen, die Reihenfolge der Pakete zu ändern.

9. Das Eindateiensystem

Bereits beim Laden des Pakets mit Umbruchoptimierung: listings

```
\usepackage{scrwfile}
```

wird `\@starttoc` von scrwfile so umdefiniert, dass davon selbst keine Datei mehr zum Schreiben angefordert und geöffnet wird. Unmittelbar vor dem Schließen der `aux`-Datei in `\end{document}` wird dann `\@writefile` so umdefiniert, dass diese Anweisung statt in die eigentlichen Hilfsdateien in eine neue Hilfsdatei mit der Endung `wrt` schreibt. Nach dem Einlesen der `aux`-Dateien wird schließlich die `wrt`-Datei abgearbeitet und zwar ein Mal für jede der Hilfsdateien, in die mit `\@writefile` geschrieben wird. Dabei muss aber nicht jede dieser Hilfsdateien gleichzeitig geöffnet sein. Stattdessen ist immer nur eine zum Schreiben geöffnet und wird auch wieder explizit geschlossen. Da dabei eine interne Schreibdatei von LaTeX wiederverwendet wird, benötigt scrwfile keine einzige eigene Schreibdatei für diese Art von Verzeichnissen.

Selbst wenn bisher nur mit einem Inhaltsverzeichnis gearbeitet wird, steht nach dem Laden des Pakets bereits eine Schreibdatei mehr für Literaturverzeichnisse, Stichwortverzeichnisse, Glossare und ähnliche Verzeichnisse, die nicht mit `\@starttoc` arbeiten, zur Verfügung. Darüber hinaus können beliebig viele Verzeichnisse, die mit `\@starttoc` arbeiten, angelegt werden.

10. Das Klonen von Dateieinträgen

Nachdem `\@writefile` für das Eindateiensystem aus dem vorherigen Abschnitt bereits so geändert wurde, dass es nicht direkt in die entsprechende Hilfsdatei schreibt, lag eine weitere Idee nahe. Beim Kopieren der `\@writefile`-Anweisungen in die `wrt`-Datei können diese auch für andere Zielendungen übernommen werden. Es sei ausdrücklich darauf hingewiesen, dass dies nur in der Dokumentpräambel funktioniert.

`\TOCclone[`⟨*Verzeichnisüberschrift*⟩`]{`⟨*Quellendung*⟩`}{`⟨*Zielendung*⟩`}`

Durch dieses Klonen von Dateieinträgen werden so ganze Verzeichnisse geklont. Dazu muss man nur die Endung der Hilfsdatei des Verzeichnisses kennen, dessen Einträge kopiert werden sollen. Zusätzlich muss man die Endung einer Zieldatei angeben. In diese werden die Einträge

dann kopiert. Natürlich kann man in dieses geklonte Verzeichnis auch zusätzliche Einträge schreiben.

Die ⟨*Zielendung*⟩ der Zieldatei wird mit Hilfe von tocbasic verwaltet. Steht eine solche Datei bereits unter Kontrolle von tocbasic wird eine Warnung ausgegeben. Anderenfalls wird mit Hilfe von tocbasic ein neues Verzeichnis für diese Endung angelegt. Die Überschrift des neuen Verzeichnisses kann man über das optionale Argument ⟨*Verzeichnisüberschrift*⟩ bestimmen.

`\listof⟨Zielendung⟩` Ausgeben kann man dieses neue Verzeichnis dann beispielsweise über die Anweisung `\listof⟨Zielendung⟩`. Die Verzeichniseigenschaften `leveldown`, `numbered`, `onecolumn` und `totoc` (siehe `\setuptoc` im Kapitel to tocbasic in the KOMA-Script-Anleitung) werden automatisch in das Zielverzeichnis übernommen, falls sie für das Quellverzeichnis bereits gesetzt waren. Die Eigenschaft `nobabel` wird für geklonte Verzeichnisse immer gesetzt, da die entsprechenden babel-Einträge in das Quellverzeichnis ohnehin bereits kopiert werden.

Beispiel: Angenommen, Sie wollen zusätzlich zum normalen Inhaltsverzeichnis eine Gliederungsübersicht, in der nur die Kapitel angezeigt werden.

```
\usepackage{scrwfile}
\TOCclone[Gliederungsübersicht]{toc}{stoc}
```

Hierdurch wird zunächst ein neues Verzeichnis mit der Überschrift »Gliederungsübersicht« angelegt. Das neue Verzeichnis verwendet die Dateiendung `stoc`. Alle Einträge in die Datei mit der Endung `toc` werden auch in dieses Verzeichnis kopiert.

Damit dieses neue Verzeichnis nun nur die Kapitelebene ausgibt, verwenden wir:

```
\addtocontents{stoc}{\protect\value{tocdepth}=0}
```

Während normalerweise erst ab `\begin{document}` Einträge in ein Verzeichnis vorgenommen werden können, funktioniert dies nach Laden von scrwfile bereits in der Dokumentpräambel. Durch die hier gezeigte unkonventionelle Art, den Zähler `tocdepth` innerhalb der Verzeichnisdatei zu ändern, bleibt diese Änderung nur für dieses Verzeichnis wirksam.

Später im Dokument wird das Verzeichnis mit der Endung `stoc` dann durch

```
\listofstoc
```

ausgegeben und zeigt nur die Teile und Kapitel des Dokuments.

Etwas schwieriger wird es, wenn das Inhaltsverzeichnis in der Gliederungsübersicht angezeigt werden soll. Dies wäre zwar mit

```
\addtocontents{toc}{%
  \protect\addxcontentsline
    {stoc}{chapter}{\protect\contentsname}%
}
```

möglich. Da jedoch alle Einträge in `toc` auch nach `stoc` kopiert werden, würde so von der Gliederungsübersicht dieser Eintrag ebenfalls übernommen. Also darf der Eintrag nicht aus der Verzeichnisdatei heraus erzeugt werden. Da das Paket tocbasic zum Einsatz kommt, kann aber

```
\BeforeStartingTOC[toc]{%
  \addxcontentsline{stoc}{chapter}
                   {\protect\contentsname}}
```

verwendet werden. Natürlich setzt dies voraus, dass die Datei mit Endung `toc` auch unter der Kontrolle von tocbasic steht. Dies ist bei allen KOMA-Script-Klassen der Fall. Näheres zur Anweisung `\BeforeStartingTOC` im Kapitel zu tocbasic in der KOMA-Script-Anleitung zu finden. Die Erklärung zu `\addxcontentsline` findet sich dort ebenfalls.

11. Hinweis zum Entwicklungsstand

Obwohl das Paket bereits von mehreren Anwendern getestet wurde und vielfach im Einsatz ist, ist es vom Autor als experimentell eingestuft. Deshalb ist es theoretisch möglich, dass insbesondere an der internen Funktionsweise des Pakets noch Änderungen vorgenommen werden. Teilweise befindet sich auch Code für zusätzliche Erweiterungen im Paket. Da jedoch keine Benutzeranweisungen existieren, mit denen diese Möglichkeiten genutzt werden könnten, wurde auf eine Dokumentation derselben verzichtet.

12. Bekannte Paketunverträglichkeiten

Wie in Abschnitt 8 bereits erwähnt, muss scrwfile einige wenige Anweisungen des LaTeX-Kerns umdefinieren. Dies geschieht nicht allein während des Ladens des Pakets, sondern vielmehr zu verschiedenen Zeitpunkten während der Abarbeitung eines Dokuments, beispielsweise vor dem Einlesen der `aux`-Datei. Das führt dazu, dass scrwfile sich nicht mit anderen Paketen verträgt, die dieselben Anweisungen ebenfalls zur Laufzeit umdefinieren.

Ein Beispiel für eine solche Unverträglichkeit ist titletoc. Das Paket definiert unter gewissen Umständen `\@writefile` zur Laufzeit um. Werden scrwfile und titletoc zusammen verwendet, ist die Funktion beider Paket nicht mehr gewährleistet. Dies ist weder ein Fehler in titletoc noch in scrwfile.

Part III.
Implementation of scrwfile

```
1 ⟨*package⟩
2 \PackageWarningNoLine{scrwfile}{%
3   THIS IS AN EXPERIMENTAL PACKAGE!\MessageBreak
4   USAGE OF THIS PACKAGE IS ON YOUR OWN RISK!\MessageBreak
5   EVERYTHING MAY HAPPEN!\MessageBreak
6   EVERYTHING MAY CHANGE IN FUTURE!\MessageBreak
7   THERE IS NO SUPPORT, IF YOU USE THIS PACKAGE!\MessageBreak
8   Maybe, it would be better not to load this package%
9 }
```

13. Options

Currently we don't need options.

14. Body

14.1. Needed Packages

Package scrbase is needed, because of using several KOMA-Script basic commands.

```
\RequirePackage{scrbase}[2015/08/29]
```

Package scrlfile is needed because of the aux-file handling and `\protected@immediate@write`.

```
\RequirePackage{scrlfile}[2010/09/30]
```

Package iftex is used to detect LuaLaTeX.

```
\RequirePackage{iftex}
```

14.2. Is is needed?

If the user uses LuaLaTeX usage of scrwfile should not be needed.

```
\ifluatex
  \PackageWarningNoLine{scrwfile}{LuaLaTeX detected.\MessageBreak
    With LuaLaTeX you should never get an error message:\MessageBreak
    \space\space‘No room for a new \string\write’.\MessageBreak
    So scrwfile could make much more harm than benefit\MessageBreak
    and using it is not recommended}%
\fi
```

14.3. LaTeX Kernel Patchs

For some features we need to patch LaTeX kernel macros. Those features and macros are:

Single handle feature means, that LaTeX will no longer need a file handle for every help file, but only one for all files. We will patch `\@starttoc` and `\@writefile` to do so.

Clone file feature means, that every write to one file may be done to another file, too. We will patch `\@writefile` to do so.

Every patch should be minimum invasive, so that files, that are not under scrwfile's control are changed as little as possible.

`\scrwfile@if@only` First of all we check, if the file should be handled by scrwfile.

```
\newcommand*{\scrwfile@if@only}[1]{%
  \begingroup
    \scr@ifundefinedorrelax{scrwfile@only}{\@tempswatrue}{%
      \@tempswafalse
      \edef\reserved@b{#1}%
      \@for\reserved@a:=\scrwfile@only\do
        {\ifx\reserved@a\reserved@b\@tempswatrue\fi}%
    }%
    \if@tempswa
```

```
      \scr@ifundefinedorrelax{scrwfile@never}{}{%
        \edef\reserved@b{#1}%
        \@for\reserved@a:=\scrwfile@never\do
          {\ifx\reserved@a\reserved@b\@tempswafalse\fi}%
      }%
    \fi
  \expandafter\endgroup
  \if@tempswa
    \expandafter\@firstoftwo
  \else
    \expandafter\@secondoftwo
  \fi
}
```

`\scrwfile@starttoc` `\scrwfile@@starttoc` This is the internal redefinition of `\@starttoc`. First of all test, if it should be used, then use it or not.

```
\newcommand*{\scrwfile@starttoc}[1]{%
  \scrwfile@if@only{#1}{\scrwfile@@starttoc}{\scrwfile@saved@starttoc}{#1}%
}
\newcommand*{\scrwfile@@starttoc}[1]{%
⟨trace⟩  \PackageInfo{scrwfile}{%
⟨trace⟩    Using my own \string\@starttoc\space for ‘#1’}%
  \begingroup
    \if@filesw
      \xdef\scrwfile@writefilelist{\scrwfile@writefilelist,#1}%
    \fi
    \@fileswfalse
    \scrwfile@saved@starttoc{#1}%
  \endgroup
}
```

`\scrwfile@writefile` `\scrwfile@@writefile` `\scrwfile@wrtout` `\scrwfile@writefilelist` This is the internal redefinition of `\@writefile`. First of all test, if it should be used, then use it or not.

```
\newcommand*{\scrwfile@writefile}[1]{%
  \scrwfile@if@only{#1}{\scrwfile@@writefile}{\scrwfile@saved@writefile}{#1}%
}
\newcommand{\scrwfile@@writefile}[2]{%
⟨trace⟩  \PackageInfo{scrwfile}{%
⟨trace⟩    Using my own \string\@writefile\space for ‘#1’}%
  \ifnum\scrwfile@wrtout>0
    \begingroup
      \@temptokena{#2}%
      \immediate\write\scrwfile@wrtout{%
        \string\@writefile{#1}{\the\@temptokena}%
      }%
```

This was the entry for the real file. But we also may have clone files:

```
      \scrwfile@process@clones{#1}%
    \endgroup
  \fi
}
\chardef\scrwfile@wrtout\z@
```

```
\newcommand*{\scrwfile@writefilelist}{}
```

`\@writefile` `\scrwfile@saved@writefile`

We have to add the single handle feature and the clone file feature to `\@writefile` and therefore save the original definition. `\scrwfile@saved@writefile` is used, whenever a file is not under scrwfile's control.

```
\newcommand*{\scrwfile@saved@writefile}{}
\BeforeClosingMainAux{%
  \ifx\scrwfile@writefilelist\@empty\else
    \let\scrwfile@saved@writefile\@writefile
    \let\scrwfile@wrtout\@partaux
    \immediate\openout\scrwfile@wrtout \jobname.wrt
    \let\@writefile\scrwfile@writefile
  \fi
}
\AfterReadingMainAux{%
  \ifx\scrwfile@writefilelist\@empty\else
    \immediate\closeout\scrwfile@wrtout
    \chardef\scrwfile@wrtout\z@
    \begingroup
      \let\@writefile\scrwfile@saved@writefile
      \@for\@currext:=\scrwfile@writefilelist\do{%
        \begingroup
          \ifx\@currext\@empty\else
            \scr@ifundefinedorrelax{tf@\@currext}{%
⟨trace⟩               \typeout{Process extension: `\@currext'}
              \immediate\openout\@partaux \jobname.\@currext
              \expandafter\let\csname tf@\@currext\endcsname\@partaux
              \@input@{\jobname.wrt}%
              \immediate\closeout\@partaux
            }{}%
          \fi
        \endgroup
      }%
    \endgroup
  \fi
}
```

Note: Here we use a LaTeX version test, because a do-nothing-definition like the one of latexrelease wouldn't be correct, if the command is not supported by the current LaTeX kernel version setting.

```
\IfLTXAtLeastTF{2020/10/01}{%
  \DeclareHookRule{enddocument/afteraux}%
                  {scrwfile}{before}{scrlayer-notecolumn}%
}{%
  \@ifpackageloaded{scrlayer-notecolumn}{%
    \PackageWarningNoLine{scrwfile}{%
      Dangerous package order detected!\MessageBreak
      As a general rule, you should load scrwfile as soon\MessageBreak
      as possible, maybe even before `\string\documentclass'\MessageBreak
      (using `\string\RequirePackage' instead of
      `\string\usepackage').\MessageBreak
      Following packages should be loaded after scrwfile:\MessageBreak
```

```
      \space - scrlayer-notecolumn}%
  }{}%
}
```

\@starttoc
\scrwfile@saved@starttoc

We have to add the single handle feature to `\@starttoc` and therefore save the original definition. `\scrwfile@saved@starttoc` is be used, whenever the file is not under scrwfile's control. Because of old versions of package hyperref, that do a hard redefinition without any care for changed definitions, we have to take care for that and cannot do a simple

```
\let\scrwfile@saved@starttoc\@starttoc
\let\@starttoc\scrwfile@starttoc
```

```
\newcommand*{\scrwfile@saved@starttoc}{}
\AtBeginDocument{%
  \begingroup
    \@ifpackageloaded{hyperref}{%
      \scr@ifundefinedorrelax{Hy@AtBeginDocument}{%
        \PackageInfo{scrwfile}{%
          Using immediate redefinition of ‘\string\@starttoc’%
        }%
        \aftergroup\@firstofone
      }{%
        \ifx\Hy@AtBeginDocumentHook\@undefined
          \PackageInfo{scrwfile}{%
            Using immediate redefinition of ‘\string\@starttoc’%
          }%
          \aftergroup\@firstofone
        \else
```

From v7.00o hyperref does not redefine `\@starttoc` any more. So we do need the usage of `\Hy@AtBeginDocument` only for versions before.

```
          \@ifpackagelater{hyperref}{2022/02/22}{% newer than v7.00n
            \PackageInfo{scrwfile}{%
              Using immediate redefinition of ‘\string\@starttoc’%
            }%
            \aftergroup\@firstofone
          }{%
            \PackageWarning{scrwfile}{%
              Outdated package ‘hyperref’ detected.\MessageBreak
              Using ‘\string\Hy@AtBeginDocument’ for redefinition of
              ‘\string\@starttoc’.\MessageBreak
              We recommend to update ‘hyperref’
            }%
            \aftergroup\Hy@AtBeginDocument
          }%
        \fi
      }%
    }{%
      \PackageInfo{scrwfile}{%
        Using immediate redefinition of ‘\string\@starttoc’%
      }%
      \aftergroup\@firstofone
```

```
    }%
  \endgroup
  {%
    \PackageInfo{scrwfile}{%
      Extending `\string\@starttoc'
    }%
    \let\scrwfile@saved@starttoc\@starttoc
    \let\@starttoc\scrwfile@starttoc
  }%
}
```

14.4. Clone TOC Feature

scrwfile may clone a TOC, that means, every entry to one file will be copied to other files, too. You must not clone recursively!

ToDo: The whole feature should be moved to tocbasic. But this would need a complete re-implementation and would result in one more write handle for every cloned file. So maybe it wouldn't be a really good idea to do so.

`\scrwfile@process@clones`

```
\newcommand*{\scrwfile@process@clones}[1]{%
  \scr@ifundefinedorrelax{scrwfile@clone@#1}{}{%
    \begingroup
      \let\@@protect\protect\let\protect\@empty\afterassignment\restore@protect
      \edef\reserved@b{\csname scrwfile@clone@#1\endcsname}%
      \edef\reserved@c{,#1}%
      \@for \reserved@a:=\reserved@b\do {%
        \@tempswatrue
        \@for \reserved@d:=\reserved@c\do {%
          \ifx\reserved@d\reserved@a\@tempswafalse\fi
        }%
        \if@tempswa
⟨trace⟩           \typeout{clone entry from `#1' to `\reserved@a'}%
          \immediate\write\scrwfile@wrtout{%
            \string\@writefile{\reserved@a}{\the\@temptokena}%
          }%
          \edef\reserved@c{\reserved@c,\reserved@a}%
        \fi
      }%
    \endgroup
  }%
}%
```

`\TOCclone` Clone the entries from the second (first mandatory) argument TOC to the third (second mandatory) argument TOC. If the first (optional) argument was given, define `\listof#3name` to this and also define `\listof#3` and clone the toc features `leveldown`, `numbered`, `onecolumn` and `totoc` of `#2` to `#3`. The toc feature `nobabel` will always be set, because the babel entries at TOC `#3` will be cloned from TOC `#2`.

Note: We use owner `TOCclone` for all cloned extensions.

```
\newcommand*{\TOCclone}[3][]{%
  \RequirePackage{tocbasic}%
  \scr@ifundefinedorrelax{scrwfile@clone@#2}{%
    \expandafter\protected@edef\csname scrwfile@clone@#2\endcsname{%
      #3,\protect\csname scrwfile@clone@#3\endcsname
    }%
  }{%
    \edef\reserved@b{\csname scrwfile@clone@#2\endcsname}%
    \expandafter\protected@edef\csname scrwfile@clone@#2\endcsname{%
      \csname scrwfile@clone@#2\endcsname,%
      #3,%
      \protect\csname scrwfile@clone@#3\endcsname
    }%
  }%
  \scr@ifundefinedorrelax{scrwfile@clone@#3}{%
    \expandafter\let\csname scrwfile@clone@#3\endcsname\@empty
  }{}%
  \Ifattoclist{#3}{%
    \PackageWarning{scrwfile}{‘#3’ already under control of
      tocbasic.\MessageBreak
      Nevertheless features will be set%
    }%
  }{%
    \addtotoclist[TOCclone]{#3}%
  }%
  \setuptoc{#3}{nobabel}%
  \IfArgIsEmpty{#1}{%
  }{%
    \@namedef{listof#3name}{#1}%
    \@namedef{listof#3}{\listoftoc{#3}}%
    \Iftocfeature{#2}{leveldown}{\setuptoc{#3}{leveldown}}{}%
    \Iftocfeature{#2}{numbered}{\setuptoc{#3}{numbered}}{}%
    \Iftocfeature{#2}{onecolumn}{\setuptoc{#3}{leveldownonecolumn}}{}%
    \Iftocfeature{#2}{totoc}{\setuptoc{#3}{totoc}}{}%
  }%
}
\@onlypreamble\TOCClone
```

⟨/package⟩

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Change History