

The extract package *

Hendri Adriaens

v1.9 (2019/09/17)

Abstract

This package can be used to (conditionally) extract specific commands and environments from a source file and write them to a target file. This can be done without significant changes to the source document, but labels inside the source can be used for more flexibility. The package also provides environments to write code directly to the target file or use code in both the source and the target file. These tools allow one to generate ready-to-run files from a source document, containing only the extracted material, in an otherwise ordinary \LaTeX run.

Contents

1 Introduction	1	6 How it works, limitations	10
2 Environment extraction	2	7 Source and examples	11
3 Command extraction	3	8 Implementation	11
4 Conditional extraction	4	References	25
4.1 Extraction with numbers . .	4	Acknowledgements	26
4.2 Extraction with labels . . .	5	Version history	26
5 Extraction tools	6	Index	27
5.1 Manual extraction	6		
5.2 Skipping extraction	8		
5.3 Miscellaneous options . . .	9		

1 Introduction

I created this package when I was working on some lecture notes with exercises in the text and wanted to generate an exercises book on the fly. David Carlisle put forward the idea to use a technique as is now implemented in this package. The package heavily uses the verbatim package [13] by Rainer Schöpf and uses the xkeyval package [1] to provide a simple and easy interface¹.

There are other packages around that provide tools for conditionally typesetting material or writing material to an external file. Let me list a few.

*This package can be downloaded from the CTAN mirrors: `/macros/latex/contrib/extract`. See `extract.dtx` for information on installing extract into your \LaTeX distribution and for the license of this package.

¹And some tools like `\XKV@ifundefined` and `\XKV@sp@deflist`. See section 8 for more information.

askinclude [12], excludeonly [9]

These packages enhance \TeX 's `\include` and `\includeonly` system to select the files that should be included in typesetting.

comment [6], verbatim [13], xcomment [14]

The first two packages define the `comment` environment and the third the `xcomment` environment. These environments ignore their body. But if the command `\xcomment` is used and supplied with a list of environments, these environments will be typeset when they appear in the body of the `xcomment` environment. The `comment` package provides the commands `\includecomment` and `\excludecomment` to do a similar job.

optional [3], version [4], versions [8]

These packages define some commands with which you can control which material should be typeset.

pagesel [11], pdfpages [10], selectp [2]

These packages only typeset certain pages.

fancyvrb [15], listings [7]

These packages (among others) provide tools to write text to an external file.

The `extract` package differs from all these packages since it extracts content and leaves the typeset version of the original document untouched. Furthermore, for simple extraction jobs, it is not necessary to make any changes to the document other than adding the `\usepackage` command. This allows for conditional extraction of commands and environments based on the number of the command or environment counted from the beginning of the document. More flexible conditional extraction can be achieved by adding labels to the source document. How all of this works will be explained in the sections to come.

2 Environment extraction

The following provides an example of the user interface of the package.

```
\usepackage[
  active,
  generate=file,
  extract-env={figure,table}
]{extract}
```

options
active
generate
extract-env

If the `active` option is not specified (or set to `false`), the package does nothing and no files are generated. If the option is specified, the package will redefine the environments `figure` and `table` so that they write their bodies (the content of the environment) to the file indicated with the `generate` option, here `file.tex`², including the `\begin{figure}` and `\end{figure}` commands. Besides that, the environments will be executed as usual. Most environments are supported (known exception is the `document` environment). When the package encounters `\begin{document}` or `\end{document}` in the source file, by default³, it will also write these commands to the target file, such that, if a suitable preamble is added, the file is ready to be run by

²If a file extension is lacking, `.tex` will be used.

³See section 5.3 for the `handles` option.

TeX. See listing 1 for an example.⁴ The package will also write a header to the target

xtrex1.tex	file.tex
<pre> \documentclass[10pt]{article} \usepackage[active, generate=file, copydocumentclass=false, extract-env=equation]{extract} \begin{extract} \documentclass[11pt]{article} \end{extract} \begin{document} Some text. \begin{equation} a^2+b^2=c^2 \end{equation} Some text. \begin{equation} x^2+y^2=z^2 \end{equation} Some text. \end{document} </pre>	<pre> \documentclass[11pt]{article} \begin{document} \begin{equation} a^2+b^2=c^2 \end{equation} \begin{equation} x^2+y^2=z^2 \end{equation} \end{document} </pre>

Listing 1: Environment extraction.

file with some information about the source and time of generation of the target file. This header can be turned off with the `no-header` option. See also section 5.3.

3 Command extraction

options
`extract-cmd`
`extract-cmdline`

This package can also extract commands. It supplies two methods to do this. See the example below.

```

\usepackage[
  active,
  generate=file,
  extract-cmd=section,
  extract-cmdline=label
]{extract}

```

The first method (accessed with the `extract-cmd` option) is based on the particular syntax of the command and hence only supports particular commands. It will read its arguments and write them, together with the original command, to the target file. Besides that, the command will be executed as usual. Currently, the commands `\chapter`, `\section`, `\subsection`, and `\subsubsection` in the standard TeX classes (and classes or packages derived from those) are supported. An optional argument to these commands, like `\chapter[short]{long}` is supported. However, the starred version `\chapter*{title}` will not be written to the file due to technical limitations. Sections 6 and 8 will explain this in more detail.

The second method (accessed with the `extract-cmdline` option) will redefine commands to write themselves and all the text following on the same line to the target

⁴See section 5.1 for the `extract` environment, section 5.3 for the `copydocumentclass` option and section 7 how to obtain the example files.

file and will also execute the entire line as with an ordinary \LaTeX run. This allows to re-define any command that is not supported by the first method, but should be applied with care. If the command is used internally in a class or style file, your document might fail to run. In particular, one should not redefine one of the commands supported by the first method with this method. See listing 2 for a working example of both methods.⁵

xtrex2.tex	file.tex
<pre> \documentclass{article} \usepackage[active, generate=file, extract-env=exercise, extract-cmd=section, extract-cmdline=label]{extract} \begin{extract*} \newtheorem{exercise}{Exercise} \end{extract*} \begin{document} \section{Theory} \label{sec:1} Some text. \section{Exercises} \begin{exercise} Use the results from section \ref{sec:1} to show that\dots \end{exercise} Some text. \end{document} </pre>	<pre> \documentclass{article} \newtheorem{exercise}{Exercise} \begin{document} \section{Theory} \label{sec:1} \section{Exercises} \begin{exercise} Use the results from section \ref{sec:1} to show that\dots \end{exercise} \end{document} </pre>

Listing 2: Command extraction.

4 Conditional extraction

4.1 Extraction with numbers

options It is also possible to conditionally extract environments and commands. After the op-
-nrs tions `extract-env`, `extract-cmd` and `extract-cmdline` are used, for each environment or command specified there, there will be a new option with the name of that environment or command and the `-nrs` postfix. This option can take a comma separated list in which you can specify which environments or macros (counting from the `\usepackage{extract}` command⁶) should be extracted. Table 1 on page 5 lists the syntax that can be used in the comma separated list. The term ‘item’ is used for ‘command or environment’. See an example in the listing below.

```

\documentclass{book}
\usepackage[
  active,
  generate=file,

```

⁵See section 5.1 for the `extract*` environment.

⁶Notice that starred commands like `\chapter*`, which won't be redefined by the `extract-cmd` option, will also not be counted.

Syntax	Meaning
$\langle x \rangle$	Item $\langle x \rangle$ will be extracted.
$-\langle x \rangle$	All items up to and including item $\langle x \rangle$ will be extracted.
$\langle x \rangle -$	All items from and including item $\langle x \rangle$ will be extracted.
$\langle x \rangle - \langle y \rangle$	All items in between $\langle x \rangle$ and $\langle y \rangle$, including $\langle x \rangle$ and $\langle y \rangle$ will be extracted.

Table 1: Syntax for conditional extraction with numbers.

```
extract-env={figure,table},
figure-nrs={-2,4-},
extract-cmd=chapter,
chapter-nrs={3-5,7}
]{extract}
\begin{document}
...
\end{document}
```

This example, when completed with content, will extract all `table` environments, figure environments 1, 2, and all figures from (and including) figure 4. It will also extract chapter commands 3 to 5 and 7.

4.2 Extraction with labels

options Conditional extraction is also possible with labels. The advantage of using labels is that
-labels output does not change (in comparison to using numbers) when commands or environments are added. The drawback is that one needs to modify the source document and add labels in the text.

`\extractionlabel` Labels are declared with the following command.

```
\extractionlabel{\name}
```

A label should be declared just before the command or environment that you want to extract. For instance

```
\extractionlabel{exer-a}
\begin{exercise}
...
\end{exercise}
```

You can reuse the same label multiple times and you can specify which items should be extracted by the options with a `-labels` prefix. This works in the same way as with numbers.

```
\documentclass{book}
\usepackage[
  active,
  generate=file,
  extract-env=exercise,
  exercise-labels={exer-a,exer-c}
]{extract}
\begin{document}
...
\end{document}
```

This example will only extract exercises that have been preceded by the declaration `\extractionlabel{exer-a}` or `\extractionlabel{exer-c}`.

When using both conditional extraction with numbers and with labels, the command or environment at hand will be extracted when at least one of the conditions is true. Find an example in listing 3.

xtrex3.tex	file.tex
<pre> \documentclass{article} \usepackage[active, generate=file, extract-env=figure, figure-nrs={1,3}, figure-labels={fig-a,fig-b}]{extract} \begin{document} Some text. \begin{figure} Figure 1. \end{figure} Some text. \extractionlabel{fig-a} \begin{figure} Figure 2. \end{figure} Some text. \extractionlabel{fig-b} \begin{figure} Figure 3. \end{figure} Some text. \extractionlabel{fig-c} \begin{figure} Figure 4. \end{figure} \end{document} </pre>	<pre> \documentclass{article} \begin{document} \begin{figure} Figure 1. \end{figure} \begin{figure} Figure 2. \end{figure} \begin{figure} Figure 3. \end{figure} \end{document} </pre>

Listing 3: Conditional extraction.

5 Extraction tools

5.1 Manual extraction

environment
extract The package provides the environment `extract`.

```

\begin{extract}
<body>
\end{extract}

```

This environment writes its body only to the target file. This can be used to generate a preamble in the target file so that you can run the generated file through \LaTeX immediately after creation. See the example in listing 1 on page 3 and the example below.

environment
extract* There also exists a starred version of the `extract` environment.

```

\begin{extract*}

```

```

<body>
\end{extract*}

```

This environment will not only write the body to the target file, but will also execute the code in the source document. This can be used to create a common preamble which holds packages and commands that will be used in both the source and the target file. See listing 2 on page 4 and listing 4 for examples. Notice that the ‘answer’ will not appear in xtrex4.dvi.

xtrex4.tex	file.tex
<pre> \documentclass{article} \usepackage[active, generate=file, extract-env=equation*]{extract} \begin{extract*} \usepackage{amsmath} \end{extract*} \begin{document} Some text. \begin{equation*} x^2+y^2=z^2 \end{equation*} \begin{extract} \$x=3\$, \$y=4\$ and \$z=5\$ satisfy this equation. \end{extract} \end{document} </pre>	<pre> \documentclass{article} \usepackage{amsmath} \begin{document} \begin{equation*} x^2+y^2=z^2 \end{equation*} \$x=3\$, \$y=4\$ and \$z=5\$ satisfy this equation. \end{document} </pre>

Listing 4: Extract environments.

`\extractline` The package also provides a command that extracts the current line.
`\extractline*`

```

\extractline
\extractline*

```

The starred version also executes the code at that line. See the example below.

```

\extractline This line should be extracted.
\extractline*This line should be extracted and executed.

```

Notice that a space is following the command `\extractline` and that no space is following the command `\extractline*`. In the first line, \TeX will eat this space, but it won't do that in the second line. If we add a space there in between the `*` and `This`, this space will be executed and extracted as well.

`\extractline` The `extract` and `extract*` environments and the macros `\extractionlabel`
`\extractline*` and `\extractionlabel*` have an optional argument for specifying the label directly.
environments

```

extract      \begin{extract}[\langle name \rangle]
extract*    \begin{extract*}[\langle name \rangle]
            \extractline[\langle name \rangle]
            \extractline*[\langle name \rangle]

```

options The options `extract-nrs` and `line-nrs` can be used to control conditional extraction of these environments and commands using numbers. Moreover, one can do conditional extractions with labels for these commands and environments as well. Use the options `extract-labels` and `line-labels` for that purpose. See also section 4 for information about conditional extraction.

See an example in listing 5. This example will demonstrate that only certain lines and environments are extracted. Note that, when running `xtrex5.tex` with \LaTeX , the output, `xtrex5.dvi`, will contain the following.

```
line 2
line 4
line 5
```

`xtrex5.tex`

`file.tex`

```
\documentclass{article}
\usepackage[
  active,
  generate=file,
  copydocumentclass=false,
  extract-labels=type-a,
  line-labels={type-a,type-c},
  line-nrs=3
]{extract}
\begin{extract}[type-a]
\documentclass{article}
\end{extract}
\begin{extract}[type-b]
\documentclass{book}
\end{extract}
\begin{document}
\parindent0pt
\extractline[type-a]line 1\\
\extractline*line 2\\
\extractline line 3\\
\extractline*[type-a]line 4\\
\extractline*[type-c]line 5\\
\extractline line 6\\
\end{document}
```

```
\documentclass{article}

\begin{document}
line 1\\
line 3\\
line 4\\
line 5\\

\end{document}
```

Listing 5: Optional labels.

5.2 Skipping extraction

environment The package also provides the `extractskip` environment.
extractskip

```
\begin{extractskip}[\langle name \rangle]
\langle body \rangle
\end{extractskip}
```

The body of this environment will not be written to the target file, but will be executed. This environment can be used to skip material in an environment which extracts its entire body. This could be the `extract` environment, but also an environment that has been redefined to be extracted using the `extract-env` option. The argument $\langle name \rangle$ is optional and contains the label.

options
extractskip-nrs
extractskip-labels

This environment can also operate conditionally as has been described in section 4 using the options `extractskip-nrs` and `extractskip-labels`. Listings 6 and 7 on page 10 will demonstrate this environment.

xtrex6.tex	file.tex
<pre> \documentclass{article} \usepackage[active, generate=file, extract-env=figure]{extract} \begin{document} \begin{figure}[!h] \begin{extractskip} \fbbox{figure 1} \end{extractskip} \fbbox{figure 2} \end{figure} \begin{extract*} \begin{itemize} \item 1 \item 2 a d \item 4 \end{itemize} \end{extract*} \end{document} </pre>	<pre> \documentclass{article} \begin{document} \begin{figure}[!h] \fbbox{figure 2} \end{figure} \begin{itemize} \item 1 \item 2 a d \item 4 \end{itemize} \end{document} </pre>

Listing 6: Skipping extraction.

5.3 Miscellaneous options

option When setting the header option to false, no header will be written to the target file.
header If the option is not specified or set to true, the package will write a header to the target file including information on when the target file was generated and which source file was used.

option This option controls whether the package will write `\begin{document}` and
handles `\end{document}` to the target file when it encounters these commands in the source document. By default, this option is set to true. When the option is set to false, the generated file can be `\inputed` or `\included` by another file immediately after production.

option This option control whether the target file should get the same `\documentclass`
copydocumentclass command as the source document (including class options). If set to false, you should specify another document class for the target file, for instance using the `extract` environment. See listing 1 on page 3 for an example.

xtrex7.tex	file.tex
<pre> \documentclass{article} \usepackage[active, generate=file, extractskip-labels=skipb]{extract} \begin{document} \begin{extract*} \begin{itemize} \begin{extractskip}[skipa] \item 1 \end{extractskip} \begin{extractskip}[skipb] \item 2 \end{extractskip} \begin{extractskip}[skipc] \item 3 \end{extractskip} \end{itemize} \end{extract*} \end{document} </pre>	<pre> \documentclass{article} \begin{document} \begin{itemize} \item 1 \item 3 \end{itemize} \end{document} </pre>

Listing 7: Skipping extraction conditionally.

6 How it works, limitations

The package works as follows. When an environment is asked to be extracted, the package will first make a backup of the environment with the XTR prefix, for instance, `XTRequation*`. After that, the package will redefine the environment to read the lines of its body verbatim (without executing), parse the lines to locate `extractskip` environments and write all lines to a temporary file and the lines that are not in an `extractskip` environment to a temporary file. After the environment is finished, the temporary file, containing for instance

```

\begin{XTRequation*}
x^2+y^2=z^2
\end{XTRequation*}

```

will be inserted in the source document using `\input`. This works, at least in theory, with most environments. Notice that this method does require a change to the `\begin` and `\end` macros provided by the \TeX kernel [5]. I had to add a hook to these commands to be able to collect code to be executed after the current environment is ended. In particular, this package will use those hooks to `\input` the original code after finishing the current group. See section 8 for more details.

Commands require a different approach. As a command can have a very specific argument structure, redefining commands safely, without distorting its original behavior, is not possible in general. I have chosen to support the most basic document structure commands as provided by standard \TeX classes, like `book` and `article`. Extraction of commands from other classes, like provided by the `koma-script` bundle, might work, but there is no guarantee.

To be more precise: when requested, the macros `\@chapter` and `\@sect` will be redefined. These commands are at the basis of all chapter and section commands. When `extract` redefines one of these commands, it first makes a backup, like `\XTR@chapter`.

After that, it will redefine the original command to read its argument(s), export them to the target file and execute the backup with the proper arguments.

An alternative to this, rather restricted method is the method accessed by the `extract-cmdline` option which redefines the commands listed there to read the entire line of text, write that to the target file and afterwards execute it with a backup copy of the original command. This method too has its drawbacks though as you can't redefine commands that are used internally in other macros.

More details on the package code can be found in section 8.

7 Source and examples

To generate this documentation, find the source of this package, `extract.dtx` in your local \TeX installation or on CTAN and perform the following steps.

```
latex extract.dtx
latex extract.dtx
bibtex extract
makeindex -s gglo.ist -o extract.gls extract.glo
makeindex -s gind.ist -o extract.ind extract.idx
latex extract.dtx
latex extract.dtx
```

If you only want to produce the package and example files from the source, then the first step is sufficient. This step will generate the package file `extract.sty` and the example files `xtrex1.tex`, `xtrex2.tex`, `xtrex3.tex`, `xtrex4.tex`, `xtrex5.tex`, `xtrex6.tex` and `xtrex7.tex`.

8 Implementation

Initializations.

```
1%<*extract>
2\NeedsTeXFormat{LaTeX2e}[1995/12/01]
3\ProvidesPackage{extract}
4 [2019/09/17 v1.9 extract content from document (HA)]
5\RequirePackage{verbatim}
6\RequirePackage{xkeyval}
7\newwrite\XTR@out
8\newwrite\XTR@tmp
9\newif\ifXTR@st
10\newif\ifXTR@skip
11\newif\ifXTR@extract

\XTR@err {\text{}}
Error macro.
12\def\XTR@err#1{\PackageError{extract}{#1}\@ehc}

\XTR@namelet {\cmd1}{\cmd2}
Version of \let for two command sequence names.
13\def\XTR@namelet#1#2{%
14 \expandafter\let\csname#1\expandafter\endcsname\csname#2\endcsname
15}
```

<i>option</i>	Options section, powered by xkeyval. Control extraction with one switch.
active	16 \define@boolkey [XTR]{extract.sty} [XTR@]{active} [true] {}
<i>option</i>	Do not create a header in the target file.
header	17 \define@boolkey [XTR]{extract.sty} [XTR@]{header} [true] {}
<i>option</i>	Extract \begin{document} and \end{document} or not.
handles	18 \define@boolkey [XTR]{extract.sty} [XTR@]{handles} [true] {}
<i>option</i>	Copy the \documentclass command to the target file.
copydocumentclass	19 \define@boolkey [XTR]{extract.sty} [XTR@]{copydocumentclass} [true] {}
<i>option</i>	Entry point for the target file name.
generate	20 \DeclareOptionX [XTR]{generate}{\lowercase{\def\XTR@file{#1}}}
<i>option</i>	Environments that should be extracted.
extract-env	21 \DeclareOptionX [XTR]{extract-env}{% 22 \def\XTR@envs{#1}% 23 \XKV@for@n{#1}\XTR@tempa\XTR@tempb 24 }
<i>option</i>	Commands that should be extracted with the ‘arguments method’.
extract-cmd	25 \DeclareOptionX [XTR]{extract-cmd}{% 26 \def\XTR@cmdsargs{#1}% 27 \XKV@for@n{#1}\XTR@tempa\XTR@tempb 28 }
<i>option</i>	Commands that should be extracted with the ‘line method’.
extract-cmdline	29 \DeclareOptionX [XTR]{extract-cmdline}{% 30 \def\XTR@cmdsline{#1}% 31 \XKV@for@n{#1}\XTR@tempa\XTR@tempb 32 } 33 \def\XTR@tempb{%
<i>options</i>	For each environment or command provide new package options that save the argument to a list cleared from redundant spaces. The -nrs options also create a ‘counter’ for counting the commands or environments. The lists and counters will be used for conditional extraction. Note that \XTR@tkey contains the key name inside the option macro (due to the use of \ProcessOptionsXi).
-nrs	34 \DeclareOptionX [XTR]{\XTR@tempa-nrs}{% 35 \expandafter\XKV@sp@deflist\csname XTR@\XKV@tkey\endcsname{##1}% 36 \XTR@namelet{XTR@\XKV@tkey @cnt}{z@}% 37 }%
-labels	38 \DeclareOptionX [XTR]{\XTR@tempa-labels}{% 39 \expandafter\XKV@sp@deflist\csname XTR@\XKV@tkey\endcsname{##1}% 40 }% 41 }
<i>options</i>	Generate options for \extractline commands.
line-nrs	42 \def\XTR@tempa{line}\XTR@tempb
line-labels	Generate options for extract environments.
<i>options</i>	43 \def\XTR@tempa{extract}\XTR@tempb
extract-nrs	Generate options for extractskip environments.
extract-labels	44 \def\XTR@tempa{extractskip}\XTR@tempb
<i>options</i>	Generate an error for unknown options.
extractskip-nrs	
extractskip-labels	45 \DeclareOptionX*{\XTR@err{Unknown option ‘\CurrentOption’}}

Initialize options.

```
46 \ExecuteOptionsX[XTR]{header=true,handles=true,copydocumentclass=true}
```

Process options.

```
47 \ProcessOptionsX[XTR]
```

```
\XTR@opentmp  Shortcut macros for much used command sequences.
\XTR@writetmp 48 \def\XTR@opentmp{\immediate\openout\XTR@tmp\jobname.xtr\relax}
\XTR@closetmp 49 \def\XTR@writetmp{\immediate\write\XTR@tmp}
\XTR@writeout 50 \def\XTR@closetmp{\immediate\closeout\XTR@tmp}
51 \def\XTR@writeout{\immediate\write\XTR@out}
```

Perform some checks on the input. Notice the use of `\XKV@ifundefined` which is equal to `\@ifundefined` if no ϵ - \TeX engine is available and which uses `\ifcsname` when it is. In the latter case, testing whether commands are defined does not create an entry in \TeX 's hash table.

```
52 \ifXTR@active
53   \XKV@ifundefined{XTR@file}{
54     \XTR@activefalse
55     \XTR@err{no file to generate; extract deactivated}
56   }{}
57   \XTR@opentmp
58   \XTR@writetmp{%
59     \string\lowercase{\string\def\string\XTR@tempa{\jobname}}%
60   }
61   \XTR@closetmp
62   \input{\jobname.xtr}
63   \ifx\XTR@tempa\XTR@file
64     \XTR@activefalse
65     \XTR@err{attempt to overwrite source file; extract deactivated}
66   \fi
67 \fi
```

`\@envdepth` Counter for depth of environments.

```
68 \newcount\@envdepth\@envdepth\z@
```

`\begin` $\{\langle environment \rangle\}$

Modify the macro `\begin` to allow adding code to a level specific hook which can be executed after `\endgroup` in `\end`. See for more info on this macro the \LaTeX source [5].

```
69 \@namedef{begin}#1{%
70   \@ifundefined{#1}%
71     {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
72     {\def\reserved@a{\def\@currenvir{#1}%
73       \edef\@currenvline{\on@line}%
74       \csname #1\endcsname}}%
75   \@ignorefalse
76   \begingroup\@endpfalse
```

Advance depth level.

```
77   \global\advance\@envdepth\@ne
```

Initialize the hook for this level.

```
78   \global\@namedef{@afterendenvhook@\romannumeral\@envdepth}{}%
79   \reserved@a}
```

`\end` $\{\langle environment \rangle\}$
 Modify `\end` to execute the code collected in the hook.

```

80 \namedef{end}#1{%
81   \csname end#1\endcsname\@checkend{#1}%
82   \expandafter\endgroup\if@endpe\@doendpe\fi

```

Copy current hook code to a temporary macro.

```

83   \expandafter\let\expandafter\reserved@a
84   \csname @afterendenvhook@\romannumeral\@envdepth\endcsname

```

Decrease the depth.

```

85   \global\advance\@envdepth\m@ne

```

Execute the hook of the current environment. This is done after decreasing the depth as to avoid level mixing problems when the hook contains another environment. This environment has to be executed at the same level as the environment in which the hook was defined since it is executed after the group and does not belong anymore to the environment in which the hook was defined.

```

86   \reserved@a\relax
87   \if@ignore\@ignorefalse\ignorespaces\fi}

```

`\AfterEndEnv` Adds code to the macros `\@afterendenvhook@i,i,i`, etc. which will be executed after the group of the current environment.

```

88 \def\AfterEndEnv{%
89   \expandafter\g@addto@macro
90   \csname @afterendenvhook@\romannumeral\@envdepth\endcsname
91 }

```

`\XTR@checkxtr` $\{\langle type \rangle\}\{\langle item \rangle\}$
 Checks whether a certain environment or command should be extracted or skipped. $\langle type \rangle$ is the type of check: for extraction or for skipping content. $\langle item \rangle$ is the name of a command or an environment.

```

92 \def\XTR@checkxtr#1#2{%
93   \@nameuse{XTR@#1false}%
94   \XTR@namelet{XTR@maketrue}{XTR@#1true}%

```

First check whether, for this macro or environment, some method of conditional extraction is used. If not, just extract.

```

95   \XKV@ifundefined{XTR@#2-nrs}{%
96     \XKV@ifundefined{XTR@#2-labels}\XTR@maketrue{}%
97   }{%

```

Advance the ‘counter’.

```

98   \begingroup
99     \expandafter\count@\csname XTR@#2-nrs@cnt\endcsname
100    \advance\count@\@ne
101    \edef\XTR@resa{\expandafter\noexpand\expandafter\gdef\expandafter
102      \noexpand\csname XTR@#2-nrs@cnt\endcsname{\the\count@}}%
103    \expandafter\endgroup\XTR@resa
104  }%
105  \@nameuse{ifXTR@#1}\else
106    \XKV@ifundefined{XTR@#2-labels}{}{%

```

If the current label is in the list for extraction, extract it.

```

107    \ifx\XTR@currentlabel\relax\else

```

```

108     \@expandtwoargs\in@{\XTR@currentlabel,}%
109     {\csname XTR@#2-labels\endcsname,}%
110     \ifin@\XTR@maketrue\fi
111   \fi
112 }%
113 \fi
114 \@nameuse{ifXTR@#1}\else
115   \XKV@ifundefined{XTR@#2-nrs}{\fi}%

```

If the current command or environment number is in the list, extract it.

```

116   \expandafter\XTR@ch@ckxtr\csname XTR@#2-nrs\expandafter
117   \endcsname\csname XTR@#2-nrs@cnt\endcsname
118 }%
119 \fi

```

Redefine `\XTR@currentlabel` to avoid extracting all following environments of this type.

```

120 \global\let\XTR@currentlabel\relax
121 }

```

`\XTR@ch@ckxtr` *<list><counter>*

Parse the *<list>* of numbers and compare each item with the *<counter>* holding the number of the current item.

```

122 \def\XTR@ch@ckxtr#1#2{%
123   \XKV@for@o#1\XTR@resa{\expandafter\XTR@ch@ck@tr\XTR@resa--\@nil#2}%
124 }

```

`\XTR@ch@ck@tr` *<x>-<y>-<z>\@nil<counter>*

Parse an element of the list. Basically, decide whether we have x, x-y, x- or -y and act accordingly.

```

125 \def\XTR@ch@ck@tr#1-#2-#3\@nil#4{%
126   \ifx\@empty#1\@empty
127     \ifnum#4>#2 \else\XTR@maketrue\fi
128   \else
129     \ifx\@empty#2\@empty
130       \ifx\@empty#3\@empty
131         \ifnum#4=#1 \XTR@maketrue\fi
132       \else
133         \ifnum#4<#1 \else\XTR@maketrue\fi
134     \fi
135   \else
136     \ifnum#4<#1 \else\ifnum#4>#2 \else\XTR@maketrue\fi\fi
137   \fi
138 \fi
139 }

```

`\extractionlabel` `\XTR@currentlabel` `\extractionlabel` saves its argument (after removing redundant spaces). This label will be used for conditional extraction. `\XTR@currentlabel` is initialized.

```

140 \def\extractionlabel{\KV@@sp\def\XTR@currentlabel}
141 \let\XTR@currentlabel\relax

```

`\extract` Define environments that write verbatim to the target file. The starred version also executes the code by writing it to a temp file and inputting it `\AfterEndEnv`, just as with `\extract*`

redefining existing environments. When the package is inactive, `extract` is equivalent to the `comment` environment and `extract*` takes its body out of the group and executes it hence acting as if `\begin{extract*}` and `\end{extract*}` were never typed.

```
142 \def\extract{\XTR@stfalse\XTR@extract}
143 \@namedef{extract*}{\XTR@sttrue\XTR@extract}
```

`\XTR@extract` Prepare verbatim reading and check for an optional argument.

```
144 \def\XTR@extract{%
145   \@bsphack
146   \let\do\@makeother\dospecials\catcode'\^M\active
147   \@testopt\XTR@extract\@nil
148 }
```

`\XTR@@extract` [*label*]

Process the optional label, define line processing and start reading verbatim. Do not extract when the package is not active. Use a temporary file to extract the body to in case this needs to be executed in the source document (`extract*` environment).

```
149 \def\XTR@@extract[#1]{%
Check state.
150   \ifXTR@active
151     \def\XTR@tempa{#1}%
152     \ifx\XTR@tempa\@nnil\else
153       \KV@sp@def\XTR@currentlabel{#1}%
154     \fi
155     \XTR@checkxtr{extract}{extract}%
156   \else
157     \XTR@extractfalse
158   \fi
159   \ifXTR@st\XTR@opentmp\fi
160   \let\verbatim@processline\XTR@processline@begin
161   \verbatim@start
162 }

163 \begingroup
164   \lccode'\!='\ \lccode'\=( '\ \lccode'\)= '\
165 \lowercase{\endgroup
```

`\XTR@processline@begin` This macro starts the reparsing of a line read by verbatim. It is possible to have `\begin{extractskip}` and `\end{extractskip}` on the same line.

```
166 \def\XTR@processline@begin{%
Initialize \@temptokena (used for temp file) and \verbatim@line (used for output
file). Save the original content of \verbatim@line for later use.
167   \@temptokena{}%
168   \edef\XTR@orig@line{\the\verbatim@line}%
169   \verbatim@line{}%
170   \expandafter\XTR@testbegin\XTR@orig@line!begin(extractskip)\@nil
171 }
```

`\XTR@testbegin` *<text1>* `\begin{extractskip}` *<text2>* `\@nil`
Checks whether `\begin{extractskip}` occurs.

```

172 \def\XTR@testbegin#1!begin(extractskip)#2\@nil{%
173   \temptokena\expandafter{\the\temptokena#1}%
174   \verbatim@line\expandafter{\the\verbatim@line#1}%
175   \def\XTR@tempa{#2}%

```

If *<text2>* empty, there is no `\begin{extractskip}`. Just write the content to file.

```

176   \ifx\XTR@tempa\@empty\XTR@processline@write\else\XKV@afterfi

```

Check the label.

```

177   \XTR@skiplabel#2[]\@nil

```

Should we skip the `extractskip` environment or not?

```

178   \XTR@checkxtr{skip}{extractskip}%

```

Switch to scanning for `\end{extractskip}` in the next line.

```

179   \let\verbatim@processline\XTR@processline@end

```

Remove some stuff that we added and continue scanning for `\end{extractskip}` on the current line.

```

180   \ifx\XTR@tempa\@nnil\XKV@afterelsefi
181     \XTR@t@stbegin#2\@nil
182   \else\XKV@afterfi
183     \expandafter\XTR@t@stbegin\XTR@tempa\@nil
184   \fi
185 \fi
186 }

```

`\XTR@skiplabel` *<text1>* [*<label>*] *<text2>* \@nil

This macro checks whether a label is present and sets `\XTR@currentlabel` if necessary.

```

187 \def\XTR@skiplabel#1[#2]#3\@nil{%
188   \def\XTR@tempa{#1}%
189   \def\XTR@tempb{#2}%
190   \ifx\XTR@tempa\@empty
191     \ifx\XTR@tempb\@empty
192       \let\XTR@tempa\@nnil
193     \else
194       \KV@@sp@def\XTR@currentlabel{#2}%
195       \XTR@sk@plabel#3\@nil
196     \fi
197   \else
198     \let\XTR@tempa\@nnil
199   \fi
200 }

```

`\XTR@sk@plabel` *<text>* [] \@nil

Remove extra brackets from input.

```

201 \def\XTR@sk@plabel#1[]\@nil{\def\XTR@tempa{#1}}

```

`\XTR@t@stbegin` *<text>* \begin{extractskip} \@nil

Remove the extra `\begin{extractskip}` and start scanning for `\end{extractskip}` in the current line.

```

202 \def\XTR@t@stbegin#1!begin(extractskip)\@nil{%
203   \XTR@testend#1!end(extractskip)\@nil
204 }

```

`\XTR@processline@end` Starts scanning for `\end{extractskip}` in case this was not on one line together with `\begin{extractskip}`. Comparable to `\XTR@processline@begin`.

```

205 \def\XTR@processline@end{%
206   \@temptokena{}%
207   \edef\XTR@orig@line{\the\verbatim@line}%
208   \verbatim@line{}%
209   \expandafter\XTR@testend\XTR@orig@line!end(extractskip)\@nil
210 }

```

`\XTR@testend` *<text1>\end{extractskip}<text2>\@nil*
Check whether `\end{extractskip}` occurs in the line.

```

211 \def\XTR@testend#1!end(extractskip)#2\@nil{%
212   \@temptokena\expandafter{\the\@temptokena#1}%

```

Skip material conditionally on labels or numbers.

```

213   \ifXTR@skip\else\verbatim@line\expandafter{\the\verbatim@line#1}\fi
214   \def\XTR@tempa{#2}%
215   \ifx\XTR@tempa\@empty\XTR@processline@write\else\XKV@afterfi

```

Switch to scanning for `\begin{extractskip}` in the next line.

```

216     \let\verbatim@processline\XTR@processline@begin

```

Continue scanning for `\begin{extractskip}` in the current line.

```

217     \XTR@t@stend#2\@nil
218   \fi
219 }

```

`\XTR@t@stend` *<text>\end{extractskip}\@nil*
Remove the redundant `\end{extractskip}` and continue scanning for the string `\begin{extractskip}` in this line.

```

220 \def\XTR@t@stend#1!end(extractskip)\@nil{%
221   \XTR@testbegin#1!begin(extractskip)\@nil
222 }}

```

`\XTR@processline@write` Writes the material to the appropriate file. If one of the tokens has become empty, it might be because the line was empty originally or because the parsing and removal of `\begin{extractskip}` and `\end{extractskip}` made it empty. In the latter case, do not write the empty line. In the former case, do write it.

```

223 \def\XTR@processline@write{%
224   \ifXTR@st\ifcat$\the\@temptokena$\else
225     \XTR@writetmp{\the\@temptokena}%
226   \fi\fi
227   \ifXTR@extract\ifcat$\the\verbatim@line$\else
228     \XTR@writeout{\the\verbatim@line}%
229   \fi\fi
230   \ifx\XTR@orig@line\@empty\XTR@writetmp{}\XTR@writeout{}\fi
231 }

```

`\endextract` Stop reading verbatim and if necessary execute the body of the environment after the
`\endextract*` `extract*` environment.

```

232 \def\endextract{\XTR@stfalse\XTR@endextract}
233 \@namedef{endextract*}{\XTR@sttrue\XTR@endextract}
234 \def\XTR@endextract{%
235   \@esphack

```

```

236 \ifXTR@st
237 \XTR@closetmp
238 \AfterEndEnv{\input{\jobname.xtr}}%
239 \fi
240 }

\extractskip The extractskip environment when it is not used inside an environment that is
\endextractskip redefined to be extracted. Hence this environment makes itself disappear, just as
extract*, but doesn't write to the output file. The trick with XTR@activefalse will
remain local.
241 \@namedef{extractskip}{\XTR@activefalse\@nameuse{extract*}}
242 \XTR@namelet{endextractskip}{endextract*}

\extractline This macro extracts all text after the macro and at the same line. First we check for an
optional star.
243 \def\extractline{%
244 \XKV@ifstar{\XTR@sttrue\XTR@extractline}%
245 {\XTR@stfalse\XTR@extractline}%
246 }

\XTR@extractline Start the group and reset all catcodes for verbatim reading.
247 \def\XTR@extractline{%
248 \begingroup
249 \let\do\@makeother\dospecials\catcode'\~M\active

Test for an optional argument. Note that, due to reset catcodes, macros won't work
in the optional argument, but that is not a real restriction, while it saves some tokens
and memory. If we want to allow for macro arguments, we need an extra macro for the
check.
250 \@testopt\XTR@@extractline\@nil
251 }

\XTR@@extractline [label][text](eof)
The workhorse that reads input until the end of the line. Use the \lowercase trick for
the definition.
252 \begingroup
253 \catcode'\~= \active\lccode'\~= '\~M
254 \lowercase{\endgroup
255 \def\XTR@@extractline[#1]#2~{%

Check state.
256 \ifXTR@active
257 \def\XTR@tempa{#1}%
258 \ifx\XTR@tempa\@nnil\else
259 \KV@@sp@def\XTR@currentlabel{#1}%
260 \fi
261 \XTR@checkxtr{extract}{line}%
262 \else
263 \XTR@extractfalse
264 \fi
265 \ifXTR@extract\XTR@writeout{#2}\fi

If we need to execute the line, the catcodes are wrong, so write it to the temporary file
and insert it again when the catcodes are reset by \endgroup.

```

```

266     \ifXTR@st\XTR@opentmp\XTR@writetmp{#2}\XTR@closetmp\fi
267 \endgroup
Insert original content.
268     \ifXTR@st
269         \input{\jobname.xtr}%
270     \fi
271 }%
272 }

```

Only define the following macros when the package is active. This branch also performs redefinitions of the macros and environments that should be extracted.

```
273 \ifXTR@active
```

Start writing the target file.

```
274 \immediate\openout\XTR@out\XTR@file\relax
```

Write header to the target file.

```
275 \ifXTR@header
```

Compute the time.

```

276 \@tempcnta\time
277 \divide\@tempcnta 60
278 \edef\XTR@tempb{%
279     \the\year/\ifnum\the\month<10 0\fi\the\month/%
280     \ifnum\the\day<10 0\fi\the\day,\the\@tempcnta:%
281 }
282 \multiply\@tempcnta 60
283 \@tempcntb\time
284 \advance\@tempcntb-\@tempcnta
285 \ifnum\@tempcntb<10
286     \xdef\XTR@tempb{\XTR@tempb0\the\@tempcntb}
287 \else
288     \xdef\XTR@tempb{\XTR@tempb\the\@tempcntb}
289 \fi
290 \begingroup

```

Save the % character.

```

291     \catcode'\%=12
292     \gdef\XTR@tempa{%%\space}
293 \endgroup

```

Write all information to the target file.

```

294 \XTR@writeout{\XTR@tempa}
295 \filename@parse\XTR@file
296 \ifx\filename@ext\relax\def\filename@ext{tex}\fi
297 \XTR@writeout{%
298     \XTR@tempa This is file, '\filename@base.\filename@ext',%
299 }
300 \XTR@writeout{%
301     \XTR@tempa generated with the extract package.^^J\XTR@tempa
302 }
303 \XTR@writeout{\XTR@tempa Generated on : \space\XTR@tempb}
304 \filename@parse\jobname
305 \ifx\filename@ext\relax\def\filename@ext{tex}\fi
306 \XTR@writeout{%

```

```

307 \XTR@tempa From source \space: \space\filename@base.\filename@ext
308 }
309 \XTR@writeout{%
310 \XTR@tempa Using options: \space\csname opt@extract.sty\endcsname
311 }
312 \XTR@writeout{\XTR@tempa}
313 \fi

```

If requested, reconstruct the `\documentclass` command using information from `xkeyval`.

```

314 \ifXTR@copydocumentclass
315 \def\XTR@tempa#1.cls\@nil{\def\XTR@tempa{#1}}
316 \expandafter\XTR@tempa\XKV@documentclass\@nil
317 \ifx\XKV@classoptionslist\@empty
318 \XTR@writeout{\string\documentclass{\XTR@tempa}}
319 \else
320 \@temptokena\expandafter{\XKV@classoptionslist}%
321 \XTR@writeout{\string\documentclass[\the\@temptokena]{\XTR@tempa}}
322 \fi
323 \fi

```

Perform redefinitions at the beginning of the document.

```

324 \AtBeginDocument{%
325 \ifXTR@handles
326 \XTR@writeout{%
327 \XTR@writeout{\string\begin{document}}%
328 \fi

```

Redefine environments.

```

329 \XKV@ifundefined{XTR@envs}{\{%
330 \XKV@for@o\XTR@envs\XTR@tempa{%

```

Check whether the environment is defined.

```

331 \XKV@ifundefined\XTR@tempa{%
332 \XTR@err{%
333 environment ‘\XTR@tempa’ not defined; extraction canceled%
334 }%
335 }{%

```

Backup the beginning of the environment.

```

336 \XTR@namelet{XTR\XTR@tempa}{\XTR@tempa}%

```

Redefine the beginning of the environment. This uses `verbatim` internally.

```

337 \@namedef{\XTR@tempa\expandafter}\expandafter{\expandafter
338 \def\expandafter\XTR@tempa\expandafter{\XTR@tempa}%

```

Check whether the current environment should be extracted. Note that `\XTR@tempa` contains the current environment name.

```

339 \XTR@checkxtr{extract}\XTR@tempa
340 \ifXTR@extract

```

If extraction is required, write to the target file and to a temporary file for inclusion afterwards.

```

341 \XTR@writeout{\XTR@opentmp
342 \@bsphack
343 \let\do\@makeother\dospecials\catcode‘\~M\active

```

```

\verbatim@processline Process macro for verbatim.
344         \def\verbatim@processline{%
\verbatim@processline is redefined here since the first line is treated specially, see
below.
345         \let\verbatim@processline\XTR@processline@begin
Write the content to the files.
346         \XTR@writeout{%
347         \string\begin{\XTR@tempa}\the\verbatim@line
348         }%
349         \XTR@writetmp{%
350         \string\begin{XTR\XTR@tempa}\the\verbatim@line
351         }%
352         }%

353         \XTR@sttrue\let\XTR@tempb\verbatim@
354         \else

Else, execute the backup of the current environment.
355         \edef\XTR@tempb{\noexpand\begin{XTR\XTR@tempa}}%
356         \fi
357         \XTR@tempb
358         }%

Backup the end of the environment.
359         \XTR@namelet{endXTR\XTR@tempa}{end\XTR@tempa}%

Redefine the end of the environment.
360         \@namedef{end\XTR@tempa\expandafter}\expandafter{\expandafter
361         \def\expandafter\XTR@tempa\expandafter{\XTR@tempa}%
362         \ifXTR@extract
363         \@esphack

Finalize writing and add the \input to the hook at the end of the current environment.
364         \XTR@writeout{\string\end{\XTR@tempa}}%
365         \XTR@writetmp{\string\end{XTR\XTR@tempa}}%
366         \XTR@closetmp
367         \AfterEndEnv{\input{\jobname.xtr}}%
368         \else

If not extracting, execute the backup of the end of the environment.
369         \edef\XTR@tempa{\noexpand\end{XTR\XTR@tempa}}%
370         \expandafter\XTR@tempa
371         \fi
372         }%
373         }%
374         }%
375         }%

Redefine commands using the arguments.
376         \XKV@ifundefined{XTR@cmdsargs}{\}{%

Once backup the current definitions.
377         \let\XTR@sect\@sect
378         \let\XTR@chapter\@chapter
379         \def\XTR@tempb{chapter}%

```

Redefine a list of macros to write themselves to the target file. Chapters and section are treated differently since they are constructed differently. `\chapter*` will not extract itself since this gives technical difficulties due to the fact that this macro is reused at several places inside other macros, taking none-character input in its argument.

```

380 \XKV@for@o\XTR@cmdsargs\XTR@tempa{%
381 \XKV@ifundefined\XTR@tempa{%
382 \XTR@err{command ‘\@backslashchar\XTR@tempa’ not defined;
383 extraction canceled%
384 }%
385 }{%

```

Check whether allowed or not.

```

386 \@expandtwoargs\in@{,\XTR@tempa,}%
387 {,chapter,section,subsection,subsubsection,}%
388 \ifin@
389 \ifx\XTR@tempa\XTR@tempb
390 \def\@chapter[#1]#2{%

```

Check whether to extract this chapter or not.

```

391 \XTR@checkxtr{extract}{chapter}%
392 \ifXTR@extract
393 \XTR@writeout{}%
394 \def\XTR@tempa{#1}%
395 \def\XTR@tempb{#2}%
396 \ifx\XTR@tempa\XTR@tempb
397 \@temptokena{#2}%
398 \else
399 \@temptokena{[#1]{#2}}%
400 \fi

```

Write to file.

```

401 \XTR@writeout{\string\chapter\the\@temptokena}%
402 \fi

```

Typeset the chapter.

```

403 \XTR@chapter[#1]{#2}%
404 }%
405 \else

```

We do a similar thing for sections created with `\@sect`.

```

406 \def\@sect#1#2#3#4#5#6[#7]#8{%
407 \@expandtwoargs\in@{,#1,}{,\XTR@cmdsargs,}%
408 \ifin@
409 \XTR@checkxtr{extract}{#1}%
410 \ifXTR@extract
411 \XTR@writeout{}%
412 \def\XTR@tempa{#7}%
413 \def\XTR@tempb{#8}%
414 \ifx\XTR@tempa\XTR@tempb
415 \@temptokena{#8}%
416 \else
417 \@temptokena{[#7]{#8}}%
418 \fi
419 \XTR@writeout{\expandafter
420 \string\csname#1\endcsname\the\@temptokena}%
421 \fi

```

```

422         \fi
423         \XTR@sect{#1}{#2}{#3}{#4}{#5}{#6}[#7]{#8}%
424     }%
425     \fi
426     \else
427         \XTR@err{unsupported command ‘\XTR@tempa’;
428             try the ‘extract-cmdline option’}%
429     \fi
430 }%
431 }%
432 }%
433 \XKV@ifundefined{XTR@cmdslne}{%}{%

```

Redefine a list of commands to write themselves and the text on the same line to the target file. This works similar to `\extractline`.

```

434     \XKV@for@o\XTR@cmdslne\XTR@tempa{%
435         \XKV@ifundefined\XTR@tempa{%

```

Check whether the command is defined.

```

436         \XTR@err{command ‘\@backslashchar\XTR@tempa’ not defined;
437             extraction canceled}%
438     }{%

```

Check whether allowed or not.

```

439         \@expandtwoargs\in@{,\XTR@tempa,}%
440         {,chapter,section,subsection,subsubsection,}%
441     \ifin@
442         \XTR@err{%
443             use the ‘extract-cmd’ option for command ‘\XTR@tempa’%
444         }%
445     \else

```

Backup the command.

```

446         \XTR@namelet{XTR\XTR@tempa}{\XTR@tempa}%

```

Redefine the command. Note that, inside the definition of the command, `\XTR@tempa` contains the command name.

```

447         \@namedef{\XTR@tempa\expandafter}\expandafter{\expandafter
448             \def\expandafter\XTR@tempa\expandafter{\XTR@tempa}%

```

Check whether this command should be extracted.

```

449         \XTR@checkxtr{extract}\XTR@tempa
450         \begingroup
451             \let\do\@makeother\dospecials\catcode‘\^^M\active
452             \XTR@extractcmdline
453         }%
454     \fi
455 }%
456 }%
457 \begingroup
458     \catcode‘\~= \active\lccode‘\~=‘\^^M

```

`\XTR@extractcmdline` *<text>**<eol>*

Workhorse for the command line extraction method. This macros reads until the next end of line and saves the content in `\XTR@tempb`.

```

459     \lowercase{\endgroup

```

```

460     \def\XTR@extractcmdline#1~{\verbatim@line{#1}\XTR@extractcmdline}%
461     }%

```

`\XTR@extractcmdline` Finalize the operation with the content of the current line. We write it to a target file and to a temporary file for execution in the current document. Note that `\XTR@tempa` still contains the current command name.

```

462     \def\XTR@extractcmdline{%
463         \XTR@writeout{}%
464         \XTR@writeout{\expandafter\string\csname\XTR@tempa
465             \endcsname\the\verbatim@line
466         }%
467         \XTR@opentmp
468         \XTR@writetmp{\expandafter\string\csname XTR\XTR@tempa
469             \endcsname\the\verbatim@line
470         }%
471         \XTR@closetmp
472     \endgroup
473     \input{\jobname.xtr}%
474 }%
475 }%
476 }

```

Finalize writing the target file.

```

477 \AtEndDocument{%
478     \ifXTR@handles
479         \XTR@writeout{}%
480         \XTR@writeout{\string\end{document}}}%
481 \fi
482 \immediate\closeout\XTR@out
483 }
484 \fi
485 \extract

```

References

- [1] Hendri Adriaens. `xkeyval` package. CTAN:/macros/latex/contrib/xkeyval.
- [2] Donald Arseneau. `selectp` package, v0.9. CTAN:/macros/latex/contrib/misc, 1992/09/25.
- [3] Donald Arseneau. `optional` package, v2.2. CTAN:/macros/latex/contrib/misc, 2001/09.
- [4] Stephen Bellantoni. `version` package. CTAN:/macros/latex/contrib/misc, 1990.
- [5] Johannes Braams, David Carlisle, Alan Jeffrey, Leslie Lamport, Frank Mittelbach, Chris Rowley, and Rainer Schöpf. The \LaTeX 2_ε sources. CTAN:/macros/latex/base, 2003.
- [6] Victor Eijkhout. `comment` package, v3.6. CTAN:/macros/latex/contrib/comment, 1999/10.

- [7] Carsten Heinz. listings package, v1.3. CTAN:/macros/latex/contrib/listings, 2004/09/07.
- [8] Uwe Lück. versions package, v0.51. CTAN:/macros/latex/contrib/versions, 2003/10/15.
- [9] Dan Luecking. excludeonly package, v1.0. CTAN:/macros/latex/contrib/misc, 2003/03/14.
- [10] Andreas Matthias. pdfpages package, v0.3e. CTAN:/macros/latex/contrib/pdfpages, 2004/01/31.
- [11] Heiko Oberdiek. pagesel package, v1.1. CTAN:/macros/latex/contrib/oberdiek, 1999/04/13.
- [12] Pablo A. Straub. askinclude package, v1.2e. CTAN:/macros/latex/contrib/misc, 1994/11/11.
- [13] Rainer Schöpf. verbatim package, v1.5q. CTAN:/macros/latex/required/tools, 2003/08/22.
- [14] Timothy Van Zandt. xcomment package, v1.2. CTAN:/macros/latex/contrib/seminar, 1993/02/12.
- [15] Timothy Van Zandt. fancyvrb package, v2.6. CTAN:/macros/latex/contrib/fancyvrb, 1998/07/17.

Acknowledgements

Thanks go to David Carlisle for proposing the technique implemented in this package to solve my initial problem.⁷ Thanks also go to Michael Kohlhase and Johannes Luber for reporting bugs and providing ideas for extensions of the package.

Version history

v1.0	<i>(2004/12/19)</i>
General: Initial release	1
v1.1	<i>(2005/01/01)</i>
General: Added conditional extraction using labels	1
Revised documentation	1
\verbatim@processline: Made to write content on same line as environment heading if so in source file	22
v1.2	<i>(2005/01/06)</i>
General: Fixed \AfterEndEnv hook for nested environments	1
Increased efficiency of the package	1
Simplified optional argument check for extract environment	1
v1.3	<i>(2005/01/12)</i>
General: Added \extractline	1
Added document-handles option	1
Added examples and documentation	1
Added line extraction method for commands	1

⁷See section 1.

Removed some input checks for more flexibility	1
Simplified package	1
Updated license information	1
\extractionlabel: Was \extractlabel	15
v1.4	<i>(2005/02/06)</i>
General: Added extractskip environment	1
Adjusted options section for xkeyval (\XKV@tkey not defined anymore)	1
Allow for any filename	1
Changed temp extension to .xtr	1
\XTR@extractcmdline: Changed temp macro to token register	24
v1.5	<i>(2005/02/08)</i>
General: Avoid using counters for conditional extraction with numbers	1
Solved bug in options section	1
v1.6	<i>(2005/02/14)</i>
General: Added optional header for extracted files	1
Improved options section	1
v1.7	<i>(2005/03/31)</i>
General: Revised documentation	1
Solved small bug with 'no-header' option	1
v1.8	<i>(2005/05/07)</i>
General: Added copydocumentclass option	1
Changed name of option document-handles to handles	1
Changed name of option no-header to header	1
Revised documentation and examples	1
\XTR@processline@write: Solved bug of not copying empty lines to the target file	18
v1.9	<i>(2019/09/17)</i>
\begin: Changed definition to robust version	13
\end: Changed definition to robust version	14

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.

Symbols	
\%	291
-labels (option)	5, 12
-nrs (option)	4, 12
\@chapter	378, 390
\@envdepth	<u>68</u> , 77, 78, 84, 85, 90
\@sect	377, 406
\{	164
\}	164
\~	146, 249, 253, 343, 451, 458
\~	253, 458
A	
active (option)	2, 12
\AfterEndEnv	<u>88</u> , 238, 367
B	
\begin	<u>69</u>
C	
copydocumentclass (option)	9, 12
E	
\end	<u>80</u>
\endextract	<u>232</u>
\endextract*	<u>232</u>
\endextractskip	<u>241</u>
environments:	
extract	6, 7
extract*	6, 7
extractskip	8
\ExecuteOptionsX	46
\extract	<u>142</u>
extract (environment)	6, 7
\extract*	<u>142</u>
extract* (environment)	6, 7
extract-cmd (option)	3, 12

extract-cmdline (option) 3, 12
 extract-env (option) 2, 12
 extract-labels (option) 8, 12
 extract-nrs (option) 8, 12
 \extractionlabel 5, 140
 \extractline 7, 243
 \extractline* 7
 \extractskip 241
 extractskip (environment) 8
 extractskip-labels (option) 9, 12
 extractskip-nrs (option) 9, 12

G

generate (option) 2, 12

H

handles (option) 9, 12
 header (option) 9, 12

I

\ifXTR@active 52, 150, 256, 273
 \ifXTR@copydocumentclass 314
 \ifXTR@extract
 11, 227, 265, 340, 362, 392, 410
 \ifXTR@handles 325, 478
 \ifXTR@header 275
 \ifXTR@skip 10, 213
 \ifXTR@st 9, 159, 224, 236, 266, 268

L

line-labels (option) 8, 12
 line-nrs (option) 8, 12

O

options:
 -labels 5, 12
 -nrs 4, 12
 active 2, 12
 copydocumentclass 9, 12
 extract-cmd 3, 12
 extract-cmdline 3, 12
 extract-env 2, 12
 extract-labels 8, 12
 extract-nrs 8, 12
 extractskip-labels 9, 12
 extractskip-nrs 9, 12
 generate 2, 12
 handles 9, 12
 header 9, 12
 line-labels 8, 12
 line-nrs 8, 12

V

\verbatim@ 353

\verbatim@line 168, 169, 174, 207, 208,
 213, 227, 228, 347, 350, 460, 465, 469
 \verbatim@processline 160, 179, 216, 344
 \verbatim@start 161

X

\XTR@extract 147, 149
 \XTR@extractcmdline 460, 462
 \XTR@extractline 250, 252
 \XTR@ch@ck@tr 123, 125
 \XTR@ch@ckxtr 116, 122
 \XTR@chapter 378, 403
 \XTR@checkxtr
 . 92, 155, 178, 261, 339, 391, 409, 449
 \XTR@closetmp .. 48, 61, 237, 266, 366, 471
 \XTR@cmdsargs 26, 380, 407
 \XTR@cmdslines 30, 434
 \XTR@currentlabel
 107, 108, 120, 140, 153, 194, 259
 \XTR@endextract 232-234
 \XTR@envs 22, 330
 \XTR@err
 12, 45, 55, 65, 332, 382, 427, 436, 442
 \XTR@extract 142, 143, 144
 \XTR@extractcmdline 452, 459
 \XTR@extractfalse 157, 263
 \XTR@extractline 244, 245, 247
 \XTR@file 20, 63, 274, 295
 \XTR@maketrue . 96, 110, 127, 131, 133, 136
 \XTR@namelet 13, 36, 94, 242, 336, 359, 446
 \XTR@opentmp ... 48, 57, 159, 266, 341, 467
 \XTR@orig@line .. 168, 170, 207, 209, 230
 \XTR@out 7, 51, 274, 482
 \XTR@processline@begin
 160, 166, 216, 345
 \XTR@processline@end 179, 205
 \XTR@processline@write .. 176, 215, 223
 \XTR@sect 377, 423
 \XTR@sk@plabel 195, 201
 \XTR@skiplabel 177, 187
 \XTR@t@stbegin 181, 183, 202
 \XTR@t@stend 217, 220
 \XTR@ttestbegin 170, 172, 221
 \XTR@ttestend 203, 209, 211
 \XTR@tmp 8, 48-50
 \XTR@writeout 48, 228, 230, 265,
 294, 297, 300, 303, 306, 309, 312,
 318, 321, 326, 327, 341, 346, 364,
 393, 401, 411, 419, 463, 464, 479, 480
 \XTR@writetmp
 .. 48, 58, 225, 230, 266, 349, 365, 468