The **newfile** package[*]

Author: Peter Wilson, Herries Press
Maintainer: Will Robertson
`will dot robertson at latex-project dot org`

2004/05/10

Abstract

The newfile package provides convenient user level commands for reading and writing new files during a LaTeX run.

Contents

1 Introduction

TeX has a maximum of 16 input and 16 output *streams* for reading and writing files. The newfile package provides means of associating several different files with a particular stream.

The newfile package requires the verbatim package [SRR99], which is one of the required packages in a LaTeX distribution.

This manual is typeset according to the conventions of the LaTeX DOCSTRIP utility which enables the automatic extraction of the LaTeX macro source files [GMS94].

Section 2 describes the usage of the package. Commented source code for the package is in Section 4.

---

[*] This file (`newfile.dtx`) has version number v1.0b, last revised 2004/05/10.

2 The newfile package

`\newoutputstream` `\newinputstream` The command `\newoutputstream{`⟨*stream*⟩`}` creates a new stream called ⟨*stream*⟩ for writing out text and commands. The ⟨*stream*⟩ should be just alphabetic characters with no spaces; for example `myout`. The command `\newinputstream{`⟨*stream*⟩`}` creates a new stream for reading from a file. The ⟨*stream*⟩ names must be unique — you cannot use the same name for both an input and an output stream.

If you try and create too many streams, TeX will tell you via an error message.

These two commands also attempt to create two other new commands, called respectively `\atstreamopen<stream>` and `\atstreamclose<stream>`. If these commands already exist then they are left untouched, otherwise they are created (like using `\providecommand{\atstreamopen<stream>}{}`). For example if if you have used `mystr` as the name of a stream (either input or output), then the macros `\atstreamopenmystr` and `\atstreamclosemystr` are defined; by default they do nothing, but you can `\renewcommand` them to do something.

`\openoutputfile` The macro `\openoutputfile{`⟨*filename*⟩`}{`⟨*stream*⟩`}` opens the file called ⟨*filename*⟩ and the output stream ⟨*stream*⟩. It then attaches the file to the stream for writing and calls the macro `\atstreamopen<stream>`. Any pre-existing contents of ⟨*filname*⟩ are deleted.

`\closeoutputstream` The macro `\closeoutputstream{`⟨*stream*⟩`}` calls the macro `\atstreamclose<stream>`, closes the output stream ⟨*stream*⟩ and closes whatever file is currently attached to ⟨*stream*⟩. It then detaches the file from the stream.

`writeverbatim` The `writeverbatim` environment takes one argument, the name of an output stream, which must be open. The contents of the environment are written out verbatim to the file currently attached to the stream.

`\addtostream` The command `\addtostream{`⟨*stream*⟩`}{`⟨*text*⟩`}` writes ⟨*text*⟩ to the file currently attached to the output stream ⟨*stream*⟩, which must be open. Any commands within ⟨*text*⟩ will be processed before being written. To prevent this, put `\protect` before any command that you do not want expanding.

`\openinputfile` The macro `\openinputfile{`⟨*filename*⟩`}{`⟨*stream*⟩`}` opens the file called ⟨*filename*⟩ and the input stream ⟨*stream*⟩. It then attaches the file to the stream for reading and calls the macro `\atstreamopen<stream>`. It is an error if ⟨*filename*⟩ can not be found.

`\closeinputstream` The macro `\closeinputstream{`⟨*stream*⟩`}` calls the macro `\atstreamclose<stream>`, closes the input stream ⟨*stream*⟩ and closes whatever file is currently attached to ⟨*stream*⟩. It then detaches the file from the stream.

`\readstream` The macro `\readstream{`⟨*stream*⟩`}` reads the contents of the file that is currently associated with the input stream ⟨*stream*⟩. This provides the same functionality as `\input{`⟨*filename*⟩`}` does.

`\readaline` The macro `\readaline{`⟨*stream*⟩`}` reads what TeX considers to be one line from the file that is currently associated with the input stream ⟨*stream*⟩. Multiple lines can be read by calling `\readaline` multiple times. A warning is issued if there are no more lines to be read (i.e., the end of the file has been reached).

`\readverbatim` The macro `\readverbatim{`⟨*stream*⟩`}` reads the contents of the file that is currently associated with the input stream ⟨*stream*⟩ as `verbatim` text. This provides the same functionality as the verbatim package's `\verbatiminput{⟨filename⟩}` command does.

`\streamvfont`

Text read in verbatim is typeset using the font specified by `\streamvfont{⟨font⟩}`. The default is `\streamvfont{\normalfont\ttfamily}`. To typeset in a smaller font, try for example, `\streamvfont{\small\ttfamily}`.

`\numbervstream`
`\marginnumbervstream`
`\streamvnumfont`

The declaration `\numbervstream` causes `\readverbatim` to number each line it reads. The declaration `\marginnumbervstream` is similar to `\numbervstream` except it puts the numbers in the margin. The `streamvline` counter is used for the line numbering. Both the numbering declarations (re)set it to zero. The numbers are typeset in the font specified by `\streamvnumfont{⟨font⟩}`. The default is `\streamvnumfont{\footnotesize}`.

`\plainvstream`

The declaration `\plainvstream` stops any numbering. The default is `\plainvstream`.

`comment`

The `comment` environment, which is part of the verbatim package, throws away everything inside the environment.

The verbatim package [SRR99] provides some other facilities, apart from the `comment` environment, and its documentation explains them.

3 Examples

3.1 Output stream

This example is inspired by the endfloat package [MG95] which provides a more sophisticated approach than shown below. The example demonstrates the use of an output stream.

Suppose that in an article class document you want all the figures to be collected at the end, but to specify them at the appropriate places in the body of the text.

⟨*ex⟩

```
\documentclass{article}
\usepackage{newfile}
...
```

Create a new output stream called `figs` and open it to write to file `figures.tex`.

```
\newoutputstream{figs}
\openoutputfile{figures.out}{figs}
...
```

Write out verbatim the first `figure` environment to the `figs` stream.

```
\begin{writeverbatim}{figs}
\begin{figure}
...
\end{figure}
\end{writeverbatim}
...
```

After the last `figure` is written to `figs`, close the `figs` output stream, which also closes the `figures.out` file.

```
\begin{writeverbatim}{figs}
```

```
\begin{figure}
...
\end{figure}
\end{writeverbatim}
\closeoutputstream{figs}
...
```

At the end `\input figures.out` to typeset the figures.

```
\input{figures.out}
\end{document}
```

3.2 An output and input stream

This example is the kind of thing that the answers package [Pif96] does rather better. The example illustrates the use of an output and an input stream.

Suppose you are writing a document that includes questions and answers and you want all the answers at the end. It is most convenient if you can write the answer to each question as it is posed, and then only print them at the end.

```
\documentclass{article}
\usepackage{newfile}
...
```

Create new output and input streams called `ans` and `ansin`. Just to demonstrate the use of the `\atstreamopen...` command, redefine `\atstreamopenansin` so that it will start a section called ‘Answers to all the questions’. Then open the `ans` stream to write to a file called `answers.all`.

```
\newoutputstream{ans}
\newinputstream{ansin}
\renewcommand{\atstreamopenansin}{\section{Answers to all the questions}}
\openoutputfile{answers.all}{ans}
...
```

For each question, write out the answer verbatim to the `ans` stream.

```
This is a question.
\begin{writeverbatim}{ans}
This is the answer to a question.
\end{writeverbatim}
...
```

At the end, close the `ans` stream and open the `ansin` stream to read from the file `answers.all`. Opening the stream will also start the new ‘Answers to all the questions’ section (from `\atstreamopenansin`). Then read from the `ansin` stream.

```
\closeoutputstream{ans}
\openinputfile{answers.all}{ansin}
\readstream{ansin}
```

Finally, close the `ansin` stream, and we are done.

```
\closeinputstream{ansin}
\end{document}
```

If desired, instead of inserting the answers at the end of the question document, `answers.all` could have been `\input` into a seperate answer document.

Along the same lines as above, perhaps the original document is a book class, with questions (and answers) at the end of each chapter. A seperate answer file could be produced for each chapter, like:

```
...
\newoutputstream{ansout}
\newinputstream{ansin}
...
\chapter{First chapter} % chapter 1
\openoutputfile{\jobname\thechapter.ans}{ansout}
```

The `\jobname` is the name of the main LaTeX document file, without the `.tex` extension, so if the name of the LaTeX source file is `mybook.tex` the above line creates a file called `mybook1.ans`.

```
...
\begin{writeverbatim}{ansout}
An answer
\end{writeverbatim}
...
\closeoutputstream{ansout}
\chapter{Another chapter} % chapter N
\openoutputfile{\jobname\thechapter.ans}{ansout}
...
\closeoutputstream{ansout}

\chapter{Answers}

\section{Chapter 1}
\openinputstream{\jobname1.ans}{ansin}
\readstream{ansin}
\closestream{ansin}
...
\section{Chapter N}
\openinputstream{\jobnameN.ans}{ansin}
\readstream{ansin}
\closestream{ansin}
...
```

The above example shows how you can associate different files with a single stream.

3.3 Multiple streams

This is a more complex example, again inspired by the endfloat package. In a book class document you want all the figures and tables to be collected at the end, but to specify them at the appropriate places in the body of the text.

```
\documentclass{book}
\usepackage{newfile}
```

```
...
```

Create and open two new output streams, one for figures (`figs`) and the other (`tabs`) for tables. For demonstration purposes, also create a new input stream called figtab.

```
\newoutputstream{figs}
\openoutputfile{figures.out}{figs}
\newoutputstream{tabs}
\openoutputfile{tables.out}{tabs}
\newinputstream{figtab}
...
```

There is a slight difficulty with this example, as in the book class, figures and table numbers start anew with each chapter and the chapter number is preprended to the sequence number. We can make a start on solving this by creating a new pseudo chapter number and changing the default definitions of the figure and table numbers (which are `\renewcommand{\thefigure}{\thechapter.\arabic{figure}}` and similarly for tables).

```
\newcounter{pseudochapter}
\renewcommand{\thepseudochapter}{\arabic{pseudochapter}}
\renewcommand{\thefigure}{\thepseudochapter.\arabic{figure}}
\renewcommand{\thetable}{\thepseudochapter.\arabic{table}}
...
```

Do the usual things at the start of the document.

```
\begin{document}
\maketitle
\tableofcontents
\listoffigures
..
```

At the start of each chapter we need to set the `pseudochapter` counter to the chapter number, and write this to the output streams so that there is a record of which chapters the figures and tables came from Normally, each chapter resets the figure and table numbers, but as these will now be at the end, we have to fake the start of chapters in the output files. We can do all of this by the `\addtostream` macro.

```
\chapter{First}
```

The next bit of code will result in

```
\setcounter{pseudochapter}{N}
```

where `N` is the number of this chapter, appearing in the output files.

```
\addtostream{figs}{\protect\setcounter{pseudochapter}{\thechapter}}
\addtostream{tabs}{\protect\setcounter{pseudochapter}{\thechapter}}
```

The next bit of code results in the following two lines in the output files:

```
\refstepcounter{chapter}
\addtocounter{chapter}{-1}
```

The first of these has the effect of resetting the figure and table counters by increasing the chapter counter by one. The second line resets the chapter counter back to its original value.

```
\addtostream{figs}{\protect\refstepcounter{chapter}}
\addtostream{figs}{\protect\addtocounter{chapter}{-1}}
\addtostream{tabs}{\protect\refstepcounter{chapter}}
\addtostream{tabs}{\protect\addtocounter{chapter}{-1}}
```

Write out verbatim each figure to the `figs` stream.

```
\begin{writeverbatim}{figs}
\begin{figure}
...
\end{figure}
\end{writeverbatim}
...
```

and write each table verbatim to the `tabs` stream.

```
\begin{writeverbatim}{tabs}
\begin{table}
...
\end{table}
\end{writeverbatim}
...
```

When we have done all the chapters, figures and tables, close the two output streams.

```
\closeoutputstream{figs}
\closeoutputstream{tabs}
```

Then if we want the figures to be typeset before the tables, open the `figtab` stream to read from `figures.out`, read the stream and close it.

```
\openinputfile{figures.out}{figtab}
\readstream{figtab}
\closestream{figtab}
```

We can use the same `figtab` stream for reading the tables.

```
\openinputfile{tables.out}{figtab}
\readstream{figtab}
\closestream{figtab}
\end{document}
```

⟨/ex⟩

Note that the endfloat package [MG95] will produce the same effect as the previous example, but much simply.

4 The package code

This package can only be used with LaTeX2e, and requires the verbatim package [SRR99], which is one of the required packages for a LaTeX distribution.

⟨*outin⟩

```
\NeedsTeXFormat{LaTeX2e}[1996/06/01]
\ProvidesPackage{newfile}[2009/09/03 v1.0c Output and input files]
\RequirePackage{verbatim}
```

To try and avoid name clashes with other packages, each internal macro in this package includes the character string ‘`stre@m`’.

`\newoutputstream`

`\newoutputstream{`⟨*stream*⟩`}` creates a new output stream called ⟨*stream*⟩. Different files may be associated with the ⟨*stream*⟩. Note that TeX permits no more than 16 output streams.

```
\newcommand{\newoutputstream}[1]{%
  \@ifundefined{#1outstre@m}%
    {\expandafter\newwrite\csname #1outstre@m\endcsname
     \csname newif\expandafter\endcsname
       \csname ifstre@m#1open\endcsname
     \global\csname stre@m#1openfalse\endcsname
     \expandafter\ifx\csname atstreamopen#1\endcsname\relax
       \global\@namedef{atstreamopen#1}{}%
     \fi
     \expandafter\ifx\csname atstreamclose#1\endcsname\relax
       \global\@namedef{atstreamclose#1}{}%
     \fi
    }%
    {\PackageError{newfile}{Output stream #1 is already defined}{\@ehc}}}

```

`\newinputstream`

`\newinputstream{`⟨*stream*⟩`}` creates a new input stream called ⟨*stream*⟩. Different files may be associated with the ⟨*stream*⟩. Note that TeX permits no more than 16 input streams.

```
\newcommand{\newinputstream}[1]{%
  \@ifundefined{#1instre@m}%
    {\expandafter\newread\csname #1instre@m\endcsname
     \csname newif\expandafter\endcsname
       \csname ifstre@m#1open\endcsname
     \global\csname stre@m#1openfalse\endcsname
     \expandafter\ifx\csname atstreamopen#1\endcsname\relax
       \global\@namedef{atstreamopen#1}{}%
     \fi
     \expandafter\ifx\csname atstreamclose#1\endcsname\relax
       \global\@namedef{atstreamclose#1}{}%
     \fi
    }%
    {\PackageError{newfile}{Input stream #1 is already defined}{\@ehc}}}

```

Some checking macros will be useful as some of the checks occur in multiple places.

`\@ifstre@mopen`

`\@ifstre@mopen{`⟨*stream*⟩`}{`⟨*TRUE code*⟩`}{`⟨*FALSE code*⟩`}` checks if stream ⟨*stream*⟩ is currently open.

```
\newcommand{\@ifstre@mopen}[3]{%
  \csname ifstre@m#1open\endcsname#2\else#3\fi}
```

`\instre@mandopen`

`\instre@mandopen{`⟨*stream*⟩`}{`⟨*TRUE code*⟩`}` checks if ⟨*stream*⟩ is an input stream and is open. If so, it executes ⟨*TRUE code*⟩.

```
\newcommand{\instre@mandopen}[2]{%
  \@ifundefined{#1instre@m}{%
    \PackageError{newfile}{#1\space is not an input stream}{\@ehc}}%
  {\@ifstre@mopen{#1}{#2}{%
    \PackageError{newfile}{Input stream #1\space is not open}{\@ehc}}}}

```

`\instre@mandclosed`

`\instre@mandclosed{`⟨*stream*⟩`}{`⟨*TRUE code*⟩`}` checks if ⟨*stream*⟩ is an input stream and is closed (not open). If so, it executes ⟨*TRUE code*⟩.

```
\newcommand{\instre@mandclosed}[2]{%
  \@ifundefined{#1instre@m}{%
    \PackageError{newfile}{#1\space is not an input stream}{\@ehc}}%
  {\@ifstre@mopen{#1}{%
    \PackageError{newfile}{Input stream #1\space is open}{\@ehc}}{#2}}}

```

`\outstre@mandopen`

`\outstre@mandopen{`⟨*stream*⟩`}{`⟨*TRUE code*⟩`}` checks if ⟨*stream*⟩ is an output stream and is open. If so, it executes ⟨*TRUE code*⟩.

```
\newcommand{\outstre@mandopen}[2]{%
  \@ifundefined{#1outstre@m}{%
    \PackageError{newfile}{#1\space is not an output stream}{\@ehc}}%
  {\@ifstre@mopen{#1}{#2}{%
    \PackageError{newfile}{Output stream #1\space is not open}{\@ehc}}}}

```

`\outstre@mandclosed`

`\outtre@mandclosed{`⟨*stream*⟩`}{`⟨*TRUE code*⟩`}` checks if ⟨*stream*⟩ is an output stream and is closed (not open). If so, it executes ⟨*TRUE code*⟩.

```
\newcommand{\outstre@mandclosed}[2]{%
  \@ifundefined{#1outstre@m}{%
    \PackageError{newfile}{#1\space is not an output stream}{\@ehc}}%
  {\@ifstre@mopen{#1}{%
    \PackageError{newfile}{Output stream #1\space is open}{\@ehc}}{#2}}}

```

`\openoutputfile`

`\openoutputfile{`⟨*filename*⟩`}{`⟨*stream*⟩`}` opens the file called ⟨*filename*⟩ and attaches it to the stream ⟨*stream*⟩ for writing. However, if the `\nofiles` command has been given the file is *not* attached to the stream. No more than one file can be attached to a stream at any given time.

```
\newcommand{\openoutputfile}[2]{%
  \outstre@mandclosed{#2}{%
    \global\@namedef{#1@filename}{#1}%
    \if@filesw
      \immediate\openout\@nameuse{#2outstre@m}=\@nameuse{#1@filename}%
    \fi
    \global\csname stre@m#2opentrue\endcsname%
    \@nameuse{atstreamopen#2}%
  }%
}

```

`\closeoutputstream`

`\closeoutputstream{`⟨*stream*⟩`}` closes the stream ⟨*stream*⟩.

```
\newcommand{\closeoutputstream}[1]{%
  \outstre@mandopen{#1}{%
    \@nameuse{atstreamclose#1}%
    \immediate\closeout\@nameuse{#1outstre@m}%
    \global\csname stre@m#1openfalse\endcsname}%
}

```

`\openinputfile`

`\openinputfile{`⟨*filename*⟩`}{`⟨*stream*⟩`}` opens the file called ⟨*filename*⟩ and attaches it to the stream ⟨*stream*⟩ for reading. The file is added to the list of files. No more than one file can be attached to a stream at any given time.

```
\newcommand{\openinputfile}[2]{%
  \IfFileExists{#1}{%                  file exists
    \instre@mandclosed{#2}{%
      \@addtofilelist{#1}%
      \global\@namedef{#1@filename}{#1}%
      \immediate\openin\@nameuse{#2instre@m}=\@nameuse{#1@filename}%
      \global\csname stre@m#2opentrue\endcsname%
      \@nameuse{atstreamopen#2}}}%
  {%                                   file not found
    \PackageError{newfile}{Can't find file #1}{\@ehc}%
  }%
}

```

`\closeinputstream`

`\closeinputstream{`⟨*stream*⟩`}` closes the stream ⟨*stream*⟩.

```
\newcommand{\closeinputstream}[1]{%
  \instre@mandopen{#1}{%
     \@nameuse{atstreamclose#1}%
     \immediate\closein\@nameuse{#1instre@m}%
     \global\csname stre@m#1openfalse\endcsname}%
}

```

`writeverbatim`

`\begin{writeverbatim}{`⟨*stream*⟩`}` writes the contents of the environment as verbatim text to the given ⟨*stream*⟩.

```
\def\writeverbatim#1{%
  \@bsphack
  \let\do\@makeother\dospecials
  \catcode`\^^M\active
  \verbatim@startline
  \verbatim@addtoline
  \verbatim@finish
  \def\verbatim@processline{%
```

```
    \immediate\write\@nameuse{#1outstre@m}{\the\verbatim@line}}%
  \verbatim@start}
\def\endwriteverbatim{\@esphack}

```

`\addtostream`

`\addtostream{`⟨*stream*⟩`}{`⟨*text*⟩`}` writes ⟨*text*⟩ to the given ⟨*stream*⟩.

```
\newcommand{\addtostream}[2]{%
  \@bsphack
  \outstre@mandopen{#1}{%
    {\let\protect\string
     \immediate\write\@nameuse{#1outstre@m}{#2}%
    }}%
  \@esphack
}

```

`\ifstre@mnoteof`
`\checkstre@mnoteof`

`\checkstre@mnoteof{`⟨*stream*⟩`}` sets `\ifstre@mnoteof` to TRUE if ⟨*stream*⟩ is not at the end of the file (i.e., it is the opposite of `\ifeof`).

```
\newif\ifstre@mnoteof
\newcommand{\checkstre@meof}[1]{%
  \stre@mnoteoftrue\ifeof\@nameuse{#1instre@m}\stre@mnoteoffalse\fi}

```

`\readstream`

`\readstream{`⟨*stream*⟩`}` reads the contents of the given ⟨*stream*⟩ as `\input` text.

```
\def\readstream#1{
  \instre@mandopen{#1}{%
    \loop \checkstre@meof{#1} \ifstre@mnoteof
      \read\@nameuse{#1instre@m} to\temptokstre@m
     \temptokstre@m
    \repeat
    }%
}

```

`\readaline`

`\readaline{`⟨*stream*⟩`}` reads what TeX considers to be one line from the given ⟨*stream*⟩ as `\input` text.

```
\def\readaline#1{
  \instre@mandopen{#1}{%
    \ifeof\@nameuse{#1instre@m}
      \PackageWarning{newfile}{No more to read from stream #1}
    \else
      \read\@nameuse{#1instre@m} to\temptokstre@m
      \temptokstre@m
    \fi
    }%
}

```

`\readverbatim`
`\stre@mverbatim@input`
`\verbatim@readstre@m`

`\readverbatim{`⟨*stream*⟩`}` reads the contents of the given ⟨*stream*⟩ as verbatim text.

The read verbatim code is a slight variation on code from the verbatim package. Most of the setup is done by the macros `\stre@mverbatim@input{`⟨*setup*⟩`}{`⟨*stream*⟩`}` and `\verbatim@readstre@m{`⟨*stream*⟩`}`. Finally, `\verbatim@read@file` is a verbatim package macro.

```
\def\readverbatim{\begingroup
  \@ifstar{\stre@mverbatim@input\relax}%
          {\stre@mverbatim@input{\frenchspacing\@vobeyspaces}}}

\def\stre@mverbatim@input#1#2{%
  \@ifstre@mopen{#2}%
    {\@verbatim #1\relax
     \def\verbatim@in@stream{\@nameuse{#2instre@m}}
     \verbatim@readstre@m{#2}\endtrivlist\endgroup\@doendpe}%
    {\PackageError{newfile}{Stream #2 is not open}{\@ehc}\endgroup}%
}

\def\verbatim@readstre@m#1{%
  \verbatim@startline
  \expandafter\endlinechar\expandafter\m@ne
  \expandafter\verbatim@read@file
  \expandafter\endlinechar\the\endlinechar\relax
  \verbatim@finish
}

```

`\plainvstream` A macro that sets `\verbatim@processline` to its default definition.

```
\newcommand{\plainvstream}{%
  \def\verbatim@processline{\the\verbatim@line\par}%
}

```

`\streamvnumfont`
`\stre@mvnfont` We need a counter for numbering lines read in verbatim.

`\streamvnumfont{`⟨*font*⟩`}` defines `\stre@mvnfont` to be ⟨*font*⟩.

```
\newcounter{streamvline}
\newcommand{\streamvnumfont}[1]{\def\stre@mvnfont{#1}}
\streamvnumfont{\footnotesize}

```

`\streamvfont`
`\verbatim@font` `\streamvfont{`⟨*font*⟩`}` defines `\verbatim@font` to use ⟨*font*⟩.

```
\newcommand{\verbatimfont}[1]{%
  \def\verbatim@font{#1%
    \hyphenchar\font\m@ne
    \let\do\do@noligs
    \verbatim@nolig@list}}
\verbatimfont{\normalfont\ttfamily}

```

`\numbervstream` `\numbervstream` puts numbers at the start of each line read verbatim.

```
\newcommand{\numbervstream}{%
  \setcounter{streamvline}{0}%
  \def\verbatim@processline{%
    \addtocounter{streamvline}{1}%
    \leavevmode
    {\stre@mvnfont \thestreamvline}\space
    \the\verbatim@line\par}%
}

```

`\marginnumbervstream`

`\marginnumbervstream` puts numbers in the margin at the start of each line read verbatim.

```
\newcommand{\marginnumbervstream}{%
  \setcounter{streamvline}{0}%
  \def\verbatim@processline{%
    \addtocounter{streamvline}{1}%
    \leavevmode
    \llap{{\stre@mvnfont \thestreamvline} \hskip\@totalleftmargin}
    \the\verbatim@line\par}%
}

```

The end of the newfile package.

```
⟨/outin⟩
```

References

[GMS94] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion.* Addison-Wesley Publishing Company, 1994.

[MG95] James Darrell McCauley and Jeff Goldberg. 'The `endfloat` package'. October, 1995.

[Pif96] Mike Piff. 'Production of solution sheets in LaTeX2e'. July, 1996.

[SRR99] Rainer Schöpf, Bernd Raichle, and Chris Rowley. 'A New Implementation of LaTeX's `verbatim` and `verbatim*` Environments'. December, 1999.

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.